



**WICHITA STATE  
UNIVERSITY**

**UNIVERSITY LIBRARIES**

## **Energy-aware scheduling with day ahead electricity pricing**

Item Type	Dissertation
Authors	Al Rubaiee, Saeed
Publisher	Wichita State University
Rights	Copyright 2015 Saeed Al Rubaiee
Download date	2026-05-13 15:34:39
Link to Item	<a href="http://hdl.handle.net/10057/12120">http://hdl.handle.net/10057/12120</a>

ENERGY-AWARE SCHEDULING WITH DAY AHEAD ELECTRICITY PRICING

A Dissertation by

Saeed Al Rubaiee

Master of Science in Engineering Management, University of South Florida, 2010

Bachelor of Science in Chemical Engineering, Tennessee Tech University, 2009

Submitted to the Department of Industrial and Manufacturing Engineering  
and the faculty of the Graduate School of  
Wichita State University  
in partial fulfillment of  
the requirements for the degree of  
Doctor of Philosophy

December 2015

© Copyright 2015 by Saeed Al Rubaiee,

All Rights Reserved

## ENERGY-AWARE SCHEDULING WITH DAY AHEAD ELECTRICITY PRICING

The following faculty members have examined the final copy of this dissertation for form and content, and recommend that it be accepted in partial fulfillment of the requirement for the degree of Doctor of Philosophy with a major in Industrial and Manufacturing Engineering.

---

Mehmet Bayram Yildirim, Committee Chair

---

Deepak Gupta, Committee Member

---

Ehsan Salari, Committee Member

---

Esra Buyuktahtakin, Committee Member

---

Visvakumar Aravinthan, Committee Member

Accepted for the College of Engineering

---

Royce Bowden, Dean

Accepted for the Graduate School

---

Kerry Wilks, Interim Dean

## ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep gratitude to my dissertation chair, Professor Mehmet Bayram Yildirim. He patiently guided me through my dissertation research, never accepting less than my best efforts. His wisdom, knowledge, and commitment to the highest standards have inspired and motivated me.

I acknowledge my committee members, Dr. Esra Buyuktahtakin, Dr. Ehsan Salari, Dr. Deepak Gupta, and Dr. Visvakumar Aravinthan, for agreeing to serve on my dissertation committee.

I also give my heartfelt appreciation to my parents and my wife, who have always supported me and encouraged me to study and get an advanced degree.

## ABSTRACT

This dissertation develops a multiobjective framework for energy-aware operations planning in order to optimize the energy cost and scheduling objectives. Using frameworks developed for different operational settings, the decision-maker has the opportunity to implement a schedule with an energy cost and other scheduling performance measures in the presence of dynamic electricity prices.

In this dissertation, mixed-integer mathematical models are developed and solved for multiobjective problems (on single or parallel machines) with minimization of both scheduling criteria and energy cost. The problems considered in this dissertation are as follows: non-preemptive single machine to minimize total tardiness and total energy cost; preemptive single machine to minimize total completion time and total energy cost; preemptive single machine with non-preemptive sequence-independent setup time to minimize total completion time and total energy cost; and non-preemptive parallel machine with load balancing, total completion time, and total energy cost.

The proposed models were solved using exact solution methods such as weighted sum and  $\varepsilon$ -constraint, as well as metaheuristic approaches such as genetic algorithms, the ant colony optimization algorithm, and the greedy randomized adaptive search to obtain good approximate sets of non-dominated solutions in a timely fashion. Based on his/her preference, the decision-maker can use a selection method such as the technique for order preference by similarity to ideal solution (TOPSIS) or multiobjective optimization on the basis of ratio analysis (MOORA) to implement a schedule among all non-dominated solutions that minimizes some other secondary objectives.

## TABLE OF CONTENTS

Chapter	Page
1. INTRODUCTION .....	1
1.1 Overview.....	1
1.2 Energy Outlook in U.S.....	3
1.3 Objective of Dissertation .....	8
1.4 Contributions of Dissertation.....	8
1.4.1 Summary of Modeling Contributions.....	10
1.4.2 Algorithmic Contributions .....	11
1.5 Outline of Dissertation.....	12
1.6 Summary.....	14
1.7 References.....	14
2. LITERATURE REVIEW .....	17
2.1 Introduction.....	17
2.2 Single-Objective and Single Machine Scheduling .....	19
2.2.1 Minimization of Total Completion Time .....	21
2.2.2 Minimization of Total Tardiness .....	22
2.3 Single Machine Scheduling with Preemptive and Setup Times .....	23
2.3.1 Single Machine Scheduling with Setup Times.....	23
2.4 Single-Objective Parallel Machine Scheduling .....	25
2.4.1 Minimization of Completion Time Objectives .....	25
2.4.2 Minimization of Deviation Using Load Balancing .....	26
2.5 Energy-Aware Operations Management.....	27
2.5.1 Energy-Aware Production Planning.....	27
2.5.2 Energy-Aware Production Planning with Classical Single Machine Scheduling .....	31
2.5.3 Peak-Load Management.....	33
2.6 Mixed-Integer Programming .....	36
2.7 Multiobjective Optimization.....	38
2.7.1 Multiobjective Combinatorial Optimization .....	40
2.8 Methods for Solving Multiobjective Combinatorial Optimization Problems.....	43
2.8.1 Exact Algorithms.....	44
2.8.2 Heuristic Algorithms .....	47
2.9 Complexity of Algorithms .....	51
2.10 Summary.....	53
2.11 References.....	53
3. MULTIOBJECTIVE MIXED-INTEGER MATHEMATICAL PROGRAMMING MODEL TO MINIMIZE TOTAL ENERGY COST AND TOTAL TARDINESS ON NON-PREEMPTIVE SINGLE MACHINE .....	65

TABLE OF CONTENTS (continued)

Chapter	Page
3.1	Introduction..... 65
3.2	Multiobjective Mixed-Integer Programming Model to Minimize Energy Cost and Total Tardiness (MMIP-MEC-TT) ..... 70
3.2.1	Notation and Formulation .....71
3.3	Using Weighted Sum Method to Solve MMIP-MEC-TT Problem ..... 73
3.4	Using Genetic Algorithms Based on (GA-1), (GA-2), and (GA-3) to Solve MMIP-MEC-TT Problem ..... 75
3.4.1	Representation (Coding).....77
3.4.2	Crossover.....77
3.4.3	Mutation .....78
3.4.4	Evaluation of Chromosome: Obtaining Approximate Pareto Front of Chromosome Using Reduced MMIP-MEC-TT .....78
3.4.5	Evaluation of Fitness Functions .....80
3.4.5.1	Fitness Function and Value for Chromosomes in GA-1 .....80
3.4.5.2	Fitness Function and Value for Chromosomes in GA-2 .....81
3.4.5.3	Selection Process in GA-1 and GA-2.....82
3.4.5.4	Fitness Function and Value for Chromosomes in GA-3 .....83
3.4.5.5	Selection Process in GA-3.....84
3.4.6	Stopping Criterion in GA-1, GA-2, and GA-3 .....85
3.5	Experimental Design and Evaluation..... 85
3.5.1	Generating Experimental Data .....86
3.5.2	Solving MMIP-MEC-TT Using Weighted Sum Method.....87
3.5.3	Genetic Algorithm Parameter Fine-Tuning.....89
3.5.3.1	Choosing Best Crossover Rate .....91
3.5.3.2	Choosing Generation Size .....92
3.5.3.3	Choosing Optimal Number of Generations.....92
3.5.3.4	Choosing Radius in Fitness Function for GA-2 .....92
3.6	Comparison of Proposed Genetic Algorithms (GA-1, GA-2, and GA-3) ..... 93
3.7	Comparison of GA-3 and Weighted Sum Method ..... 95
3.8	Comparison of Performance of GA-3 Algorithm with Random Number of Generations ..... 96
3.9	Using Sub-Optimal Solutions in Chromosome Evaluation ..... 98
3.10	Selecting Solution from Pareto Front for Implementation: A Case Study ..... 102
3.10.1	Selection of Alternative Using TOPSIS Method .....102
3.11	Conclusion ..... 105
3.12	References..... 107
4.	MULTIOBJECTIVE ANT COLONY ALGORITHM FOR PREEMPTIVE SCHEDULING AND ENERGY-COST OPTIMIZATION FOR SINGLE MACHINE PROBLEM WITH TOTAL COMPLETION TIME.....111

TABLE OF CONTENTS (continued)

Chapter	Page
4.1	Introduction..... 111
4.2	Multiobjective Mixed-Integer Programming Model to Minimize Energy Cost and Total Completion Time (MMIP-MEC-TCT)..... 117
4.2.1	Notation and Formulation .....118
4.3	Weighted Sum Method to Solve MMIP-MEC-TCT Problem..... 120
4.4	Solution to MMIP-MEC-TCT Model Using Multiobjective Ant Colony Algorithms Based on (ACO-DR) and (ACO-DRC) ..... 121
4.4.1	Construction of Graph .....122
4.4.2	Constraints.....123
4.4.3	Pheromone Trail.....123
4.4.4	Heuristic Information .....124
4.4.5	Solution Construction.....124
4.4.6	State Transition Rule .....126
4.4.7	Local Updating Rule .....127
4.4.8	Obtaining Approximate Pareto Front of Tour Using Reduced MIP .....127
4.4.9	Evaluation of Fitness Functions .....132
4.4.9.1	Fitness Function and Value for Tours in ACO-DR.....132
4.4.9.2	Fitness Function and Value for Tours in ACO-DRC .....133
4.4.10	Selection Process in ACO-DR .....134
4.4.11	Selection Process in ACO-DRC.....135
4.4.12	Global Updating Rule.....136
4.4.13	Stopping Criteria .....137
4.5	Experimental Design and Evaluation..... 137
4.6	Generation of Experimental Data ..... 137
4.6.1	Solving MMIP-MEC-TCT Using Weighted Sum Method .....138
4.6.2	Parameter Fine-Tuning.....139
4.6.2.1	Choosing Appropriate $\alpha$ , $\beta$ and $q_0$ Parameters.....140
4.6.2.2	Choosing Population Size .....140
4.6.2.3	Choosing Appropriate $\rho_1$ and $\rho_g$ Parameters.....141
4.7	Comparison of ACO Algorithm with-and-without Theorem 1 ..... 142
4.8	Comparison of ACO-DR Algorithm, ACO-DRC Algorithm, and WSM..... 143
4.9	Impact of Electricity Rate Structure on Performance of Metaheuristics ..... 147
4.10	Selecting Solution from Pareto Front for Implementation: A Case Study ..... 149
4.11	Conclusion ..... 154
4.12	References..... 156
5.	MULTIOBJECTIVE MIP OPTIMIZATION MODEL FOR PARALLEL MACHINE SCHEDULING WITH LOAD BALANCING, TOTAL ENERGY COST, AND TOTAL COMPLETION TIME.....160
5.1	Introduction..... 160

TABLE OF CONTENTS (continued)

Chapter	Page
5.2	Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost, Total Completion Time, and Deviation from Load Balancing on Parallel Machines MMIP-MO-TE-TC-LB-PM..... 166
5.2.1	Notation and Formulation .....167
5.3	Using $\epsilon$ -constraint Method to Solve MIP-MO-TE-TC-LB-PM Problem..... 170
5.4	Solution to MIP Model Using a Multiobjective GRASP and GA-DRC Metaheuristics..... 172
5.4.1	The Proposed Multiobjective GRASP Algorithm.....173
5.4.1.1	Construction Phase.....175
5.4.1.2	Local Search Procedure.....177
5.4.2	The Proposed Multiobjective GA-DRC Algorithm .....178
5.4.2.1	GA-DRC Overview.....179
5.4.2.2	Representation (Coding).....179
5.4.2.3	Crossover.....180
5.4.2.4	Mutation .....181
5.4.2.5	Evaluation of Chromosome: Obtaining Approximate Pareto Front .....182
5.4.2.6	Evolution of Fitness Function .....182
5.4.2.7	Selection Operator.....183
5.5	Experiment Design and Evaluation ..... 184
5.5.1	Generation of Experiments Data .....184
5.6	Solving MMIP-MO-TE-TC-LB-PM Using $\epsilon$ -constraint Method..... 185
5.7	Parameter Fine-Tuning ..... 187
5.8	Comparison of GRASP, GA-DRC Algorithms and $\epsilon$ -constraint Method ..... 191
5.9	Conclusion ..... 195
5.10	Reference ..... 197
6.	MINIMIZING TOTAL COMPLETION TIME AND ENERGY COST ON PREEMPTIVE SINGLE MACHINE WITH SEQUENCE-INDEPENDENT SETUP TIMES.....201
6.1	Introduction..... 201
6.2	Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Completion Time MMIP-MO-TE-TC-SM-SIST..... 207
6.2.1	Notation and Formulation .....207
6.3	Using $\epsilon$ -constraint Method to Solve MIP-MO-TE-TC-LB-PM Problem..... 211
6.4	Solution to MIP Model Using Multiobjective GADR and GARC Algorithms .... 213
6.4.1	Representation (Coding).....214
6.4.2	Crossover.....215
6.4.3	Mutation .....216
6.4.4	Evaluation of Chromosome: Obtaining Approximate Pareto Front of Chromosome Using Reduced MIP .....217

## TABLE OF CONTENTS (continued)

Chapter	Page
6.4.5	Fitness Function and Value for Chromosomes in GARD Algorithm ...219
6.4.5.1	Selection Process in GARD Algorithm.....220
6.4.6	Fitness Function and Value for Chromosomes in GARC Algorithm .221
6.4.6.1	Selection Process in GARC Algorithm .....222
6.5	Experimental Design and Evaluation..... 223
6.5.1	Generation of Experimental Data.....224
6.5.2	Solving MMIP-MO-TE-TC-SM-SIST Using $\epsilon$ -constraint Method... 225
6.6	Parameter Fine-Tuning ..... 226
6.7	Comparison of GADR, GARC Algorithms, and $\epsilon$ -constraint Method..... 230
6.8	Conclusion ..... 236
6.9	References..... 237
7.	CONCLUSIONS AND FUTURE RESEARCH .....241
7.1	Conclusions of This Dissertation ..... 241
7.1.1	Conclusion in Chapter 3 .....241
7.1.2	Conclusion in Chapter 4.....242
7.1.3	Conclusion in Chapter 5.....243
7.1.4	Conclusion in Chapter 6.....244
7.1.5	Summary of Contributions of This Dissertation .....245
7.2	Future Research ..... 246
7.2.1	Maintenance Planning and Scheduling with Energy-Aware Scheduling .....246
7.2.2	Reliability Studies in Machine Inserting On/Off Time with Energy- Aware Scheduling .....248
7.2.3	Multiobjective Scheduling with Sequence-Dependent Setup Times ....249
7.2.4	Simple Heuristic for Reducing Number of Variables and Constraints, and Generating Approximate Pareto Front .....250
7.2.5	Consideration of Multi-Machine, Multiobjective Scheduling with Energy-Aware Scheduling .....251
7.3	References..... 251

## LIST OF TABLES

Table	Page
1.1 Contribution Summary.....	9
2.1 Review of Energy-Aware Operations Management Studies .....	35
3.1 Features of Multiobjective Optimization Instances .....	88
3.2 Evaluation of Non-Dominated Solutions for GAs.....	93
3.3 Results of GA-3 and WSM.....	95
3.4 Decision (Pay-Off) Matrix ( $\theta_{i\phi}$ ).....	103
3.5 Similarity Index ( $D_i$ ) Values .....	105
4.1 Numerical Example .....	132
4.2 Results of WS-MMIP-MEC-TCT.....	139
4.3 Candidate Set .....	141
4.4 GD Metric Performance Depending on Parameters and Number of Jobs .....	141
4.5 ACO-DRC with-and-without Theorem 1 .....	143
4.6 Results of the WSM and Applied Algorithms .....	145
4.7 Results of the ACO-DRC Algorithm.....	148
4.8 Decision Matrix with Initial Values.....	150
4.9 Ranking of Alternatives Applying Ratio System.....	153
5.1 Features of Multiobjective Optimization Instances and Results of $\epsilon$ -constraint Method .....	187
5.2 Performance Depending on $\alpha$ Parameter and PHF = 0.75 .....	188
5.3 Performance Depending on $\alpha$ Parameter and PHF = 0.50 .....	188
5.4 Performance Depending on Crossover Rates and PHF = 0.75 .....	189
5.5 Performance Depending on Crossover Rates and PHF = 0.5 .....	190

LIST OF TABLES (continued)

Table	Page
5.6 Performance Depending on Number of Individuals in a Generation and PHF = 0.75 ....	190
5.7 Performance Depending on Number of Individuals in a Generation and PHF = 0.5 .....	190
5.8 Performance Depending on Number of Generation and PHF = 0.75 .....	191
5.9 Performance Depending on Number of Generations and PHF = 0.5 .....	191
5.10 Pairwise Comparison of GRASP, GA-DRC Algorithms and $\epsilon$ -constraint Method .....	193
6.1 Numerical Example .....	215
6.2 Features of Multiobjective Optimization Instances and Results of $\epsilon$ -constraint Method .....	226
6.3 Performance Depending on Crossover Rates and PHF = 0.75 and 0.5 .....	229
6.4 Performance Depending on Number of Individuals in a Generation and PHF = 0.75 and 0.5.....	229
6.5 Performance Depending on Number of Generations and PHF = 0.75 and 0.5.....	229
6.6 Pairwise Comparison of GARD, GARC Algorithms and $\epsilon$ -constraint Method .....	235

## LIST OF FIGURES

Figure	Page
1.1 Primary energy use by end-use sector, 2011–2040 (quadrillion Btu) [5].....	3
1.2 Electricity sales and power sector generating capacity, 1949–2040 (indexes, 1949 = 1.0) [5] .....	4
1.3 Total energy production and consumption, 1980–2040 (quadrillion Btu) [8].....	5
1.4 U.S. electricity demand growth, 1950–2040 (% , three-year moving average) [5].....	6
2.1 Efficient set (left) and dominated set (right).....	41
2.2 Non-dominated set with ideal, and anti-ideal .....	42
2.3 Relation between objective function value vectors .....	43
3.1 Flowchart of GA-1, GA-2, and GA-3 .....	76
3.2 Crossover operator .....	77
3.3 Mutation operator.....	78
3.4 Process of chromosome selection for generating next population.....	82
3.5 Non-dominated levels and computation of crowding distance.....	84
3.6 Process of selection of chromosomes for generating the next population.....	85
3.7 Prices signal (\$/kWh) vs. time (hr).....	87
3.8 Non-dominated set of solutions for instances 3 and 4 by Using WSM.....	88
3.9 Execution time vs. number of jobs with PHF = 0.75.....	89
3.10 Execution time vs. number of jobs with PHF = 0.5.....	89
3.11 Measure of SSC performance of set of non-dominated solutions .....	91
3.12 MID performance relative to number of jobs: (a) crossover rate, (b) number of individuals, (c) number of generations, and (d) sharing parameters.....	91

LIST OF FIGURES (continued)

Figure	Page
3.13	Box-and-whisker plot of GAs performance measures with 95% median notch.....94
3.14	Non-dominated solutions in jobs 40 from GAs in Table 3.2 and WSM in Table 3.3 .....94
3.15	Performance metrics for GA-3 and WSM vs number of generations in GA-3 .....97
3.16	Performance metrics for GA-3-MOTL, GA-3, and WSM vs number of generations .....100
3.17	Pareto front and selected solution by TOPSIS.....102
4.1	Structure of MMIP-MEC-TCT-MOACOA .....122
4.2	Construction graph for ACO algorithms of size $n = 5$ .....123
4.3	A pairwise interchanging of jobs $j$ and $i$ .....130
4.4	Theorem 1 in sequence of individual time slots for different jobs over 12 units of planning horizon .....133
4.5	Non-dominated levels and computation of crowding distance.....134
4.6	Determining fitness function of each tour .....135
4.7	Process of tour selection for generating next ant population .....136
4.8	Price signal (\$/kWh) vs. time (hr).....138
4.9	Measure of SSC performance of a combined set of non-dominated solutions.....146
4.10	Pareto exact and approximation solutions obtained by WSM and applied algorithms, respectively, for instance 2 from Table 4.6.....147
4.11	Pareto fronts for 40 random jobs with optimized parameter and optimal Pareto front with MOORA selection .....149
4.12	Diagram of MOORA .....150
5.1	Illustration of extreme points for three-objective problem .....172
5.2	Multiobjective GRASP algorithm.....147

LIST OF FIGURES (continued)

Figure	Page
5.3 The distance between a solution and the non-dominated solutions.....	176
5.4 Illustration of job neighborhood .....	178
5.5 Flow chart of GA-DRC.....	180
5.6 Illustration of the crossover operator .....	181
5.7 Illustration of the mutation operator .....	182
5.8 Non-dominated levels and computation of crowding distance.....	183
5.9 Process of selection of chromosomes for generating the next population.....	184
5.10 Prices signal (\$/kWh) vs. time (hr.) (i.e., example of time-of-use tariffs).....	185
5.11 Pareto front for problem $n = 30$ and $PHF = 0.5$ using $\epsilon$ -constraint method, GA-DRC, and GRASP .....	195
6.1 Crossover operator .....	216
6.2 Mutation operator.....	217
6.3 Process of selection of chromosomes for generating the next population.....	221
6.4 Non-dominated levels and computation of crowding distance.....	222
6.5 Process of selection of chromosomes for generating the next population.....	223
6.6 Prices signal (\$/kWh) vs. time (hr.) (i.e., example of time-of-use tariffs).....	224
6.7 Measure of performance of a set of non-dominated solutions.....	227
6.8 Pareto fronts for problem $n = 30$ and $PHF = 0.75$ using $\epsilon$ -constraint method, GARC, and GARD .....	235

## LIST OF ABBREVIATIONS

ACO	Ant Colony Optimization
ACO-DR	Ant Colony Optimization Based on Dominance Rank
ACO-DRC	Ant Colony Optimization Based on Dominance Rank and Crowding Distance Comparison
CPU	Central Processing Unit
DiM	Diversification Metric
ER	Error Rate
ESS	Efficient Set Spacing
GA	Genetic Algorithm
GA-1	Genetic Algorithm Based on Dominance Ranking Procedure
GA-2	Genetic Algorithm Based on Weighted Sum Aggregation
GA-3	Genetic Algorithm Based on Dominance Ranking Procedure and Crowding Distance Comparison
GA-3-MOTL	Genetic Algorithm-3-MIP Optimizer Time Limit
GADR	Genetic Algorithm Based on Dominance Ranking Procedure
GA-DRC	Genetic Algorithm Based on Dominance Rank Procedure and Crowding Distance Comparison
GAMS	General Algebraic Modeling System
GARC	Genetic Algorithm Based on Dominance Ranking Procedure and Crowding Distance Comparison
GD	Generational Distance
GRASP	Greedy Random Adaptive Search Procedure
MCDM	Multiple-Criteria Decision-Making
MID	Mean Ideal Distance

## LIST OF ABBREVIATIONS (continued)

MIP	Mixed-Integer Programming
MMIP-MEC-TCT	Multiobjective Mixed-Integer Programming Model to Minimize Energy Cost and Total Completion Time
MMIP-MEC-TCT-LB-PM	Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost, Total Completion Time, and Load Balancing on a Parallel Machine
MMIP-MEC-TCT-MOACOA	Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Completion Time Using Multiobjective Ant Colony Optimization Algorithm
MMIP-MEC-TT	Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Tardiness
MMIP-MEC-TT-R	Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Tardiness-Reduced
MMIP-MO-TE-TC-SM-SIST	Multiobjective Mixed-Integer Mathematical Model to Minimize Total Energy Cost and Total Completion Time on a Single Machine with Sequence-Independent Setup Times
MMIP-MO-TE-TC-SM-SIST-R	Multiobjective Mixed-Integer Mathematical Model to Minimize Total Energy Cost and Total Completion Time on a Single Machine with Sequence-Independent Setup Times-Reduced
MOCOP	Multiobjective Combinatorial Optimization Problem
MOMP	Multiobjective Mathematical Programming
MOORA	Multiobjective Optimization on Basis of Ratio Analysis
No. Pareto	Number of Solutions on Pareto Front
NP-Hard	Non-deterministic Polynomial-Time Hard (No Known Algorithm Can Solve This Problem in Polynomial Time)
NSGA-II	Non-Dominated Sorting Genetic Algorithm
QM	Quality Metric
RCL	Restricted Candidate List
SAC	Size of Area Covered

LIST OF ABBREVIATIONS (continued)

SSC	Size of Space Covered
SPT	Shortest Processing Time
SRPT	Shortest Remaining Processing Time
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution
WSM	Weighted Sum Method
WS-MMIP-MEC-TCT	Weighted Sum Method Used to Solve Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Completion Time
WS-MMIP-MEC-TCT-R	Weighted Sum Method Used to Solve Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Completion Time-Reduced
WS-MMIP-MEC-TT	Weighted Sum Method Used to Solve Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Tardiness
WS-MMIP-MEC-TT-R	Weighted Sum Method Used to Solve Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Tardiness-Reduced

## LIST OF NOTATIONS

$N$	Set of Jobs
$j$	Jobs, $j \in N$
$i$	Jobs, $i \in N$
$T$	Set of Planning Horizon
$t$	Time Interval, $t \in T$
$t'$	Time Interval, $t' \in T$
$M$	Big M
$E_t$	Electric Price Signal at $t$
$d_j$	Due Date of Job $j$
$p_j$	Process Time of Job $j$
$C_j$	Completion Time of Job $j$
$\ell_j$	Tardiness of Job $j$ , $\ell_j = \max\{0, C_j - d_j\}$
$x_{jt}$	Equals 1 if Job Starts at Time $t$
$\delta_{ji}$	Equals 1 if Job $j$ is Processed before Job $i$
$\gamma_{jt}$	Equals 1 if $x_{jt}$ Equals 0
$h_{jt'}$	Equals 1 if Job Processes Overtime $t'$
$k$	Number of Criteria Functions
$w_k$	Weights
$z_k^N$	Nadir Point (Upper Bound)
$z_k^U$	Utopia Point (Lower Bound)
$\xi$	Generation

## LIST OF NOTATIONS (continued)

$\iota$	Solution
$\#_{\iota\xi}$	Number of Solutions in Generation
$\Theta_{\iota\xi}$	Fitness Value
$d_{\varepsilon_1\varepsilon_2}$	Normalized Distance between Two Solutions
$\psi$	Chromosome
P	Total Processing Time
TF	Tardiness Factor
RDD	Relative Range of Due Date
<i>PHF</i>	Planning Horizon Factor
$\mathcal{D}$	Total Number of Non-Dominated Solutions
$\sigma_{sh}$	Sharing Parameter
$T'$	Subset of Planning Horizon
$C_j$	Completion Time of Job $j$
$\tau_{(i,u)}$	Pheromone Trail
$x$	Decision Vector
$X$	Parameter Space
$G$	Complete Graph
$A$	Arc
$\eta_{(a,u)}$	Inverse of Heuristic Information
$(d_{a,u})$	Heuristic Information
$S_k(\alpha)$	Set of Feasible Nodes

## LIST OF NOTATIONS (continued)

$\beta$	Relative Importance
$q$	Value Chosen Randomly with Uniform Probability in $[0,1]$
$\alpha$	Node
$b$	Node
$\gamma$	Artificial Ant
$\alpha$	Parameter to Adjust Randomness of GRASP Algorithm
$\varepsilon$	Eps-Parameter
$Q$	Number of Solutions

# CHAPTER 1

## INTRODUCTION

### 1.1 Overview

The goal of many industrial sectors is to decrease the cost of production by any means possible while satisfying environmental regulations and ensuring quality and customer satisfaction [1]. In the last 60 years, the consumption of energy by the industrial sector has virtually doubled [2]. Today, industrial sectors, which include manufacturing, agriculture, mining, and construction, consume about half of the world's energy [3]. The U.S. consumes about one-third of the world's energy used, contributes about 28% of greenhouse gas emissions [4], and spends about \$200 billion per year on energy costs alone [5]. In China, manufacturers consume about 50% of the entire electricity produced in the country and contribute to at least 26% of the carbon dioxide emissions [6]. In Germany, manufacturers are responsible for about 47% of the total national electricity consumption and generate about 20% of the total carbon dioxide emissions [7].

A key feature of this dissertation is the opportunity for sectors that have access to dynamic electricity prices, which vary from hour to hour, to be able to respond to these dynamic prices by reducing the power demand when prices are high and increasing the power demand when prices are low. Consequently, they would pay less for electricity consumption over the long run.

In many facilities, different jobs arrive at the resource (machine) to be processed. Most jobs are released to the machine with known deterministic processing times and due dates (they are part of an order, and it is known when the facility will receive these jobs to be processed). Also, characteristics of the machine, such as setup times on the machine between two different jobs, are known. Using the framework developed in this dissertation to schedule the machines could result

in a significant reduction in total energy cost while maintaining a suitable level of performance according to other scheduling objectives, such as total completion time, total tardiness, etc.

Energy-aware operations planning has been an area of interest, especially in computer systems. From the perspective of production planning and scheduling in manufacturing applications, developing decision models including the energy cost aspect is critical in order to achieve an energy-efficient system. When a schedule is designed, usually the energy cost of that schedule is assumed to be constant or ignored. The problem addressed in this dissertation focuses on including the total energy cost objective into the scheduling problem or, more precisely, in the developing methods (e.g., metaheuristics and optimization algorithms) that consider energy cost as a way to minimize the schedule.

The objective of this dissertation is to develop a multiobjective framework for energy-aware operations planning in order to optimize both energy cost and the scheduling objective. Designing efficient methods for solving multiobjective scheduling problems with energy cost minimization as one of the objectives may result in solutions that not only save money but also help the environment by using energy-conscious scheduling techniques. Some scheduling objectives that may be considered in this framework are total completion time (sum of the completion time of all jobs), total tardiness (sum of the tardiness of all jobs where tardiness is defined as the lateness of a job with due dates), load balancing for multi-machine problems, and sequence-dependent setup times. The output of the framework may be when to process a job on the machine(s), when to schedule setup times (i.e., the work it takes to prepare the machine to perform a job), and when to keep machines off and on to minimize the total energy cost and scheduling criteria at the same time.

## 1.2 Energy Outlook in U.S

Over the last decade, industrial sectors have focused on being energy-efficient and environmentally sustainable. Indeed, the United States' industrial carbon dioxide emissions will begin to surpass emissions from the transportation sector in the middle of the next decade for the first time since the late 1990s [8]. Additionally, approximately one-third of the total delivered energy in the U.S. in 2012, 23.6 quadrillion Btu, was consumed in the industrial sector, which includes manufacturing, agriculture, construction, and mining. Consequently, the total industrial energy consumption delivered will grow to 30.2 quadrillion Btu in 2040, which is 1.5 quadrillion Btu or 5%, higher than the amount reported by the *Annual Energy Outlook's 2013* (AEO 2013) reference case projection [8]. Therefore, the U.S. industrial sector is set to become the largest energy consuming sector by 2018 and will remain so for the rest of the projection period [8].

Figure 1.1 shows that the total primary energy consumption, including fuels used for electricity generation, will grow by 0.3 percent per year from 2011 to 2040, to 107.6 quadrillion Btu in 2040 [5].

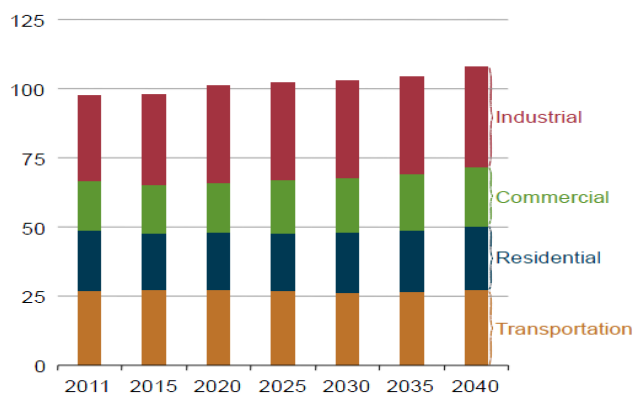


Figure 1.1: Primary energy use by end-use sector, 2011–2040 (quadrillion Btu) [5]

The largest growth, 5.1 quadrillion Btu from 2011 to 2040, will be in the industrial sector. For example, the energy used for heat and power in energy-intensive industries will grow from 11.5 quadrillion Btu in 2012 to 13.1 quadrillion Btu in 2040, averaging 0.9% increase per year from

2012 to 2025 and 0.1% increase per year from 2025 to 2040 [5]. The industrial sector was more severely affected than other end-use sectors by the 2007–2009 economic downturns. The increase in industrial energy consumption from 2008 through 2040 is expected to be 3.9 quadrillion Btu [5].

On the other hand, growth in electricity-generating capacity parallels growth in the end-use demand for electricity. Unexpected variations in demand or sudden changes affecting capacity investment decisions can, however, cause imbalances that may take years to resolve. Figure 1.2 summarizes the relative changes in total generating capacity and electricity demand during the 1950s through the 2040s.

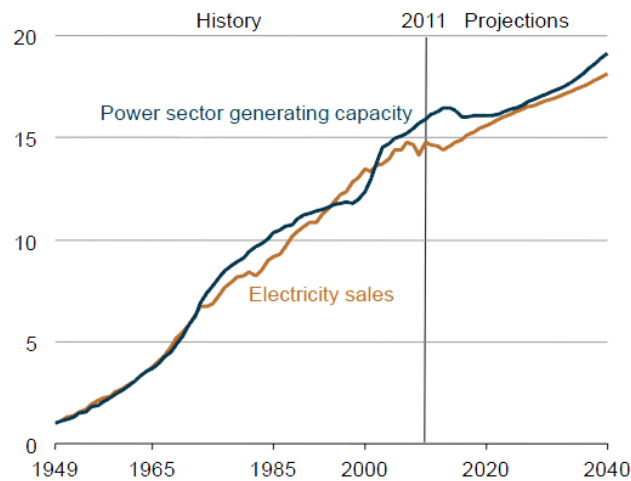


Figure 1.2: Electricity sales and power sector generating capacity, 1949–2040 (indexes, 1949 = 1.0) [5]

It is obvious that in the 2000s, demand and capacity were imbalanced and followed by a period of growth in electricity demand that exceeded capacity growth. Then, in the late 2000s, a boom in construction of new natural gas-fired plants commenced, balancing capacity. As a result, the industrial sectors are now required to not only adopt a strategy with minimal environmental impact but also consider sustainability practices as an incentive for innovation. Furthermore, industrial sectors must become more aware of whether the products they produce are considered a sustainable

source. Finally, corresponding production operations must guarantee minimum environmental impact. Denying sustainability practices in the long term will increase the electricity-generating capacity and energy prices.

Figure 1.3 shows the United States’ total energy production and consumption and the net imports of energy decline in absolute terms. This decline reflects the increased domestic production of natural gas along with a reduction in demand resulting from rising energy prices [8].

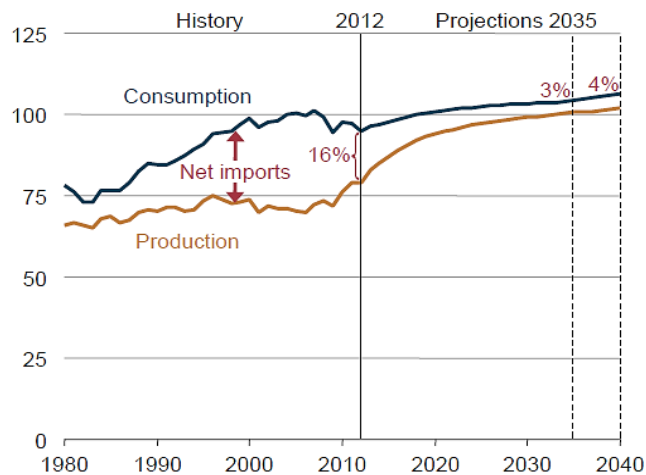


Figure 1.3: Total energy production and consumption, 1980–2040 (quadrillion Btu) [8]

Figure 1.4 illustrates the United States’ electricity demand growth remaining relatively slow, because the increasing demand for electricity services is satisfied by efficiency gains from new appliance efficiency standards and investments in energy-efficient equipment [5]. Total electricity demand is projected to grow by 28 percent (0.9 percent per year), from 3,839 billion kilowatt-hours in 2011 to 4,930 billion kilowatt-hours in 2040. Electricity sales to the industrial sector will grow by 17 percent alone, to 1,145 billion kilowatt-hours in 2040 [5]. Consequently, the growing electricity sales have exercised more force on the industrial sectors resulting from rising energy prices at the same time. The industrial sectors, therefore, have to reduce energy consumption in order to reduce energy costs and become more environmentally friendly.

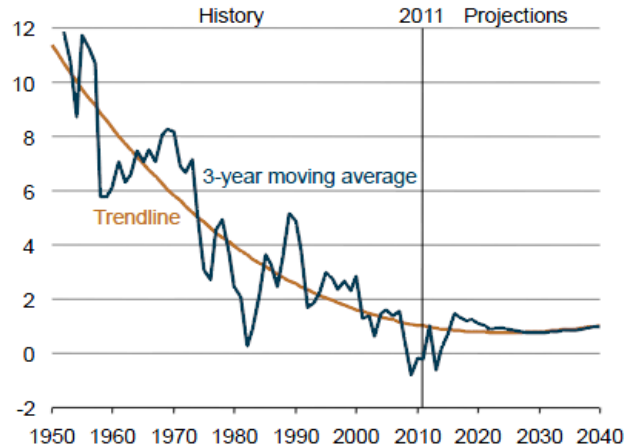


Figure 1.4: U.S. electricity demand growth, 1950–2040 (% , three-year moving average) [5]

In fact, retail electricity prices use differential pricing policies with significantly higher power costs, depending on whether the power demand exceeds a base load [9], or fuel prices increase. However, the relationship between retail electricity prices and demand capacity or fuel prices is complex, and many factors (e.g., share of fuel generation in a region, level of cost associated with distribution systems, and number of customers who purchase power directly from wholesale power markets) influence the degree to which and the timeframe over which they are related. As a result, the impacts from changing those factors can significantly affect retail electricity prices. Therefore, balancing energy-efficiency, power demand, and production targets is challenging. The research presented in this dissertation takes an energy-aware operations production planning and scheduling approach in order to minimize total energy cost within a manufacturing facility while considering multiple objectives including total energy cost and scheduling criteria at the same time.

For all these reasons, it is very important to utilize energy resources in such a way that the total energy cost is minimized. Limiting energy resources and increasing the population leads to higher energy prices. In the long term, this trend is expected to continue as a result of growing demand around the world. Drake et al. [10] show that machine start-up and machine idling

consume a significant amount of energy. For example, compressors in an industrial setting consume about 50% of the maximum power when they are idle [11]. Gutowski et al. [12] point out that in a mass production environment, such as the Toyota Motor Corporation, 85.2% of the energy is consumed for activities that are not directly related to production processes. Mouzon et al. [3] point out that there is a significant amount of energy savings if non-bottleneck machines are turned off instead of kept in idle mode for a long period of time. Fang et al. [13] consider the problem of processing jobs under time-of-use electricity tariffs. They assume that with time-of-use tariffs, energy prices vary hourly, which is typically announced a day ahead or an hour ahead of their production processes (i.e., price structures are used to shift electricity use from on-peak hours to off-peak hours).

The initial steps in reducing the energy demand for industrial production are to gather the current energy costs and to analyze the potential to reduce energy costs [14]. The possibility for energy saving in industrial sectors lies not only in continuously increasing the energy-efficiency of production processes, logistics, and products' life cycles, but also in developing novel energy monitoring and management approaches [15]. Industrial sectors, therefore, are required to have detailed knowledge of the energy behavior of their components, energy consumption of their production process, and methods to analyze and evaluate design alternatives. Understanding these facets will allow industrial managers to respond accurately about the required energy at which time and place, and thus support their decisions with respect to dynamic changes in production planning, energy prices, and environmental impacts.

Recently, several approaches have been proposed to minimize energy cost, such as exploiting the variable pricing of electricity. It is well known that electricity is an efficient and safe way to move energy from one place to another [16]. For example, advanced metering and

monitoring can help manage the balance between the supply and demand of electricity, and the reliability and efficiency of electrical power grids (i.e., providing variable pricing) [13].

Over the past five years, investment in energy-efficiency in most countries has largely been driven by higher energy prices. Thus, the increasing price of energy has exercised more force on industrial sectors. However, utilizing the dynamic pricing and significant limitations on peak energy will allow detailed manufacturing scheduling and control systems to have considerable influence on energy consumption and associated costs [17].

### **1.3 Objective of Dissertation**

The main objective of this dissertation is to develop multiobjective solution frameworks to minimize the total energy cost in an industrial environment under different constraints and scheduling preferences. These frameworks will be useful in minimizing energy costs in a single- or multiple-machine environment where multiple jobs are available at the same time.

The proposed frameworks use the multiobjective optimization theory to minimize the total energy cost as well as other scheduling criteria. They provide decision-makers or production managers with new energy-efficient approaches that they can implement with their usual scheduling objectives to plan jobs on machines.

### **1.4 Contributions of Dissertation**

The objective of this dissertation is to design a mixed-integer multiobjective mathematical problem to minimize the total energy cost and scheduling objectives. The contributions of this dissertation are summarized in Table 1.1.

Table 1.1: Contribution Summary

Environment	Objective	Mathematical Model	Solution Algorithm	Performance Measurement	Multiple-Criteria Decision Analysis
Non-preemptive Single Machine	Total Tardiness	MIP	GA	No. Pareto DiM MID	TOPSIS
	Total Energy Cost		WSM	QM CPU	
Preemptive Single Machine	Total Completion Time	MIP	ACO	GD ESS SSC	MOORA
	Total Energy Cost		WSM		
Non-preemptive Parallel Machine	Total Completion Time	MIP	GRASP	No. Pareto MID QM CPU	
	Total Energy Cost		GA		
	Load Balancing		$\epsilon$ -constraint Method		
Preemptive Single Machine with Non-preemptive Sequence-Independent Setup Time	Total Completion Time	MIP	GA	No. Pareto SAC GD QM ER CPU	
	Total Energy		$\epsilon$ -constraint Method		

The abbreviations in Table 1.1 are defined as follows:

- **ACO:** Ant Colony Optimization
- **CPU:** Central Processing Unit
- **DiM:** Diversification Metric
- **ER:** Error Rate
- **ESS:** Efficient Set Spacing

- **GA:** Genetic Algorithm
- **GD:** Generational Distance
- **GRASP:** Greedy Random Adaptive Search Procedure
- **MID:** Mean Ideal Distance
- **MIP:** Mixed-Integer Programming
- **MOORA:** Multiobjective Optimization on the Basis of Ratio Analysis
- **No. Pareto:** Number of Pareto Front
- **QM:** Quality Metric
- **SAC:** Size of Covered Area
- **SSC:** Size of the Space Covered
- **TOPSIS:** Technique for Order Preference by Similarity to Ideal Solution
- **WSM:** Weighted Sum Method

#### **1.4.1 Summary of Modeling Contributions**

The consideration of energy cost together with a scheduling objective while planning jobs on a machine is the main contribution of this dissertation. The designed mathematical models include two or three objectives. The first mathematical model is to minimize the total tardiness and total energy cost under a set of constraints on a *non-preemptive single* machine. The second mathematical model minimizes the total completion time and total energy cost on a *preemptive single* machine. The third mathematical model aims to minimize the total energy cost, total completion time, and load balancing on *non-preemptive parallel* machines. The last mathematical model will minimize the total completion time and total energy cost on a *preemptive single* machine with *non-preemptive sequence-independent setup time*. These mathematical models are useful in manufacturing planning and scheduling on a single machine or multiple machines.

## 1.4.2 Algorithmic Contributions

When the mathematical models are designed, approaches to solve these models are utilized in order to find efficient solutions (i.e., optimal Pareto front). The problems that we consider in this dissertation are multiobjective mixed-integer mathematical models which are complex and non-deterministic polynomial-time hard (NP-hard) problems (i.e., no known algorithm can solve this problem in polynomial time). Therefore, in order to obtain near-optimal solutions in a reasonable amount of central processing unit (CPU) time, metaheuristics are developed for each model.

Genetic algorithms (GAs) are developed to determine the approximate Pareto front when the scheduling objective is total tardiness in a non-preemptive single machine environment and a total completion time in a preemptive single machine environment with setup times. An ant colony optimization (ACO) algorithm is proposed for a single machine preemptive scheduling problem with a total completion time objective. Finally, a multiobjective greedy random adaptive search procedure (GRASP) approach solves the model when the scheduling objective is total completion time.

To evaluate the efficiency of the proposed metaheuristics, a weighted sum method (WSM) is utilized to solve multiobjective optimization problems in order to obtain the approximate Pareto front in all models. Finally, in all cases, a framework to select the best solution among all non-dominated solutions based on decision-maker preferences is designed based on the multiobjective selection methods such as technique for order preference by similarity to ideal solution (TOPSIS) and multiobjective optimization on the basis of ratio analysis (MOORA).

## 1.5 Outline of Dissertation

The organization of this dissertation is as follows: Chapter 2 provides a literature review on scheduling, energy-aware scheduling, the complexity of scheduling, mixed-integer programming (MIP), multiobjective optimization, and the methods for solving multiobjective optimization problems.

In Chapter 3, a multiobjective mixed-integer mathematical programming model on a non-preemptive single machine to minimize total tardiness and total energy cost under time-of-use tariffs with varying energy prices is developed. This mathematical model is solved to global Pareto-optimal (complete and exact solutions) using the WSM. In addition, a multiobjective GA based on dominance rank (GA-1), weighted sum aggregation (GA-2), and dominance rank crowding distance comparison (GA-3) are proposed to approximate the optimal Pareto front and provide decision-makers with a set of non-dominated solutions from which they are seeking the most preferred solutions of the model. The parameters of the proposed GAs are fine-tuned depending on detailed experimental results. Each GA is compared against each other and the WSM using several performance measures in order to obtain more insight into the algorithm's dynamics. In a case study, the GA-3 algorithm is used to illustrate the TOPSIS method in order to assist the decision-maker in choosing the most-efficient schedule with an appropriate energy-cost level.

Chapter 4 presents a scheduling problem on a preemptive single machine to minimize the total completion time and total energy cost under time-of-use electricity tariffs, where energy prices vary hourly and are announced a day ahead. This problem is modeled using a multiobjective mixed-integer mathematical programming model, and is solved by using either the WSM or a multiobjective ant colony algorithm based on dominance rank (ACO-DR) or based on dominance rank and crowding distance comparison (ACO-DRC). The parameters of the proposed ant colony

algorithms are fine-tuned based on detailed experimental results with a varying number of jobs using a specific performance measure in the experimental setup. In addition, the performance of the proposed ant colony algorithms are compared to each other using several performance measures in order to clearly explain how Pareto fronts can be characterized. The ACO-DRC is illustrated in the case study, and the MOORA method is used to assist the decision-maker in choosing the most: the most appropriate, efficient and cost-effective schedule.

Chapter 5 presents another scheduling problem on a non-preemptive parallel machine to minimize the total completion time, load balance and total energy cost under time-of-use tariffs. This problem is modeled using multiobjective a mixed-integer mathematical programming model and is solved via several methods including the  $\varepsilon$ -constraint method, multiobjective GRASP, and multiobjective GA based on dominance rank procedure and crowding distance comparison (GA-DRC). The  $\varepsilon$ -constraint method is used to solve the model and obtain the global Pareto-optimal (complete and exact solutions). After fine-tuning the proposed metaheuristic algorithms, an analysis and detailed experimental results are provided to evaluate the performance of the algorithms using several performance measures, which maintain the quality of solutions.

Chapter 6 presents a preemptive scheduling problem on a single machine to minimize the total completion time and total energy cost of time-of-use electricity tariffs with varying energy prices. Each job has a non-preemptive sequence-independent setup time, which is performed only once before the job is first processed on the machine. This problem is modeled using a multiobjective mixed-integer mathematical programming model. Several methods, including the  $\varepsilon$ -constraint method, multiobjective GA based on dominance rank (GADR), and the multiobjective GA based on a dominance rank and crowding distance comparison (GARC) are proposed to solve the model. After fine-tuning the proposed GAs, they are compared to each other and to the  $\varepsilon$ -

constraint method using several performance measures in order to gain insight into the algorithms' performance.

In Chapter 7, the conclusions drawn from each problem and the proposed research are presented. The proposed research includes the development of maintenance planning, a reliability model of the machine under sporadic on/off cycles (times), and machines with sequence-dependent and sequence-independent setup times with energy concerns. In addition, simple heuristics or methods to obtain an approximate Pareto front without complex calculations (i.e., dispatch rules) for the scheduling problem are also proposed.

## **1.6 Summary**

This chapter demonstrates the motivation for the proposed dissertation work. After the problems and models are defined, the main contributions of this dissertation work are addressed. Finally, an outline of this dissertation is presented.

## **1.7 References**

- [1] A. Gungor and S.M. Gupta, "Issues in environmentally conscious manufacturing and product recovery: A survey," *Computers & Industrial Engineering*, vol. 36, no. 4, Sept. 1999, pp. 811-853.
- [2] K. Fang, N. Uhan, F. Zhao, and J.W. Sutherland, "A new shop scheduling approach in support of sustainable manufacturing," in *Glocalized Solutions for Sustainability in Manufacturing*, Springer, June 2011, pp. 305-310.
- [3] G. Mouzon, M.B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *International Journal of Production Research*, vol. 45, no. 18-19, Oct. 2007, pp. 4247-4271.
- [4] G. Mouzon and M.B. Yildirim, "A framework to minimise total energy consumption and total tardiness on a single machine," *International Journal of Sustainable Engineering*, vol. 1, no. 2, March 2008, pp. 105-116.
- [5] U.S. Energy Information Agency, "Annual Energy Outlook 2013," [Online: Annual Report], Nov. 2013; [cited: May 2013], available from World Wide Web: <http://www.eia.gov/forecasts/aeo/>.

- [6] Y. Liu, H. Dong, N. Lohse, S. Petrovic, and N. Gindy, "An investigation into minimising total energy consumption and total weighted tardiness in job shops," *Journal of Cleaner Production*, vol. 65, no. 15, Feb. 2013, pp. 87–96.
- [7] M.A. Salido, J. Escamilla, F. Barber, A. Giret, D. Tang, and M. Dai. "Energy-aware parameters in job-shop scheduling problems," in *GREEN-COPLAS 2013: IJCAI 2013 Workshop on Constraint Reasoning, Planning and Scheduling Problems for a Sustainable Future*, vol. 1, May 2013, pp. 44-53.
- [8] U.S. Energy Information Agency, "Annual Energy Outlook 2014," [Online: Annual Report], Aug. 2014; [cited: May 2014], available from World Wide Web: <http://www.eia.gov/forecasts/aeo/>.
- [9] A. Bruzzone, D. Anghinolfi, M. Paolucci, and F. Tonelli, "Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops," *CIRP Annals-Manufacturing Technology*, vol. 61, no. 1, 2012, pp. 459-462.
- [10] R. Drake, M.B. Yildirim, J.M. Twomey, L.E. Whitman, J.S. Ahmad, and P. Lodhia. "Data collection framework on energy consumption in manufacturing," in Wichita State University Libraries SOAR: Shocker Open Access Repository, Wichita State University, May 2006.
- [11] M.B. Yildirim and G. Mouzon, "Single machine sustainable production planning to minimize total energy consumption and total completion time using a multiple objective genetic algorithm," *IEEE Transactions on Engineering Management*, vol. 59, no. 4, 2012, pp. 585-597.
- [12] T. Gutowski, C. Murphy, D. Allen, D. Bauer, B. Bras, T. Piwonka, P. Sheng, J. Sutherland, D. Thurston, and E. Wolff, "Environmentally benign manufacturing: Observations from Japan, Europe and the United States," *Journal of Cleaner Production*, vol. 13, no. 1, Feb. 2005, pp. 1-17.
- [13] K. Fang, N.A. Uhan, F. Zhao, and J.W. Sutherland. "Scheduling on a single machine under time-of-use electricity tariffs," May 2014, [cited: Feb. 2015]; available from World Wide Web: [http://www.optimization-online.org/DB\\_FILE/2014/04/4313.pdf](http://www.optimization-online.org/DB_FILE/2014/04/4313.pdf).
- [14] A. Pechmann and I. Schöler, "Optimizing energy costs by intelligent production scheduling," in *Glocalized Solutions for Sustainability in Manufacturing*, Springer Berlin Heidelberg, Dec. 2011, pp. 293-298.
- [15] N. Weinert, S. Chiotellis, and G. Seliger, "Methodology for planning and operating energy-efficient production systems," *CIRP Annals-Manufacturing Technology*, vol. 60, no. 1, 2011, pp. 41-44.
- [16] P. Detti, C. Hurkens, A. Agnetis, and G. Ciaschetti, "Optimal packet-to-slot assignment in mobile telecommunications," *Operations Research Letters*, vol. 37, no. 4, 2009, pp. 261-264.

- [17] D. Trentesaux and V. Prabhu, "Sustainability in manufacturing operations scheduling: Stakes, approaches and trends," in *Advances in Production Management Systems. Innovative and Knowledge-Based Production Management in a Global-Local World*, Springer Berlin Heidelberg, July 2014, pp. 106-113.

## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 Introduction

Pinedo [1] states that at the beginning of the 20th century, Henry Gantt and other pioneers studied scheduling in manufacturing. In the early fifties, the first applications were published in the *Naval Research Logistics Quarterly*. During the sixties, integer programming formulations for scheduling problems received considerable attention. In the early seventies, Richard Karp published a landmark paper in complexity theory. After that, research focused on the complexity hierarchy of scheduling problems. In the eighties, stochastic scheduling problems received a large amount of attention. Also, personal computers had begun to generate usable schedules in manufacturing facilities that were designed by industrial engineers, operations researchers, and computer scientists. More details about scheduling in general may be found in the books of Brucker [2] and Leung et al. [3].

As a research area, scheduling has applications in all environments where scarce resources must be allocated over time, e.g., in production planning, telecommunications, operations systems, and industry. Typically, one only needs the best allocation, which is based on specified criterion. In machine scheduling problems, tasks are referred to as jobs. The machine can process only one job at a time. However, one needs to assign jobs to the machine and determine the order in which they are processed. Additionally, the work assigned should be processed on time, without overload, or finished as soon as possible. Jobs may be categorized into non-preemptive jobs and preemptive jobs. If the execution of a job is started and cannot be interrupted until it is entirely processed, then this is referred to as non-preemptive job scheduling. If the execution of a job can be interrupted at any time and restarted at a later time in favor of another job, then this is called preemptive job

scheduling. If the execution of a job does not depend on the sequence of job assignments on the machine, then this is referred to as an independent job; otherwise, it is called a dependent job.

In operations research, scheduling is well organized in theory and practice. Scheduling is categorized by a nearly unlimited number of applications and problem types. As a result, it has become a field of study in its own right—a hybrid created from the disciplines of industrial engineers, operational researchers, computer scientists, and business managers.

Today, scheduling is considered as a theory of decision-making, which plays a vital role in manufacturing and service industries. Furthermore, it has become a necessity for survival in the marketplace. Supply designs, for example, must meet a production environment that has been committed to their systems, and failure to do so may result in a significant loss of goodwill. Consequently, jobs must be scheduled in such a way as to use the available resources (machines) efficiently.

Today, with advances in computers and software, more complex problems can be solved in a timely fashion to obtain either exact or near-global optimal solutions. Nessah and Kacem [4] utilize an exact algorithm called a branch-and-bound method to minimize a single-objective scheduling problem on a single machine scheduling problem. Keha et al. [5] study the computational performance of four different MIP formulations for a single-objective on various single machine scheduling problems. Ferrolho and Crisostomo [6] utilize a metaheuristic called a genetic algorithm to provide good solutions for minimizing a single-objective scheduling problem on a single machine scheduling problem. Al-Turki et al. [7] develop a tabu search-based solution procedure to minimize a single objective on a single machine scheduling problem.

In the eighties, researchers began to explore multiobjective scheduling using multiple-criteria decision-making (MCDM) methods. These methods provide solutions that optimize more

than one, usually conflicting, objectives. MCDM can be classified into selection and optimization problems. If a choice must be made between a set of finite possible solutions, then this is called a selection problem. If solutions belonging to the optimal non-dominated solution (Pareto front) must be identified among a set of infinite possible solutions, then this is called an optimization problem.

Today, the growth in price of and end-use demand for electrical energy have resulted in greater efforts to minimize energy consumption. Moreover, carbon dioxide emissions in a manufacturing environment are regulated by the government and cannot exceed a certain defined level. As a result, energy scheduling is considered in a manufacturing environment where the objective is energy savings.

The remainder of this chapter is organized as follows: Section 2.2 reviews the single-objective scheduling literature and surveys total completion time and total tardiness objectives. Single machine scheduling with preemptive and setup times, and parallel machine scheduling are reviewed in sections 2.3 and 2.4, respectively. Section 2.5 provides a review of energy-aware scheduling. Section 2.6 defines mixed-integer programming. Multiobjective optimization is defined in section 2.7. Methods for solving multiobjective combinatorial optimization problems and the complexity of algorithms are reviewed in sections 2.8 and 2.9, respectively.

## **2.2 Single-Objective and Single Machine Scheduling**

Scheduling jobs on a single machine are widely applicable in real life. The single-objective single machine scheduling problem has been studied in depth relative to various objectives. For an overview of these objectives, one can study the surveys of Leung [8] and Queyranne and Schulz [9]. Based on Graham et al. [10], a machine scheduling problem is defined by a triplet  $\alpha|\beta|\gamma$  as follows:

- $\alpha$  describes the number and type of machine available;  $\alpha = \mathbf{1}$  species a single machine;  $\alpha = \mathbf{P}$  represents several identical machines, i.e., the processing time of a job does not depend on the machine the job is processed on; and  $\alpha = \mathbf{R}$  represents several unrelated machines, i.e., the processing time of a job is different depending on which machine the job is processed.
- $\beta$  provides additional constraints of the problem. For example,  $p_j = p_i$  describes the processing times of all jobs as being equal.
- $\gamma$  defines the optimality criterion of the problem.

Generally, in machine scheduling, each job  $j$  from a given set  $N = \{1, \dots, n\}$  is characterized by certain parameters, some of which might be stochastic. Common parameters are the following:

- Processing time  $p_j$ , or the length of the job.
- Release date  $r_j$ , or the earliest point in time when the job becomes available to be executed.
- Deadline or due date  $d_j$ , or ideally the time when the job should be completed.
- Weight  $w_j$ , or the value of the job.

Mellor [11] studied 27 different objectives that can be used in scheduling, the most common of are the following:

- Total completion time  $\sum_j^n C_j$ , or sum of the completion time of all jobs.
- Maximum completion time (makespan)  $C_{max}$ , or the completion time of the last job.
- Maximum lateness  $L_{max}$ , or the maximum difference between the completion time of a job and its due date.

- Total tardiness  $\sum_j^n T_j$ , or the sum of the tardiness of all jobs where tardiness is defined as the positive part of its lateness (due date minus release date).
- Number of tardy jobs, or number of jobs completed after their due date.

Traditional constraints can be described as follows:

- Preemption may not be allowed.
- Two jobs cannot be executed at the same time.
- A job cannot be executed before its release date.
- A setup time might be required between certain types of jobs.

For example, a single machine environment described as  $1|r_j|\sum_j^n T_j$  denotes a single machine system with job  $j$  entering a system of different release dates, and the criterion is minimizing the weighted total tardiness.

Single machine scheduling is important for various reasons. In practice, more complicated machine environments are frequently decomposed into subproblems that deal with single machines. The obtained results for single scheduling not only generate visions into the single machine environment, but also they generate a basis for heuristics that are valid to more complicated machine environments.

The following sections provide a more detailed review of total completion time and total tardiness objectives.

### **2.2.1 Minimization of Total Completion Time**

The minimization of total completion time has been studied in depth for various scheduling problems. The completion time of job  $j$  ( $C_j$ ) is the time at which processing of the latest operation of the job is completed. According to Little's law, minimizing the total completion time will minimize the work in process level. Also, the result of the total completion time is a measure of

the total cycle time and also the total inventory holding cost for a given schedule [12]. When the release dates are equal, the shortest processing time (SPT) rule minimizes the total completion time [13]. Very simply stated, jobs are scheduled in the order of increasing processing time in a single batch. In the preemptive case, if jobs have equal weights and different release dates (i.e., the jobs are unweighted,  $w_j = 1$ ), then the shortest remaining processing time (SRPT) rule works [14].

### 2.2.2 Minimization of Total Tardiness

The total tardiness objective is used to evaluate the job scheduling performance in the manufacturing environment to determine the total lateness of tasks. This objective can be defined as

$$\min \sum_{j=1}^n \max(C_j - d_j, 0),$$

where  $C_j$  is the completion time of job  $j$ , and  $d_j$  is the due date of job  $j$ . The objective is to minimize the sum of all jobs' lateness over the planning horizon. If the job is scheduled on time or early, then the associated tardiness is zero. This problem is NP-hard [15]. Lawler [16] develops a pseudo-polynomial dynamic programming algorithm that relies on the decomposition of the problem. Potts and Van [17] utilize the branch-and-bound method to solve the total tardiness problem on a single machine. In further research, Potts and Van [18] propose an exact algorithm solution and dynamic programming to solve the total tardiness problem on a single machine. Mouzon [19] develops heuristics, local search heuristics, and decomposition heuristics to obtain good solutions in a short amount of time. In further research, Yildirim and Mouzon [20] develop dominance rules and heuristic rules to accelerate heuristic solutions.

## **2.3 Single Machine Scheduling with Preemptive and Setup Times**

Preemptive single machine scheduling problems are usually very complex to solve in a reasonable amount of time, even with advanced computers and computational software [21]-[22]. Lawler [23] proposes dynamic programming to minimize the number of late jobs on a preemptive single machine. Bülbül et al. [24] utilize the lower-bound method to solve the preemption in single machine scheduling. Nessah and Kacem [4] utilize exact enumeration algorithms when an NP-hard scheduling problem is reduced to a relaxed problem and propose a heuristic based on the dominance properties, which are provided to be very efficient to prune the search tree. Batsyn et al. [25] present an efficient high-quality heuristic based on the weighted shortest remaining process time rule to solve a preemptive single machine scheduling problem.

In the next section, a summary of the literature on single machine scheduling with setup times is provided.

### **2.3.1 Single Machine Scheduling with Setup Times**

Setup times are defined as the work it takes to prepare the machine to perform a job [26]. Setup times play a vital role in scheduling. The decision to obtain tools, return and adjust tools, clean up, inspect materials, and produce multiple products with common machines results in the need for changeover and setup activities. Reducing setup times provides several benefits: expenses are reduced; lead times are reduced; output, competition, and profitability are increased; competition and output are increased; faster changeover and deliveries are increased; and inventory levels and total cost curves are reduced.

The majority of scheduling research assumes that setup times are negligible or part of the processing times [27]. Considering setup times independently from processing times allows operations to be achieved simultaneously and machine utilization to be improved.

The importance of integrating setup times in scheduling research has been investigated since the mid-1960s [28]. Panwalkar et al. [29] find that around 75% of industrial managers need to process setup times independently from processing times. Krajewski et al. [30] observe that implementing the Kanban scheduling system, planning for manufacturing requirements, or shaping the manufacturing environment can reduce setup costs and lot size simultaneously. Setup reduction is the most effective means to lower inventory levels and improve satisfaction. Flynn [31] points out that incorporating scheduling with setup times leads to an increased output capacity in cellular manufacturing. In an automated manufacturing line with robots, Kogan and Levner [32] realize that processing setup times independently leads to a significant reduction in makespan, or total length of the schedule.

In production scheduling, Liu and Chang [33] show that setup times consume more than 20% of the available resource capacity. For example, in a printed circuit board assembly, Trovinger and Bohn [34] point out that about 50% of the effective capacity can be lost due to setup activities. They emphasize that implementing a setup time reduction approach can reduce setup times by more than 80% and increase profits by \$1.8 million per year.

One of the common types of setup times is sequence-dependent, that is, the importance of setup times strongly depends on both current and immediately preceding jobs on the same machine. Chang et al. [35] point out that the time, raw materials, and equipment necessary to prepare for the next job in a factory depend on the preceding job; therefore, setup times are sequence-dependent. Lin and Liao [36] point out that when a machine is changed over from jobs in one class to jobs in another class, a sequence-dependent setup time is required for the changeover. Gupta and Smith [37] propose two heuristics—a GRASP and a problem space-based local search heuristic to minimize the total tardiness on a single machine. Chang et al. [38] design

a mathematical programming model with logical constraints to minimize the total weighted tardiness on a single machine with release dates and sequence-dependence. They propose a heuristic algorithm to solve the problem in a reasonable amount of CPU time. Lee and Asllani [39] present MIP and a GA for a single machine with sequence-dependence, and with minimization of the number of the tardy jobs as the primary objective and minimization of makespan as the secondary objective. Rabadi et al. [40] propose a branch-and-bound algorithm for the earliness-tardiness machine scheduling problem, with a common due date, and sequence-dependent setup times. They show that a problem with up to 25 jobs can be solved by the algorithm in a reasonable period of time.

## **2.4 Single-Objective Parallel Machine Scheduling**

Over the last decade, parallel machine scheduling problems have been intensively discussed by researchers. Parallel machine scheduling involves two types of decisions (sequencing and assignment), whereas single machine scheduling has only a sequencing decision. Complexity of the parallel machine scheduling problem grows exponentially, making problems intractable [41]. Therefore, many heuristic algorithms have been developed to solve these problems, yielding good solutions.

### **2.4.1 Minimization of Completion Time Objectives**

In the literature, completion time objectives, such as makespan, have been extensively considered more than any other criteria. Makespan is the longest period of time required to complete a job. In the non-preemptive case, when all release dates are equal, this problem is NP-hard. In the preemptive case, the optimal is given by the maximum between the sum of the processing times divided by the number of machines and the maximum processing time. Herrmann and Lee [42] study makespan on parallel machine scheduling with setup times. Ghomi and

Ghazvini [43] propose an algorithm based on pairwise interchanges to approximately solve the makespan objective on a parallel machine. Tang and Luo [44] develop a local search, based on cyclic change, which can find the optimal solution 83% of the time to solve the maximum completion time objective on a parallel machine.

Total completion time is another widely considered criterion for multi-machine scheduling problems. For example, Yildirim et al. [45] discuss the total completion time problem with load balancing and sequence-dependent setup times using a GA to solve the model in different instances. Duman et al. [46] propose a four-phase solution procedure to solve the total completion time problem with load balancing and sequence-dependent setup times considering the problem of scheduling the casting lines of an aluminum casting and processing plant to determine a near-optimal schedule.

#### **2.4.2 Minimization of Deviation Using Load Balancing**

Efficient use of parallel machines usually requires the redistribution of jobs during the execution of applications. Load balancing, or workflow, is used to remove the bottleneck in a manufacturing line and reduce the idle time. The usual balancing procedure attempts to equalize the assigned workload on all machines. Yildirim et al. [45] define load balancing as the difference between the total processing times on one machine and the average processing time per machine. Koltai [47] proposes an aggregate capacity model for a workload balancing procedure to reduce complexity in a flexible manufacturing system. Farkas et al. [48] develop a heuristic balancing procedure based on aggregate capacity analysis for smoothing a schedule over the short term. Kumar and Shanker [49] compare various balancing objectives to devise an imbalancing measurement in a flexible manufacturing system. Hayrinen et al. [50] discuss several scheduling algorithms in generalized flexible flow lines that have similar machine allocations and sequencing

phases. Saad et al. [51] develop a multiobjective optimization technique based on simulation and a taboo-search algorithm for loading and scheduling cellular manufacturing systems.

## **2.5 Energy-Aware Operations Management**

Energy-awareness has been an area of interest, especially in computer systems. From the perspective of production planning and scheduling in manufacturing applications, developing decision models including the energy consumption aspect is critical to achieving an energy-efficient system. In the specialized literature, cost, time, and quality objectives have been extensively studied. However, research regarding reducing energy consumption in a manufacturing system via production planning and scheduling approaches has been rather limited. In this section, studies that incorporate the objective of “total energy consumption minimization” into production planning and scheduling decisions are discussed.

### **2.5.1 Energy-Aware Production Planning**

Energy-efficiency is an indispensable productivity objective in manufacturing systems due to rising energy costs and increasing pressure to reduce energy consumption costs and environmental impacts (i.e., CO<sub>2</sub> emissions). Production planning and management approaches are potential tools that increase energy-efficiency and allow manufacturing to achieve proper output with less energy consumption [52]. For this reason, many articles have been written on approaches to increasing energy-efficiency in various levels of manufacturing. Schmidt et al. [53] present a methodology for the reliable prediction of energy consumption of arbitrary manufacturing processes, using consumption models to aid in the calculation of the product carbon footprint and validate measures to improve energy-efficiency in production. Mustafaraj et al. [54] develop a methodology that accurately and flexibly determines the auxiliary and value-added electricity in manufacturing operations. Shrouf and Miragliotta [55] present a framework to

support the integration of energy data into a company's information technology in order to improve efficiency. Mousavi et al. [56] develop an integrated conceptual framework to model the energy consumption of a production system, where the outcome potentially leads to a more streamlined process plan and a more energy-efficient production system. May et al. [57] provide a seven-step method to support manufacturing companies in order to develop energy-based performance indicators. They enable energy-related information to assess the ability of manufacturing companies to reach their energy-efficiency goals. Fysikopoulos et al. [58] propose a generic energy-efficiency approach at four levels of manufacturing—process, machine, production line, and factory. Based on the aforementioned publications, research aims to incorporate energy consumption costs in different measures and methodologies. In brief, they found significant improvement in the potential of energy-efficiency in energy consumption cost models and methods.

Devoldere et al. [59] discuss the potential for energy improvement measures, they emphasize the importance of operations during non-productive times, and they explain that total energy consumption is not only a function of production rate (i.e., dependent energy) but also the fixed independent energy consumed during a machine's idle or standby mode. They implement a time study to categorize the percentages of energy consumed during non-productive and productive periods. Dietmair and Verl [60] propose a generic method to model the energy consumption behavior of machines and plants based on a statistical discrete event formulation, and they describe how energy consumption, and thus energy-efficiency of machines and production systems, relates to the way they are operated. This framework informs decision-makers about energy-efficiency throughout the life cycle of a machine in the early design stages and also in real-time, tactical, and strategic decision-making processes.

Recently, time-varying electricity prices have become an important factor in defining energy consumption costs. Because electricity prices vary hourly, they represent a tremendous opportunity to minimize energy costs by shifting electricity usage from high-peak hours to low-peak or mid-peak hours. For example, the wholesale electricity market is open to anyone who connects to the grid, and anyone who is willing to receive electricity prices at different altitudes, depending on the time of day (i.e., retail electricity prices that vary hourly to reflect dynamics in the market). Such time-of-use tariffs provide real-time electricity pricing over time that represents an interesting challenge to minimize the total energy cost in a planning and scheduling problem. For example, Zhang et al. [61] develop a mathematical model to minimize the electricity cost and carbon footprint under time-of-use tariffs without compromising production throughput. In further research, Zhang et al. [62] use a distributed optimization approach to minimize the total electricity cost as a function of the manufacturing schedule, whereby energy-efficient scheduling is subject to real-time electricity pricing. Cheng et al. [63] investigate a new bi-objective single machine batch scheduling problem with a time-of-use policy to improve the productivity and minimize the total electricity cost, whereas Fang et al. [64] consider the problem of scheduling jobs on a single machine to minimize the total electricity cost of processing these jobs under time-of-use electricity tariffs. Ghobeity and Mitsos [65] optimize the operation of seawater reverse osmosis in order to minimize the total energy cost, and their results show significant electricity and production cost-saving potential.

Newman et al. [66] utilize energy consumption and other environmental metrics in their framework to optimize process planning for CNC machining. Seow and Rahimifard [67] study the concept of lean energy, which enables the manufacturing industry to use the most energy-efficient processes and activities within their production facilities. They consider the product viewpoint and

energy-consumption data at both plant and process levels in order to provide a breakdown of energy used during production, and they propose a novel approach for modelling energy flows within a manufacturing system. Moon et al. [68] propose a hybrid generic algorithm to solve the unrelated parallel machine problem with the same due date by inserting idling time to reduce total electricity costs. Furthermore, Escamilla et al. [69] develop a GA to solve an extended version of the job-shop scheduling problem, wherein machines consume different amounts of energy to process jobs at different rates.

Herrmann et al. [70] study energy consumption in a flow-oriented manufacturing system and define energy-efficiency as the ratio of production output and input energy to compare different production planning scenarios. Solding and Tollander [71] discuss the adoption of the energy consumption concept into a simulation of production planning for a production system as a possible performance measure, considering overhead as well as direct and indirect types of power usage, and they claim that companies are able to develop more energy-efficient production plans and reduce their overall costs. Herrmann and Thiede [72] present a process chain simulation to decrease the energy requirement and waste at all layers of the production process, machine, production system, and technical building services.

Weinert et al. [52] propose an “EnergyBlock” planning methodology that breaks the production process into operations with a known duration and amount of energy consumption in order to calculate total energy consumption more accurately in production planning, and scheduling. EnergyBlocks describe the machine behavior during loading and unloading states, and are identified by machine type, process parameter, and duration of each operation state. As a result, EnergyBlocks represent the amount of energy consumed in each operating state per part and machine. EnergyBlocks sequences, which indicate the production process of a product on a given

machine, calculate the total amount of energy consumption. Wang et al. [73] point out that discrete event simulation performs a significant role in evaluating performance of the production plan using a multi-granularity state chart model to control energy consumption of the entire production process on a CNC machine with the energy consumption states of setup (power off), idle, busy, and down.

Lin and Sun [74] propose a Markov decision model of energy control decisions and system state evaluations to achieve energy-efficiency in typical manufacturing systems with multiple machines and buffers to dynamically control energy consumption, considering both energy states and production constraints. They utilize an approximate algorithm in order to obtain near-optimal real-time solutions. He et al. [75] discuss an event graph modeling approach to analyze the task-oriented energy consumption in the machining process with seven states of start (when a new job arrives), task assignment, arrival machine, start machining, end machining, idle waiting, and end (when a job is finished). They show that different flow schemas influence energy consumption and productivity, and a tradeoff needs to be achieved between these two measurements.

### **2.5.2 Energy-Aware Production Planning with Classical Single Machine Scheduling**

This section presents some scheduling methodologies at various levels in order to reduce total energy consumption. Mouzon et al. [76] investigate the single machine job-shop scheduling problem to minimize total energy consumption. They observe that non-bottleneck machines could be turned off until needed, rather than leaving them idle, which could lead to 80% savings in total energy consumption. They also propose several dispatching rules to control the machines and also save energy when the availability of jobs is not known in advance. Furthermore, they observe that in this environment, forecasting the arrival of jobs is an important aspect of deciding if the machine should be turned off or stay idle. As a result, an artificial neural network-based method is proposed

to predict the next arrival based on preceding arrivals, which later is used as input for dispatching rules to minimize power consumption.

Changing the order of jobs and when machines are turned off and on can provide significant savings in energy consumption in a manufacturing facility. Mouzon [19] studies a single machine scheduling problem, develops a framework to model and solve a multiobjective model to minimize total tardiness and total energy consumption, and utilizes a greedy random adaptive search metaheuristic in order to obtain an approximate optimal Pareto front. Similarly, Yildirim and Mouzon [20] develop a mathematical model to minimize energy consumption and total completion time on a single machine, and develop a multiobjective GA in order to obtain an approximate Pareto front. In order to expedite the computational time, they also incorporate a heuristic to obtain Pareto optimal solutions for a sequence of jobs instead of solving a multiobjective linear programming subproblem. Shrouf et al. [77] consider variable energy prices during one day to develop a mathematical model to minimize energy consumption costs for a single machine during production scheduling. Dai et al. [78] present a multiobjective total energy consumption and makespan job-shop problem to describe an energy-aware integrated process plan and schedule. Duerden et al. [79] present a methodology for modifying a manufacturing production schedule with the goal of minimizing variance in production-line energy consumption. Ding et al. [80] consider a permutation flow shop scheduling problem with the objective of minimizing total carbon emissions and total length of the schedule (makespan).

Frequent turn off and turn on can have a significant impact on machine failures. Liu et al. [81] investigate the job-shop scheduling problem to minimize the total electricity consumption and total weighted tardiness using a non-dominated sorting GA to obtain a Pareto front. They argue that the proposed framework can be applied across existing legacy systems with minor investment.

May et al. [82] focus on developing different policies in order to control the behavior of machine components; therefore, energetic states at which machines should be operated are dependent upon period duration and power requirements of the various states.

Salido et al. [83] focus on the classical job-shop scheduling problem in a dynamic setting, taking into account energy consumption, robustness, and makespan, whereby each operation must be executed on a single machine that works at different speeds. In particular, they observe that there is a clear relationship between robustness and energy-efficiency, and a tradeoff between robustness, energy-efficiency, and makespan. He and Liu [84] propose procedures for integrating energy consumption and environmental impacts into the operation of machining processes, and they develop a mixed-integer model that minimizes total energy consumption and makespan in a parallel machine setting.

In an integrated production planning and scheduling study, Wang et al. [85] point out that a paint shop in an automotive assembly plant consumes a considerable amount of energy. With the help of quality evaluation models, an optimal batch size, and a vehicle-sequence scheduling procedure, significant energy savings is achieved with no additional investment and no changes to existing planning processes.

### **2.5.3 Peak-Load Management**

This section discusses the literature on peak-load management, where peak power consumption is analyzed with respect to production planning and scheduling decisions. From an energy-management approach perspective, developing strategies for balancing energy at various levels within the manufacturing system in order to control power and energy demands is beneficial. Ashok [86] investigates a production planning problem to minimize operating costs including energy consumption, load shifting, and maximum demand charges with respect to production,

process storage, and equipment constraints, resulting in a model that could also be used in a time-varying-tariff energy cost to decrease electricity bills.

Fang et al. [87] develop a general multiobjective mixed-integer linear programming model for optimizing a job-shop schedule with makespan and peak power load objectives, while varying the operation speed of machines as an independent variable to affect peak load and energy consumption. Fang et al. [88] study a job-shop scheduling problem consisting of two machines to optimize the makespan, peak power load, and carbon footprint using a mixed-integer multiobjective model. Their approach considers unlimited storage between two machines with controllable process speeds. Xu et al. [89] focus on a power-peak-related energy-aware scheduling problem and present an MIP model and simulation to ensure a global optimum.

Bruzzzone et al. [90] present an MIP model to optimally plan energy savings for a given job schedule generated by an advanced planning and scheduling system that minimizes total tardiness and makespan. They incorporate the peak power minimization objective into the above schedule while accepting possible inferior solutions for tardiness and makespan. The proposed approach changes the jobs' timetable while keeping the job assignments and sequences fixed. Nilsson and Söderström [91] investigate the influence of time-dependent electricity costs on industrial production planning, pointing out that it is profitable to shift the electricity demand from high-rate periods to low-rate periods. In another study, Nilsson [92] shows that electricity generation also benefits when peaks in the electricity demand shift to time periods with a lower electricity price.

Pechmann and Schöler [93] notice that the energy consumption cost is calculated with respect to customer service charge, energy charge, power charge (defined by peak demand), and power factor charges. They show that one remedy to minimize peak energy and associated energy costs might be rearranging the orders of product processing on the machine(s). They discuss the

application of intelligent production scheduling software like E-PPS, which recommends postponing the processing of some products in order to limit the energy peak and avoid associated costs. With the review of energy-aware operations management studies, the relevant research papers based on a few characterizations are categorized in Table 2.1.

Table 2.1: Review of Energy-Aware Operations Management Studies

Characteristic	Objectives	
	Total Energy Consumption	Peak Energy (also considering total energy consumption)
Order of job known and job start time optimized	Wang et al. [85] Seow and Rahimifard [67]	Ashok [86] Bruzzone et al. [90] Fang et al. [87] Fang et al. [88] Herrmann and Thiede [72] Weinert et al. [52]
Both order of job and job start time determined/optimized	Mouzon et al. [76] Mouzon and Yildirim [94] Yildirim and Mouzon [20] He et al. [75] Herrmann et al. [95] Heilala et al. [96] Herrmann et al. [97] Thiede et al. [70] Weinert et al. [52] Wang et al. [73] Liu et al. [98] Lin and Sun [74] Salido et al. [83] Newman et al. [66]	Pechmann and Schöler [93]
Statistical discrete event formulation used to model energy consumption	Devoldere et al. [59] Dietmair and Verl [60]	Weinert et al. [52] Fang et al. [87] Fang et al. [88] Herrmann and Thiede [72] Pechmann and Schöler [93]
Dynamic procedure used to model energy consumption to evaluate production rate over time	Herrmann et al. [70] He and Liu [84] Solding and Petku [99] Solding and Thollander [71]	

## 2.6 Mixed-Integer Programming

In this dissertation, mixed-integer programming is used to model energy-aware operations. Integer programming refers to methods used to solve optimization problems containing discrete or integer variables. Such variables are used to model indivisibilities, on-off decisions to select, play, buy, eliminate, and so on. Integer programming problems arise in biology, medicine, sports, national security, transportations, and scheduling of a maintenance team or electricity production.

MIP is a division of the extensive field of mathematical programming. Many combinatorial optimization problems including machine-scheduling problems can be modeled and solved by using MIP. MIP involves a problem of minimizing (or maximizing, depending on the type of the objective) a linear objective function involving many variables that contain unknown quantities or decisions that are to be optimized, subject to linear constraints and integrity restrictions on the part of variables [100].

Assume the following mathematical program:

$$\max\{cx: Ax \leq b, x \geq 0\}$$

where  $A$  is an  $m$ -by- $n$  matrix,  $c$  is an  $n$ -dimensional row vector,  $b$  is an  $m$ -dimensional column vector, and  $x$  is an  $n$ -dimensional column vector of variables or unknowns. If all variables are integers, then this becomes a linear integer program (LIP) written as

$$\begin{aligned} \max & cx \\ & Ax \leq b \\ & x \geq 0 \text{ and integer,} \end{aligned}$$

If some but not all variables are integers, then we have a linear mixed-integer program written as

$$\begin{aligned} \max & cx + hy \\ & Ax + Gy \leq b \\ & x \geq 0, y \geq 0 \text{ and integer,} \end{aligned}$$

where  $A$  again is an  $m$ -by- $n$  matrix,  $G$  is an  $m$ -by- $p$  matrix,  $h$  is a  $p$  row-vector, and  $y$  is a  $p$  column-vector of integer variables.

In the context of linear and MIP problems, decisions that must be made are subject to certain system requirements and restrictions. Constraints enforce these requirements and restrictions in the model. Each constraint requires that a linear function of the decision variables is either equal to, less than or equal to, or more than or equal to a scalar value. Moreover, all linear programming problems can be transformed into an equivalent minimization (or maximization) problem with non-negativity variables and equality constraints.

Several researchers interchange the term “integer” with “binary” or “0 – 1” when variables are restricted to take on the value of either 0 or 1. Note that a solution that satisfies all constraints is called a feasible solution. If the feasible solution achieves the best objective function value (according to whether one is minimizing or maximizing), then it is called an optimal solution. In some cases, no solution exists in MIP, and the MIP problem itself is called infeasible. In another respect, no optimal solution can be obtained in some feasible MIP problems due to the possibility of obtaining infinitely good objective function values with feasible solutions. Such problems are considered to be unbounded.

This dissertation focuses on modeling and solving scheduling problems by introducing multiobjective mixed-integer scheduling optimization. Modeling MIP problems is more of an art than a science, since a three-step looped process is involved. The first step defines a set of decision variables that represent choices that must be optimized. The second step defines the statement of constraints in the model. The third step defines the statement of an objective function. The last two steps can be completed in either order. The work of Wolsey [100] provides a fundamental integer

programming method and theory, while the updated work of Nemhauser and Wolsey [101] discusses integer programming and combinatorial theory in detail.

## **2.7 Multiobjective Optimization**

The optimal solution and all related information (shadow prices, etc.) for one performance criterion (single objective function) in convex mathematical programming problems is guaranteed. In contrast, there is no optimal solution that simultaneously optimizes all performance objectives in the multiobjective mathematical programming (MOMP). Ordinarily, decision-makers are seeking the most preferred solution [102]. The concept of Pareto optimality or efficiency contains the best solutions in MOMP, in contrast to the optimal solution of one performance objective. The set of Pareto optimal (or efficient, non-dominated) solutions is called the Pareto set. From the perspective of mathematical theory, weak solutions are not desired in MOMP, since they may be dominated by other efficient solutions. They will not be included in the Pareto set. It is important to remember that decision-makers are seeking the most preferred solutions among the non-dominated (Pareto optimal) solutions of MOMP.

As late as the eighties, scheduling took into consideration only one performance objective. However, real-life problems typically involve more than one objective. If only one objective is taken into account, then no matter what objective is considered, the outcome is probably unbalanced. However, it is important to measure the quality of a solution on all important objectives, which has led to the development of the area of multiobjective scheduling.

This section focuses on the basic concept of multiobjective scheduling. Assume that two performance objectives must be considered. It can be assumed that without loss of generality, both objectives are to be minimized. It is fortunate that there will be no such scheduling that achieves the minimum value for both performance objectives simultaneously, which means that the quality

of at least one of the two objectives must be achieved. If one performance objective is preferred more than the other, then a search to find the optimum value with respect to the preferred objective is undertaken, and then selection from among the set of optimum schedules (preferred objective) of the one that performs better than the other. Such an approach is called lexicographical optimization, in which performance objectives are arranged in order of importance.

If the lexicographical optimization generates an unbalanced schedule because no objective dominates, then simultaneous optimization can be a better alternative. Three different approaches in simultaneous optimization have been distinguished by Evans [103] and Fry et al. [104]: a priori optimization, interaction optimization, and a posteriori optimization. In a priori optimization, both objectives are grouped into a single composite objective function under consideration (constraints), after which an optimum solution is obtained for this one problem as a whole. For example, both objectives can be adjusted to specific goals or weights in order to make a single composite objective function. A priori methods allow the decision-maker to define his/her preferences before the solution process. The criticism about a priori optimization is that it is very difficult to know what this composite single objective function looks like, and it is very difficult for the decision-maker to know beforehand and be able to accurately evaluate the performance of setting those goals or weights to express his/her preferences into a single composite objective function. In interactive optimization, the decision-maker interacts and swaps with phases of calculation in order to drive the search with his/her preferences towards the most preferred solution. Interactive optimization is very cooperative with the decision-maker. One criticism about interactive optimization is that the decision-maker cannot see the entire picture of efficient solutions or even an approximation of it. In a posteriori optimization, efficient solutions of MOMP are generated, and then the decision-maker selects the most preferred solution among them. It is

obvious that a posteriori optimization is the most difficult of the three approaches. Therefore, it is assumed from this point forward that a posteriori optimization is what is considered when referring to simultaneous optimization without further specification.

### **2.7.1 Multiobjective Combinatorial Optimization**

Ordinarily, most problems involve more than one objective to be achieved. Multiobjective combinatorial optimization problems (MOCOPs) optimize more than one performance objective. However, the concern here is to obtain a feasible solution that minimizes (or maximizes, if that is the goal) a specific objective function vector, depending on some notion of optimality. For instance, the traveling salesman problem can involve more than one performance objective, such as minimizing the total distance between cities, traveling time, total cost, and so on [105].

Scheduling is concerned with all situations in which scarce resources must be allocated to tasks over time in order to meet specific objectives, and where the values of some but not all variables are integers (i.e., binaries) due to resource indivisibility. However, MOCOP models can be designated as integer programming models, where some or all decisions can take only a finite number of alternative possibilities [106].

The aim of the MOCOP is to obtain one or more optimal solutions in an apparent discrete problem space. Steps in the MOCOP are as follows:

- Obtain a set of instances.
- For each set, obtain a finite set  $\mathcal{S}$  of feasible solutions.
- Assign a performance objective to each set and each feasible solution.
- Obtain an objective function value for each performance objectives.

The ambition of any optimization process is to find a feasible solution for each set, which is called the global optimum. These minimize (or maximize, if that is the goal) the performance

objective. When the performance objective is to be minimized, a solution  $s \in S$  is a global optimum, if and only if there is no other solution  $s' \in S$  such that  $f(s') < f(s)$ .

In order to avoid conflict by having different performance objectives, it is necessary to define the optimal solution sought. This involves defining two fundamental concepts: efficiency and non-dominance. A general form of the MOCOP can be written as

$$\underset{s \in S}{\text{Min}} (f_1(s), \dots, f_p(s)),$$

where  $S \subset R^n$  is a feasible set, and  $f: R^n \rightarrow R^p$  is an objective function vector with  $P$  objectives. The image of the feasible set in the objective spaces can be expressed by  $Y = f(S) \subset R^p$ . A solution  $s \in S$  is a global optimum (efficient), if and only if there is no other solution  $s' \in S$  such that  $f_i(s') \leq f_i(s)$  for every  $i = 1, \dots, p$  and  $f_j(s') < f_j(s)$  for some  $j$ . A solution  $s$  can be called a non-dominated point if it is the efficient solution. The set of all efficient solutions  $s \in S$  is called the efficient set, and the set of all non-dominated points  $f(s) \in Y$  is called the Pareto front. The efficient solution indicates the solutions  $s$  in the decision space, whereas non-dominance is used for objective vectors  $f(s) \in R^p$  in the objective space. Figure 2.1 shows that if  $s$  is efficient, then  $f(s) = (f_1(s), \dots, f_p(s))$  is called non-dominated.

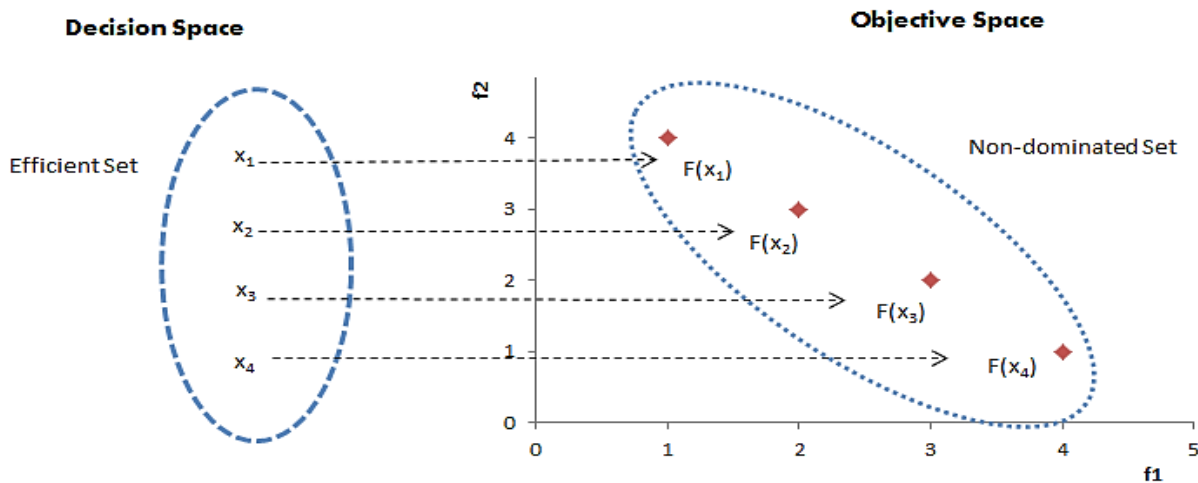


Figure 2.1: Efficient set (left) and dominated set (right)

In addition, concepts of the “ideal or Nadir solution” and the “anti-ideal or Utopia solution” are relevant in multiobjective optimization. The ideal solution is an artificial solution, which represents the best of both objectives, while the anti-ideal solution represents the worst of both objectives. Both solutions are used to define the upper (ideal/Nadir) and lower (anti-ideal/Utopia) bound for the non-dominated set. These bounds provide an evaluation of the range of values that non-dominated points can estimate. Figure 2.2 illustrates a non-dominated set, and the respective ideal and anti-ideal points.

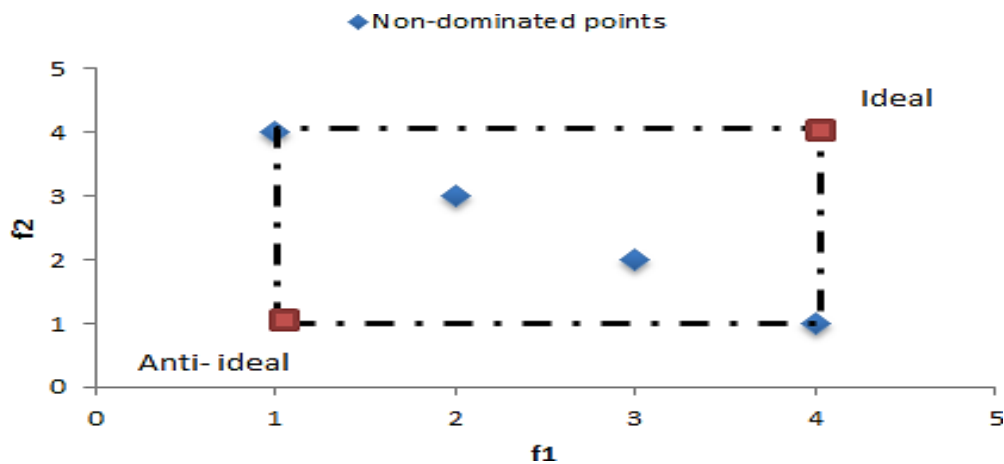


Figure 2.2: Non-dominated set with ideal and anti-ideal points

Ehrgott and Gandiblexu [107] note that the MOCOP differs from the traditional single-objective in several ways. In many cases, the MOCOP is very hard to solve with exact algorithms [108]. The goal of solving MCOPs is to obtain solutions (Pareto optimal) that cannot be improved in one objective without degenerating their performance in at least one of the others. The knowledge of all these optimal solutions allows the decision-maker to see the whole picture (tradeoff) among the different objectives while making decisions. In terms of dominance, the relation between objective function value vectors of any two feasible solutions  $s$  and  $s'$  can be shown as follows:

- If  $f(\mathbf{s}) < f(\mathbf{s}')$ , then  $f(\mathbf{s})$  dominates  $f(\mathbf{s}')$ , i.e.,  $f(\mathbf{s}) \neq f(\mathbf{s}')$  and  $f_i(\mathbf{s}) \leq f_i(\mathbf{s}')$ ,  $i = 1, \dots, p$ .
- If  $f(\mathbf{s}) \leq f(\mathbf{s}')$ , then  $f(\mathbf{s})$  weakly dominates  $f(\mathbf{s}')$ , i.e.,  $f_i(\mathbf{s}) \leq f_i(\mathbf{s}')$ ,  $i = 1, \dots, p$ .
- If  $f(\mathbf{s}) < f(\mathbf{s}')$ , then  $f(\mathbf{s})$  strictly dominates  $f(\mathbf{s}')$ , i.e.,  $f_i(\mathbf{s}) < f_i(\mathbf{s}')$ ,  $i = 1, \dots, p$ .

Figure 2.3 illustrates the following solution (a) dominates (a'), solution (b) strictly dominates (b'), and solution (c) weakly dominates (c').

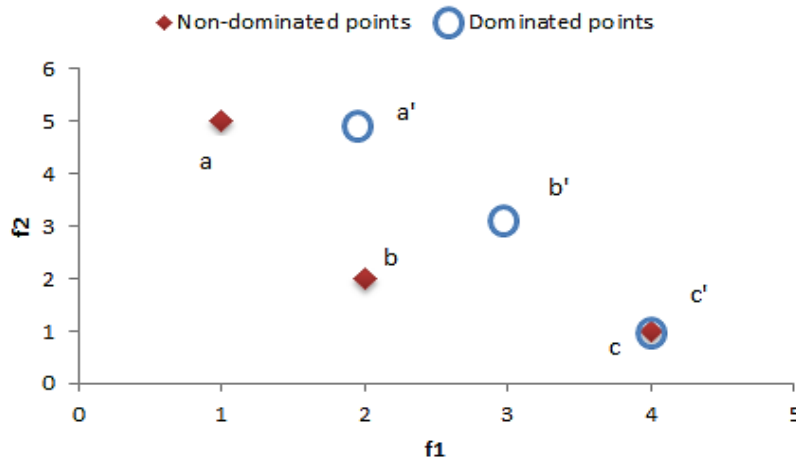


Figure 2.3: Relation between objective function value vectors

Ehrgott [109] points out that the Pareto global optimum solution can be defined as follows: a solution  $s \in S$  is Pareto global optimum solution, if and only if there is no  $s' \in S$  such that  $f(\mathbf{s}') < f(\mathbf{s})$ . Additionally, the Pareto global optimum set can be defined as follows:  $S' \subseteq S$  is a Pareto global optimum set, if and only if it contains only all Pareto global optimum solutions. In addition, this Pareto global optimum set is ordinarily difficult to obtain. After all, it would be better to have an approximation to that set in a reasonable amount of time, where the objective function value vectors rebounded should be non-dominated.

## 2.8 Methods for Solving Multiobjective Combinatorial Optimization Problems

The goal of solving MOCOPs is to obtain a set of the non-dominated solutions, or at least an approximation of it, more quickly. Existing algorithms can be categorized into two major

classes: exact algorithms and approximate (heuristic) algorithms. Exact algorithms can guarantee that the a solution of an MOCOP will be an entirely optimal solution. The drawback is that no exact algorithms can guarantee that such a solution will be found in a reasonable amount of time, since the running time increases dramatically with the instance size. On the other hand, approximate (heuristic) algorithms can be utilized for solving larger instances in a timely fashion and good solutions in a limited amount of time. In single-objective and multiobjective scheduling problems, metaheuristics such as the GA, ACO, GRASP, and tabu search can be utilized.

### **2.8.1 Exact Algorithms**

The operations research literature has cultivated various exact techniques to solve several real-life problems such as branch-and-cut, branch-and-price, and branch-and-cut-and-price linear and integer programming methods. These methods are an implementation of the branch-and-bound algorithm in which linear programming is used to obtain valid bounds during the construction of a search tree. Ruzika and Wiecek [110] provide a comprehensive survey and analyze separately the case of two objective functions, and the case strictly more than two objective functions. Ehrgott [109] discusses different techniques related to multiple objectives integer programming. Also, T'kindt et al. [111] discuss multiobjective optimization approaches.

The dynamic programming method uses the divide-and-conquer principle, and solves many subproblems in which solutions are combined together to form the final solution [112]. Similarly, the branch-and-bound algorithm divides the problem into smaller sets and finds a lower bound for each set in the minimization case. Lawler and Wood [113] provide a detailed survey on the branch-and-bound algorithm. Furthermore, elimination methods are designed for solving specific problems by applying search strategies and reducing the possible solution space.

The multiobjective problem is often solved by converting it into an aggregated scalar objective function. This method, known as the weighted sum method [114], multiplies each objective function by a weighting factor and sums up all terms as

$$\min_{x \in X} \sum_{k=1}^K w_k f_k(x) = w_1 f_1(x) + w_2 f_2(x)$$

where  $k$  is the number of criteria functions,  $x$  is the decision vector,  $X$  is the parameter space, and  $w_k$  stands for weights. Without loss of generality, when this optimization problem is solved, a point in the efficient set is obtained. Systematically repeating the process for various weights defines some representation of the efficient set (Pareto front).

Another solution technique to multiobjective optimization is the  $\varepsilon$ -constraint method, designed by Chankong and Haimes [115]. This method is the most widely used multiobjective solution method in terms of generation (a posteriori) optimization. This method optimizes one of the objective functions while transforming the other objective function into constraints with an upper bound of  $\varepsilon$ , where  $\varepsilon$  is a parameter that varies over iterations as shown below [116]

$$\begin{aligned} & \text{Min} \quad f_1(x) \\ & \text{st} \quad \\ & \quad \quad f_2(x) \leq \varepsilon_2 \\ & \quad \quad x \in S \end{aligned}$$

where  $x$  is the vector of the decision variables,  $f_1(x)$  and  $f_2(x)$  are the objective functions,  $\varepsilon_2$  is on the right-hand side of the constrained objective function, and  $S$  is the feasible region for a two-objective problem. By solving the problem for a different value of  $\varepsilon$ , the Pareto front (efficient solutions) of the problem is obtained. Mavrotas [116] proposes a novel version of the  $\varepsilon$ -constraint method that avoids the production of a weak Pareto-optimal solution and accelerates the process by avoiding redundant iterations. In further research, Mavrotas and Florios [117] use the  $\varepsilon$ -

constraint method to generate Pareto-optimal solutions (i.e., to produce the exact and complete Pareto set, which includes all possible Pareto-optimal solutions) in multiobjective mathematical programming problems with discrete variables. Using the  $\varepsilon$ -constraint method, Salari et al. [118] develop a box algorithm that sequentially generates weak Pareto-optimal points using the  $\varepsilon$ -constraint method. Brown et al. [119] propose a multiobjective security game model and develop iterative- $\varepsilon$ -constraints in order to generate the Pareto front.

The exact Pareto-optimal front can be found by solving a sequence of constrained single-objective optimization problems. First, the payoff table is calculated by using lexicographic optimization of the objective functions. In this approach, the first objective function is optimized, and then, using lexicographic optimization of the objective functions, the second possible alternative objective function is optimized, and so on. The  $f_1(x)$  function is selected to be optimized, obtaining  $\min f_1(x) = z_1^*$ . Then,  $f_2(x)$  is selected to be optimized by adding the constraint  $f_1(x) = z_1^*$ , in order to keep the optimal solution of the first optimization. In conclusion, using lexicographic optimization, results will be more meaningful and adequately describe the Pareto-optimal front [120].

After calculating the payoff table, the objective space is discretized to equal intervals, which are used as values of  $\varepsilon_p$ , where  $P$  is the number of objective functions, in the  $\varepsilon$ -constraint method. As a result, the multiobjective mathematical problem is transferred to the  $\varepsilon$ -constraint equivalent, which is on the right-hand side of the  $\varepsilon_p$ , in order to obtain the Pareto-optimal solutions by varying the parameter  $\varepsilon_p$ . Finally, the Pareto-optimal solutions are mapped onto the above constraint using a parametric problem on  $\varepsilon_p$ , and so on. For example, solving the problem for a different value of  $\varepsilon_p$  will generate a finite number of Pareto-optimal solutions as a target.

### 2.8.2 Heuristic Algorithms

As discussed earlier, there is a need for heuristic algorithms that are capable of finding a near-optimal solution in a timely fashion. Generally, heuristic algorithms are simple procedures that determine a good approximate Pareto front in a reasonable amount of time for multiobjective problems. At any rate, a well-designed heuristic algorithm is able to generate a solution that is at least approximate to an optimal solution [121]. The only drawback is that the quality of the solution obtained cannot be guaranteed. There are three different types of heuristics: (1) those where dispatching rules that select the next job to be scheduled on the basis of when a job is arriving or completing, (2) local search, and (3) decomposition.

Local search heuristics, the most effective approach, are structured to drive the search towards the Pareto front (efficient solutions). They are most effective because of the accessibility and intuitiveness of the concepts behind them, the first step being to start from an initial solution. The next step is to search for an improved solution in an appropriately defined neighborhood of the current solution. The last step is to substitute the initial solution iteratively (constantly) with the improved solution. This procedure is actually intuitive. Based on the objective function, the neighbored solutions of the current solution are assessed, and if the new solution is better than the current, then it is substituted. The stop criterion occurs once a better neighbored solution can no longer be obtained. Note that typically these heuristics are not supposed to generate the same outcome for different runs because they are based on a randomized search process, which generates a different solution every time. They are easier to implement than exact algorithms, and they have a strong ability to generate better solutions for solving large instances of NP-hard problems.

Several researchers and pioneers have proposed local search heuristics (approximations) for multiobjective problems. They can be applied to many problems with small modifications,

such as GAs [122], greedy heuristic algorithms [123], neural networks, [124], simulated annealing, [125], ACO, [126], and tabu search [127]. Ehrgott and Gandibleux [107] discuss different techniques related to multiple objectives. In this dissertation, a GA is utilized in Chapters 3, 5, and 6, ACO is utilized in Chapter 4, and the GRASP is utilized in Chapter 5 to solve different multiobjective scheduling problems.

Constructive algorithms are utilized to generate a good-looking (better) initial solution for the subsequent application of local search algorithms. In some order, they generate all possible solutions from scratch by adding to the initial empty partial-solution components until the whole solution is completed. This method is fast, but the quality is not as good as those generated by local search algorithms [128].

The genetic algorithm concept has been applied successfully to optimize several kinds of real-life problems. GAs generate very good solutions for solving large instances and hard combinatorial optimization problems, and they are inspired by the Charles Darwin's theory of evolution, which states that individuals that are more adaptable can improve their chance of survival in a specific environment.

The GA has three main stages: recombination, mutation, and selection. In the recombination stage, two chromosomes (parents) are chosen, and they generate two new chromosomes, called offspring. The concept behind the crossover operator is to transfer the good features of the parent solutions to the offspring. The mutation operator is used to change some characteristics of the chromosomes by performing random neighborhood moves. Finally, the selection operator uses a deterministic mechanism to choose the best set of offspring chromosomes, which have higher fitness values.

The efficiency of the GA method mainly depends on the shape of the Pareto global optimum regions (solutions) among the presence of constraints and other features. However, the GA needs to be executed iteratively to find all Pareto optimal solutions. In other respects, population-based features are concrete approaches that are used in GAs to solve multiobjective optimization problems. This is due to the population-based feature ability to obtain multiple optimal solutions in a single simulation run. In fact, a set of individuals is called a population, and the population designates feasible solutions. Each individual in a population has a fitness value, according to its non-domination level. Selection of those individuals with higher fitness generates a search direction towards the Pareto-optimal region.

In order to understand GAs, it is important to analyze the methodology of assigning the fitness solution and keeping the diversification of solutions. Several researchers have presented different GA approaches for solving multiobjective optimization problems with a different framework for the rank and selection procedures. The ranking procedure involves assigning a single fitness value to each solution based on its position in the objective space. After that, a selection of a set of solutions is generated by a stochastic sampling procedure based on the solution ranking. Ordinarily, the selections of solutions close to the Pareto front have higher fitness, which is considered to be a better solution. Srinivas and Patnaik [129] provide a detailed survey of this GA heuristic.

Another powerful metaheuristic technique, the ant colony optimization algorithm has become the most widely used technique to solve combinatorial optimization problems. In addition, it has shown a great potential to cope with multiobjective optimization problems. The basic mechanism of the ACO algorithm is that a colony of artificial ants cooperates in finding good solutions to combinatorial optimization problems. The ants' foraging behavior shifts; when ants

look for food, they are capable of finding the shortest path from a food source to their nest without using visual cues [130] by exploiting pheromone information [131]. The key to ants' effectiveness is their pheromones, which is a chemical substance. While traveling from a food source to their nest, and vice versa, ants deposit pheromone substances on the ground. They are capable of smelling the pheromone and following it. New ants will probably prefer to follow the path of stronger pheromone concentrations. This, in turn, increases the number of ants choosing the shortest path. Finally, all ants will be following the shortest path from a food source to their nest.

Colomi et al. [132] propose the first example of the ACO algorithm. Dorigo and Stützle [133] propose the ACO as a new metaheuristic approach for solving hard combinatorial optimization problems in the literature. Bauer et al. [134] propose an ACO application for the single machine total weighted tardiness problem. McMullen [135] proposes the ACO approach to address a just-in-time sequencing problem with multiple objectives. Leguizamón and Coello [136] describe the most relevant and recent developments on the use of the ACO variant for solving multiobjective optimization.

Another metaheuristic, the GRASP, is an iterative process with a construction phase and a local search phase [137]. In the construction phase, an initial solution is generated by randomly adding a piece of the solution that meets certain criteria (i.e., order of jobs is fixed) to the solution in construction. In the local search phase, the newly constructed solution is locally searched to find a better solution according to the objective in a defined neighborhood until local optimality or the satisfied stopping criterion is achieved [138]. The definition of neighborhood is important in the design of the GRASP to ensure convergence of the local search to a local optimum. Also, the best solution is kept over the iteration. Resende and Ribeiro [137] provide an extensive review of this heuristic. Armentano and De Araujo [139] develop a GRASP heuristic to solve the minimization

of the total tardiness on a single machine with setup times. Rocha et al. [140] use a GRASP heuristic to obtain a quick upper bound to utilize in the branch-and-bound algorithm. In another application, Robertson [141] uses GRASP implementations for the multidimensional assignment problem.

## 2.9 Complexity of Algorithms

A branch of mathematics is known as computational complexity analysis, which provides a formal means for evaluating the hardness of an algorithm. In order to develop an appreciation of why some mathematical problems cannot be solved optimally, it is necessary to go from finding the best solution to finding simply a good solution [111]. In problem classes, mathematical problems can be divided into the following two classes according to their complexity:

- Class  $P$  problems are problems that can be solved by algorithms whose computational time grows as a polynomial function of problem size.
- NP-hard problems are those that cannot be solved by any known polynomial algorithm; therefore, the time to find a solution to these problems grows exponentially with problem size. Although it has not been definitively proven, there is no polynomial algorithm for solving NP-hard problems; many eminent mathematicians have tried and failed.

Class  $P$  problems are easy to solve, while NP-hard problems are difficult to solve, and some NP-hard problems are more difficult than others. Efficient algorithms have generated good approximate solutions for some NP-hard problems, but other NP-hard problems are even difficult to solve approximately with efficient algorithms. In scheduling problems, complexity results have been collected by Brucker and Knust [142].

An NP-hard problem means that there is no known algorithm that can solve it in polynomial time. A polynomial algorithm is a procedure for solving a problem that, for any possible instance,

generates the correct answer in a finite amount of time. For example, the simplex method with an anti-cycling rule is an algorithm for solving the linear programming problem. Karmarkar's algorithm is an algorithm introduced by Karmarkar [143], which is applied to solve a problem in polynomial time if its computing time is polynomially bounded by the encoding size of the input. Moreover, Karmarkar's algorithm is a polynomial algorithm used for linear programming but the simplex method is not used for linear programming. Klee and Minty [144] provide a list of instances where the simplex method requires an exponential number of iterations to find the optimal solution.

In order to understand what is implied in polynomial and exponential algorithms, consider a single machine sequencing problem with three jobs. There are many ways to sequence all three jobs. Any one of the three jobs can be the first in the order, which leaves any one of the others to be second in the order, and only one to be last in the order. Therefore, the sequencing number is  $3 \times 2 \times 1 = 6$ , which is expressed as  $(3!)$ , or "3 factorial." However, if the best sequence with respect to some criteria is sought, then it is important to consider implicitly six alternatives. Due to the exponential growth of the factorial function, the amount of time required to obtain the optimal solution also grows exponentially with problem size.

There are three approaches to solve NP-hard problems:

- Approximation algorithms are polynomial algorithms generating approximate solutions whose quality can be estimated a priori.
- Heuristic algorithms generate solutions and do not provide any estimates of their quality.

Their main advantage is reaching a good solution quite fast. Their main drawback is that it is impossible to determine how good the solution is. The most extensive class of such algorithms are called "local search" or metaheuristic algorithms.

- Enumeration algorithms are not polynomial algorithms and are intended to generate an optimal solution of a problem. Typically, there is no guarantee that a solution to a problem will be found in a certain amount of time. If there is no optimal solution found, an enumeration algorithm can ordinarily provide some solution. The most extensive class of enumeration algorithms are branch-and-bound algorithms.

This dissertation focuses on devising heuristic algorithms for solving a multicriteria scheduling problem by introducing multiobjective scheduling optimization problems.

## 2.10 Summary

This chapter provided a literature review on single machine and parallel machine scheduling, as well as solution methods for solving multiobjective scheduling optimization. Interest here focused on total tardiness, total completion time, energy cost, setup times, and load balancing in scheduling. The application of these objectives in different fields was summarized.

## 2.11 References

- [1] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, July 1995, pp. 12-65.
- [2] P. Brucker, *Scheduling Algorithms*, 2nd ed., vol. 3, Springer-Verlag, May 2007, pp. 55-233.
- [3] J. Leung, L. Kelly, and J.H. Anderson, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, 3rd ed., CRC Press, Inc., Aug. 2004, pp. 35-254.
- [4] R. Nessah and I. Kacem, "Branch-and-bound method for minimizing the weighted completion time scheduling problem on a single machine with release dates," *Computers & Operations Research*, vol. 39, no. 3, June 2012, pp. 471-478.
- [5] A.B. Keha, K. Khowala, and J.W. Fowler, "Mixed integer programming formulations for single machine scheduling problems," *Computers & Industrial Engineering*, vol. 56, no. 1, Sept. 2009, pp. 357-367.
- [6] A. Ferrolho and M. Crisostomo. "Single machine total weighted tardiness problem with genetic algorithms," in *IEEE/ACS International Conference on Computer Systems and Applications, AICCSA '07*, vol. 1, July 2007, pp. 1-8.

- [7] U. Al-Turki, C. Fedjki, and A. Andijani, "Tabu search for a class of single machine scheduling problems," *Computers & Operations Research*, vol. 28, no. 12, May 2001, pp. 1223-1230.
- [8] J.Y. Leung, *Algorithms, Models, And Performance Analysis*, 1st ed., vol. 1, CRC Press, Jun. 2001, pp. 33-121.
- [9] M. Queyranne and A.S. Schulz, "Polyhedral approaches to machine scheduling," vol. 1, TU, Fachbereich 3, Sept. 1994, pp. 52-192.
- [10] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: A Survey," in *Annals of Discrete Mathematics*, E.L. Johnson, P.L. Hammer and B.H. Korte, editors, Elsevier, Aug. 1979, pp. 287-326.
- [11] P. Mellor, "A review of job shop scheduling," *Operational Research Quarterly*, vol. 17, no. 2, July 1966, pp. 161-171.
- [12] W.J. Hopp and M.L. Spearman, *Factory Physics*, 4 ed., vol. 1, Waveland Press, Dec. 2011, pp. 12-98.
- [13] M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 6 ed., vol. 1, Springer Science & Business Media, March 2012, pp. 81-281.
- [14] K.R. Baker and D. Trietsch, *Principles of Sequencing and Scheduling*, John Wiley & Sons, Oct. 2009, pp 47-186.
- [15] J. Du and J.Y.T. Leung, "Minimizing total tardiness on one machine is NP-hard," *Mathematics of Operations Research*, vol. 15, no. 3, Feb. 1990, pp. 483-495.
- [16] E.L. Lawler, "A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness," *Annals of Discrete Mathematics*, vol. 1, no. 1, Sept. 1977, pp. 331-342.
- [17] C.N. Potts and L.N. Van, "A branch and bound algorithm for the total weighted tardiness problem," *Operations Research*, vol. 33, no. 2, Dec. 1985, pp. 363-377.
- [18] C.N. Potts and L. Van, "Dynamic programming and decomposition approaches for the single machine total tardiness problem," *European Journal of Operational Research*, vol. 32, no. 3, April 1987, pp. 405-414.
- [19] G.C. Mouzon, Operational methods and models for minimization of energy consumption in a manufacturing environment, PhD dissertation, Department of Industrial and Manufacturing Engineering, Wichita State University, Wichita, Kansas, Dec 2008.
- [20] M.B. Yildirim and G. Mouzon, "Single machine sustainable production planning to minimize total energy consumption and total completion time using a multiple objective genetic algorithm," *IEEE Transactions on Engineering Management*, vol. 59, no. 4, July 2012, pp. 585-597.

- [21] B. Chen, C.N. Potts, and G.J. Woeginger, "A review of machine scheduling: Complexity, algorithms and approximability," in *Handbook of Combinatorial Optimization*, Springer, Dec. 1999, pp. 1493-1641.
- [22] J. Labetoulle, E.L. Lawler, J.K. Lenstra, and A. Rinnooy Kan, "Preemptive scheduling of uniform machines subject to release dates," *Stichting Mathematisch Centrum. Mathematische Besliskunde*, vol. 1, no. 1, BW 166/82, 1982, pp. 1-20.
- [23] E.L. Lawler, "A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs," *Annals of Operations Research*, vol. 26, no. 1, 1990, pp. 125-133.
- [24] K. Bülbül, P. Kaminsky, and C. Yano, "Preemption in single machine earliness/tardiness scheduling," *Journal of Scheduling*, vol. 10, no. 4-5, 2007, pp. 271-292.
- [25] M. Batsyn, B. Goldengorin, P.M. Pardalos, and P. Sukhov, "Online heuristic for the preemptive single machine scheduling problem of minimizing the total weighted completion time," *Optimization Methods and Software*, vol. 29, no. 5, Sept. 2014, pp. 1-9.
- [26] R.W. Conway, W.L. Maxwell, and L.W. Miller, *Theory of Scheduling*, Courier Dover Publications, Oct. 2012, pp 26-87.
- [27] A. Allahverdi, C. Ng, T.E. Cheng, and M.Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *European Journal of Operational Research*, vol. 187, no. 3, Dec. 2008, pp. 985-1032.
- [28] A. Allahverdi and H. Soroush, "The significance of reducing setup times/setup costs," *European Journal of Operational Research*, vol. 187, no. 3, Feb. 2008, pp. 978-984.
- [29] S. Panwalkar, R. Dudek, and M. Smith. "Sequencing research and the industrial scheduling problem," in *Symposium on the Theory of Scheduling and its Applications*, Springer Berlin Heidelberg, June 1973, pp. 29-38.
- [30] L.J. Krajewski, B.E. King, L.P. Ritzman, and D.S. Wong, "Kanban, MRP, and shaping the manufacturing environment," *Management Science*, vol. 33, no. 1, June 1987, pp. 39-57.
- [31] B. Flynn, "The effects of setup time on output capacity in cellular manufacturing," *International Journal of Production Research*, vol. 25, no. 12, Dec. 1987, pp. 1761-1772.
- [32] K. Kogan and E. Levner, "A polynomial algorithm for scheduling small-scale manufacturing cells served by multiple robots," *Computers & Operations Research*, vol. 25, no. 1, Feb. 1998, pp. 53-62.
- [33] C.Y. Liu and S.C. Chang, "Scheduling flexible flow shops with sequence-dependent setup effects," *Robotics and Automation, IEEE Transactions on*, vol. 16, no. 4, July 2000, pp. 408-419.
- [34] S.C. Trovinger and R.E. Bohn, "Setup Time Reduction for Electronics Assembly: Combining Simple (SMED) and IT-Based Methods," *Production and Operations Management*, vol. 14, no. 2, Feb. 2005, pp. 205-217.

- [35] P.C. Chang, J.C. Hsieh, and Y.-W. Wang, "Genetic algorithms applied in BOPP film scheduling problems: minimizing total absolute deviation and setup times," *Applied Soft Computing*, vol. 3, no. 2, Oct. 2003, pp. 139-148.
- [36] H.T. Lin and C.J. Liao, "A case study in a two-stage hybrid flow shop with setup time and dedicated machines," *International Journal of Production Economics*, vol. 86, no. 2, Sept. 2003, pp. 133-143.
- [37] S.R. Gupta and J.S. Smith, "Algorithms for single machine total tardiness scheduling with sequence dependent setups," *European Journal of Operational Research*, vol. 175, no. 2, Jan. 2006, pp. 722-739.
- [38] T.Y. Chang, F.D. Chou, and C.-E. Lee, "A heuristic algorithm to minimize total weighted tardiness on a single machine with release dates and sequence-dependent setup times," *Journal of the Chinese institute of industrial engineers*, vol. 21, no. 3, June 2004, pp. 289-300.
- [39] S.M. Lee and A.A. Asllani, "Job scheduling with dual criteria and sequence-dependent setups: mathematical versus genetic programming," *Omega*, vol. 32, no. 2, 2004, pp. 145-153.
- [40] G. Rabadi, M. Mollaghasemi, and G.C. Anagnostopoulos, "A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time," *Computers & Operations Research*, vol. 31, no. 10, Oct. 2004, pp. 1727-1751.
- [41] A. Agnetis, J.C. Billaut, S. Gawiejnowicz, D. Pacciarelli, and A. Soukhal, "Parallel Machine Scheduling Problems," in *Multiagent Scheduling*, Springer, Sept. 2014, pp. 189-215.
- [42] J.W. Herrmann and C.Y. Lee, "On parallel machine scheduling with operator-constrained setups," *Institute For System Research*, Oct. 1994.
- [43] S.F. Ghomi and F.J. Ghazvini, "A pairwise interchange algorithm for parallel machine scheduling," *Production Planning & Control*, vol. 9, no. 7, 1998, pp. 685-689.
- [44] L. Tang and J. Luo, "A new ILS algorithm for parallel machine scheduling problems," *Journal of Intelligent Manufacturing*, vol. 17, no. 5, 2006, pp. 609-619.
- [45] M.B. Yildirim, E. Duman, K.K. Krishnan, and K. Senniappan, "Parallel machine scheduling with load balancing and sequence dependent setups," in *Wichita State University Libraries SOAR: Shocker Open Access Repository*, Wichita State University, Nov. 2006.
- [46] E. Duman, M.B. Yildirim, and A.F. Alkaya, "Scheduling continuous aluminium casting lines," *International Journal of Production Research*, vol. 46, no. 20, Nov. 2008, pp. 5701-5718.

- [47] T. Koltai, "Aggregate production planning of flexible manufacturing systems using the concept of operation types," in *University of Michigan, Business School, Research Support*, June 1998.
- [48] A. Farkas, T. Koltai, and K.E. Stecke, "Workload balancing using the concept of operation types," *University of Michigan Business School Working Paper*, Sep1999, pp. 99-002.
- [49] N. Kumar and K. Shanker, "Comparing the effectiveness of workload balancing objectives in FMS loading," *International Journal of Production Research*, vol. 39, no. 5, Dec. 2001, pp. 843-871.
- [50] T. Hayrinen, M. Johnsson, T. Johtela, J. Smed, and O. Nevalainen, "Scheduling algorithms for computer-aided line balancing in printed circuit board assembly," *Production Planning & Control*, vol. 11, no. 5, Nov. 2000, pp. 497-510.
- [51] S.M. Saad, A. Baykasoglu, and N.N. Gindy, "A new integrated system for loading and scheduling in cellular manufacturing," *International Journal of Computer Integrated Manufacturing*, vol. 15, no. 1, May 2002, pp. 37-49.
- [52] N. Weinert, S. Chiotellis, and G. Seliger, "Methodology for planning and operating energy-efficient production systems," *CIRP Annals-Manufacturing Technology*, vol. 60, no. 1, July 2011, pp. 41-44.
- [53] C. Schmidt, W. Li, S. Thiede, S. Kara, and C. Herrmann, "A methodology for customized prediction of energy consumption in manufacturing industries," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 2, no. 2, Oct. 2015, pp. 163-172.
- [54] G. Mustafaraj, J. Cosgrove, M.J. Rivas-Duarte, F. Hardiman, and J. Harrington, "A methodology for determining auxiliary and value-added electricity in manufacturing machines," *International Journal of Production Research*, vol. 53, no. 17, Dec 2015, pp. 1-13.
- [55] F. Shrouf and G. Miragliotta, "Energy management based on Internet of things: practices and framework for adoption in production management," *Journal of Cleaner Production*, vol. 100, no. 1, Aug. 2015, pp. 235-246.
- [56] S. Mousavi, S. Thiede, W. Li, S. Kara, and C. Herrmann, "An integrated approach for improving energy efficiency of manufacturing process chains," *International Journal of Sustainable Engineering*, Jan. 2015, pp. 1-14.
- [57] G. May, I. Barletta, B. Stahl, and M. Taisch, "Energy management in production: A novel method to develop key performance indicators for improving energy efficiency," *Applied Energy*, vol. 149, no. 1, July 2015, pp. 46-61.
- [58] A. Fysikopoulos, G. Pastras, T. Alexopoulos, and G. Chryssolouris, "On a generalized approach to manufacturing energy efficiency," *The International Journal of Advanced Manufacturing Technology*, vol. 73, no. 9-12, 2014, pp. 1437-1452.

- [59] T. Devoldere, W. Dewulf, W. Deprez, B. Willems, and J.R. Duflou, "Improvement potential for energy consumption in discrete part production machines," in *Advances in Life Cycle Engineering for Sustainable Manufacturing Businesses*, Springer, Nov. 2007, pp. 311-316.
- [60] A. Dietmair and A. Verl, "A generic energy consumption model for decision-making and energy efficiency optimisation in manufacturing," *International Journal of Sustainable Engineering*, vol. 2, no. 2, Oct. 2009, pp. 123-133.
- [61] H. Zhang, F. Zhao, K. Fang, and J.W. Sutherland, "Energy-conscious flow shop scheduling under time-of-use electricity tariffs," *CIRP Annals-Manufacturing Technology*, vol. 63, no. 1, March 2014, pp. 37-40.
- [62] H. Zhang, F. Zhao, and J.W. Sutherland, "Energy-efficient scheduling of multiple manufacturing factories under real-time electricity pricing," *CIRP Annals-Manufacturing Technology*, vol. 64, no.1, July 2015, pp. 41-44.
- [63] J. Cheng, F. Chu, W. Xia, J. Ding, and X. Ling. "Bi-objective optimization for single machine batch scheduling considering energy cost," in *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on IEEE*, Nov. 2014, pp. 236-241.
- [64] K. Fang, N.A. Uhan, F. Zhao, and J.W. Sutherland. "Scheduling on a single machine under time-of-use electricity tariffs," May 2014, [cited: Feb. 2015]; available from World Wide Web: [http://www.optimization-online.org/DB\\_FILE/2014/04/4313.pdf](http://www.optimization-online.org/DB_FILE/2014/04/4313.pdf).
- [65] A. Ghobeity and A. Mitsos, "Optimal time-dependent operation of seawater reverse osmosis," *Desalination*, vol. 263, no. 1, Oct. 2010, pp. 76-88.
- [66] S.T. Newman, A. Nassehi, R. Imani-Asrai, and V. Dhokia, "Energy efficient process planning for CNC machining," *CIRP Journal of Manufacturing Science and Technology*, vol. 5, no. 2, Sept. 2012, pp. 127-136.
- [67] Y. Seow and S. Rahimifard, "A framework for modelling energy consumption within manufacturing systems," *CIRP Journal of Manufacturing Science and Technology*, vol. 4, no. 3, Dec. 2011, pp. 258-264.
- [68] J.Y. Moon, K. Shin, and J. Park, "Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency," *The International Journal of Advanced Manufacturing Technology*, vol. 68, no. 1-4, March 2013, pp. 523-535.
- [69] J. Escamilla, M.A. Salido, A. Giret, and F. Barber, "A Metaheuristic Technique for Energy-Efficiency in Job-Shop Scheduling," *Constraint Satisfaction Techniques for Planning and Scheduling*, Nov. 2014, pp. 42-52.
- [70] C. Herrmann, S. Thiede, S. Kara, and J. Hesselbach, "Energy oriented simulation of manufacturing systems—Concept and application," *CIRP Annals-Manufacturing Technology*, vol. 60, no. 1, June 2011, pp. 45-48.

- [71] P. Solding and P. Thollander. "Increased energy efficiency in a Swedish iron foundry through use of discrete event simulation," in *Simulation Conference, 2006. WSC 06. Proceedings of the Winter IEEE*, Dec. 2006, pp. 1971-1976.
- [72] C. Herrmann and S. Thiede, "Process chain simulation to foster energy efficiency in manufacturing," *CIRP Journal of Manufacturing Science and Technology*, vol. 1, no. 4, 2009, pp. 221-229.
- [73] J.f. Wang, S.q. Li, and J.-h. Liu. "A multi-granularity model for energy consumption simulation and control of discrete manufacturing system," in *The 19th International Conference on Industrial Engineering and Engineering Management*, Springer, June 2013, pp. 1055-1064.
- [74] L. Li and Z. Sun, "Dynamic Energy Control for Energy Efficiency Improvement of Sustainable Manufacturing Systems Using Markov Decision Process," *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, vol. 43, no. 5, 2013, pp. 1195-1205.
- [75] Y. He, B. Liu, X. Zhang, H. Gao, and X. Liu, "A modeling method of task-oriented energy consumption for machining manufacturing system," *Journal of Cleaner Production*, vol. 23, no. 1, Oct. 2012, pp. 167-174.
- [76] G. Mouzon, M.B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *International Journal of Production Research*, vol. 45, no. 18-19, July 2007, pp. 4247-4271.
- [77] F. Shrouf, J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *Journal of Cleaner Production*, vol. 67, no. 15, March. 2014, pp. 197-207.
- [78] M. Dai, D. Tang, Y. Xu, and W. Li, "Energy-aware integrated process planning and scheduling for job shops," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 229, no. 1, Oct. 2014, pp. 13-26.
- [79] C. Duerden, L. Shark, G. Hall, and J. Howe. "Genetic algorithm based modification of production schedule for variance minimisation of energy consumption," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, Oct. 2014
- [80] J.-Y. Ding, S. Song, and C. Wu, "Carbon-efficient scheduling of flow shops by multi-objective optimization," *European Journal of Operational Research*, vol. 248, no. 3, Feb. 2015, pp. 758-771.
- [81] Y. Liu, H. Dong, N. Lohse, S. Petrovic, and N. Gindy, "An investigation into minimising total energy consumption and total weighted tardiness in job shops," *Journal of Cleaner Production*, vol. 65, Feb. 2014, pp. 87-96.
- [82] G. May, B. Stahl, M. Taisch, and V. Prabhu, "Multi-objective genetic algorithm for energy-efficient job shop scheduling," *International Journal of Production Research*, vol. 45, no. 18-19, Jan 2015, pp. 1-19.

- [83] M.A. Salido, J. Escamilla, F. Barber, A. Giret, D. Tang, and M. Dai. "Energy-aware parameters in job-shop scheduling problems," in *GREEN-COPLAS 2013: IJCAI 2013 Workshop on Constraint Reasoning, Planning and Scheduling Problems for a Sustainable Future*, vol. 1, May 2013, pp. 44-53.
- [84] Y. He and F. Liu, "Methods for integrating energy consumption and environmental impact considerations into the production operation of machining processes," *Chinese Journal of Mechanical Engineering*, vol. 23, no. 4, Oct. 2010, pp. 428.
- [85] J. Wang, J. Li, and N. Huang. "Optimal scheduling to achieve energy reduction in automotive paint shops," in *ASME 2009 International Manufacturing Science and Engineering Conference*, American Society of Mechanical Engineers, Dec. 2009, pp. 161-167.
- [86] S. Ashok, "Peak-load management in steel plants," *Applied energy*, vol. 83, no. 5, 2006, pp. 413-424.
- [87] K. Fang, N. Uhan, F. Zhao, and J.W. Sutherland, "A new shop scheduling approach in support of sustainable manufacturing," in *Glocalized Solutions for Sustainability in Manufacturing*, Springer, March 2011, pp. 305-310.
- [88] K. Fang, N. Uhan, F. Zhao, and J.W. Sutherland, "A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction," *Journal of Manufacturing Systems*, vol. 30, no. 4, Oct. 2011, pp. 234-240.
- [89] F. Xu, W. Weng, and S. Fujimura. "Energy-efficient scheduling for flexible flow shops by using MIP," in *IIE Annual Conference. Proceedings*, Institute of Industrial Engineers-Publisher, Nov. 2014, pp. 1040.
- [90] A. Bruzzone, D. Anghinolfi, M. Paolucci, and F. Tonelli, "Energy-aware scheduling for improving manufacturing process sustainability: A mathematical model for flexible flow shops," *CIRP Annals-Manufacturing Technology*, vol. 61, no. 1, July 2012, pp. 459-462.
- [91] K. Nilsson and M. Söderström, "Industrial applications of production planning with optimal electricity demand," *Applied energy*, vol. 46, no. 2, Sept. 1993, pp. 181-192.
- [92] K. Nilsson, "Industrial production planning with optimal electricity cost," *Energy conversion and management*, vol. 34, no. 3, Nov. 1993, pp. 153-158.
- [93] A. Pechmann and I. Schöler, "Optimizing energy costs by intelligent production scheduling," in *Glocalized Solutions for Sustainability in Manufacturing*, Springer, Dec. 2011, pp. 293-298.
- [94] G. Mouzon and M.B. Yildirim, "A framework to minimise total energy consumption and total tardiness on a single machine," *International Journal of Sustainable Engineering*, vol. 1, no. 2, June 2008, pp. 105-116.
- [95] C. Herrmann, L. Bergmann, S. Thiede, and A. Zein, "Framework for integrated analysis of production systems," in *Advances in Life Cycle Engineering for Sustainable Manufacturing Businesses*, Springer, March 2007, pp. 195-200.

- [96] J. Heilala, S. Vatanen, H. Tonteri, J. Montonen, S. Lind, B. Johansson, and J. Stahre. "Simulation-based sustainable manufacturing system design," in *Simulation Conference, WSC 2008 Winter IEEE*, Oct. 2008, pp. 1922-1930.
- [97] C. Herrmann, S. Thiede, J. Stehr, and L. Bergmann, "An environmental perspective on Lean Production," in *Manufacturing Systems and Technologies for the New Frontier*, Springer, June 2008, pp. 83-88.
- [98] Y. Liu, H. Dong, N. Lohse, S. Petrovic, and N. Gindy, "An investigation into minimising total energy consumption and total weighted tardiness in job shops," *Journal of Cleaner Production*, vol. 65, no. 1, Feb. 2013, pp. 87-96.
- [99] P. Solding and D. Petku. "Applying energy aspects on simulation of energy-intensive production systems," in *Proceedings of the 37th conference on Winter simulation*, Winter Simulation Conference, May 2005, pp. 1428-1432.
- [100] L.A. Wolsey, *Integer Programming*, vol. 42, New York: Wiley, Nov. 1998, pp 42-96.
- [101] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, March 1999, pp. 61-124.
- [102] H. Hoogeveen, "Multicriteria scheduling," *European Journal of Operational Research*, vol. 167, no. 3, Dec. 2005, pp. 592-623.
- [103] G.W. Evans, "An Overview of Techniques for Solving Multiobjective Mathematical Programs," *Management Science*, vol. 30, no. 11, Nov. 1984, pp. 1268-1282.
- [104] T.D. Fry, R.D. Armstrong, and H. Lewis, "A framework for single machine multiple objective sequencing research," *Omega*, vol. 17, no. 6, June 1989, pp. 595-607.
- [105] L.F. Paquete, *Stochastic Local Search Algorithms for Multiobjective Combinatorial Optimizations: Methods and Analysis*, IOS Press, Inc., May 2006, pp 72-264.
- [106] E. Hossain, D. Niyato, and Z. Han, *Dynamic Spectrum Access and Management in Cognitive Radio Networks*, Cambridge University Press, Nov. 2009, pp 26-84.
- [107] M. Ehrgott and X. Gandibleux, "Approximative solution methods for multiobjective combinatorial optimization," *Top*, vol. 12, no. 1, July 2004, pp. 1-63.
- [108] M. Ehrgott, "Hard to say it's easy — four reasons why combinatorial multiobjective programmes are hard," in *Research and Practice in Multiple Criteria Decision-making* ed., Y. Haimmes and R. Steuer, Editors, Springer Berlin Heidelberg, March 2000, pp. 69-80.
- [109] M. Ehrgott, *Multicriteria Optimization*, Springer Berlin Heidelberg, Aug. 2005, pp. 32-108.
- [110] S. Ruzika and M.M. Wiecek, "Approximation methods in multiobjective programming," *Journal of Optimization Theory and Applications*, vol. 126, no. 3, June 2005, pp. 473-501.
- [111] V. T'kindt, H. Scott, and J.C. Billaut, *Multicriteria Scheduling: Theory, Models And Algorithms*, Springer Science & Business Media, Dec. 2006, pp 62-267.

- [112] M. Pinedo, *Planning and Scheduling in Manufacturing and Services*, vol. 3: Springer, Aug. 2005, pp 96-157.
- [113] E.L. Lawler and D.E. Wood, "Branch-and-bound methods: A survey," *Operations Research*, vol. 14, no. 4, 1966, pp. 699-719.
- [114] M. Caramia and P. Dell'Olmo, *Multi-Objective Management In Freight Logistics: Increasing Capacity, Service Level And Safety With Optimization Algorithms*, Springer Science & Business Media, Oct. 2008, pp. 68-208.
- [115] V. Chankong and Y. Haimes, "Multiobjective decision-making: theory and methodology," *New York: Noth-Holland*, vol. 1, no. 1, April 1983, pp. 59-93.
- [116] G. Mavrotas, "Effective implementation of the  $\epsilon$ -constraint method in Multi-Objective Mathematical Programming problems," *Applied Mathematics and Computation*, vol. 213, no. 2, July 2009, pp. 455-465.
- [117] G. Mavrotas and K. Florios " A novel version of the  $\epsilon$ -constraint method for finding the exact Pareto set in Multi-Objective Integer Programming problems," in *GAMS Optimization Software*, Dec. 2011.
- [118] E. Salari, J. Wala, and D. Craft, "Exploring trade-offs between VMAT dose quality and delivery efficiency using a network optimization approach," *Physics in medicine and biology*, vol. 57, no. 17, July 2012, pp. 5587.
- [119] M. Brown, B. An, C. Kiekintveld, F. Ordóñez, and M. Tambe, "An extended study on multi-objective security games," *Autonomous agents and multi-agent systems*, vol. 28, no. 1, Oct. 2014, pp. 31-71.
- [120] M. Caramia and P. Dell'Olmo, *Multi-Objective Management In Freight Logistics*, Springer Science & Business Media, June 2008, pp 56-94.
- [121] Z. Michalewicz and D.B. Fogel, *How to Solve It: Modern Heuristics*, Springer, March 2004, pp 38-69.
- [122] J.D. Schaffer, *Some experiments in machine learning using vector evaluated genetic algorithms (artificial intelligence, optimization, adaptation, pattern recognition)*, PhD dissertation, Dep. of Industrial Engineering, Vanderbilt University, T.N. , May 1984.
- [123] M.J. Rosenblatt and Z. Sinuany-Stern, "Generating the Discrete Efficient Frontier to the Capital Budgeting Problem," *Operations Research*, vol. 37, no. 3, Nov. 1989, pp. 384-394.
- [124] B. MALAKOOTI, J. WANG, and E.C. TANDLER, "A sensor-based accelerated approach for multi-attribute machinability and tool life evaluation," *The International Journal Of Production Research*, vol. 28, no. 12, Oct. 1990, pp. 2373-2392.
- [125] P. Serafini, " Simulated annealing for multi objective optimization problems," chap. 4 in *Multiple Criteria Decision-making*, Springer, New York, Nov. 1994, pp. 283-292.

- [126] M. Dorigo, V. Maniezzo, and A. Coloni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, March 1996, pp. 29-41.
- [127] X. Gandibleux, N. Mezdaoui, and A. Fréville, "A tabu search procedure to solve multiobjective combinatorial optimization problems," in *Advances In Multiple Objective And Goal Programming*, Springer, May 1997, pp. 291-300.
- [128] H. Hoos and T. Stützle, *Stochastic Local Search: Foundations and Applications*, Morgan Kaufmann Publishers Inc., Oct 2004, pp 86-302.
- [129] M. Srinivas and L.M. Patnaik, "Genetic algorithms: A survey," *Computer*, vol. 27, no. 6, Sept. 1994, pp. 17-26.
- [130] B. Hölldobler, *The Ants*, Harvard University Press, Sept. 1990, pp 64-112.
- [131] M. Dorigo and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, July 1997, pp. 53-66.
- [132] A. Coloni, M. Dorigo, and V. Maniezzo. "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142: Paris, France, Aug. 1991, pp. 134-142.
- [133] M. Dorigo and T. Stützle, "The ant colony optimization metaheuristic: Algorithms, applications, and advances," in *Handbook of Metaheuristics*, Springer, Dec. 2003, pp. 250-285.
- [134] A. Bauer, B. Bullnheimer, R.F. Hartl, and C. Strauss. "An ant colony optimization approach for the single machine total tardiness problem," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2: IEEE, Oct. 1999
- [135] P.R. McMullen, "An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives," *Artificial Intelligence in Engineering*, vol. 15, no. 3, June 2001, pp. 309-317.
- [136] G. Leguizamón and C.A.C. Coello, "Multi-objective ant colony optimization: A taxonomy and review of approaches," *Integration of Swarm Intelligence and Artificial Neural Network*, July 2011, pp. 67-94.
- [137] M.G. Resende and C.C. Ribeiro, "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications," in *Handbook of Metaheuristics*, Springer, Oct. 2010, pp. 283-319.
- [138] M.G. Resende, "Greedy randomized adaptive search procedures greedy randomized adaptive search procedures (GRASP)," *Encyclopedia of optimization*, July 2009, pp. 1460-1469.

- [139] V.A. Armentano and O.C.B. De Araujo, "Grasp with memory-based mechanisms for minimizing total tardiness in single machine scheduling with setup times," *Journal of Heuristics*, vol. 12, no. 6, Oct. 2006, pp. 427-446.
- [140] P.L. Rocha, M.G. Ravetti, and G.R. Mateus. "The metaheuristic GRASP as an upper bound for a branch and bound algorithm in a scheduling problem with non-related parallel machines and sequence-dependent setup times," in *Proceedings of the 4th EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-Heuristics*, vol. 1, Oct. 2004, pp. 62-67.
- [141] A.J. Robertson, "A set of greedy randomized adaptive local search procedure (GRASP) implementations for the multidimensional assignment problem," *Computational Optimization and Applications*, vol. 19, no. 2, Aug. 2001, pp. 145-164.
- [142] P. Brucker. and S. Knust., *Complexity Results For Scheduling Problems*, University of Osnabrueck, Dec. 2009, pp. 57-195.
- [143] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, March 1984, pp. 373-395.
- [144] V. Klee and G.J. Minty, "How good is the simplex algorithm?," *Inequalities, III, Proc. Third Sympos (Univ. California, CA, 1969)*, Academic Press, New York, vol., no., May 1972, pp. 159 - 175.

## CHAPTER 3

### MULTIOBJECTIVE MIXED-INTEGER MATHEMATICAL PROGRAMMING MODEL TO MINIMIZE TOTAL ENERGY COST AND TOTAL TARDINESS ON NON-PREEMPTIVE SINGLE MACHINE

#### Abstract

Energy-aware scheduling in a manufacturing environment with real-time energy pricing is a challenging problem. The purpose of this chapter is to study a scheduling problem on a non-preemptive single machine to minimize the total tardiness and total energy cost under time-of-use electricity tariffs with varying energy prices, which is a multiobjective mixed-integer mathematical programming model. This proposed model is solved via several methods including WSM, and multiobjective GAs based on dominance rank (GA-1), weighted sum aggregation (GA-2), and dominance rank and crowding distance comparison (GA-3). All metaheuristics solve the model and obtain an approximate Pareto front in a reasonable amount of time. Detailed experimental results evaluating the performance of the proposed GA algorithms are provided. A case study illustrates how the results of this multiobjective model could be utilized in decision-making to select a solution among all solutions on the Pareto front by using the TOPSIS method.

**Keywords:** Green manufacturing, Energy-aware scheduling, Time-of-use electricity tariffs, Multiobjective optimization, Genetic algorithm

#### 3.1 Introduction

Energy-aware decision-making and operations management are currently gaining significant momentum, especially by individuals and companies who champion sustainability. Companies are not only improving their product performance but also optimizing production processes to improve energy consumption in order to manage environmental challenges [1]. In the last 60 years, the consumption of energy by the industrial sector has virtually doubled [2]. Today,

industrial sectors, including manufacturing, agriculture, mining, and construction, consume about half of the world's energy [3]. The United States consumes about one-third of the world's energy used, contributes about 28% of greenhouse gas emissions [4], and spends about \$200 billion per year on energy costs alone [5]. In addition, industrial energy consumption is expected to increase 40% by 2030 [2]. In China, manufacturers consume about 50% of the entire electricity produced and contribute to at least 26% of carbon dioxide emissions [6]. In Germany, manufacturers are responsible for about 47% of the total national electricity consumption and generate about 20% of the total carbon dioxide emissions [1]. With better operational planning, manufacturing industry may reduce electricity consumption significantly.

Investment in energy-efficiency in most countries has largely been driven by higher energy prices and also interest in sustainable living. Many countries have urged citizens and manufacturing environments to reduce their energy consumption and improve their environmental performance. Therefore, energy is no longer considered an additional cost but rather a valuable and manageable resource.

Most existing research on manufacturing energy consumption has focused on developing more energy-efficient machines or machining processes [2]. In addition, energy requirements for manufacturing equipment operations have also been investigated. Drake et al. [7] show that machine start-up and machine idling consume a significant amount of energy. For example, compressors in an industrial setting consume about 50% of the maximum power when they are idle [8]. Mouzon et al. [3] point out that there is a significant amount of energy savings if non-bottleneck machines are turned off instead of keeping them in idle mode for a long period of time. As a result, in a mass production environment, more than 85% of the energy is consumed for activities that are not directly related to production processes [2], suggesting that energy-saving

efforts may be lost on manufacturing systems. Consequently, significant attention has been paid to energy-aware manufacturing planning in order to reduce energy costs in manufacturing systems via production scheduling [9].

Shop floor schedules can significantly affect energy consumption, energy cost, and environmental impacts at the level of individual machines [2]. The aim of this chapter is to provide a framework for sustainable and green manufacturing by accomplishing energy consumption savings without any major equipment investment. The chapter focuses its efforts on operations that could provide significant savings in addition to reductions that may be gained by designing more energy-efficient machines with less power demand and energy consumption. The goal here is to minimize the scheduling objective, total tardiness, and total energy consumption (energy) cost in a non-preemptive single machine setting under time-of-use tariffs. This problem is modeled as a multiobjective mixed-integer optimization problem, which, when solved, will yield a set of non-dominated solutions (Pareto front). The methods utilized to solve this problem are as follows: (1) WSM to obtain an exact Pareto front and (2) multiobjective GAs based on dominance rank (GA-1), weighted sum aggregation (GA-2), and dominance rank and crowding distance comparison (GA-3), in order to obtain an approximate Pareto front. Moreover, the exact method Pareto front is used as a benchmark to evaluate the performance of proposed GA algorithms that generate an approximate Pareto front. Finally, the TOPSIS method is used to select the most appropriate solution based on certain criteria to illustrate how this multiobjective framework could be utilized to help decision-makers choose a final solution to be implemented.

The scheduling objective, i.e., minimization of total tardiness on a single machine, is an NP-hard problem [10]. Potts and Van [11] utilize the branch-and-bound method to solve the total tardiness problem on a single machine. In further research, Potts and Van [12] propose an exact

algorithm solution and dynamic programming to solve the total tardiness problem on a single machine.

The energy cost minimization objective aims to find a schedule for the jobs on a single machine such that the total energy consumption cost is minimized. In a manufacturing environment, there has been significant interest in reducing the energy consumption. Mouzon et al. [3] investigate the single machine job-shop scheduling problem to minimize total energy consumption, and observe that non-bottleneck machines could be turned off until needed, rather than left idle, which could lead to 80% savings in total energy consumption. Dai et al. [13] propose an energy-efficient model for flexible flow shop scheduling.

Shrouf et al. [14] consider variable energy prices during one day, and propose a mathematical model to minimize energy consumption costs for single machine production scheduling during production processes. Zhang et al. [15] develop and use a time-indexed integer programming formulation to identify manufacturing scheduling that minimizes electricity cost and the carbon footprint under time-of-use tariffs without compromising production throughput and changing the sequence of jobs. In addition, Zhang et al. [16] use a distributed optimization approach to minimize the total electricity cost as a function of manufacturing schedule subject to real-time electricity pricing. Fang et al. [17] consider the problem of scheduling jobs on a single machine to minimize the total electricity cost of processing these jobs under time-of-use electricity tariffs, in which energy prices vary hourly and are typically announced a day ahead or an hour ahead. Cheng et al. [18] investigate a bi-objective single machine batch scheduling problem with time-of-use policy to improve productivity and minimize the total electricity cost.

Multiobjective scheduling methods determine solutions that optimize more than one, often conflicting, objectives using the multiple criteria decision-making theory [19]. In addition,

multiobjective metaheuristics, such as GAs, tabu search, and simulated annealing optimization, have been studied to find an approximate optimal Pareto front in a reasonable amount of time. Mouzon and Yildirim [4] propose a mathematical model and a multiobjective greedy random adaptive search metaheuristic that minimizes total tardiness and total energy consumption of a single machine scheduling problem. In further research, Yildirim and Mouzon [8] propose a multiobjective GA to minimize energy consumption and total completion time on a single machine. In order to expedite the computational time required, they also integrate a heuristic to obtain Pareto optimal solutions for a sequence of jobs instead of solving a multiobjective linear programming subproblem. Liu et al. [6] study the job-shop scheduling problem and utilize a non-dominated sorting GA (NSGA-II) to minimize the total tardiness and electricity consumption. Chang et al. [20] utilize the NSGA-II to solve single-objective and multiobjective flow shop scheduling problems, respectively. Varadharajan and Rajendran [21] propose a multiobjective simulated annealing algorithm to solve permutation flow shop scheduling with the total flow time of jobs and makespan objectives. Eren and Güner [22] study a single machine bi-criteria problem with improving ability as a result of repeating the same or similar tasks, which is known as the learning effect, and they develop a technique called the discrete differential evaluation algorithm. Baykasoğlu et al. [23] model a multiobjective job-shop scheduling problem and develop a metaheuristic based on a multiple dispatching rule and tabu search.

The key contributions of this chapter are as follows: (1) energy cost with a scheduling objective while planning jobs on a non-preemptive single machine is considered; (2) a multiobjective mixed-integer mathematical problem to minimize total energy cost and total tardiness is proposed; (3) three multiobjective GAs to obtain a near-optimal Pareto front in a timely

fashion are developed; (4) analysis and detailed experimental results evaluating the performance of the algorithms, which greatly maintains the quality of solutions, are provided.

The organization of this chapter is as follows: In sections 3.2 and 3.3, the multiobjective model is defined, and a WSM is utilized to evaluate the performance of the heuristic. In section 3.4, GA metaheuristic algorithms are proposed to solve the multiobjective problem. Section 3.5 presents the experimental design and some performance measures for evaluating the algorithm. Section 3.6 discusses results of the proposed GA algorithms. Section 3.7 discusses results of the best GA algorithm and the WSM. In section 3.8, a comparison of performance of the best GA algorithm with a random number of generations is discussed. Section 3.9 discusses the GA algorithm with increasing speed of the CPLEX solver, thus limiting the solution time. In section 3.10, a case study is developed to illustrate the best GA algorithm. Finally, section 3.11, presents the conclusion and future work of this research.

### **3.2 Multiobjective Mixed-Integer Programming Model to Minimize Energy Cost and Total Tardiness (MMIP-MEC-TT)**

The MMIP-MEC-TT problem is a single machine scheduling problem with jobs having known deterministic processing times ( $p_j$ ) and due dates ( $d_j$ ), which are available at the same time (i.e.  $r_j = 0$ ). Under time-of-use electricity tariffs, the electricity prices vary over times. The machine can process one job at a time, and once the job process is started, it cannot be interrupted, i.e., jobs are non-preemptive. The objectives here are to minimize the total energy cost and total tardiness. MMIP-MEC-TT is formulated as a mixed-integer programming problem. Note that a multiobjective problem is NP-hard when one of the objectives being solved is NP-hard [19].

To model MMIP-MEC-TT, the time-index variable formulation is utilized, whereby the planning horizon is discretized into intervals of one unit length. The time-index variable

formulation for non-preemptive single machine scheduling problems was presented by Sousa and Wolsey [24].

### 3.2.1 Notation and Formulation

The notation to model MMIP-MEC-TT is as follows:

#### Sets

$N$  Jobs,  $\{1, \dots, n\}$

$T$  Planning horizon,  $\{1, \dots, |T|\}$

#### Indices

$j, i$  Jobs,  $j, i \in N$

$t, t'$  Time interval,  $t, t' \in T$

#### Parameters

$M$  A large number

$E_t$  Electric price signal at  $t$

$d_j$  Due date of job  $j$

$p_j$  Process time of job  $j$

#### Variables

$C_j$  Completion time of job  $j$

$\ell_j$  Tardiness of job  $j$ ,  $\ell_j = \max\{0, C_j - d_j\}$

$x_{jt} = \begin{cases} 1, & \text{if job starts at time } t \\ 0, & \text{otherwise} \end{cases}$

$\delta_{ji} = \begin{cases} 1, & \text{if job } j \text{ is processed before job } i \\ 0, & \text{otherwise} \end{cases}$

$\gamma_{jt} = \begin{cases} 1, & \text{if binary variable } x_{jt} \text{ equals } 0 \\ 0, & \text{otherwise} \end{cases}$

$$h_{jt'} = \begin{cases} 1, & \text{if job processes over time } t' \\ 0, & \text{otherwise} \end{cases}$$

The proposed time-indexed MMIP-MEC-TT model is a mathematical program with multiple objectives and several constraints:

$$\text{Minimize } \sum_{j=1}^n \ell_j \quad (3.1)$$

$$\text{Minimize } \sum_{j=1}^n \sum_{t \in T} E_t h_{jt} \quad (3.2)$$

$$\sum_{t=1}^{|T|-p_j+1} x_{jt} = 1 \quad \forall j \in N \quad (3.3)$$

$$\sum_{j=1}^n \sum_{t'=\max(0,t-p_j+1)}^t x_{jt'} \leq 1 \quad \forall t \in T \quad (3.4)$$

$$C_j \geq \sum_{t=1}^{|T|-p_j+1} (t-1+p_j)x_{jt} \quad \forall j \in N \quad (3.5)$$

$$\ell_j \geq C_j - d_j \quad \forall j \in N \quad (3.6)$$

$$x_{jt} \leq M(1-\gamma_{jt}) \quad \forall j \in N; \forall t \in T \quad (3.7)$$

$$-(h_{jt'} - 1) \leq M\gamma_{jt} \quad \forall j \in N; t' \geq t \text{ and } t' \leq (t+p_j-1) \quad (3.8)$$

$$C_j + p_i \leq C_i + M(1-\delta_{ji}) \quad \forall j, i \in N \text{ and } j < i \quad (3.9)$$

$$C_i + p_j \leq C_j + M\delta_{ji} \quad \forall j, i \in N \text{ and } j < i \quad (3.10)$$

$$C_j \leq |T| \quad \forall j \in N \quad (3.11)$$

$$C_j \geq 0 \quad \forall j \in N \quad (3.12)$$

$$\ell_j \geq 0 \quad \forall j \in N \quad (3.13)$$

$$x_{jt} \in \{0, 1\} \quad \forall j \in N; \forall t \in T \quad (3.14)$$

$$\gamma_{jt} \in \{0, 1\} \quad \forall j \in N; \forall t \in T \quad (3.15)$$

$$h_{jt'} \in \{0, 1\} \quad \forall j \in N; \forall t \in T \quad (3.16)$$

$$\delta_{ji} \in \{0, 1\} \quad \forall j, i \in N \text{ and } j < i \quad (3.17)$$

The first objective (3.1), minimizes the total tardiness, and the second objective (3.2) minimizes the total energy cost. Using the time-index variables, the assignment constraint set (3.3) states that each job must start exactly once, whereas the capacity constraint (3.4) ensures that the machine can handle, at most, one job during each time interval. Constraint (3.5) provides the completion time of each job. Constraint (3.6) defines the tardiness of each job. Constraints (3.7)

and (3.8) ensure that if job  $j$  starts at  $t$ , then the machine processes job  $j$  from the start time  $t$  until the completion time of  $(t + p_j - 1)$ . This information is used to calculate the energy cost of processing job  $j$ . Constraints (3.9) and (3.10) are disjunctive constraints, which enforce that either job  $j$  is processed before job  $k$  or job  $k$  is processed before job  $j$  for any pair of jobs. Constraint (3.11) defines the boundary of completion time over the planning horizon  $T$ . Furthermore, (3.12) and (3.13) are non-negativity constraints, and (3.14) to (3.17) are integrality constraints. Note that the constraints for MMIP-MEC-TT could also be modeled in terms of the start-time variables.

MMIP-MEC-TT has  $2|N| + 3|N||T| + |N|^2$  variables and  $4|N| + |T| + 2|N||T| + \sum_{j=1}^N (p_j(|T| - p_j + 1) + p_j(\frac{p_j-1}{2}))$  constraints for a problem of size  $|T|$  and  $|N|$ , where  $|T| \geq \sum_{j=1}^n p_j$ . Obviously, if either  $|T|$  or the number of jobs increases, then the size of the problem increases significantly.

This multiobjective scheduling and pricing problem, MMIP-MEC-TT, can be solved exactly using multiobjective programming solution techniques such as the WSM,  $\epsilon$ -constraint method, goal programming, or multilevel programming, or it can be solved approximately by using metaheuristic methods such as GAs, simulating annealing, or tabu search [25].

In the next section, we will utilize the WSM to illustrate the mathematical model and gain some insight into the problem. Then we will propose several multiobjective GAs to solve larger-sized problems in a reasonable amount of time.

### 3.3 Using Weighted Sum Method to Solve MMIP-MEC-TT Problem

The WSM used to solve multiobjective optimization problems converts the multiobjective problem into an aggregated scalar single objective function (WS-MMIP-MEC-TT) by multiplying each objective function by a weighting factor and summing up all terms as

$$\min_{x \in X} \sum_{k=1}^K w_k f_k(x)$$

where  $k$  is the number of criteria functions,  $x$  is the decision vector,  $X$  is the parameter space, and  $w_k$  stands for weights. Without loss of generality, the solution for this optimization problem defines a point in the efficient set. Systematically repeating the process for various weights defines some representation of the efficient set (i.e., Pareto front). However, because different objective functions can have different orders of magnitude, the normalization of objectives is needed in order to obtain a Pareto optimal solution that is consistent with the weights selected by the decision-maker [25]. Therefore, each objective function is normalized using optimal function values in the Nadir point ( $z_k^N = \max_k(f_k(x))$ ), which is an upper bound for that objective to all solutions in the Pareto optimal set, and the Utopia point ( $z_k^U = \min_k(f_k(x))$ ), which is the lower bound of the Pareto set optimal. Nadir and Utopia points provide an interval where the optimal objective functions may vary within the Pareto optimal set [26]. Note that the maximum  $f_k^{max}$  and minimum  $f_k^{min}$  values of the  $k^{th}$  objectives must be obtained individually. The normalized values of each objective function  $f_k$  are computed as

$$f'_k = \frac{f_k - z_k^U}{z_k^N - z_k^U}$$

This transformation maps all  $f_k(x)$  to  $[0, 1]$  Hence, we can characterize the exact Pareto set by trying all possible sets of weight combinations for problems with a convex non-dominated Pareto front set (i.e., problem whose non-dominated set has a convex shape) [27]. When a limited set of weights is considered while solving WS-MMIP-MEC-TT, an approximate Pareto front is obtained.

To determine an approximate Pareto front for large-sized problems in a reasonable amount of time, a multiobjective GA, a multiobjective metaheuristic approach, is proposed.

### **3.4 Using Genetic Algorithms Based on (GA-1), (GA-2) and (GA-3) to Solve MMIP-MEC-TT Problem**

In this section, multiobjective GAs are proposed for solving the multiobjective optimization problem MMIP-MEC-TT to determine an approximate Pareto front in a reasonable amount of time. The first GA is based on dominance rank (GA-1), the second GA is based on weighted sum aggregation (GA-2) and the third GA is based on dominance rank and crowding distance comparison (GA-3). The advantages of GAs are that they determine an approximate Pareto front for a large-sized problem in a reasonable amount of time, and they allow flexibility when rescheduling is needed as the result of disruptions on the shop floor or changes in the manufacturing environment. In this implementation, at each algorithmic iteration, the GAs provide an approximate Pareto optimal solution as well as information on when to start each job, in order to calculate the scheduling and energy costs for that particular solution.

GAs are inspired from evolution theory: a population generates new offspring via crossovers and mutations, and new generations are formed via the survival-of-the-fittest principle. GAs are widely used to solve combinatorial problems and have also been implemented to solve problems with multiple objectives. Van Veldhuizen and Lamont [28] provide a detailed literature review on multiobjective GAs. Meza et al. [29] implement a multiobjective GA to the power generation expansion problem with a special fitness function to obtain a well-distributed near-optimal Pareto front.

In practical, GAs have three main stages: recombination (crossover), mutation, and selection. In the recombination stage, two chromosomes (parents) are chosen to generate new chromosomes, or offspring. Mutation results in establishing a new solution from the current

chromosome by changing the order of the two jobs. Recombination and mutation operations are used to generate new solutions (i.e., generate additional child chromosomes) within the search space based on the variation of existing ones. Selection is where the algorithm decides which solutions will carry over to the next generation. The complete cycle of these stages corresponds to one iteration. Figure 3.1 presents the general procedure to determine a Pareto front for this multiobjective optimization problem using GA-1, GA-2, and GA-3.

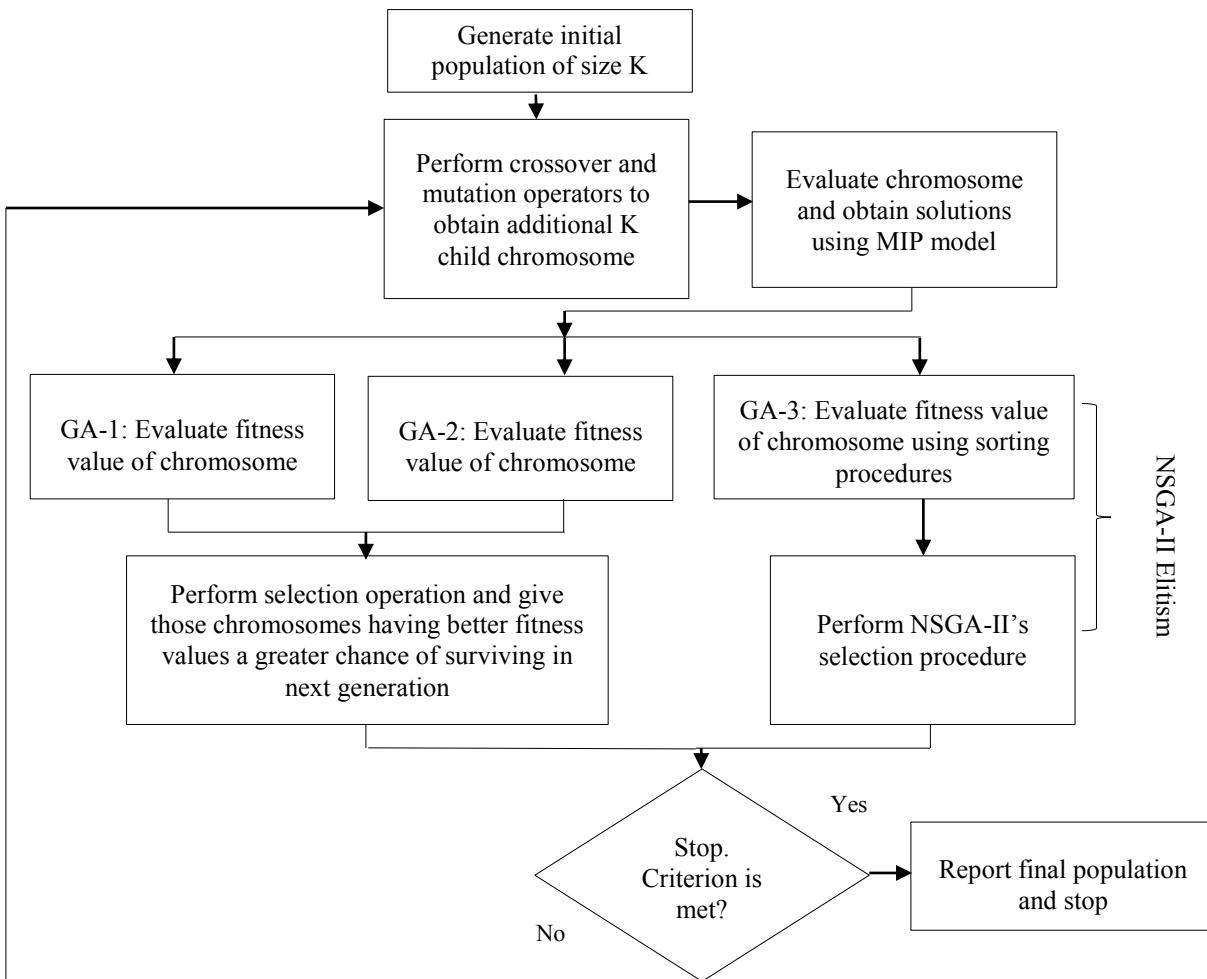


Figure 3.1: Flowchart of GA-1, GA-2, and GA-3

Note that all proposed GA algorithms are using the same recombination operators (i.e., crossover and mutation operators), while they differ in the evaluation of solution quality (i.e.,

fitness function and fitness value) and selection operator. Different components of the algorithms are discussed in more detail in the following subsections.

### 3.4.1 Representation (Coding)

In this implementation, a chromosome provides information about the order/sequence of jobs: a problem with  $n$  jobs is represented by a chromosome of  $n$  genes (i.e., an array with  $n$  cells). For example, a chromosome representing a five-job problem [3 1 4 2 5] will process all jobs in the order of 3, 1, 4, 2, and 5.

### 3.4.2 Crossover

The crossover operator selects two chromosomes as input and generates two child chromosomes (offspring) by randomly selecting a crossover point (see Figure 3.2).

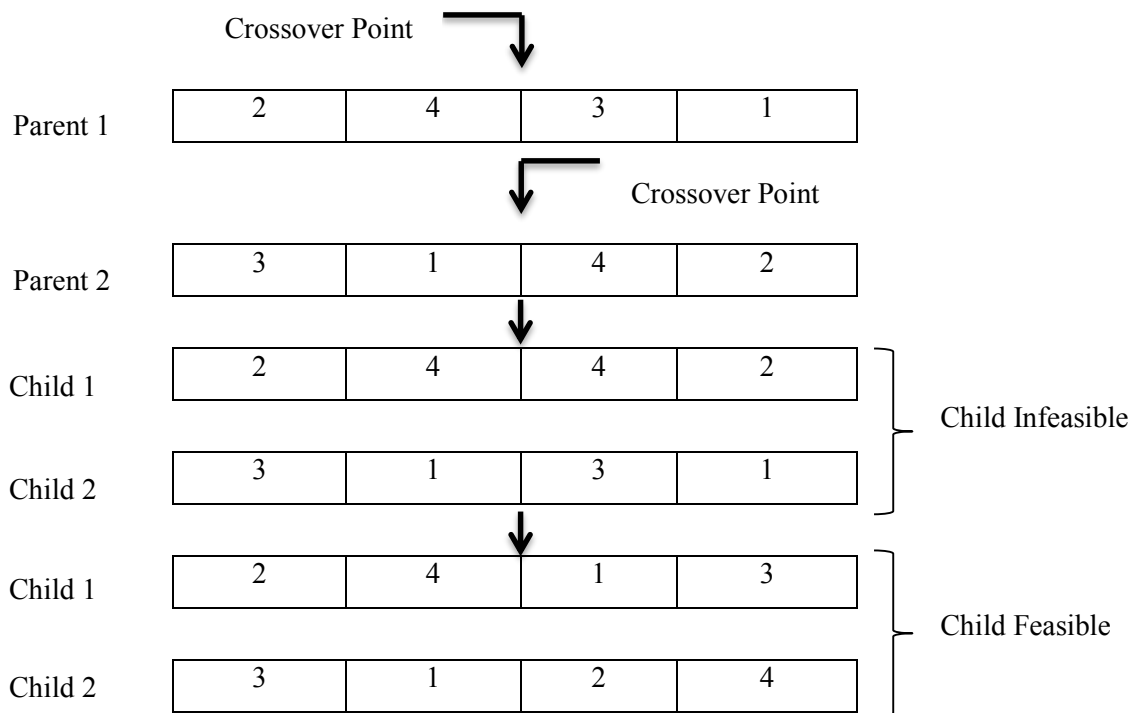


Figure 3.2: Crossover operator

As shown in Figure 3.2, the crossover operation is utilized on two chromosomes [2 4 1 3] and [3 1 4 2] by having the crossover point between the second and third genes. Each child receives

the first part of its chromosome from one of the parents and the second part of its chromosome from the other parent. Note that, for example, the second child in this example is [3 1 3 1], which is infeasible since jobs 3 and 1 are scheduled twice, whereas jobs 2 and 4 are not scheduled yet. To correct the infeasibility, the following procedure is utilized: in the child chromosome, the genes that cause infeasibility (i.e., repeated jobs) are replaced with jobs that have not appeared in the chromosome in a lexicographic order. The infeasible child [3 1 3 1] results in [3 1 2 4] when this procedure is utilized.

### 3.4.3 Mutation

The mutation operator selects a single chromosome as input and generates one new chromosome (offspring) by randomly exchanging the order of two jobs. The mutation operation is used to avoid the population of chromosomes from becoming too similar to each other, and slowing or even stopping the evolution process. Figure 3.3 illustrates the operation of the mutation operator, where the location of jobs 2 and 3 are changed.

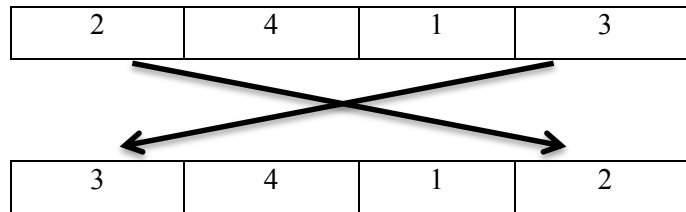


Figure 3.3: Mutation operator

### 3.4.4 Evaluation of Chromosome: Obtaining Approximate Pareto Front of Chromosome Using Reduced MMIP-MEC-TT

In a given chromosome, the order of jobs is known. It is then necessary to determine the starting time of each job in order to have information to evaluate each objective function. When the order of jobs is known, MMIP-MEC-TT reduces to the following multiobjective mixed-integer programming model: MMIP-MEC-TT-R, where R refers to reduce model. Here the disjunctive

constraints (3.9 and 3.10) are removed, and constraint set (3.26) is added to ensure that two jobs cannot be processed at the same time:

$$\text{Minimize } \sum_{j=1}^n \ell_j \quad (3.18)$$

$$\text{Minimize } \sum_{j=1}^n \sum_{t \in T} E_t h_{jt} \quad (3.19)$$

$$\sum_{t=1}^{|T|-p_j+1} x_{jt} = 1 \quad \forall j \in N \quad (3.20)$$

$$\sum_{j=1}^n \sum_{t'=\max(0,t-p_j+1)}^t x_{jt'} \leq 1 \quad \forall t \in T \quad (3.21)$$

$$C_j \geq \sum_{t=1}^{|T|-p_j+1} (t-1+p_j)x_{jt} \quad \forall j \in N \quad (3.22)$$

$$\ell_j \geq C_j - d_j \quad \forall j \in N \quad (3.23)$$

$$x_{jt} \leq M(1-\gamma_{jt}) \quad \forall j \in N; \forall t \in T \quad (3.24)$$

$$-(h_{jt'} - 1) \leq M\gamma_{jt} \quad \forall j \in N; t' \geq t \text{ and } t' \leq (t + p_j - 1) \quad (3.25)$$

$$C_i - p_i \geq C_j \quad \forall j, i \in N \text{ and } j < i \quad (3.26)$$

$$C_j \leq |T| \quad \forall j \in N \quad (3.27)$$

$$C_j \geq 0 \quad \forall j \in N \quad (3.28)$$

$$\ell_j \geq 0 \quad \forall j \in N \quad (3.29)$$

$$x_{jt} \in \{0,1\} \quad \forall j \in N; \forall t \in T \quad (3.30)$$

$$\gamma_{jt} \in \{0,1\} \quad \forall j \in N; \forall t \in T \quad (3.31)$$

$$h_{jt'} \in \{0,1\} \quad \forall j \in N; \forall t \in T \quad (3.32)$$

MMIP-MEC-TT-R has  $2|N| + 3|N||T|$  variables and  $3|N| + |T| + 2|N||T| + \sum_{j=1}^N (p_j(|T| - p_j + 1) + p_j(\frac{p_j-1}{2}))$  constraints for a problem of size  $T$  and  $|N|$ , where  $T \geq \sum_j^n p_j$ .

Solving MMIP-MEC-TT-R will provide the set of non-dominated solutions corresponding to the known chromosome (i.e., given order of jobs). When the order of jobs is fixed, the number of variables and constraints are reduced by  $|N|^2$  and  $|N|$ , respectively. Note that in this

formulation,  $C_j$ ,  $p_j$ ,  $d_j$  and  $\ell_j$  indicate completion time, process time, due date, and tardiness of job in position  $j$ , respectively, in the sequence defining the job order.

The MMIP-MEC-TT-R is solved by using the WSM, resulting in a single-objective mixed-integer program—WS-MIP-MEC-TT-R—as

$$\min_{x \in X} \sum_{k=1}^K w_k f_k(x) = w_1 \sum_{j=1}^n \ell_j + w_2 \sum_{j=1}^n \sum_{t \in T} E_t h_{jt}$$

MMIP-MEC-TT-R is a mixed-integer linear programming problem. For any combination of weights, a set of non-dominated solutions for that particular chromosome can be obtained. For any given iteration, this process generates an approximate Pareto front for each chromosome in the current population.

### 3.4.5 Evaluation of Fitness Functions

Fitness functions are used to evaluate the quality of solutions in the current population. Each individual has a fitness value, which is assigned according to its dominance. A solution determines the job order, completion time and tardiness of all jobs, and total energy cost of the completed job. Since a chromosome may characterize one or more solutions, the best measure of the fitness function over the set of solutions that a chromosome characterizes is kept to evaluate the chromosome.

Since multiobjective optimization problems try to optimize the components of a vector-valued cost function, and the solution to the problem is a family of points known as the Pareto optimal set, each proposed GA uses a unique fitness function for each solution.

#### 3.4.5.1 Fitness Function and Value for Chromosomes in GA-1

The fitness function in GA-1 evaluates the solution based on the dominance ranking procedure (i.e., number of individuals by which an individual is dominated) [30]. Let  $\Theta_{i\xi}$  be the

fitness function value for a solution  $\iota$  in generation  $\xi$ . The rank of the solution is equal to  $1 + \#_{\iota\xi}$ , where  $\#_{\iota\xi}$  is the number of solutions that dominates a solution  $\iota$  in generation  $\xi$ . If the solution is non-dominated, then its rank is 1. Note that the higher rank, the poorer the solution [8]. This can be written as

$$Fitness_{\iota}(\Theta_{\iota\xi}) = \frac{1}{rank(\Theta_{\iota\xi})} = \frac{1}{1 + \#_{\iota\xi}}$$

Note that the fitness function value for a chromosome  $\psi$ ,  $Fitness_{\psi}$ , is the sum of the fitness function value for each solution obtained from a chromosome  $\psi$  using WS-MIP-MEC-TT-R.

### 3.4.5.2 Fitness Function and Value for Chromosomes in GA-2

The fitness function in GA-2 evaluates the solution based on the concept of sharing function, which is implemented by degrading the fitness of each individual in proportion to the number of individuals located in its neighborhood (i.e., number of non-dominated solutions in the neighborhood of an individual) [31]. In the implementation, the goal is to obtain a Pareto front that is dispread and not clustered; therefore, a sharing parameter ( $\sigma_{sh}$ ) is used to determine the maximum distance from an individual for another individual to be considered part of the neighborhood. We compute a sharing function as

$$\phi(d_{\varepsilon_1\varepsilon_2}) = \begin{cases} 1 - \left(\frac{d_{\varepsilon_1\varepsilon_2}}{\sigma_{sh}}\right)^{\alpha}, & d_{\varepsilon_1\varepsilon_2} < \sigma_{sh} \\ 0, & otherwise \end{cases}$$

where  $\phi$  is the sharing value,  $d_{\varepsilon_1\varepsilon_2}$  is the normalized distance between two solutions  $\varepsilon_1$  and  $\varepsilon_2$ ,  $\sigma_{sh}$  is the extent of the sharing parameter (typically chosen between 0.01 and 0.1), and  $\alpha$  is a positive scaling factor (usually less than one). In addition,  $d_{\varepsilon_1\varepsilon_2}$  is computed as

$$d_{\varepsilon_1 \varepsilon_2} = \left[ \sum_{k=1}^2 \frac{(f_k^{\varepsilon_1} - f_k^{\varepsilon_2})^2}{(f_k^{\max} - f_k^{\min})} \right]^{\frac{1}{2}}$$

where  $f_k^{\max}$  and  $f_k^{\min}$  are the maximum and minimum values, respectively, of the  $k^{\text{th}}$  objectives in the current generation, and  $f_k^{\varepsilon_1}$  is the value of objective  $k$  of solution  $\varepsilon_1$ . Note that the  $f_k^{\max}$  and  $f_k^{\min}$  values must be obtained individually. The second fitness function is defined as

$$Fitness_2(\Theta_{l\xi}) = \frac{Fitness_1(\Theta_{l\xi})}{\sum_{\varepsilon=1}^{N_\xi} \phi(d_{l\varepsilon})}$$

where  $N_\xi$  is the number of solutions located in generation  $\xi$ . Note that the fitness function value for a chromosome  $\psi$ ,  $Fitness_\psi$ , is the sum of the fitness function value for each solution obtained from a chromosome  $\psi$  using WS-MIP-MEC-TT-R.

### 3.4.5.3 Selection Process in GA-1 and GA-2

The selection process aims to reflect nature's survival of the fittest. As mentioned previously, in a GA, a chromosome with a better fitness function will earn more chances to survive in the next generations. In this type of evaluation of a fitness function, the roulette wheel system, which selects parents for the next generation [32], is utilized. This process is shown in Figure 3.4. Note that the chromosomes with higher fitness function values have a higher probability of surviving in the next generation.

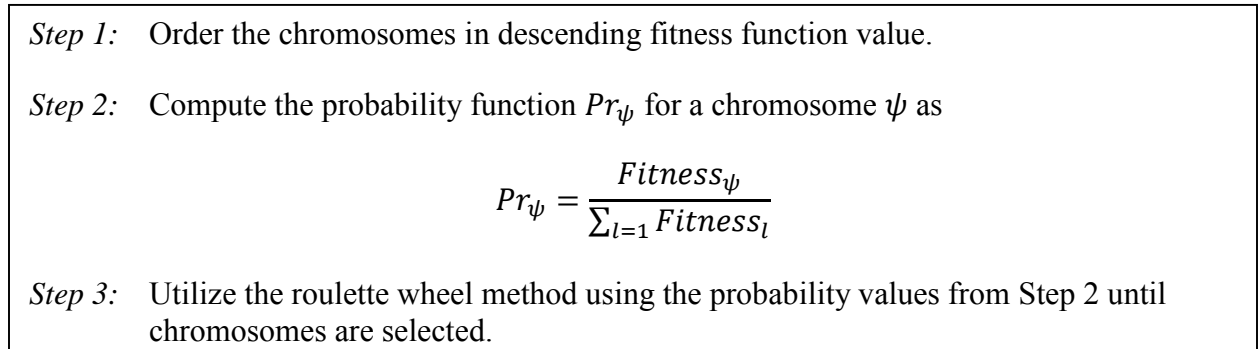


Figure 3.4: Process of chromosome selection for generating next population

#### **3.4.5.4 Fitness Function and Value for Chromosomes in GA-3**

The fitness function in GA-3 is inspired from the non-dominated sorting GA [33]. NSGA-II is used to solve multiobjective optimization problems. It has very effective sorting procedures, ensures elitism, and no sharing parameter requires to be chosen a priori. In the implementation here, the non-dominated sorting procedure and crowding distance sorting procedure are used to evaluate the quality of solutions in the current GA-3 population.

The fitness function in GA-3 uses the Pareto dominance concept of individuals to guide the search process and return the Pareto optimal set as the best result. It has two main operators: the non-dominated sorting procedure and the crowding distance sorting procedure, which evaluate the quality of individuals (solutions) in the current population.

In practice, a non-dominated sorting procedure ranks individuals in different Pareto fronts (i.e., an individual is non-dominated if none of the objective functions can be improved in value without degrading some of the other objective values). The first Pareto front (Level 1) is exactly the non-dominated set in the current population, and the second Pareto front (Level 2) is dominated by individuals in the first Pareto front only and the Pareto front builds so on (see Figure 3.5). Each individual in each Pareto front (level) are assigned fitness (rank) values based on the Pareto front to which they belong. In practice, individuals in the first Pareto front are given a fitness value of 1 and individuals in the second Pareto front are assigned a fitness value of 2, and so on. As shown in Figure 3.5, note that the level in which an individual is located represents the rank, which is the most important factor of its fitness. As a result, an individual with lower rank is preferable.

In addition to the fitness value, a new parameter called crowding distance is computed for each individual. The crowding distance measures how close an individual is to its neighbors (see Figure 3.5). In practice, an individual's crowding distance is defined as the sum of the normalized

distance between its right and left neighbors for each objective function. The first and last individuals (extreme solutions) have a crowding distance equal to infinity. In Figure 3.5, the perimeter of the cuboid is estimated by using the nearest neighbor as the vertices [33]. As a result, the crowding distance procedure guarantees diversity of the population (i.e., large average crowding distance results in better diversity in the population). In this implementation, each individual is sorted based on rank and crowding distance value.

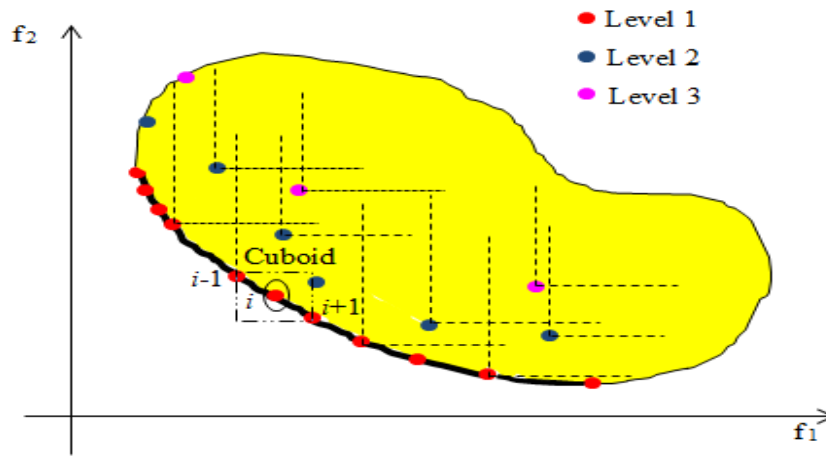


Figure 3.5: Non-dominated levels and computation of crowding distance

### 3.4.5.5 Selection Process in GA-3

In the implementation, chromosomes (parents) are selected from the population by using binary tournament selection based on two comparison rules: (1) the lower rank to which the individual belongs, the better the solution; and (2) if two individuals have the same ranking, then an individual with greater crowding distance has a better solution because the area to which that individual belongs is less crowded. This process is described in Figure 3.6. Note that the chromosomes with lower rank have better solutions, if they have the same Pareto front (i.e., rank), then the chromosome with greater crowding distance has a better solution and surviving in the next generation.

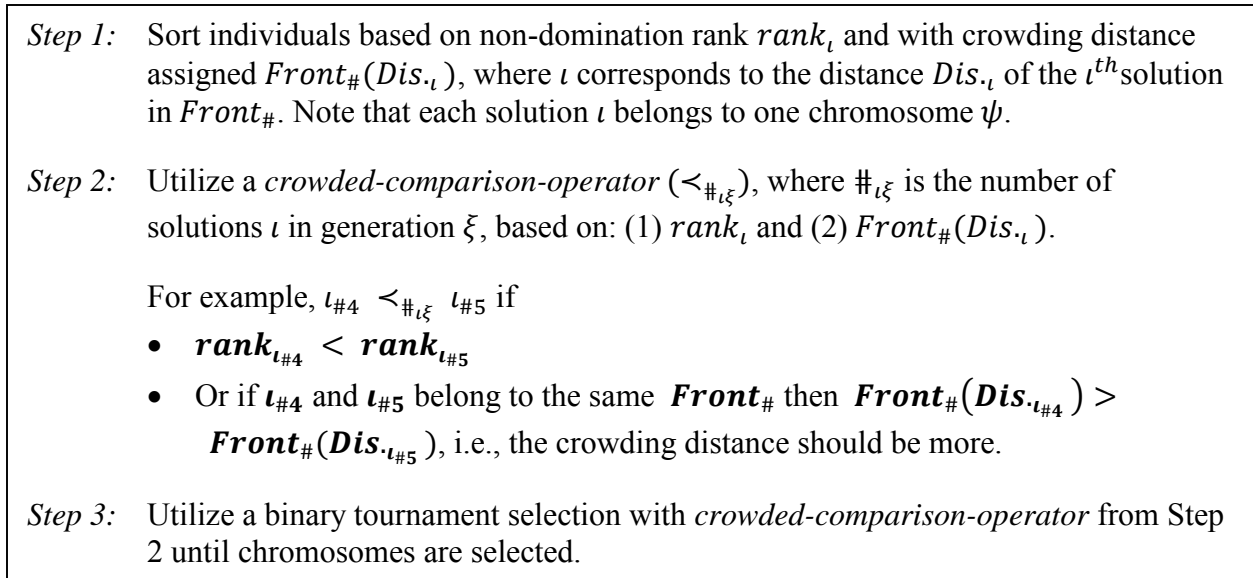


Figure 3.6: Process of selection of chromosomes for generating the next population

### 3.4.6 Stopping Criterion in GA-1, GA-2 and GA-3

When the algorithm hits the maximum number of generations, then the algorithm stops. In this stage, the approximate Pareto front is obtained. In the next section, we will present the experimental design and evaluation of performance and effectiveness of the proposed GA algorithms. Also, the comparison of the proposed GA algorithms are discussed to solve the MMIP-MEC-TT problem.

### 3.5 Experimental Design and Evaluation

The General Algebraic Modeling System (GAMS) is used to model the WS-MMIP-MEC-TT problem, and CPLEX 12.5 is used to find the exact Pareto front in order to solve the problem [34]. MATLAB R2010a language is used to evaluate the performance and effectiveness of the presented GAs, and IBM ILOG CPLEX 12.5 via MATLAB is utilized to solve the resulting subproblems. The computational experiments are performed on an Intel i5 2.27GHz machine with 4 GB of memory and a Windows 7 operating system.

### 3.5.1 Generating Experimental Data

The number of jobs  $|N|$  and size of the planning horizon  $|T|$  have a major influence on the performance of the algorithm, since the larger the size of the problem the more CPU time that is required. The problems are constructed by using varying combinations of the four parameters: number of jobs ( $n$ ), planning horizon ( $T$ ), processing times ( $p_j$ ), and due dates ( $d_j$ ).

In the experimentation, problems with 5, 10, 20, 30, 40, and 50 jobs are generated. For each job, an integer processing time is obtained randomly from a uniform distribution between 1 and 10. An integer due date ( $d_j$ ) is obtained from the uniform distribution  $\left[ P \left( 1 - TF - \frac{RDD}{2} \right), P \left( 1 - TF + \frac{RDD}{2} \right) \right]$ , where  $P$  is the total processing time, i.e.,  $P = \sum_j^n p_j$ ,  $TF$  is the average tardiness factor, and  $RDD$  is the relative range of the due date. The  $TF$  and  $RDD$  determine approximately the expected proportion of jobs that will be tardy in a random sequence of jobs. They are set at a level of 0.5, because the total tardiness objective used is likely to be harder to solve when job due dates are between 25% and 75% quartile of the total processing time/planning horizon [35]. An integer planning horizon  $|T|$  is computed as  $|T| = \sum_j^n p_j / PHF$ , where  $PHF$  is the planning horizon factor, which takes two values: 0.50 for a longer planning horizon and 0.75 for a shorter planning horizon. The electricity prices,  $E_t$ , are generated from three uniform distributions [1, 5], [6, 10], [11, 16] over three different periods of time to create low, moderate, and high variation (see Figure 3.7).

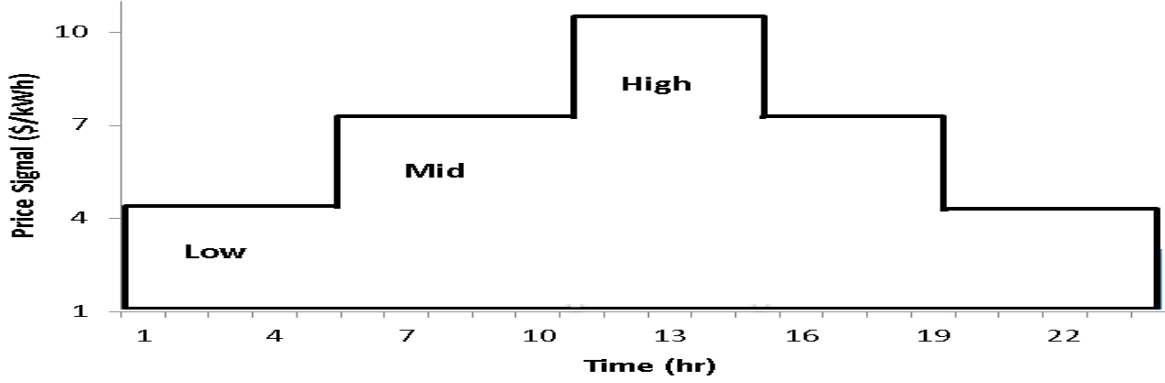


Figure 3.7: Prices signal (\$/kWh) vs. time (hr)

### 3.5.2 Solving MMIP-MEC-TT Using Weighted Sum Method

A set of ten test problems are randomly generated for testing and gaining insight into the MMIP-MEC-TT multiobjective optimization problem using the WSM. WS-MMIP-MEC-TT converts the multiobjective problem into an aggregated scalar objective function by multiplying each objective function by a weighting factor and summing up all terms as

$$\min_{x \in X} \sum_{k=1}^K w_k f_k(x) = w_1 \sum_{j=1}^n \ell_j + w_2 \sum_{j=1}^n \sum_{t \in T} E_t h_{jt}$$

to obtain a convex combination of objectives where  $w_1 + w_2 = 1$ ,  $0 \leq w_1 \leq 1$  and  $0 \leq w_2 \leq 1$  and to determine one particular optimal solution point on the Pareto front. The WSM then changes weights systemically, and each different aggregated scalar objective function optimization determines a different optimal solution to form an approximate optimal Pareto front. The WSM has also been used to solve the subproblems in the metaheuristic solution framework to obtain an approximate Pareto front in a GA.

Table 3.1 presents the results of the WSM for each problem, including the amount of execution time (CPU) and the number of non-dominated points obtained. Figure 3.8 shows the non-dominated set for some of these instances to illustrate the Pareto-optimal front for the MMIP-MEC-TT problem using the WSM (WS-MMIP-MEC-TT). Figures 3.9 and 3.10 illustrate the

average CPU time for instances of problems with different problem sizes. When the number of jobs increases, the WSM requires a significant amount of CPU time to determine the Pareto-optimal front (i.e., non-dominated solution set), since the search space for the problem increases as a function of problem size. CPU time is also a function of the number of weight combinations that are used. However, this factor may have significant impact on the quality of the Pareto front. Note that, in this implementation, the number of weight combinations is set to 10.

Table 3.1: Features of Multiobjective Optimization Instances

Instance	Number of Jobs	Planning Horizon Factor (PHF)	CPU (Sec)	Number of Non-Dominated Solutions
1	5	0.50	12.7	19
2	5	0.75	3.4	8
3	10	0.50	13,249.3	32
4	10	0.75	1,216.00	17
5	20	0.50	18,493.3	20
6	20	0.75	16,854.6	16
7	30	0.50	23,746.8	23
8	30	0.75	21,784.3	19
9	40	0.50	30,854.2	27
10	40	0.75	29,542.2	12

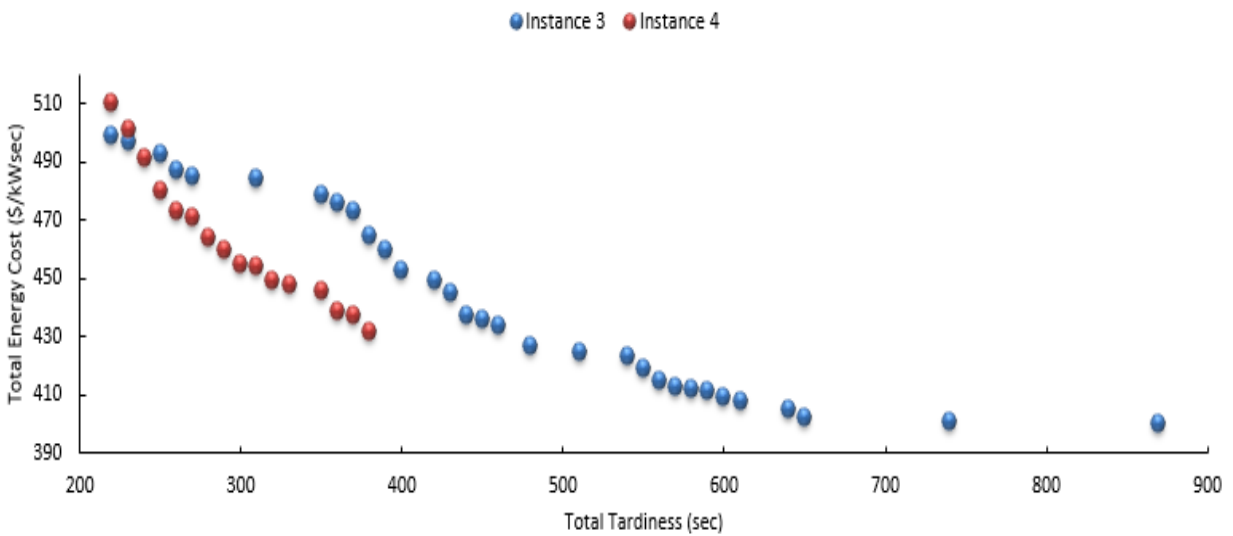


Figure 3.8: Non-dominated set of solutions for instances 3 and 4 by Using WSM

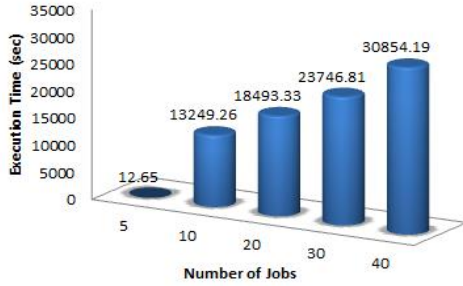


Figure 3.9: Execution time vs. number of jobs with PHF = 0.75

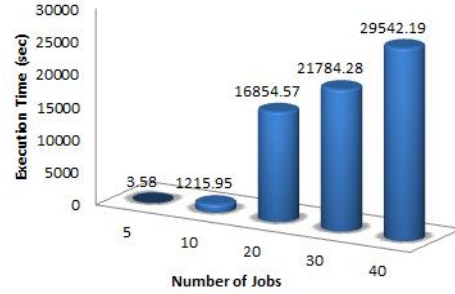


Figure 3.10: Execution time vs. number of jobs with PHF = 0.5

### 3.5.3 Genetic Algorithm Parameter Fine-Tuning

Selecting good parameters to run the GAs can play a vital role in determining a good approximate Pareto front in a timely fashion. For this goal, fine-tuning of the parameters, such as crossover rate, mutation rate, generation size, number of generations, radius  $\sigma_{sh}$  in the aggregation-based fitness function, and type of fitness function, should be performed. A set of four test problems with different number of jobs are randomly generated to test the quality of the GA solutions. In the following subsections, we will apply the GAs to examples with jobs from set  $n = \{20, 30, 40, 50\}$ . Initially, the population size  $K$  is assumed to be 15, the number of generations is 20, and the crossover and mutation rates are 80% and 20%, respectively. Each problem is run with the same parameters ten times, and the average is taken as the measure of performance. Based on these values, the goal here is to optimize the GA parameters one at a time, because considering all possible combinations of parameters may take a significant amount of time.

In order to assess and compare with different parameters, different fitness functions, and different algorithms one or more of the following performance measurement tools can be used:

- Number of Solutions on Pareto Front (No. Pareto): This tool counts the total number of non-dominated solutions that are achieved by an algorithm. The algorithm with a higher total number of non-dominated solutions is desirable.

- Diversification Metric (DiM): This metric determines the diversity of non-dominated solutions that are achieved by an algorithm [36]. Diversity can be computed by:

$$DiM = \left[ \sum_{k=1}^K (\| f_k^{max} - f_k^{min} \|)^2 \right]^{1/2}$$

where  $\| f_k^{max} - f_k^{min} \|$  is the Euclidean distance between the maximum  $f_k^{max}$  and minimum  $f_k^{min}$  values of the  $k^{th}$  objectives in the current generation  $\xi$ . Note that the maximum  $f_k^{max}$  and minimum  $f_k^{min}$  values must be obtained individually. The algorithm with a higher diversity value has a better fitness.

- Mean Ideal Distance (MID): This tool determines the nearness among the Pareto solutions and ideal point (0,0) [37], and can be computed by:

$$MID = \frac{\sum_{k=1}^K \sqrt{f_k^2 + f_k^2}}{\mathcal{D}}$$

where  $\mathcal{D}$  indicates the total number of non-dominated solutions. The algorithm with a lower value has a better fitness.

- Quality Metric (QM): In order to calculate this metric, among all non-dominated solutions achieved by the algorithms, a combination Pareto set is constructed, and the percentage of the solution belonging to each test is computed. The algorithm with a higher-quality value has a better performance.
- Size of Space Covered (SSC): This tool computes the area of objective function space covered by non-dominated vector represents a rectangle defined by points  $(\mathbf{0}, \mathbf{0})$  and  $(\mathbf{e}_1, \mathbf{e}_2)$ , where  $\mathbf{e}_1$  and  $\mathbf{e}_2$  represent a non-dominated solution. In Figure 3.11, for example, the area covered is the sum of the area of rectangles a, b, and c. The algorithm with a higher quality value has a better performance. In the result, the higher value of SSC

may result in better diversity performance (i.e., non-dominated solutions are distributed in Pareto optimal set over the global dominated set in objective space).

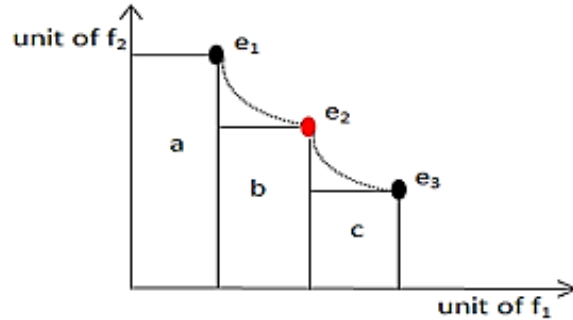


Figure 3.11: Measure of SSC performance of set of non-dominated solutions

### 3.5.3.1 Choosing Best Crossover Rate

While constructing solutions, the crossover rate is increased from 0 to 1 in increments of 0.1 in iterative runs. In each run, only one of the crossover rates is tested. Figure 3.12(a) shows that a crossover rate of 0.8 leads to good results for the problems with a different numbers of jobs.

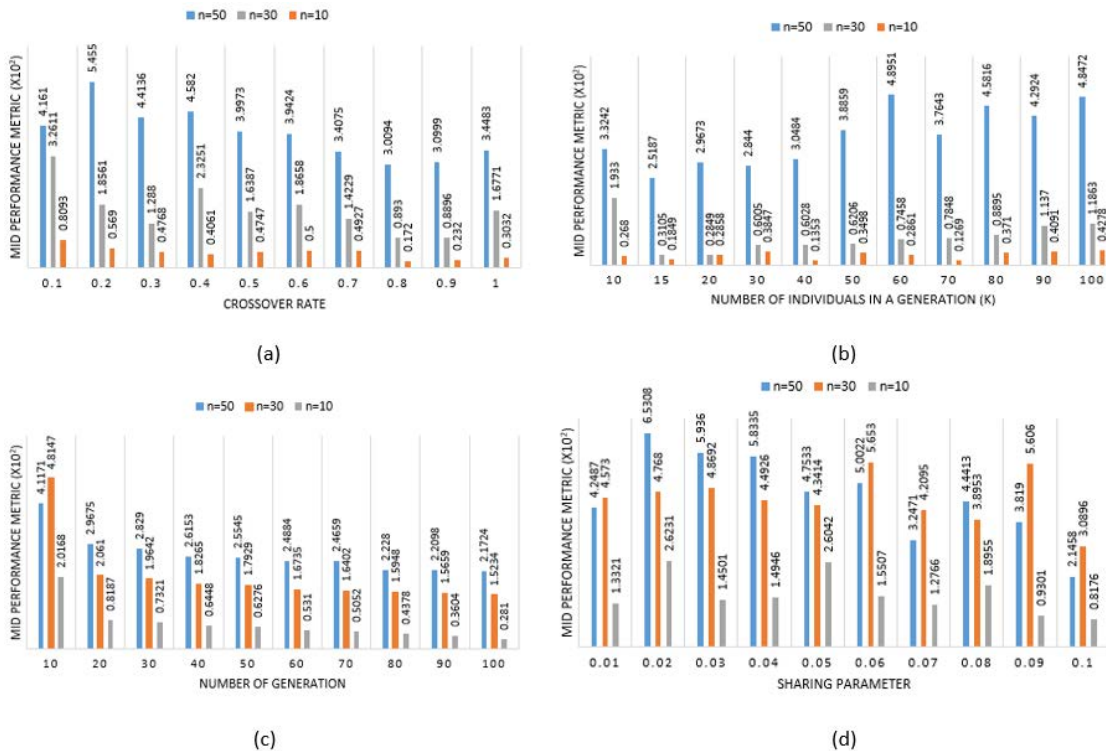


Figure 3.12: MID performance relative to number of jobs: (a) crossover rate, (b) number of individuals, (c) number of generations, and (d) sharing parameters

### 3.5.3.2 Choosing Generation Size

Using the optimal crossover rate of 0.8 while keeping all other parameters constant, the same type of experiment can be run to determine the optimal number of individuals  $K$  in a generation as a function of number of jobs to be processed. Test results show that there is a trade-off between the generation size and CPU time of an individual iteration. Therefore, a smaller generation size of 15, which yields reasonable quality solutions, is chosen. See Figure 3.12(b).

### 3.5.3.3 Choosing Optimal Number of Generations

While keeping the crossover rate and number of individuals in a generation at their optimal values, the number of generations is varied in each run (i.e., by increasing the number of generations from 10 to 100 in increments of 10). Figure 3.12(c) shows that the optimum number of generations is the maximum number of 100. A possible reason for this is the logic behind the GA, which states that the more generations, the better the solution but the longer the CPU time. Therefore, the number of generations is limited to 20.

### 3.5.3.4 Choosing Radius in Fitness Function for GA-2

The second fitness function developed with an aggregation-based weighted sum uses a sharing parameter  $\sigma_{sh}$ , which is the radius of the circle that covers the non-dominated solution in a given area (i.e., number of solutions located in its neighborhood through use of the sharing function). With appropriate choices of the sharing parameter  $\sigma_{sh}$  and generation size, sharing directs the genetic search toward locating optimal solutions corresponding to most of the weighting combinations. By varying the sharing parameter  $\sigma_{sh}$  between 0.01 and 0.1 in increments of 0.01, while keeping the other parameters at their fine-tuning values determined by other experiments, the results shown in Figure 3.12(d) indicate that there seems to be an optimum sharing parameter  $\sigma_{sh}$  around 0.1, according to the performance measurement considered.

### 3.6 Comparison of Proposed Genetic Algorithms (GA-1, GA-2, and GA-3)

With the fine-tuning of parameters, the proposed GAs are compared while varying the number of jobs. Table 3.2 shows that GA-3 performs better than either GA-1 or GA-2, according to all performance measures.

Table 3.2: Evaluation of Non-Dominated Solutions for GAs

Number of Jobs	Rank-Based GA-1					Aggregation-Based GA-2					Domination-Based GA-3				
	No. Pareto	DiM	MID	QM	CPU	No. Pareto	DiM	MID	QM	CPU	No. Pareto	DiM	MID	QM	CPU
20	16	359.7	5879.3	0.216	608.36	20	427.7	5781.8	0.382	583.74	23	635.3	5642.9	0.402	429.25
30	18	632.4	6813.5	0.184	715.73	19	781.9	6794.8	0.328	631.81	27	937.5	6625.2	0.489	547.36
40	14	715.6	4391.0	0.118	1152.30	17	807.5	4228.3	0.206	973.77	23	2489.7	4131.5	0.676	882.94
50	17	758.4	5931.6	0.106	1925.80	23	1083.5	5853.9	0.218	1736.25	29	1862.0	5804.4	0.677	1386.27

In Figure 3.13, the effectiveness between the results are statistically analyzed with respect to the four performance measures and CPU time by conducting Tukey's 95% confidence interval. As can be seen, there are statistically significant differences between the proposed GA algorithms in DiM performance according to no overlapping in the confidence intervals. As a result, the GA-3 has a higher diversity. Also, Figure 3.13 shows that there are no statistically significant differences between the proposed GA algorithms in the QM and MID performances, but GA-3 has a better performance in the number of Pareto solutions compared to GA-1 and GA-2. A possible reason for why the GA-3 performs the best is that it is using the non-dominated sorting procedure and crowding distance sorting procedure, which allow it to maintain enough non-dominated solutions in the final GA population. Figure 3.13 also shows that the difference is not significant according to overlapping in the confidence intervals in CPU performance, but GA-3 shows a slightly better performance compared with other algorithms.

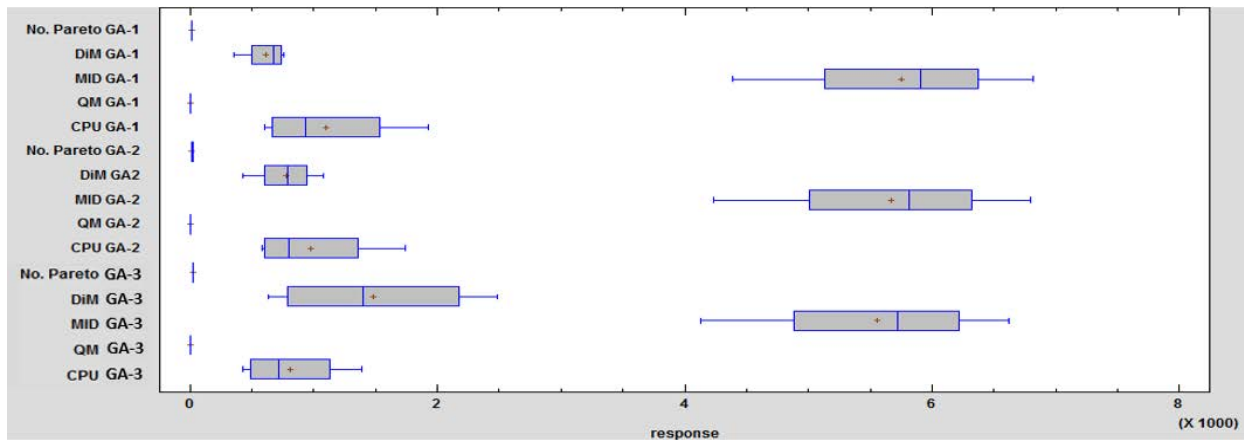


Figure 3.13: Box-and-whisker plot of GAs performance measures with 95% median notch

Figure 3.14 shows the comparison of Pareto fronts for a 40-job instance. It is clear that GA-3 solutions dominate the other algorithms. Therefore, we can conclude that GA-3 performs the best in solving MMIP-MEC-TT. The above results show the ability of GA-3 in finding diverse solutions in the front and converging to the true Pareto front while having a better spread and more non-dominated solutions in the Pareto-optimal region than the other algorithms. GA-3 is at least as good as GA-1 and GA-2, according to most performance indices, or better in the DiM metric.

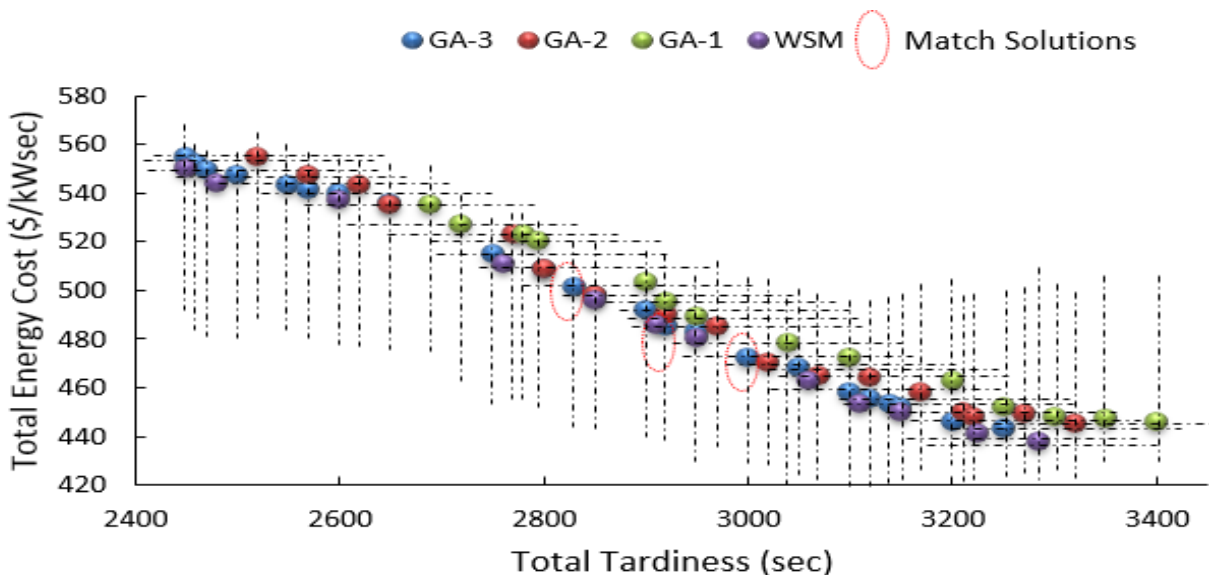


Figure 3.14: Non-dominated solutions in job 40 from GAs in Table 3.2 and WSM in Table 3.3

### 3.7 Comparison of GA-3 and Weighted Sum Method

The GA-3 algorithm is implemented to obtain an approximate solution at a reasonable amount of computational time without any guarantee of optimality. A remarkable feature of GA-3 is that it may not return the same results twice, since it is based on randomized iterations that use different random chromosomes. Taking these features into consideration, the best way to evaluate the quality of a GA-3 is by comparing its Pareto front with the Pareto-optimal front obtained by the WSM.

In order to compare both methods, GA-3 is applied using the best optimized parameters on the same instances used to characterize the performance of the weighted sum method. Since GA-3 may not return the same results, the algorithm is run twice for each instance, and the average is taken as the measure of performance. The computational time required to obtain the Pareto fronts is also recorded. Table 3.3 summarizes the results obtained with GA-3 and the WSM. Table 3.3 shows that GA-3 is able to obtain more non-dominated solutions than the WSM. Also, it is clear that GA-3 is able to obtain non-dominated solutions in about 2.6% of the WSM's CPU time. For example, for instances 2 and 4, it can be seen that the difference between their execution times is more than three hours with the WSM, while this is about seven minutes with GA-3.

Table 3.3: Results of GA-3 and WSM

Instance	Number of Jobs	PHF	GA-3			WSM		
			CPU (sec)	No. Pareto	QM	CPU (sec)	No. Pareto	QM
1	10	0.5	398.76	38	0.179	13,249.3	32	0.821
2	20	0.75	429.24	23	0.273	16,854.6	16	0.727
3	30	0.5	547.36	27	0.178	23,746.8	23	0.851
4	40	0.75	882.94	23	0.250	29,542.2	12	0.750

In Table 3.3, the percentage of solutions for GA-3 and WSM indicates what percentage of Pareto front solutions are from each method, if the Pareto fronts from both methods are combined. Figure 3.14 shows that GA-3 solutions have an identical local Pareto-optimal from the exact solutions found in the WSM (see match solutions on the figure), as well as identical solutions from other algorithms. However, GA-3 has some problems in finding the whole global Pareto-optimal solutions. A possible reason for this is that the MMIP-MEC-TT model is formulated as a mixed-integer programming, a matter that is also observed in the work of Deb et al. [33]. Another reason could be the number of generations selected (i.e., the more generations, the better the solution but the longer the CPU time). It can be concluded that GA-3 is able to find a good Pareto approximation at the WSM's Pareto-optimal front. For example, in instance 4, 25% of the final Pareto front comes from GA-3. Although the numbers of non-dominated solutions are similar, on average, 25% of the combined Pareto-optimal solutions are from GA-3. In conclusion, although GA-3 cannot find the optimal Pareto front, it obtains an approximate Pareto front in minimal computational CPU time (see Figure 3.14).

### **3.8 Comparison of Performance of GA-3 Algorithm with Random Number of Generations**

This section compares the performance of the GA-3 algorithm with a random number of generations using a set of 30 jobs with  $PHF = 0.5$ . In order to clearly explain how GA-3's Pareto front can be characterized with respect to DiM, MID, CPU, SSC, QM, and the number of solutions on Pareto front performance measurement metrics, the GA-3 is run for a long period of time, and the performance metrics are compared with the WSM.

In this implementation, the number of generations (iterations) is set to 1,000 in order to sufficiently observe GA-3 dynamics. Figure 3.15 illustrates the comparison between GA-3 and WSM solutions versus the different number of generations in the GA-3 algorithm. Figure 3.15

show that when GA-3 is run more often, it generally performs better. However, as the number of generations is increased, its solution quality becomes better.

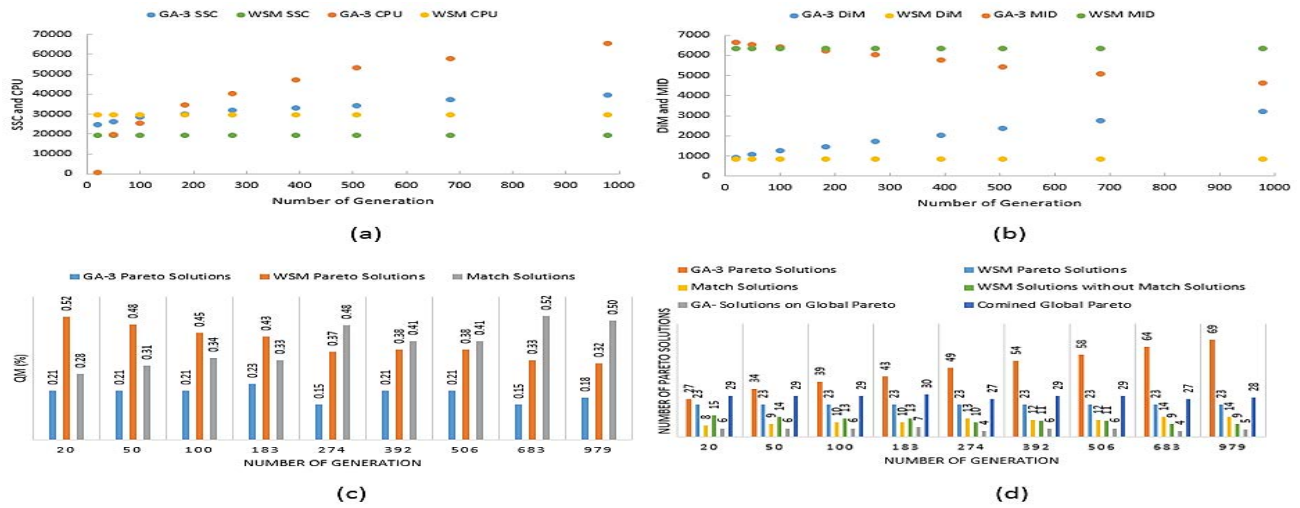


Figure 3.15: Performance metrics for GA-3 and WSM vs number of generations in GA-3

Figure 3.15(a) illustrates the comparison between the amount of CPU time and the SSC metric: GA-3 is able to report a better SSC metric than the WSM in the expense of higher CPU time. These results can be intuitively anticipated because the logic behind the GA is that the greater the number of generations, the longer the CPU time and the better the solution quality with respect to the SSC metric.

Figure 3.15(b) shows the DiM and MID metrics. DiM metric shows that GA-3 obtains a diversity of solutions with a high number of generations, whereas MID metric shows that GA-3 ensures the convergence of the neighbor search to a local optimum. In a comparison of the GA-3 with the WSM, Figure 3.15(b) also shows that GA-3 is able to perform slightly better in the DiM metric than in the WSM at 20 and 50 number of generations. However, when the number of generations grows, GA-3 does better than the WSM with respect to the DiM metric. For the MID metric, the WSM is able to obtain better solutions than GA-3 at 20, 50, and 100 generations, but when the number of generations increases more than 100, GA-3 performs better.

Figure 3.15(c) represents the evaluation of the percentage of solution from GA-3 and WSM in the final combined Pareto front. Results show that most of the solutions in the final Pareto front come from the WSM, although the percentage of solutions from GA-3 slightly increases with an increasing number of generations. However, the percentage of solutions from WSM remains greater than the percentage of solutions from GA-3 in the final combined Pareto-optimal front. For example, when the number of generations is 100, 21% of the final Pareto front comes from the GA-3, whereas 45% of the final Pareto comes from the WSM. Although the number of non-dominated solutions are similar (i.e., GA-3 has a similar local Pareto-optimal solutions from the global solutions found in the WSM), on average, 34% of the combined Pareto-optimal solutions are from the GA-3 and the WSM.

Figure 3.15(d) represents the evaluation of the number of solutions on the Pareto front from GA-3 and WSM. It can be clearly observed that when the number of generations increases, the GA-3 is able to obtain more approximate solutions on the Pareto front. Also, one interesting aspect of these results is that the combined Pareto-optimal solutions have an identical local-optimal solutions found in the GA-3 and WSM solutions.

It should be noted that the goal of GA-3 is to determine an approximate Pareto front for a large-sized problems in a reasonable amount of time in order to schedule and reschedule jobs when there are disruptions on the shop floor or changes in the manufacturing environment. As a result, the number of generations can be limited to 20 generations, since it leads to good results and finds an approximate Pareto front in a timely fashion.

### **3.9 Using Sub-Optimal Solutions in Chromosome Evaluation**

The GA-3 is solving a mixed-integer mathematical subproblem, MMIP-MEC-TT-R, using an IBM ILOG CPLEX 12.5 solver via MATLAB to find solutions on the Pareto front to evaluate

each chromosome in a population. The CPLEX solver uses a branch-and-cut algorithm to solve mixed-integer problems that may require significant amount of CPU time and physical memory to find the optimal solution. When the time it takes to determine the optimal solution is analyzed, the CPLEX optimizer may find a very good quality solution (e.g., a solution less than 0.5% away from the optimal solution) in 5% of the time it finds and verifies the optimal solution.

This section experiments with the impact of having sub-optimal solutions for the MMIP-MEC-TT-R on overall quality of the approximate Pareto front obtained by the GAs. In order to determine sub-optimal solutions, the time that the CPLEX's MIP optimizer takes in the branch-and-cut algorithm is limited: the CPLEX solver can stop if the number of branch-and-cut iterations has reached a limit or the CPU time reaches a limit. Note that the time limit is not cumulative but applies to each iteration of the GA. For example, if a time limit of 15 seconds is set and MIP optimizations are called twice, then there could be a total of, at most, 30 seconds of running time if each call consumes its maximum time limit. In this implementation, the maximum time is set at 10 seconds for each iteration (i.e., MIP optimizer takes 10 seconds to process one iteration). Note that the time limit for MIP optimization does not apply to the sum of all steps in GA-3, such as crossover and mutation operators, fitness function, selection, and internal calls to the MIP optimizer.

In order to compare the performance of GA-3 with the MIP optimizer time limit (GA-3-MOTL), the GA-3-MOTL is applied on the same instances used to characterize the performance of the GA-3 (which solves up to optimality and is without time limit), and the performance metrics are compared with the WSM as well. In this implementation, the number of generations is set to 20, 50, and 100, in order to obtain insight into the GA-3-MOTL dynamics. Figure 3.16 illustrates a performance comparison of GA-3-MOTL, GA-3, and WSM. Figure 3.16(a) shows that when the

number of generations increases, the GA-3 requires dramatically more computation time than the GA-3-MOTL for a similar problem. Also, GA-3-MOTL is able to report better performance in CPU time than WSM.

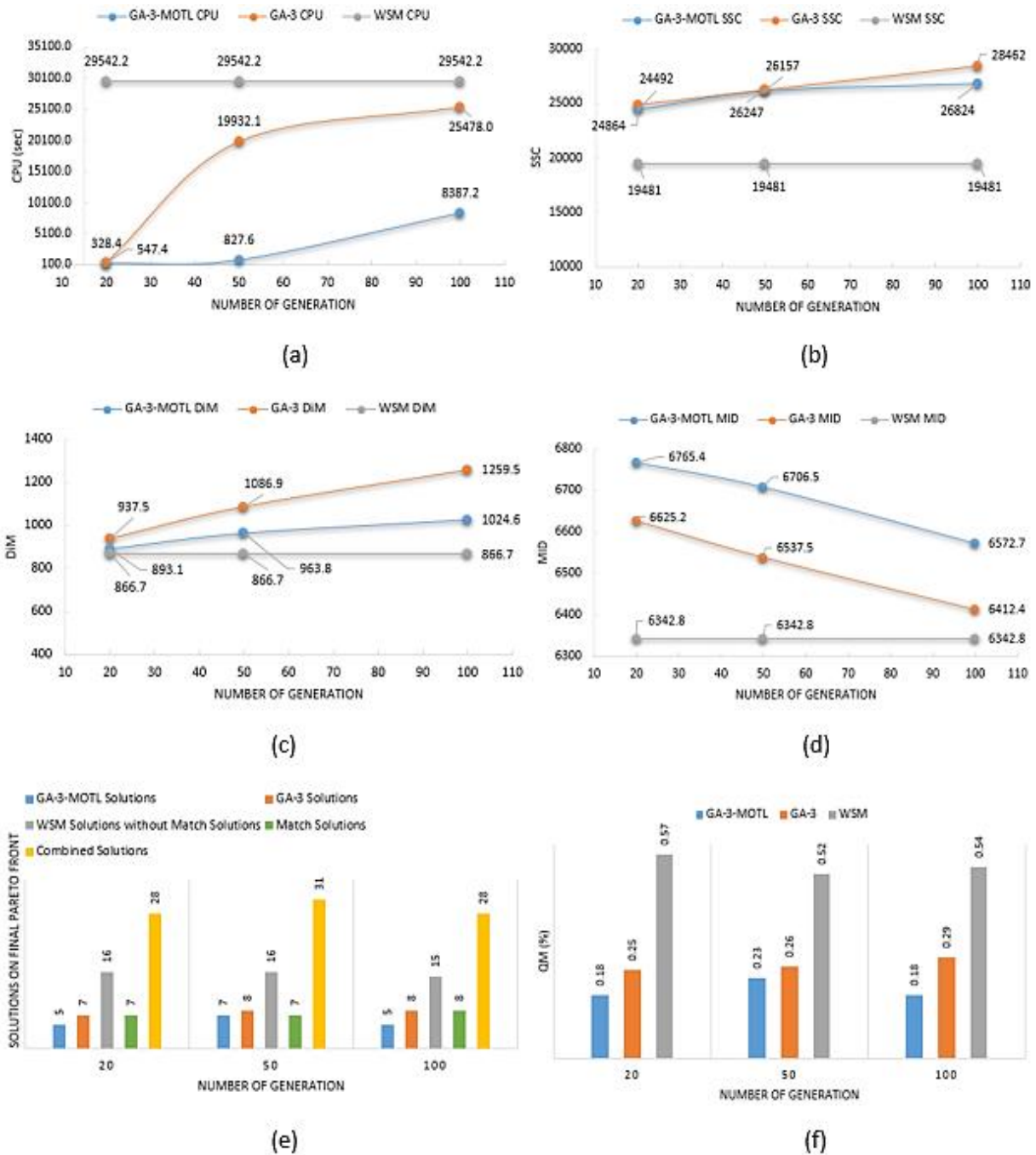


Figure 3.16: Performance metrics for GA-3-MOTL, GA-3, and WSM vs number of generations

Figure 3.16(b) shows a comparison of the SSC metric at 20 and 50 number of generations. GA-3-MOTL is able to come closer to GA-3, but when the number of generations increases, the GA-3 is able to report a better SSC metric than GA-3-MTOL and WSM. Figures 3.16(c) and (d) show the DiM and MID metrics, respectively. GA-3 performs a higher diversity of solutions than GA-3-MOTL, whereas the MID metric shows that GA-3 performs a higher MID metric than GA-3-MOTL (i.e., GA-3 always ensures convergence of the neighbor search more than GA-3-MTOL).

In the comparison of GA-3-MOTL with WSM, Figure 3.16(c) illustrates that GA-3-MOTL is able to perform better in the DiM metric than WSM, whereas Figure 3.16(d) shows that WSM is able to perform better in the MID metric than GA-3-MOTL.

Figure 3.16 (e) shows the evaluation of the number of solutions on the Pareto-optimal front from GA-3-MOTL, GA-3 and WSM: GA-3 slightly obtains more solutions than GA-3-MTOL. On the other hand, WSM always obtains more solutions in the global Pareto-optimal front than GA-3 and GA-3-MTOL. Figure 3.16 (f) indicates the percentage of solution from GA-3-MOTL, GA-3, and WSM in the global Pareto-optimal front and shows that GA-3 obtains a little bit higher percentage than GA-3-MOTL whereas WSM always performs better in QM than GA-3 and GA-3-MOTL. For example, in number of generations 50, 26% of the solutions on the global Pareto-optimal front comes from GA-3 whereas 23% and 52% comes from GA-3-MTOL and WSM respectively. Finally, we can conclude that generally GA-3-MTOL is able to solve more number of generation in small amount of time comparing with GA-3 and WSM. Also, in the results, GA-3-MOTL is able to come closer to the performance of the GA-3 in DiM, QM, and number of solutions on the global Pareto-optimal front metrics.

### 3.10 Selecting Solution from Pareto Front for Implementation: A Case Study

In this section, the GA-3 algorithm is run on a case study using the optimized parameters. Here, 40 jobs are available simultaneously at time zero and are sequenced on a non-preemptive single machine. Figure 3.17 shows the obtained Pareto fronts by the GA-3. Solutions on the Pareto fronts are well distributed, and there is no cluster of solutions. Each solution represents the sequence of jobs with total tardiness and total energy cost. This wide range of solutions allows the decision-maker to select the best solution. So the question that needs to be addressed is which solution should be selected. The TOPSIS method is used to select the best solution among all non-dominated solutions.

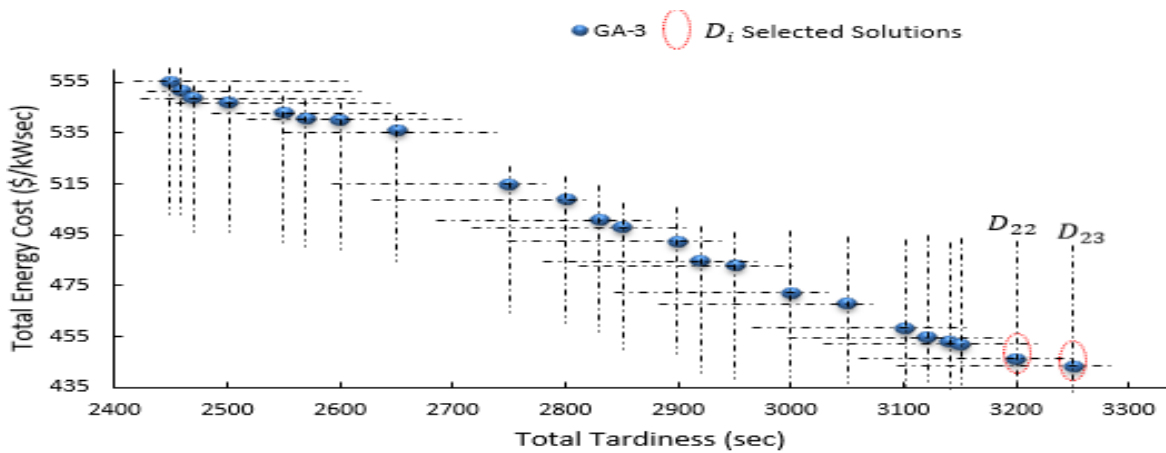


Figure 3.17: Pareto front and selected solution by TOPSIS

#### 3.10.1 Selection of Alternative Using TOPSIS Method

The TOPSIS method operates on the assumption that the preferred solution (alternative) should simultaneously be closest to the ideal solution and farthest from the non-ideal solution. This method uses an index, which combines the closeness of an alternative to the ideal solution with its remoteness from the non-ideal solution. The alternative that maximizes this index value is the preferred alternative. The decision-maker must structure the problem into criteria ( $\varphi$ ) and alternatives ( $i$ ). The decision-maker must also provide the weight of each criteria using one of the

multicriteria decision-making methods. Here, the weight must be determined using weights from the ranking method, as shown in the following equation [38]:

$$\lambda_{\varphi} = \frac{k - r_{\varphi} + 1}{\sum_{\varphi=1}^{\varphi=k} (k - r_{\varphi} + 1)}$$

where  $r_{\varphi}$  represents the rank of the criterion ( $\varphi$ ), and  $k$  is the number of criteria.

In the case here, the criteria taken into the consideration are ranked as follows: (1) total energy cost, (2) total number of tardy jobs, and (3) total completion time. The measure decision (payoff) matrix ( $\theta_{i\varphi}$ ) in criterion ( $\varphi$ ) for alternative ( $i$ ) is presented in Table 3.4.

Table 3.4: Decision (Pay-Off) Matrix ( $\theta_{i\varphi}$ )

Alternative ( $i$ )	Criteria ( $\varphi$ )		
	Total Completion Time	Total Tardy Jobs	Total Energy Cost
1	6697	9	555
2	6703	9	552
3	6717	12	549
4	6724	14	547
5	6731	15	543
6	6737	15	541
7	6741	17	540
8	6748	19	536
9	6753	20	515
10	6762	20	509
11	6768	22	501
12	6773	22	498
13	6781	22	492
14	6787	24	485
15	6792	24	183
16	6804	27	472
17	6817	29	468
18	6824	31	458
19	6832	31	455
20	6837	31	453
21	6844	32	452
22	6849	33	446
23	6857	34	443
<b>Rank</b>	3	2	1

The following is the weight vector ( $\lambda$ ):

$$\lambda^T = [0.5, 0.33, 0.17]$$

The next step is to normalize the entire decision matrix as shown in the following equation:

$$R_{i\varphi} = \frac{\theta_{i\varphi}}{\sqrt{(\sum_i \theta_{i\varphi}^2)}}$$

Then, the weighted decision matrix is computed by using the following equation

$$Q_{i\varphi} = \lambda_{\varphi} R_{i\varphi}$$

The next step is to identify the ideal and non-ideal solutions ( $I$  and  $N$ ) by using the weighted matrix as

$$I = (\max Q_{i\varphi}, \forall i; \varphi)$$

$$N = (\min Q_{i\varphi}, \forall i; \varphi)$$

Based on these solutions, separation measures for each solution are calculated as

$$P_i^I = \left[ \sum_{\varphi} (Q_{i\varphi} - I_{\varphi})^2 \right]^{1/2}, i = 1, \dots, i$$

$$P_i^N = \left[ \sum_{\varphi} (Q_{i\varphi} - N_{\varphi})^2 \right]^{1/2}, i = 1, \dots, i$$

where  $P_i^I$  is the distance of the alternative solution from the ideal solution, and  $P_i^N$  is the distance of the same solution from the non-ideal solution. TOPSIS categorizes the preferred solution by minimizing the similarity index ( $D_i$ ), defined below (note that all solutions are ranked by their index values; a solution with a higher index value is preferred over another index's values):

$$D_i = \frac{P_i^N}{(P_i^I + P_i^N)}, i = 1, \dots, i$$

Table 3.5 shows the obtained similarity index  $D_i$  values, where the decision-maker is able to rank them in decreasing order, and the preferred solution is the one with the maximum value of  $D_i$ .

Table 3.5: Similarity Index ( $D_i$ ) Values

$D_i$	Value	Rank
$D_1$	0.24	17
$D_2$	0.23	18
$D_3$	0.27	16
$D_4$	0.31	15
$D_5$	0.33	13
$D_6$	0.32	14
$D_7$	0.38	12
$D_8$	0.44	11
$D_9$	0.46	9
$D_{10}$	0.45	10
$D_{11}$	0.52	7
$D_{12}$	0.52	7
$D_{13}$	0.51	8
$D_{14}$	0.58	6
$D_{15}$	0.58	6
$D_{16}$	0.67	5
$D_{17}$	0.72	4
$D_{18}$	0.75	2
$D_{19}$	0.74	3
$D_{20}$	0.74	3
$D_{21}$	0.75	2
$D_{22}$	0.76	1
$D_{23}$	0.76	1

Using the results from Table 3.5, the decision-maker can select the best solution among all non-dominated solutions based on his/her preference. According to the decision-maker's preference and TOPSIS, the best solution is a number  $D_{22}$  or  $D_{23}$  (see Table 3.5 and highlighted solution in Figure 3.17).

### 3.11 Conclusion

The growing attention toward energy-aware practices in sustainable practices and the need to decrease the energy cost in industry provided the impetus for this study and ultimately to providing a tool for manufacturing professionals to use to reduce energy costs via production

scheduling. Here, a multiobjective mixed-integer mathematical model is proposed to minimize the total tardiness and total energy cost on a single machine setting. The WSM was used to illustrate the mathematical model and gain insight into the multiobjective problem. Due to the computational complexity involved in solving the mathematical model (i.e., problem is NP-hard), an efficient multiobjective GA to solve larger-sized problems in a reasonable amount of CPU time was proposed in order to obtain an approximate set of non-dominated solutions. The proposed multiobjective GA-3 outperformed GA-1 and GA-2 with respect to the quality of the approximate Pareto front. Also, GA-3 was able to find a good Pareto approximation at the WSM's Pareto-optimal front while maintaining a diverse set of solutions on the instances tested, and the algorithm also outperformed the WSM with respect to computational CPU time for obtaining the approximate Pareto front.

Given a chromosome, one has information on the order of jobs. When the order of jobs is known, the original model of MMIP-MEC-TT was reduced to the multiobjective mixed-integer mathematical model MMIP-MEC-TT-R, in which the number of variables and constraints were reduced by  $|N|^2$  and  $|N|$ , respectively. This reduction accelerated the heuristic solutions.

Extensive computational experimentation was undertaken to test the performance and effectiveness of the developed GAs by varying several parameter values. It was observed that selecting good parameters can play a vital role in the CPU's running time. Furthermore, results for GA-3 performed the best in solving the MMIP-MEC-TT-R compared to GA-1 and GA-2. Also, results showed the abilities of GA-3 to find diverse solutions in the front and to converge to the true Pareto front with a better spread and more non-dominated solutions in the Pareto-optimal region, comparison to other algorithms. After fine-tuning the GA-3 (i.e., for the optimal rate of crossover, size of generation, and number of generations), the algorithm was illustrated on a case

study. A TOPSIS method was used to assist the decision-maker in choosing the most efficient schedule with an appropriate energy cost level.

A direct extension of this research would be to solve the MMIP-MEC-TT using other exact solution techniques that can generate supported and non-supported Pareto-optimal solutions such as the  $\epsilon$ -constraint method, since the WSM can only generate the supported Pareto-optimal solutions (i.e., solutions that lie on the convex envelop of the Pareto frontier). Another research direction would be to investigate other problems with conflicting objectives on various operating environments (i.e., with different scheduling objectives, having sequence-dependent setup times, or on multi-machine settings). For example, the formulation may change significantly when other scheduling objectives are considered on a single machine setting, but the proposed solution approach will be quite similar, except for the mixed-integer linear programming, which is used to generate the approximate Pareto front and may change in the GA based on the type of scheduling objective that is employed. A more detailed experiment to determine if there is a clear pattern between parameters of the experimental design and the objective could be designed.

### 3.12 References

- [1] M.A. Salido, J. Escamilla, F. Barber, A. Giret, D. Tang, and M. Dai. "Energy-aware parameters in job-shop scheduling problems," in *GREEN-COPLAS 2013: IJCAI 2013 Workshop on Constraint Reasoning, Planning and Scheduling Problems for a Sustainable Future*, vol. 1, May 2013, pp. 44-53.
- [2] K. Fang, N. Uhan, F. Zhao, and J.W. Sutherland, "A new shop scheduling approach in support of sustainable manufacturing," in *Glocalized Solutions for Sustainability in Manufacturing*, Springer, March 2011, pp. 305-310.
- [3] G. Mouzon, M.B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *International Journal of Production Research*, vol. 45, no. 18-19, July 2007, pp. 4247-4271.
- [4] G. Mouzon and M.B. Yildirim, "A framework to minimise total energy consumption and total tardiness on a single machine," *International Journal of Sustainable Engineering*, vol. 1, no. 2, June 2008, pp. 105-116.

- [5] U.S. Energy Information Agency, "Annual Energy Outlook 2013," [Online: Annual Report], Nov. 2013; [cited: May 2013], available from World Wide Web: <http://www.eia.gov/forecasts/aeo/>.
- [6] Y. Liu, H. Dong, N. Lohse, S. Petrovic, and N. Gindy, "An investigation into minimising total energy consumption and total weighted tardiness in job shops," *Journal of Cleaner Production*, vol. 65, Feb. 2014, pp. 87-96.
- [7] R. Drake, M.B. Yildirim, J.M. Twomey, L.E. Whitman, J.S. Ahmad, and P. Lodhia. "Data collection framework on energy consumption in manufacturing," in Wichita State University Libraries SOAR: Shocker Open Access Repository, Wichita State University, May 2006.
- [8] M.B. Yildirim and G. Mouzon, "Single machine sustainable production planning to minimize total energy consumption and total completion time using a multiple objective genetic algorithm," *IEEE Transactions on Engineering Management*, vol. 59, no. 4, July 2012, pp. 585-597.
- [9] N. Weinert, D. Rohrmus, and S. Dudeck, "Energy-aware production planning based on energyblocks in a Siemens AG generator plant," in *Sustainable Manufacturing*, Springer Berlin Heidelberg, July 2012, pp. 211-216.
- [10] J. Du and J.Y.T. Leung, "Minimizing total tardiness on one machine is NP-hard," *Mathematics of Operations Research*, vol. 15, no. 3, Feb. 1990, pp. 483-495.
- [11] C.N. Potts and L.N. Van, "A branch and bound algorithm for the total weighted tardiness problem," *Operations Research*, vol. 33, no. 2, Dec. 1985, pp. 363-377.
- [12] C.N. Potts and L. Van, "Dynamic programming and decomposition approaches for the single machine total tardiness problem," *European Journal of Operational Research*, vol. 32, no. 3, April 1987, pp. 405-414.
- [13] M. Dai, D. Tang, A. Giret, M.A. Salido, and W. Li, "Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 5, Oct 2013, pp. 418-429.
- [14] F F. Shrouf, J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *Journal of Cleaner Production*, vol. 67, no. 15, March. 2014, pp. 197-207.
- [15] H. Zhang, F. Zhao, K. Fang, and J.W. Sutherland, "Energy-conscious flow shop scheduling under time-of-use electricity tariffs," *CIRP Annals-Manufacturing Technology*, vol. 63, no. 1, March 2014, pp. 37-40.
- [16] H. Zhang, F. Zhao, and J.W. Sutherland, "Energy-efficient scheduling of multiple manufacturing factories under real-time electricity pricing," *CIRP Annals-Manufacturing Technology*, vol. 64, no.1, July 2015, pp. 41-44.

- [17] K. Fang, N.A. Uhan, F. Zhao, and J.W. Sutherland. "Scheduling on a single machine under time-of-use electricity tariffs," May 2014, [cited: Feb. 2015]; available from World Wide Web: [http://www.optimization-online.org/DB\\_FILE/2014/04/4313.pdf](http://www.optimization-online.org/DB_FILE/2014/04/4313.pdf).
- [18] J. Cheng, F. Chu, W. Xia, J. Ding, and X. Ling. "Bi-objective optimization for single machine batch scheduling considering energy cost," in *2014 International Conference on IEEE Control, Decision and Information Technologies (CoDIT)*, Nov. 2014, pp. 236-241.
- [19] V. T'kindt, H. Scott, and J.C. Billaut, *Multicriteria Scheduling: Theory, Models and Algorithms*, Springer Science & Business Media, Dec. 2006, pp 62-267.
- [20] P.-C. Chang, S.-H. Chen, C.-Y. Fan, and C.-L. Chan, "Genetic algorithm integrated with artificial chromosomes for multi-objective flowshop scheduling problems," *Applied Mathematics and Computation*, vol. 205, no. 2, Nov. 2008, pp. 550-561.
- [21] T. Varadharajan and C. Rajendran, "A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs," *European Journal of Operational Research*, vol. 167, no. 3, Dec. 2005, pp. 772-795.
- [22] T. Eren and E. Güner, "A bicriteria flowshop scheduling with a learning effect," *Applied Mathematical Modelling*, vol. 32, no. 9, Sept. 2008, pp. 1719-1733.
- [23] A. Baykasoğlu, L. Özbakır, and T. Dereli. "Multiple dispatching rule based heuristic for multi-objective scheduling of job shops using tabu search," in *Proceedings of MIM: 5th International Conference on Managing Innovations in Manufacturing (MIM)*, Sept. 2002.
- [24] J. Sousa and L. Wolsey, "A time indexed formulation of non-preemptive single machine scheduling problems," *Mathematical Programming*, vol. 54, no. 1-3, Feb. 1992, pp. 353-367.
- [25] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, *Multiobjective optimization: Interactive and evolutionary approaches*, Springer Science & Business Media, Oct. 2008, pp. 9-45.
- [26] O. Grodzevich and O. Romanko. "Normalization and other topics in multi-objective optimization," Oct. 2006, [cited: March 2013]; available from Word Web: <http://www.maths-in-industry.org/miis/233/1/fmipw1-6.pdf>.
- [27] I.Y. Kim and O. De Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Structural and Multidisciplinary Optimization*, vol. 29, no. 2, Sept. 2005, pp. 149-158.
- [28] D.A. Van Veldhuizen and G.B. Lamont. "On measuring multiobjective evolutionary algorithm performance," in *Proceedings of the 2000 Congress on Evolutionary Computation*, vol. 1, Oct. 2000, pp. 204-211.
- [29] J.L.C. Meza, M.B. Yildirim, and A.S.M. Masud, "A model for the multiperiod multiobjective power generation expansion problem," *IEEE Transactions on Power Systems*, vol. 22, no. 2, Sept. 2007, pp. 871-878.

- [30] C.M. Fonseca and P.J. Fleming. "Genetic algorithms for multiobjective optimization: Formulation discussion and generalization," in *ICGA*, vol. 93, Nov. 1993, pp. 416-423.
- [31] P. Hajela and C.-Y. Lin, "Genetic search strategies in multicriterion optimal design," *Structural Optimization*, vol. 4, no. 2, Oct. 1992, pp. 99-107.
- [32] H. Pohlheim, the GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab," June 1994 – March 2006.
- [33] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, Dec. 2002, pp. 182-197.
- [34] I. ILOG, "Inc. CPLEX 12.5 User Manual," Somers, NY, USA, May 2012.
- [35] M.S. Akturk and M.B. Yildirim, "A new dominance rule for the total weighted tardiness problem," *Production Planning & Control*, vol. 10, no. 2, Oct. 1999, pp. 138-149.
- [36] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report, June 2008.
- [37] M. Rabiee, M. Zandieh, and P. Ramezani, "Bi-objective partial flexible job shop scheduling problem: NSGA-II, NPGA, MOGA and PAES approaches," *International Journal of Production Research*, vol. 50, no. 24, March 2012, pp. 7327-7342.
- [38] R. Janssen and M. V. Herwijnen, *Multiobjective Decision Support for Environmental Management*, Kluwer Academic Dordrecht, The Netherlands, Dec.1992, pp 57-196.

## CHAPTER 4

### MULTIOBJECTIVE ANT COLONY ALGORITHM FOR PREEMPTIVE SCHEDULING AND ENERGY COST OPTIMIZATION FOR SINGLE MACHINE PROBLEM WITH TOTAL COMPLETION TIME

#### Abstract

Energy-aware in sustainable practices is gaining significant momentum: companies are not only improving their product quality, but also optimizing the production processes to improve energy consumption (i.e., minimizing energy cost) in order to manage environmental challenges which contribute to global climate change. This chapter models a scheduling problem on a preemptive single machine to minimize the total completion time and total energy cost under time-of-use electricity tariffs, where energy prices vary hourly, which is typically announced a day ahead. The problem is modeled using a multiobjective mixed-integer mathematical programming model, and solved by using either weighted sum approach using CPLEX or a multiobjective ACO based on a dominance rank (ACO-DR) or a multiobjective ACO based on a dominance rank procedure and crowding distance comparison (ACO-DRC) to obtain an approximate Pareto-front. The results show that ACO-DRC algorithm outperforms ACO-DR algorithm with respect to solutions quality. Finally, the MOORA method is utilized to select a solution.

**Keywords:** Preemptive single machine, Energy-aware scheduling, Time-of-use electricity tariffs, Multiobjective optimization, Ant Colony Optimization algorithm

#### 4.1 Introduction

Manufacturing has been historically one of the biggest energy consumers of fossil and renewable energy including coal, oil, nuclear, wind and hydropower and currently accounts for about one-half of the world's total energy consumption [1]. In addition, manufacturing is responsible for 90% of industry energy consumption and 84% of greenhouse gas emissions [2].

Moreover, manufacturing energy consumption, which was at 175 quadrillion Btu in 2006, is projected to increase 40% by 2030 [3]. Indeed, the consumption of energy by the manufacturing sector has virtually doubled over the last 60 years [3]. In the United States, manufacturing sector is the largest energy consumer and consumes about one-half of the world's total energy consumption. Moreover, the manufacturing sector contributes 27% of the U.S. greenhouse gas emissions. Thus, the manufacturing sector become the second after the transportation sector in term of greenhouse gas emissions.

Manufacturing activities play a major role in industrial energy consumption [4]. Despite the fact that the machining performs a small portion of the whole product life cycle, reducing the energy consumed during this state is recently determined as the significant tasks to improve sustainability in manufacturing [5]. For example, Chen et al. [6] note that the painting processes at automotive assembly plants consumed about 16% of the \$700 million energy expenditures in 37 U.S. automobile manufacturing plants. While all the energy consumed and waste produced by manufacturing affect both economic and environmental impacts, it is important for decision-makers to work on utilizing energy-efficiently and effectively because it is less expensive to save energy than to produce that same amount of energy [1]. In a result, energy-efficient scheduling via intelligent operational methods may provide an immediate opportunity to decrease energy intensity of a manufacturing process. For example, utilizing the time-of-use electricity tariffs during production scheduling will make detailed manufacturing scheduling on the energy consumption and associate cost.

Giving increasing energy prices over the last ten years and efforts for sustainability, the energy savings would be more significant in nowadays especially if it could be accomplished without any major equipment investment, focusing efforts on operations instead of focusing solely

on process efficiency by developing and designing more energy-efficient machines to reduce the power and energy demands of machines and tools [7]. One of the strategies, in this chapter, we focus our attention in a preemptive single machine problem, where a machine can response to insert machine “off/on” times over planning horizon and in a time-of-use electricity tariffs, where energy prices vary hourly over planning horizon, which is typically announced a day ahead. Therefore, this strategy integrates different control policies: the first control request switches the machine “off” wherein inserting machine off time is occurring. The second request switches the machine “on” wherein the job (or part of it) is resuming.

Key features of this study are: (1) the opportunity for sectors to have access to time-of-use electricity tariffs, where energy prices vary hourly, (2) sectors that see and respond to energy prices can reduce the demand when prices are high and increase the demand when prices are low, and (3) equipment that processes jobs with allowing preemptions such as industrial battery charging stations or computer data processing jobs. The first feature can be applied via smart metering technology whereas the second feature can be applied via production scheduling. Consequently, manufacturing sectors would pay less for electricity consumption over the long run.

Some research efforts focused on the problem of scheduling start-up and shutdown of machines to minimize total energy consumption. Drake et al. [8] show that machine start-up and machine idling consume a significant amount of energy. Consequently, in a mass production environment, more than 85% of the energy is consumed for activities that are not directly related to the production processes [9]. For example, compressors in an industrial setting consume about 50% of the maximum power when they are idle [10]. In a research study, Prabhu et al. [11] propose an analytical model based on queuing theory to include energy control for waste reduction, and calculate the time interval for switching the machine off during idle period with respect to energy

savings. Chang et al. [12] provide a systematic method to search for energy saving opportunities and strategies in a machining line under random failures.

Mouzon et al. [13] develop a scheduling methodology to reduce the energy consumption of manufacturing equipment, and discover that if non-bottleneck machines are turned off instead of kept idle until the arrival of the next job up to 80% savings in total energy consumption could be saved. Fang et al. [3] develop a general multiobjective mixed-integer linear program that optimizes shop scheduling to consider both productivity and energy-related criteria and consider idle time in the product flow determination by changing the speed of operation which also impacts energy consumption. Chen et al. [6] propose a constrained optimization model for turning machines on/off in a production line using a Markov chain model assuming that have Bernoulli reliability model.

In this chapter, the main contribution is to consider the total energy cost objective and scheduling objective on a preemptive single machine. We assume that, the preemptive single machine is able to respond to inserting machine on/off time while scheduling is designed. Furthermore, when the mathematical model is developed, methods to solve the model need to be introduced in order to find solutions belonging to the global optimal Pareto front. Therefore, we propose a multiobjective mixed-integer mathematical model to minimize the total completion time and total energy cost by inserting machine on/off time on a single machine setting, where preemptions are allowed, (i.e., the processing of a job can be interrupted at any time and restarted at a later time in favor of another job with no cost). Also, we propose two multiobjective ACO algorithms in order to obtain an approximate set of non-dominated solutions and utilize the MOORA method to select the most appropriate solution based on various criteria.

Extensive research has been devoted to the preemptive single machine scheduling problem. However, little attention has been paid to the problem of inserting machine on/off time on a preemptive single machine setting where a multiobjective mathematical model is used. Such problems arise in network routing and battery-operated system such as laptops or mobile phone. For example, Swaminathan and Chakrabarty [14] present a control system to reduce energy consumption and extend battery lifetime and show that by changing the state of the machines (e.g., on/off, etc.), there can be a significant reduction in energy consumption. Inserting machine on/off time occurs when a machine is kept waiting (i.e. the machine is turned off for a predetermined amount of time) for the arrival of a job (or a part of it) that is whether or not being processed on a machine. This may be desirable in many situations such as during the delay of a job (or a part of it) when there are significant amounts of energy costs for early completion (i.e. there can be a significant amount of energy savings when idle machines are turned off for a certain amount of time). Kanet and Sridharan [15] define inserting idle time as a feasible schedule in which a machine is kept idle at a time when it could begin processing another operation. In the context of production scheduling, they provide a taxonomy of environments in which inserting idle time scheduling is relevant.

A scheduling problem that is widely studied in the literature is the minimization of total completion time. Completion time of a job is the time when a machine has finished processing a job. When all release dates are equal (i.e. all the jobs are available at the beginning,  $r_j = 0$ ) and the processing time is not interrupted (i.e. preemptions are not allowed), the SPT rule minimizes the total completion time [16]. When one of these assumptions is omitted (i.e. processing time is interrupted due to inserting machine on/off time), the total completion time problem is NP-hard [17].

To solve difficult optimization problems, metaheuristic approaches are widely utilized. ACO algorithms are used to solve combinatorial optimization problems including multiobjective optimization problems. The basic mechanism of the ACO algorithm is that a colony of artificial ants cooperates in finding good solutions to combinatorial optimization problems. The ants' foraging behavior shifts; when ants venture out for food, they are capable of finding the shortest path from a food source to their nest without using visual cues [18] by exploiting pheromone information [19]. The key of the ants' effectiveness is their pheromones, which is a chemical substance. While traveling from a food source to their nest and vice versa, ants deposit pheromone substances on the ground. Ants can smell this pheromone substance and follow it. New ants will probably prefer to follow the path of stronger pheromone concentrations. This, in turn, increases the number of ants choosing the shortest path. Finally, all ants will be following the shortest path from a food source to their nest.

Colomi et al. [20] propose the first example of the ACO algorithm. Dorigo and Stützle [21] propose the ACO as a new metaheuristic approach for solving hard combinatorial optimization problems in the literature. Bauer et al. [22] propose an ACO application for the single machine total weighted tardiness problem. McMullen [23] proposes the ACO approach to address a just-in-time sequencing problem with multiple objectives. Leguizamón and Coello [24] describe the most relevant and recent developments on the use of the ACO variant for solving multiobjective optimization.

In summary, our contributions are as follows: (1) consider energy cost with a scheduling objective while planning jobs on a preemptive single machine, (2) utilize time-of-use electricity tariffs, where energy prices vary hourly, which is typically announced a day ahead, (3) consider sustainability practices to balance the energy demands and generating capacity (i.e., reducing the

demand when energy prices are high and increase the demand when energy prices are low), (4) design a multiobjective mixed-integer mathematical problem to minimize total energy cost and total completion time, (5) develop two multiobjective ACO algorithms: one is based on a dominance rank (ACO-DR) and second based on a dominance rank procedure and crowding distance comparison (ACO-DRC) to obtain near Pareto-optimal front in a timely fashion, and (6) enable the decision-maker to utilize the energy price-aware to make detailed manufacturing schedule.

The organization of this chapter is as follows: In sections 4.2 and 4.3, the multiobjective model is defined, and a WSM is utilized to solve the model and evaluate the performance of the heuristic. In section 4.4, two ACO algorithms are proposed to solve the multiobjective problem. Section 4.5 discusses experimental design and evaluation. Section 4.6 presents experimental design, generation of the experimental data and some performance measures for evaluating the algorithm. Section 4.7 compares the results of ACO algorithm with and without incorporating a shortest remaining process time. Section 4.8 discusses the results of the applied algorithms and WSM. Section 4.9 discusses the change of energy cost under time-of-use electricity tariffs (i.e., change of the electricity rate structure). In section 4.10, a case study is developed to illustrate the optimized ACO-DRC algorithm. Finally, in section 4.11, a conclusion and future work of the research is presented.

#### **4.2 Multiobjective Mixed-Integer Programming Model to Minimize Energy Cost and Total Completion Time (MMIP-MEC-TCT)**

The MMIP-MEC-TCT problem is a single machine scheduling problem with known deterministic processing times ( $p_j$ ) and all jobs are available at the same time (i.e.  $r_j = 0$ ). The electricity pricing is varying hourly, which is typically announced a day ahead. A machine can process one job at a time and once the job process is started, it can be interrupted and restarted at

a later time, i.e., jobs are preemptive. The objectives here are to minimize the total energy cost and total completion time. Note that a multiobjective problem is NP-hard when one of the objectives being solved is NP-hard [25]. The MMIP-MEC-TCT is formulated as a mixed-integer programming problem.

To model MMIP-MEC-TCT, the time-indexed variable formulation is utilized, whereby the planning horizon is discretized into intervals of one unit time length.

#### 4.2.1 Notation and Formulation

The notation used in the problem statement and through the chapter is as follows:

##### Sets

$N$  Jobs,  $\{1, \dots, n\}$

$T$  Planning horizon,  $\{1, \dots, t\}$

$T'$  Subset of planning horizon,  $T' \subseteq T$

##### Indices

$j, i$  Jobs,  $j, i \in N$

$t, t'$  Time interval,  $t \in T$  and  $t' \in T'$

##### Parameters

$E_t$  Electric price signal at  $t$

$p_j$  Processing time of job  $j$

##### Variables

$C_j$  Completion time of job  $j$

$x_{jt} = \begin{cases} 1, & \text{if the job (or part of job) is assigned to time unit } t \\ 0, & \text{otherwise} \end{cases}$

Note that the jobs are labeled in such a way that if  $p_i \leq p_j$  then  $i \leq j$ .

The proposed mixed-integer programming model MMIP-MEC-TCT is a mathematical program with multiple objectives and several constraints.

$$\text{Minimize } \sum_{j=1}^n C_j \quad (4.1)$$

$$\text{Minimize } \sum_{j=1}^n \sum_{t=1}^{|T|} E_t x_{jt} \quad (4.2)$$

$$\sum_{t=1}^{|T|} x_{jt} = p_j \quad \forall j \in J \quad (4.3)$$

$$\sum_{j=1}^n x_{jt} \leq 1 \quad \forall t \in T \quad (4.4)$$

$$C_j \geq t x_{jt} + 1 \quad \forall j \in N; t \in T \quad (4.5)$$

$$C_j \leq |T| \quad \forall j \in N \quad (4.6)$$

$$C_j \geq 0 \quad \forall j \in N \quad (4.7)$$

$$x_{jt} \in \{0,1\} \quad \forall j \in N; t \in T \quad (4.8)$$

The first objective (4.1) minimizes the total completion time and the second objective (4.2) minimizes energy cost. The satisfaction constraint set (4.3) states that the whole processing time of every job is satisfied; the capacity constraint (4.4) ensures that the machine can handle at most one job during each time interval. Constraint (4.5) provides the completion time of each job. Constraint (4.6) defines the boundary of completion time for each job over time horizon  $|T|$ . Further, Constraint (4.7) is the non-negativity and set (4.8) is the integrality constraint.

MMIP-MEC-TCT has  $|N| + |N||T|$  variables and  $2|N| + |T| + |N||T|$  constraints for a problem of size  $|T|$  and  $|N|$  where  $|T| > \sum_j^n p_j$ . Obviously, if either  $|T|$  or the number of jobs  $N$  increases, then the size of the problem also increases significantly. Note that these constraints could also be written in terms of the start-time variables.

This multiobjective scheduling and pricing problem, MMIP-MEC-TCT, can be solved by using multiobjective programming solution techniques such as the WSM,  $\epsilon$ -constraints method,

goal programming, or multilevel programming [26] or it can be solved approximately by using metaheuristic methods such as GAs, simulating annealing, and tabu search [27].

In the next section of this chapter, we will present the WSM to obtain an exact Pareto front to the multiobjective mathematical model. Next we will propose an efficient multiobjective ACO-DR and ACO-DRC algorithms to solve larger sized problems in a reasonable amount of time.

### 4.3 Weighted Sum Method to Solve MMIP-MEC-TCT Problem

The WSM used to solve multiobjective optimization problems converts the multiobjective problem into an aggregated scalar objective function (WS-MMIP-MEC-TCT) by multiplying each objective function by a weighting factor and summing up all terms as

$$\begin{aligned} \min_{x \in X} \sum_{k=1}^K w_k f_k(x) &= w_1 f_1(x) + w_2 f_2(x) \\ &= w_1 \sum_{j=1}^n C_j + w_2 \sum_{j=1}^n \sum_{t=1}^T E_t h_{jt} \end{aligned}$$

where  $k$  is the number of criteria functions,  $x$  is decision vector,  $X$  is the parameter space, and  $w_k$  is weights. Without loss of generality, the solution for this optimization problem defines a solution in the set of the Pareto front. In practical, repeating the process for various weight values defines some representation of the Pareto front. However, as different objective functions can have different orders of magnitude, the normalization of objectives is needed in order to obtain a Pareto optimal solution consistent with the weights selected by the decision-maker [28]. Therefore, each objective function is normalized using optimal function values in the Nadir point ( $z_k^N = \max_k(f_k(x))$ ) which is an upper bound for that objective to all solutions in the Pareto optimal set and the Utopia point ( $z_k^U = \min_k(f_k(x))$ ) which is the lower bound of the Pareto set optimal. Nadir and Utopia points provide an interval where the optimal objective functions may vary within

the Pareto optimal set [29]. Note that the maximum  $f_k^{max}$  and minimum  $f_k^{min}$  values of the  $k^{th}$  objectives must be obtained individually. The normalized values of each objective function  $f_k$  are computed as

$$f'_k = \frac{f_k - z_k^U}{z_k^N - z_k^U}$$

This transformation maps all possible objective function values to  $[0, 1]$ . Hence, we can characterize the exact (complete) Pareto set by trying all possible sets of weight combinations, for problems with a convex non-dominated Pareto front set (i.e. problem whose non-dominated set has a convex shape) [30]. When a limited set of weights are considered while solving MMIP-MEC-TCT, an exact Pareto front can be obtained.

To determine an approximate Pareto front for large-sized problems in reasonable amount of time, a multiobjective ACO, a multiobjective metaheuristic approach, is proposed.

#### **4.4 Solutions to MMIP-MEC-TCT Model Using Multiobjective Ant Colony Algorithms Based on (ACO-DR) and (ACO-DRC)**

In this section, multiobjective ACO algorithms, based on Swarm Intelligence methods, are proposed for solving multiobjective optimization problem to obtain an approximate Pareto front. Generally, ACO algorithm was inspired by some principles of the behavior simulation of social insects such as ants. In our implementation, the first ACO-DR is based on the dominance ranking procedure, and the second ACO-DRC is based on the dominance ranking procedure and crowding distance comparison. These types of algorithms are used for generating multiple solutions in a single run. In those ACO algorithms, they determine an approximate Pareto front for a large-sized problem in a reasonable amount of time, and they allow flexibility when rescheduling is needed as the result of disruptions on the shop floor or changes in the manufacturing environment. At each algorithmic iteration, the ACO algorithms provide an approximate Pareto solution as well as

information on when to start each job (or part of it) to calculate the scheduling and energy costs for that particular solution. The structure of the proposed ACO algorithms are based on the ACO for the MMIP-MEC-TCT scheduling problem given in Figure 4.1. Note that ACO-DR and ACO-DRC are using the same solution components, but they differ in the evaluation of solution quality (i.e., fitness function and fitness value) and selection process. In the next subsections, we discuss different components of the algorithms in more details.

1. Initialization: Initialize pheromone trails, heuristic information, and parameters.
2. Iterative Loop:
  - 2.1 Use colony of ants to determine on time (starting) and off time (end) over time horizon.
  - 2.2 Construct a complete tour for each ant (and *repeat*):
    - Apply state transition rule to select next node.
    - Apply local updating rule.
    - Obtain approximate Pareto front of tour using reduced WS-MMIP-MEC-TCT.
    - Evaluate fitness value of tours *until* complete solution is constructed.
  - 2.3 Apply global update rule
  - 2.4 Perform selection operation
  - 2.5 Determine Pareto front
3. Cycle: If maximum number of iterations is realized, then stop; else go to step 2.

Figure 4.1: Structure of MMIP-MEC-TCT-MOACOA

#### 4.4.1 Construction of Graph

The ACO algorithms for MMIP-MEC-TCT model can be represented by a complete graph  $G = (N, A)$  with  $N$  being the set of nodes representing the times (e.g.  $N = |T|$ ), and  $A$  being the set of arcs. Each arc  $(a, b) \in A$  is assigned a value (i.e., it can be an actual time moment a price signal)  $d_{a,b}$ , which represents the value at node  $b$ . The total number of arcs is  $(N(N - 1))$ , (see Figure 4.2). On this graph, the goal of an artificial ant is to visit  $n = \sum_j^n p_j$  nodes of  $G$  exactly one, where each artificial ant is initially put on a random chosen start-time (node) and each step iteratively assigns zero to unvisited nodes and 1 to visited nodes of its partial tour. The unvisited nodes are considered to be inserting machine off time, where the machine is turned off for a certain

amount of time. For instance, with  $T = 5$  and  $\sum_j^n p_j = 3$ , the graph has five sets of nodes ( $N = 5$ ) and twenty possibilities for the arcs. Each artificial ant can visit only three out of five nodes. When an artificial ant completes a tour, the visiting node yields the time moment that a job can be assigned to. Furthermore, the visited nodes take value 1 and unvisited nodes take value 0 (e.g. [10110]). Value 1 indicates where a job can be executed and value 0 indicates where the inserting machine off time can be allocated. One tour shows the assignment of execution times and inserted machine off time over the planning horizon.

#### 4.4.2 Constraints

Walks on the construction graph  $G$  have to satisfy the constraints given by equations (4.3) and (4.4) to obtain a valid assignment. One way of generating such an assignment is by an artificial ant's walk which iteratively switches from one node to another node without repeating any node to find routes that correspond to better solutions.

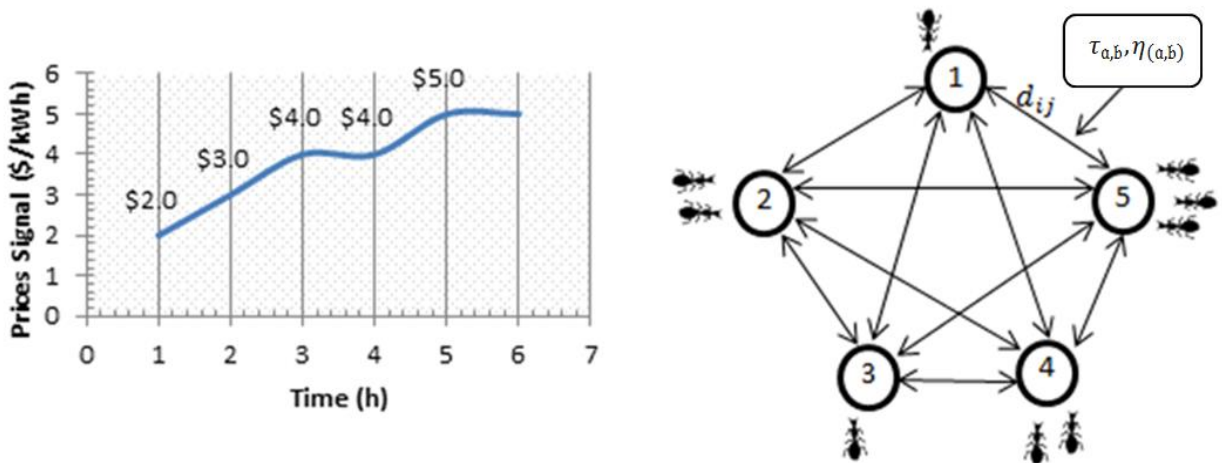


Figure 4.2: Construction graph for ACO algorithms of size  $n = 5$

#### 4.4.3 Pheromone Trail

During the construction of a tour, ants repeatedly have to take the following two basic decisions: (1) choose the node to assign next job based on actual time moment (i.e., node with lowest time indices) in order to minimize the completion time and (2) choose the lower energy

cost node moment (i.e., node with lowest electricity price indices) in order to minimize the energy cost. Pheromone trail information can be associated with any of the two decisions: an appropriate sequence for node assignments in order to minimize the total completion time or to minimize the total energy cost. As a result, a pheromone level  $\tau_{\alpha,b}$  will represent the desirability of assigning node  $b$  to node  $\alpha$ . Note that in our ACO algorithms, the initial pheromone level is calculated by  $\tau_0 = \frac{1}{\Theta_{l\xi}}$ , where  $\Theta_{l\xi}$  is the fitness function value for a solution  $l$  in tour  $\xi$ .

#### **4.4.4 Heuristic Information**

Similarly, heuristic information can be associated with any of the two decisions, on total completion time or total energy cost. For example, heuristic information could bias node assignment towards those nodes that have small time—in other words, this will minimize completion time, and bias the choice of nodes in such a way that small assignment costs are incurred—in other words, this will minimize energy costs (i.e., the machine only needs a relatively small amount of cost to perform the job). In our ACO algorithms and while constructing the schedule, we apply both decisions in random iterations in one run in order to generate different solutions.

#### **4.4.5 Solution Construction**

Construction of a solution can be performed by choosing the components to add to the partial solution from among those that, as explained above, satisfy the constraint with a probability biased by the pheromone trails and heuristic information.

In each iteration, an artificial ant determines starting “on” time and “off” time for all schedules. Each artificial ant repeatedly applies the state transition rule to select one of the decisions from the heuristic information until a complete tour is constructed. When building a tour, both the heuristic information and the pheromone amount are used to determine the on time and

off time (nodes) to be chosen. Section (4.4.6) will explain in detailed how the state transition rule can be applied.

While constructing a tour, an artificial ant also decreases the amount of pheromones between selected nodes in order to dynamically change the attractiveness of arcs by applying the local updating rule to vary other ants' visits (schedule) and to avoid leading to local optima. The local updating rule will be explained in section (4.4.7).

The tour represents a feasible schedule of the on/off times for the machine and has a fixed energy cost. However, the tour does not provide any information on which job is actually scheduled (i.e., the sequence of jobs/parts of jobs). Thus, in a given tour, one has information on the on/off times for the machine which also implies a fixed energy cost. One needs to determine the sequence of each job in order to have information to evaluate each objective function. In section (4.4.8), the process of evaluating each objective function will be addressed.

In each iteration, the fitness function is used to evaluate the quality of the solutions in the current population. The selection operator is capable of preserving those tour that have better fitness values and a better chance of leading to local optima. Based on the best value of the fitness function, the global updating rule is applied to increase pheromone values on solution components between nodes of the best solution up to the current iteration and decrease pheromone values between other nodes. Thus, all ants will focus on a better tour (schedule). Section (4.4.9) shows fitness function and value in the proposed algorithms. Sections (4.4.10) and (4.4.11) discuss the process of selection. And, section (4.4.12) shows how the fitness function value (best value) serves in the global updating rule. Finally, section (4.4.13) presents the stopping criterion in the algorithm.

For any artificial ant  $\gamma$ , while finding appropriate solution for the problem, the set of candidate nodes that may be visited in the reminder of the tour are defined as a tabu list. In an

artificial ant's tabu, those nodes that have been visited (e.g. the nodes that have been assigned already based on heuristic information) are excluded from the choice through the use of a tabu list. This is applied by the state transition rule. Until the last node is selected, the procedure is repeated. Once the algorithm reaches its stopping criterion, the obtained solutions have a well-distributed near-optimal Pareto front, which will then be used in the selection of the best appropriate solution using the MOORA method.

#### 4.4.6 State Transition Rule

In the process of choosing the next node to move, the artificial ant  $\gamma$  in node  $\mathbf{b}$  selects the node  $\mathbf{a}$  to move by applying the following state transition rule:

$$\mathbf{b} = \begin{cases} \arg \max_{u \in S_k(\mathbf{a})} \{[\tau_{(\mathbf{a},u)}]^\alpha [\eta_{(\mathbf{a},u)}]^\beta\} & \text{if } q \leq q_0 \\ \mathfrak{R} & \text{Otherwise} \end{cases}$$

where,  $\tau_{(\mathbf{a},u)}$  is the amount of pheromone trail on arc  $(\mathbf{a}, u)$ ,  $\eta_{(\mathbf{a},u)} = 1/d_{\mathbf{a},u}$  is the inverse of the heuristic information ( $d_{\mathbf{a},u}$ ) between node  $\mathbf{a}$  and node  $u$ .  $S_k(\mathbf{a})$  is the set of feasible nodes to be selected by artificial ant  $\gamma$  in node  $\mathbf{a}$ . Note that the set of feasible nodes not contained in tabu  $\alpha$  is a parameter that allows us to control the relative importance of the pheromone trail ( $\alpha > 0$ );  $\beta$  is a parameter that determines the relative importance of heuristic information ( $\beta > 0$ ).  $q$  is a value chosen randomly with uniform probability in  $[0,1]$ ; and  $q_0$  is a parameter that determines the relative importance of exploitation versus exploration ( $0 \leq q_0 \leq 1$ ).  $\mathfrak{R}$  is a random variable selected according to the following random-proportional rule probability distribution, which is the probability with which that artificial ant  $\gamma$  chooses to move from node  $\mathbf{a}$  to node  $\mathbf{b}$ :

$$P_k(\mathbf{a}, \mathbf{b}) = \begin{cases} \frac{[\tau_{(\mathbf{a},u)}]^\alpha [\eta_{(\mathbf{a},u)}]^\beta}{\sum_{u \in S_k(\mathbf{a})} [\tau_{(\mathbf{a},u)}]^\alpha [\eta_{(\mathbf{a},u)}]^\beta} & \text{if } \mathbf{b} \in S_k(\mathbf{a}) \\ 0 & \text{Otherwise} \end{cases}$$

When an artificial ant in node  $a$  chooses a node  $b$  to move to, it generates a random number  $q$ . If  $q \leq q_0$ , then the best node is chosen according to the following equation:

$$\eta_{(a,u)} = \frac{1}{d_{(a,u)}} \quad (a \neq u)$$

where,  $d_{iu}$  is the heuristic information between node  $i$  and node  $u$ .

#### 4.4.7 Local Updating Rule

While constructing a tour, an artificial ant decreases the pheromone trail level between selected nodes in order to dynamically change the attractiveness of arcs by applying the following local updating rule:

$$\tau_{(a,b)} = (1 - \rho_l) \cdot \tau_{(a,b)} + \rho_l \cdot \tau_0$$

where,  $\tau_0$  is the initial pheromone level and  $\rho_l$  ( $0 < \rho_l < 1$ ) is the local pheromone evaporating parameter. In this way, every time an artificial ant selects an arc, it becomes slightly less desirable. Ants, therefore, adapt pheromone information better. Without local updating, all ants would explore a narrower neighborhood of the best previous tour. The pheromone associated with this arc is adjusted every time the artificial ant chooses a node.

#### 4.4.8 Obtaining Approximate Pareto Front of a Tour Using Reduced MIP

While constructing a tour, an artificial ant generates one solution. By giving a random tour, a feasible schedule of start (on) and off times for the machine are fixed. Therefore, the fixed start (on) times is a subset of time horizon (i.e., a tour represents the subset of time horizon,  $T' \subseteq T$ ). When this information is known, the original model of MMIP-MEC-TCT reduces to the following multiobjective mixed-integer mathematical model (MMIP-MEC-TCT-R) where R means reduce model.

$$\text{Minimize } \sum_{j=1}^n C_j \quad (4.9)$$

$$\text{Minimize } \sum_{j=1}^n \sum_{t' \in T'} E_t x_{jt'} \quad (4.10)$$

$$\sum_{t' \in T'} x_{jt'} = p_j \quad \forall j \in J \quad (4.11)$$

$$\sum_{j=1}^n x_{jt'} = 1 \quad \forall t' \in T' \quad (4.12)$$

$$C_j \geq t' x_{jt'} + 1 \quad \forall j \in N; t' \in T' \quad (4.13)$$

$$C_j \leq |T| \quad \forall j \in N \quad (4.14)$$

$$C_j \geq 0 \quad \forall j \in N \quad (4.15)$$

$$x_{jt'} \in \{0, 1\} \quad \forall j \in N; t' \in T' \quad (4.16)$$

where the capacity constraint (4.12) ensures that the machine can handle only one job (or part of it) during each time interval, i.e.,  $t' \in T'$ . MMIP-MEC-TCT-R has  $|N| + |N||T'|$  variables and  $2|N| + |T'| + |N||T'|$  constraints for a problem of size  $T'$  and  $|N|$  where  $T' = \sum_j^n p_j$  and  $T' \subseteq T$ . Solving the MMIP-MEC-TCT-R will provide the set of non-dominated solution(s) corresponding to a known tour (i.e. a given machine on/off time). Note that in this formulation,  $C_j$  and  $p_j$  indicate completion time and process time of job in position  $j$  in the sequence defining the orders of the jobs.

A random tour which represents a subset of time horizon has a fixed total energy cost. Below we provide a theoretical proof showing the local optimality of the total energy cost function for given a random tour:

**Proposition 1:** For a given a complete tour (i.e., a tour represents the subset of time horizon,  $T' \subseteq T$ ), the total energy cost is constant.

**Proof:** The total energy cost function (equation 4.10) is:

$$\sum_j^n \sum_{t' \in T'} E_t x_{jt'} = \sum_{t' \in T'} E_t \sum_{j=1}^n x_{jt'} \quad (4.17)$$

The capacity constraint ensures the machine can handle only one job (or part of it) at any given time interval (i.e.,  $t' \in T'$ ):

$$\sum_{j=1}^n x_{jt'} = 1 \quad (4.18)$$

Therefore, from equations (4.17) and (4.18), we can prove that the total energy cost function is fixed as follows:

$$\sum_j \sum_{t' \in T'} E_t x_{jt'} = \sum_{t' \in T'} E_t \sum_{j=1}^n x_{jt'} = \sum_{t' \in T'} E_t \quad (4.19)$$

Since proposition 1 proves that for each tour the total energy cost function is fixed. So, we only solve the total completion time objective function to obtain the Pareto optimal solution corresponding to an ant's tour.

SPT or SRPT provides optimal solutions to total completion time problem without any inserted idle time. The question is if such rules can also provide an optimal solution for the problem that we are investigating in this chapter.

In SRPT rule, we schedule all jobs in the order of increasing  $p_j$ . Note that, this information may change while a job is being processed in especially preemptive job environments with unequal release dates. Thus, a job may release to process with a large processing time and low priority respectively, but after some partial processing, it will have a smaller remaining processing time and higher priority respectively. Therefore, while constructing a new sequence, at any time, we process the unfinished job with the SRPT among the available jobs at this time moment. So, we sort jobs in such a way that if  $p_i \leq p_j$  then  $i \leq j$ . Consequently, the computational complexity of our heuristic is determined by  $O(n \log n)$  on each run (i.e., complexity of a sorting algorithm).

Below we provide some theoretical results showing the local optimality of this rule for the considered problem with use of inserting machine on/off time.

**Theorem 1** In an optimal schedule, the total completion time is minimized by scheduling all jobs with respect to SRPT sequence for a MMIP-MEC-TCT-R problem with inserted on/off time and equal release time ( $t = 0$ ).

**Proof:** We prove by interchanging two neighboring jobs (or job parts). Consider two consecutive jobs  $(i, j)$  that are available at the same time ( $r_t = 0$ ),  $i < j$  and  $p_i \leq p_j$ . Let  $S$  represent an optimal schedule that is not the SRPT sequence, then  $p_j \geq p_i$ . Consider the schedule  $S'$  obtained by interchanging the jobs  $(i, j)$  (see Figure 4.3). We can show that  $\sum_{k \in S'} C_k^{S'} < \sum_{k \in S} C_k^S$ , contradicting the optimality of  $S$ .

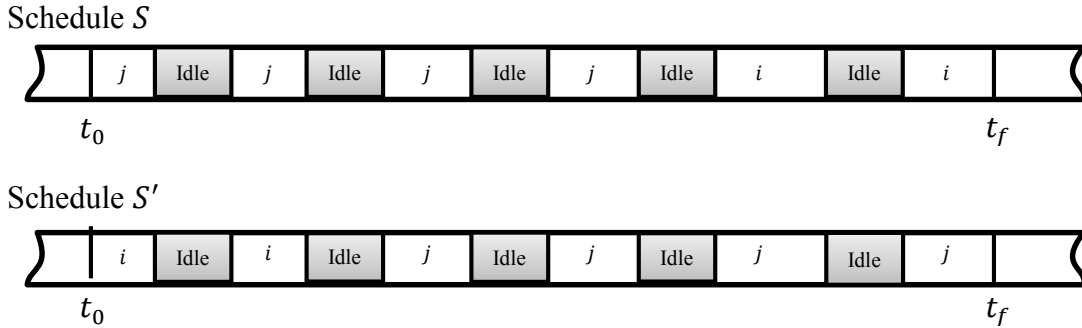


Figure 4.3: A pairwise interchanging of jobs  $j$  and  $i$

Let  $C_j^S$  and  $C_j^{S'}$  denote the completion times of job  $j$  in  $S$  and  $S'$ , respectively. We observe that (1) the first job starts at the same time point regardless the order of the jobs. Furthermore, the second job finishes at the same time point regardless the order of the jobs; and (2) idle ( $I$ ) time periods are fixed over timeline and total idle period is equal to  $I_{of} = t_f - t_0 - p_i - p_j$  where  $t_0$  is the starting time of the first job and  $t_f$  is the finishing time of the second job. Therefore, we have

$$C_j^S = t_{t_0} + p_j + I_{t_0j}$$

$$C_i^{S'} = t_{t_0} + p_i + I_{t_0i}$$

Also, we have  $p_j = p_i + \Delta$ , where  $\Delta$  is the difference between  $p_j$  and  $p_i$ , and  $\Delta \geq 0$  since  $p_j \geq p_i$ . Therefore, when we move over the timeline  $p_i$  amount of time, we encounter  $I_{t_{0i}}$ , for additional  $\Delta$  we may encounter some more idle periods  $\varphi \geq 0$ , i.e.,  $I_{t_{0j}} \geq I_{t_{0i}}$  and  $I_{t_{0j}} = I_{t_{0i}} + \varphi$ , i.e., Now, let's compare the total completion time for each schedule which only differs in the order of jobs  $i$  and  $j$ :

$$\begin{aligned}\sum_{k \in S} C_k^S &= \sum_{k \in S \setminus \{i,j\}} C_k^S + C_j^S + C_i^S = \sum_{k \in S \setminus \{i,j\}} C_k^S + t_0 + p_j + I_{t_{0j}} + t_f \\ \sum_{k \in S'} C_k^{S'} &= \sum_{k \in S' \setminus \{i,j\}} C_k^{S'} + C_i^{S'} + C_j^{S'} = \sum_{k \in S' \setminus \{i,j\}} C_k^{S'} + t_0 + p_i + I_{t_{0i}} + t_f\end{aligned}$$

Since  $\sum_{k \in S \setminus \{i,j\}} C_k^S = \sum_{k \in S' \setminus \{i,j\}} C_k^{S'}$ ,  $p_j = p_i + \Delta$ , and  $I_{0j} = I_{0i} + \varphi$ ,

$$\sum_{k \in S} C_k^S = \sum_{k \in S' \setminus \{i,j\}} C_k^S + t_0 + p_i + \Delta + I_{0i} + \varphi + t_f$$

As a result,  $\sum_{k \in S'} C_k^{S'} < \sum_{k \in S} C_k^S$ .

To illustrate the above theorem by means of a numerical example, consider the data on Table 4.1. The SRPT ( $S'$ ) solution with inserted on/off time is also given in Figure 4.4. All jobs are available to be processed at the same time moment ( $t = 0$ ). In Figure 4.4, schedule  $S$  is not the SRPT sequence. Let's investigate job  $j$  and  $i$ . We see that job  $j$  is being processed at time  $t = 6$  while there exists an unfinished job  $i$  such that; the job  $i$  is also available at the same time  $t = 6$ , and  $p_j \geq p_i$ .

Now, construct a new sequence,  $S'$  by processing job  $i$  first, and processing job  $j$  second. Since  $p_i \leq p_j$  is the newly scheduled job, job  $i$  completes before the time when either job  $j$  or  $i$  is completed in the old schedule  $S$ . That is  $C_i^{S'} = 8$  which is less than  $\min\{C_j^S = 10, C_i^S = 12\}$ . Also note that  $C_j^{S'} = 12$  which is equal to  $\max\{C_j^S = 10, C_i^S = 12\}$ . So the jobs should be

scheduled in the increasing order of the processing time  $p_i$ . Thus, the total completion time decreases by justifying the SRPT rule.

Table 4.1: Numerical Example

$n$	$k$	$j$	$i$
$p_n$	2	3	2
$w_n$	3	4	2

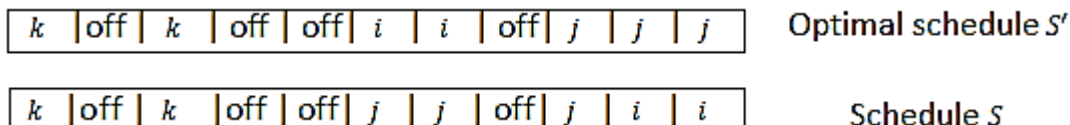


Figure 4.4: Theorem 1 in sequence of individual time slots for different jobs over 12 units of planning horizon

#### 4.4.9 Evaluation of Fitness Functions

Fitness functions are used to evaluate the quality of the solutions (i.e., the current ant population). Each solution has a fitness value, which is assigned according to its dominance. A solution determines the order of the jobs, the completion time of all jobs, and the energy cost of the job completed. Since a tour may characterize one or more solutions, the best measure of the fitness function over the set of solutions that a tour characterizes is kept to evaluate the tour.

Since multiobjective optimization problems try to optimize the components of a vector-valued cost function, and the solution to the problem is a family of points known as the Pareto optimal set, each proposed ACO algorithm uses a unique fitness function for each solution. In the next subsections, we discuss different fitness functions and selection operators of the ACO algorithms in more details.

##### 4.4.9.1 Fitness Function and Value for Tours in ACO-DR

The fitness function in ACO-DR evaluates the solution based on the dominance ranking procedure (i.e., number of solutions by which an solution is dominated) [31]. Let  $\Theta_{i\xi}$  be the fitness function value for a solution  $i$  in tour  $\xi$  corresponding to ant's tour. The rank of the solution is

equal to  $1 + \#_{i\xi}$ , where  $\#_{i\xi}$  denotes the number of solutions that dominates a solution  $i$  in tour  $\xi$ . If the solution is non-dominated, then its rank is 1. Note that the higher the rank, the poorer the solution. This can be written as

$$Fitness_1(\Theta_{i\xi}) = \frac{1}{rank(\Theta_{i\xi})} = \frac{1}{1 + \#_{i\xi}}$$

The fitness function value for a tour  $\psi$ ,  $Fitness_\psi$ , is the sum of the fitness function value for each solution obtained from a tour  $\psi$  using MMIP-MEC-TCT-R model. Note that, in order to obtain a better solution and keep the solution components consistent, the fitness function evaluates all solutions each time the MMIP-MEC-TCT-R generates a new solution in its current iteration.

#### 4.4.9.2 Fitness Function and Value for Tours in ACO-DRC

The fitness function in ACO-DRC is inspired from the non-dominated sorting GA [32]. NSGA-II is a popular GA for solving the multiobjective optimization. It has a better sorting algorithm, incorporating elitism and does not require the choosing of a sharing parameter a priori.

In our implementation, we use the non-dominated sorting procedure and crowding distance sorting procedure to evaluate the quality of solutions in the current ant population. The non-dominated sorting procedure ranks the solution in different Pareto fronts (see Figure 4.5), and the crowding distance sorting procedure measures the density of solutions in the solution space (i.e., calculates the convergence of solution in each front and preserves the diversification of the algorithm). At each iteration, these two procedures evaluate the solutions in the Pareto fronts.

In Figure 4.5, the solutions in level 1 (i.e., rank 1) is exactly the non-dominated set in the current population, and the solution in level 2 (i.e., rank 2) is dominated by solutions in level 1 only and the solutions in levels builds so on. Each solution in each level is assigned fitness (rank) values based on the level (Pareto front) to which they belong. In practice, solutions in the level 1 are given a fitness value of 1 and solutions in the level 2 are assigned fitness value as 2 and so on.

As shown, in Figure 4.5, note that the level in which a solution is located represents the rank, which is the most important factor of its fitness. In a result, a solution with lower rank is preferable.

In addition to the fitness value, a new parameter called crowding distance is computed for each solution. The crowding distance measures how close a solution is to its neighbors (see Figure 4.5). In our implementation, a solution's crowding distance is defined as the sum of the normalized distance between its right and left neighbors for each objective function. For the first and last solutions (extreme solutions) have a crowding distance equal to infinity. In Figure 4.5, the perimeter of the cuboid is estimated by using the nearest neighbor as the vertices [32]. In a result, the crowding distance procedure guarantees diversity of the population (i.e., large average crowding distance results in better diversity in the population). Finally, each individual is sorted based on rank and crowding distance value.

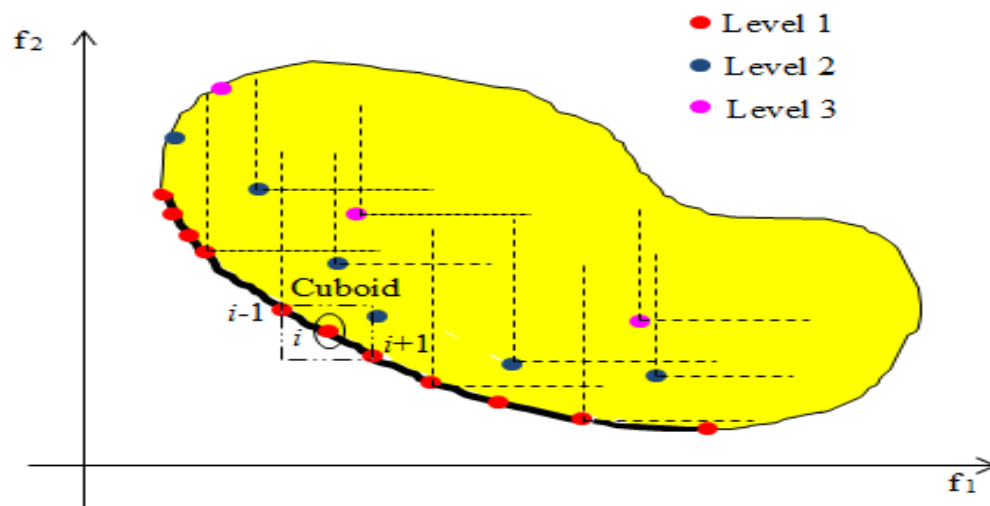


Figure 4.5: Non-dominated levels and computation of crowding distance

#### 4.4.10 Selection Process in ACO-DR

The selection process aims to select the best solution among all available solutions. A tour with a better fitness function will earn more chances to be selected. This type of selection process

was generated by some principles of the metaheuristic algorithms such as GA. We utilize the roulette wheel method, which we chose in order to select the best fitness function for the tour solutions based on rank [33]. This process is described in Figure 4.6.

Note that, the roulette wheel method is performed sequentially according to the tabu list in order to prevent escaping tours. The tours with higher fitness function values have a higher probability to be selected every time.

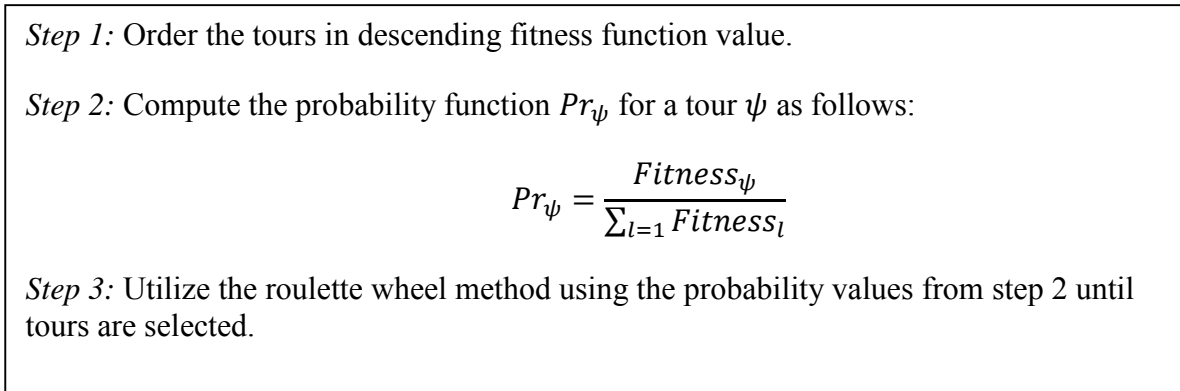


Figure 4.6: Determining fitness function of each tour

#### 4.4.11 Selection Process in ACO-DRC

The selection is based on two comparison rules (i.e., binary tournament selection): (1) the lower rank to which the solution belongs to, the better the solution; and (2) if two solutions have the same ranking, then a solution with greater crowding distance has a better solution because the area to which that solution belongs is less crowded. This process is described in Figure 4.7. Note that the tours with lower rank have better solutions, if they have the same level (i.e., Pareto front), then a tour with greater crowding distance has a better solution and surviving in the next generation.

*Step 1:* Sort individuals based on non-domination rank  $rank_{\iota}$  and with crowding distance assigned  $Front_{\#}(Dis_{\cdot\iota})$ , where  $\iota$  corresponds to the distance  $Dis_{\cdot\iota}$  of the  $\iota^{th}$  solution in  $Front_{\#}$ . Note that each solution  $\iota$  belongs to one chromosome  $\psi$ .

*Step 2:* Utilize a *crowded-comparison-operator* ( $<_{\#_{i\xi}}$ ), where  $\#_{i\xi}$  is the number of solutions  $\iota$  in generation  $\xi$ , based on: (1)  $rank_{\iota}$  and (2)  $Front_{\#}(Dis_{\cdot\iota})$ .

For example,  $\iota_{\#4} <_{\#_{i\xi}} \iota_{\#5}$  if

- $rank_{\iota_{\#4}} < rank_{\iota_{\#5}}$
- Or if  $\iota_{\#4}$  and  $\iota_{\#5}$  belong to the same  $Front_{\#}$  then  $Front_{\#}(Dis_{\cdot\iota_{\#4}}) > Front_{\#}(Dis_{\cdot\iota_{\#5}})$  i.e. the crowding distance should be more.

*Step 3:* Utilize a binary tournament selection with *crowded-comparison-operator* from step 2 until chromosomes are selected.

Figure 4.7: Process of tour selection for generating next ant population

#### 4.4.12 Global Updating Rule

After evaluating the quality of the solutions in the current iteration, we perform the global updating rule. This aims to increase the pheromone values on those solution components that have a better fitness function value. The pheromone trail level is updated as

$$\tau_{(a,b)} = (1 - \rho_g) \cdot \tau_{(a,b)} + \rho_g \cdot \Delta\tau_{(a,b)}$$

where,  $\rho_g$  ( $0 < \rho_g < 1$ ) is the global pheromone evaporating parameter and  $\Delta\tau_{(a,b)}$  is computed by the following equation:

$$\Delta\tau_{(a,b)} = \begin{cases} (\Theta_{i\xi})^{-1} & \text{if } (a, b) \in \text{best tour} \\ 0 & \text{otherwise} \end{cases}$$

where,  $\Theta_{i\xi}$  is the fitness function value of the best solution  $\iota$  in tour  $\xi$  up to the current iteration.

After the search, the pheromone trail level of the new solution is updated proportionally to the improvement of the fitness function value. In this way, we avoid repeating the same solution over and over again and explore a stopping criterion which limits having all ants choose the same route every time.

#### 4.4.13 Stopping Criteria

The ACO algorithms are run until the maximum number of iterations. In this step, the approximate Pareto front is obtained. In the next section, we will present the experimental design and evaluation of performance and effectiveness of the proposed ACO algorithms.

#### 4.5 Experimental Design and Evaluation

We solve the WS-MMIP-MEC-TCT problem using GAMS and solved using CPLEX 12.5 [34]. The performance and effectiveness of the ACO algorithms are evaluated by MATLAB R2010a language code. The computational experiments are performed on an Intel i5 2.27GHz machine with 4GB of memory and a Windows 7 operating system.

#### 4.6 Generation of Experimental Data

The problems are constructed by using varying combinations of three parameters: the number of jobs ( $n$ ), the planning horizon ( $T$ ), and the processing times ( $p_j$ ). Note that, the number of jobs  $|N|$  and size of planning horizon  $|T|$  have a major influence on the performance of the algorithm, since the larger the size of the problem, the more Central Processing Unit (CPU) time it takes.

In the experimentation, problems with 5, 10, 15 and 20, jobs are generated. For each job, an integer processing time is obtained randomly from a uniform distributed between  $[1, 10]$  An integer planning horizon  $|T|$  is computed as  $|T| = \sum_j^n p_j / PHF$ , where  $PHF$  is the planning horizon factor which takes two values: 0.50 for longer planning horizon and 0.75 for shorter planning horizon. The electricity prices  $E_t$  are generated from three uniform distributions  $[1, 5]$ ,  $[6, 10]$ ,  $[11, 16]$  over three different periods of time to create low, moderate, and high variation (see Figure 4.8). Note that, the total number of energy profile is  $E_{profile} =$

$$\frac{|T|!}{(\sum_{j=1}^n p_j)! (|T| - \sum_{j=1}^n p_j)!}$$

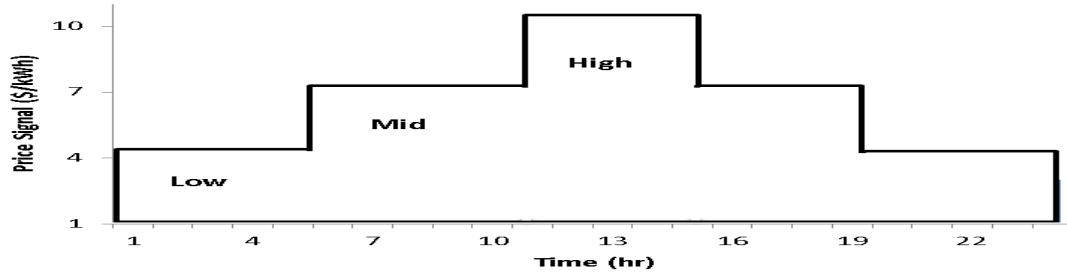


Figure 4.8: Price signal (\$/kWh) vs. time (hr)

#### 4.6.1 Solving MMIP-MEC-TCT Using Weighted Sum Method

A set of four test problems in Table 4.2 are randomly generated for testing and gaining insight into MMIP-MEC-TCT multiobjective optimization problem using the WSM (WS-MMIP-MEC-TCT). We generate 20 instances (i.e., five different instances for each test), solve every instance individually, and measure the average time in seconds.

WS-MMIP-MEC-TCT converts the multiobjective problem into an aggregated scalar objective function by multiplying each objective function by a weighting factor and summing up all terms. Each aggregated scalar objective function optimization determines one particular optimal solution point on the Pareto front. The WSM then changes weights systemically, and each different aggregated scalar objective function optimization determines a different optimal solution.

Table 4.2 presents the results of WSM and measure an average for each problem with the amount of execution time, and the number of non-dominated points obtained. While all generated instances for  $n = 5$  are solved in less than 16,000 second, for  $n = 10$  there is one instance, which WSM has failed to solve to optimality within its time limit, for  $n = 15$  there are already 2 timeouts, and for  $n = 20$  there are 3 timeouts since CPLEX is running out of memory while solving the model. It is clear that the larger the size of the problem the more CPU time it takes, i.e., when the number of jobs increases, the WSM requires a significant amount of CPU time to determine the approximate Pareto front (i.e., the non-dominated solution set), since the search space for the problem increases in terms of variables and constraints.

The CPU time can be decreased if fewer number of combinations of weights are used (i.e., we will not need as many single-objective mixed-integer problems to be solved). However, this will have significant impact on the quality of the Pareto front.

Table 4.2: Results of WS-MMIP-MEC-TCT

$n$	Timeouts	Execution Time (second)		Number of Non-dominated Points	
		Min	Max	Min	Max
5	0	12,243	16,000	27	31
10	1	30,004	>40,000	30	33
15	2	33,102	>40,000	33	36
20	3	37,200	>40,000	12	18

#### 4.6.2 Parameter Fine-Tuning

In our ACO algorithms implementation, the initial artificial ant population size in the colony is  $k_{max} = 20$  and the maximum number of iteration is  $t_{max} = 1000$ . In order to find the best combination for  $\alpha, \beta, \rho_l, \rho_g$ , and  $q_0$ , and also to analysis the impact of these parameters on the computational complexity of ACO algorithms, an experimental design is performed. We apply the ACO algorithms on examples with jobs from set  $n = \{10, 15, 20\}$ . Initially, parameters are set as  $\alpha = 1, \beta = 2, \rho_l = 0.1, \rho_g = 0.1$ , and  $q_0 = 0.9$ . We run each test with same parameters ten times and take the average as the measure of performance. We optimize the parameters of ACO algorithms one at a time, because considering all possible combinations of parameters is very expensive.

So, we change only one parameter at a time until an initial estimate of the appropriate values is obtained while keeping all other parameters constant. The appropriate candidate set for each parameter is the parameter set that yields the appropriate overall performance of the computational complexity of our proposed algorithms.

In order to assess and compare with different parameters, a Generational Distance (GD)

metric [35], found in the literature, is used to measure the performance of the algorithm's solution. GD evaluates the convergence performance of the algorithm. The metric is calculated just before a change occurs. Table 4.3 summarizes these sets that obtained from test results.

- GD

This metric defines as a way of estimating how “far” the elements (i.e., non-dominated solutions) are in global Pareto front from those in global Pareto front and is computed as

$$GD = \frac{\sqrt{\sum_{l=1}^L d_l^2}}{D}$$

where  $D$  is the total number of non-dominated solutions in global Pareto front and  $d_l$  is the Euclidean distance (measure in objective space) between each of these and the nearest member of global Pareto front. Since we consider minimization objectives, the performance of an algorithm is better when the measure has smaller values. If a value of  $GD = 0$ , then all elements generated are in global Pareto front. Therefore, any other value indicates how “far” we are from the global Pareto front of our problem.

#### 4.6.2.1 Choosing Appropriate $\alpha$ , $\beta$ and $q_0$ Parameters

While constructing the solutions, we increase the  $\alpha$ ,  $\beta$  and  $q_0$  parameters from the lowest to the highest value as shown in Table 4.3. In each run, only one value of the parameters is changed and all other parameters are kept constant. Test results in Table 4.4 show that the  $\alpha$ ,  $\beta$  and  $q_0$  parameters of 1, 0.5 and 0.9 lead to good results for the problems with a different numbers of jobs.

#### 4.6.2.2 Choosing Population Size

After  $\alpha$ ,  $\beta$  and  $q_0$  parameters are utilized, we vary the population size from 20 – 50 in increments of 10. The results in Table 4.4 show that the more population size, the better the solution but the longer the CPU time. We decided to limit the population size in the colony to  $k_{max} = 20$  to have solutions in a reasonable amount of time.

### 4.6.2.3 Choosing Appropriate $\rho_l$ and $\rho_g$ Parameters

Using the  $\alpha, \beta, q_0$  and population size at their appropriate value, we increase the  $\rho_l$  and  $\rho_g$  parameters from the lowest to the highest value as shown in Table 4.3. In each run, only  $\rho_l$  or  $\rho_g$  is varied and keep all other parameters at their optimal. Test results in Table 4.4 show that the  $\rho_l$  and  $\rho_g$  parameters of 0.3 for each lead to good results for the test problems.

Table 4.3: Candidate Set

Parameter	Candidate set
$\alpha$	(0.5, 1, 2, 5)
$\beta$	(0.5, 1, 2, 5)
$\rho_l$	(0, 0.1, 0.3, 0.5)
$\rho_g$	(0, 0.1, 0.3, 0.5)
$q_0$	(0, 0.25, 0.75, 0.9)

Table 4.4: GD Metric Performance Depending on Parameters and Number of Jobs

Parameter	Parameter Values	Number of Jobs		
		10	15	20
$\alpha$	0.50	0.283	0.296	0.304
	1	0.239	0.269	0.260
	2.00	0.314	0.329	0.333
	3.00	0.399	0.365	0.354
$\beta$	0.50	0.234	0.247	0.267
	1	0.292	0.318	0.324
	2	0.341	0.347	0.363
	3	0.352	0.368	0.368
$q_0$	0	0.384	0.388	0.388
	0.25	0.379	0.381	0.383
	0.75	0.310	0.314	0.316
	0.90	0.247	0.251	0.251
$k_{max}$	20	0.416	0.419	0.423
	30	0.404	0.408	0.414
	40	0.381	0.381	0.395
	50	0.293	0.298	0.302
$\rho_l$	0	0.287	0.289	0.289
	0.10	0.295	0.298	0.302
	0.30	0.248	0.253	0.264
	0.50	0.296	0.314	0.320

Table 4.4 (continued)

Parameter	Parameter Values	Number of Jobs		
		10	15	20
$\rho_g$	0	0.285	0.288	0.307
	0.10	0.294	0.315	0.324
	0.30	0.239	0.420	0.423
	0.50	0.302	0.308	0.311

#### 4.7 Comparison of ACO Algorithm with-and-without Theorem 1

In this section, the performance and quality of the ACO algorithm with-and-without the theorem 1 in the solution process are compared. After fine-tuning the parameters, we apply the ACO algorithm on the same instances used to characterize the performance of the WSM. In the first approach, we use the theorem 1 (SRPT method) in the solution process to obtain the heuristic solutions whereas in the second approach, the CPLEX solver is used to minimize the total completion time objective. Note that, we run each instance twice and take the average as the measure of performance. We also record the computational time required to solve each instance and the number of solutions obtained from each approach. For comparisons, we use the GD metric, CPU time and the number of non-dominated solution to measure the performance and quality of the applied algorithms. The results are presented in Table 4.5. GD metric shows that ACO with CPLEX is able to perform slightly better results than ACO with theorem 1 whereas CPU time and number of non-dominated solution show that ACO with theorem 1 is able to outperform the ACO algorithm with CPLEX in all test problems. A possible reason for this is that the theorem 1 uses SRPT heuristic to solve the total completion time objective to optimality whereas the CPLEX uses a branch-and-cut algorithm to solve MIP model by generating many subproblems which can require significant amount of CPU time and physical memory.

Table 4.5: ACO-DRC with-and-without Theorem 1

$n$	PHF	ACO with Theorem 1			ACO with CPLEX		
		CPU (second)	Non-dominated Points	GD	CPU (second)	Non-dominated Points	GD
5	0.5	205	31	0.51	720	27	0.43
10	0.75	216	34	0.68	960	29	0.51
15	0.5	292	36	0.53	1260	32	0.47
20	0.75	347	37	0.48	1620	35	0.38

#### 4.8 Comparison of the ACO-DR Algorithm, ACO-DRC Algorithm and WSM

We implement the ACO-DR and ACO-DRC algorithms to solve the multiobjective mathematical model and obtain an approximate solution at a reasonable amount of computational time without any guarantee of optimality. A remarkable feature of ACO-DR and ACO-DRC algorithms is that they may not return the same results twice, since they are based on randomized iterations that use different random tours. Taking these features into consideration, the best way to evaluate the quality of the applied algorithms is by comparing their Pareto front with the Pareto-optimal front obtained by the WSM. In order to do the comparison, we applied the ACO-DR and ACO-DRC algorithms using the fine-tune parameters on the same instances used to characterize the performance of the WSM. Also, two performance metrics: Efficient Set Spacing (ESS) [36], and Size of the Space Covered (SSC) [37] are used to measure the performance and see the dynamics of the applied algorithms. ESS evaluates the diversity performance of an algorithm; while SSC can measure the size of the global dominated set in objective space.

- ESS

This metric measures the distribution of elements in Pareto optimal set over the non-dominated region and is computed as

$$ESS = \sqrt{\frac{1}{e-1} \sum_{i=1}^e (\bar{d} - d_i)^2}$$

where  $d_i = \min_j \{|f_1^i - f_1^j| + |f_2^i - f_2^j|\}$ ,  $j = 1, \dots, e$ , and  $\bar{d}$  refers to the mean of all  $d_i$  and  $e$  is the number of elements of the Pareto optimal set found so far. If a value of  $ESS = 0$ , then the algorithm provides an ideal distribution of the elements of non-dominated vectors (i.e., all elements of  $e$  are equally spaced from one another).

- SSC

This metric estimates the size of the global dominated set in objective space. It computes the area of objective function space covered by the non-dominated solutions vector. In our problem, each dominated vector represents a rectangle defined by the points  $(0,0)$  and  $(f_1(x_i), f_2(x_i))$ , where  $f_1(x_i)$  and  $f_2(x_i)$  represent a non-dominated solution (see Figure 4.9). Therefore, *SSC* is computed as the union of the areas of all the rectangles that correspond to the non-dominated solutions. An algorithm resulting in low values of *SSC* may have a high convergence performance. In contrast, the higher values of *SSC* may result in better diversity performance (i.e., non-dominated solutions are distributed in Pareto optimal set over the non-dominated region).

The ACO-DR and ACO-DRC algorithms are run twice for each instance, and the average is taken as the measure of performance. The computational time required to obtain the Pareto fronts is also computed. Table 4.6 summarizes the results obtained by the applied algorithms and the WSM.

Table 4.6: Results of the WSM and Applied Algorithms

Instance	Number of Jobs	PHF	ACO-DR					ACO-DRC					Weighted Sum Method					Match Solutions
			CPU (sec)	Number of Non-dominated Solutions	Percentage of Solutions (%)	ESS	SSC	CPU (sec)	Number of Non-dominated Solutions	Percentage of Solutions (%)	ESS	SSC	CPU (sec)	Number of Non-dominated Solutions	Percentage of Solutions (%)	ESS	SSC	
1	10	0.75	240	30	14	4.937	27957	216	34	29	5.246	29375	>40,000	15	43	3.892	24931	14
2	20	0.5	366	33	5	7.695	33064	347	37	14	7.870	34131	>40,000	18	63	5.953	28795	18
3	30	0.75	417	35	18	6.326	29358	382	36	23	6.387	32717	>40,000	12	36	2.748	22467	23

Table 4.6 indicates that the applied algorithms are able to obtain more non-dominated solutions than the WSM. Also, it is clear that the applied algorithms are able to obtain non-dominated solutions in about 0.8% of the WSM CPU time. For example, for instance 1 and 3, we observe that the difference between their execution time is more than ten hours, with the WSM, while this is about three minutes with ACO-DR algorithm.

Table 4.6 also shows that, the WSM achieves better ESS performance metric than the applied algorithms. In contrast, the applied algorithms achieve better SSC performance metric than the WSM. A possible reason for this is that a less number of non-dominated solutions enable the WSM to result a low value of SSC, but meanwhile it leads to low diversity among the elements in Pareto-optimal set over the non-dominated region. This means that the applied algorithms perform better respect to high diversity.

The ACO-DRC algorithm results better than the ACO-DR algorithm in solving MMIP-MEC-TCT-R model (see Table 4.6). From Table 4.6, it is clear that the ACO-DRC algorithm achieves better number of non-dominated solutions, CPU time, and SSC performance metrics than ACO-DR algorithm. A possible reason for the ACO-DRC performs the best is that the ACO-DRC has characteristics (i.e., sorting methods) that allow it to maintain enough non-dominated solutions

in the final ant population. However, in order to gain more insight into these results, from Table 4.6, the percentage of solutions for the applied algorithms and WSM indicates what percentage of Pareto front solutions are from each method if Pareto fronts from all methods are combined. Figure 4.10 illustrates the combined Pareto front from the applied algorithms and WSM for instance 2 (i.e., consider only the non-dominated solutions that form the local Pareto-optimal sets). For example, in instance 2, 14% of the combined Pareto front comes from the ACO-DRC algorithm, whereas 63% and 5% come from the WSM and ACO-DR algorithm respectively. Although the number of non-dominated solutions are similar (i.e., the proposed algorithms have a match local Pareto-optimal solutions from the global solutions found in the WSM and also against each other), on average, 18% of the combined Pareto-optimal solutions are from the proposed algorithm and WSM. In conclusion, we can conclude that the ACO-DRC algorithm outperforms the performance of the ACO-DR algorithm with respect to solution quality and also outperforms the WSM with respect to computational CPU time for obtaining the approximate Pareto front.

The next section discusses the results of the ACO-DRC algorithm while changing the electricity rate structure.

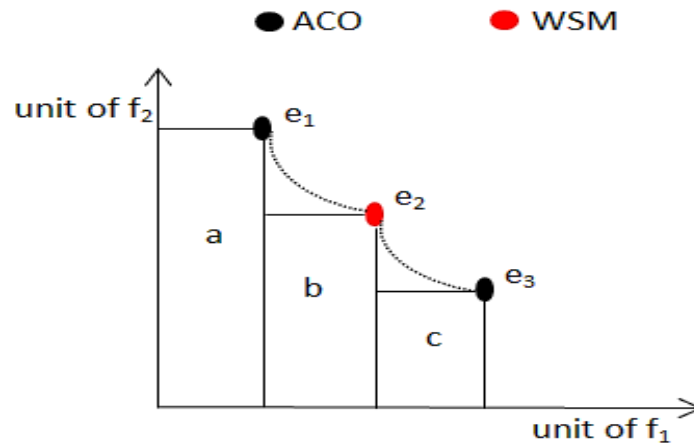


Figure 4.9: Measure of SSC performance of a combined set of non-dominated solutions

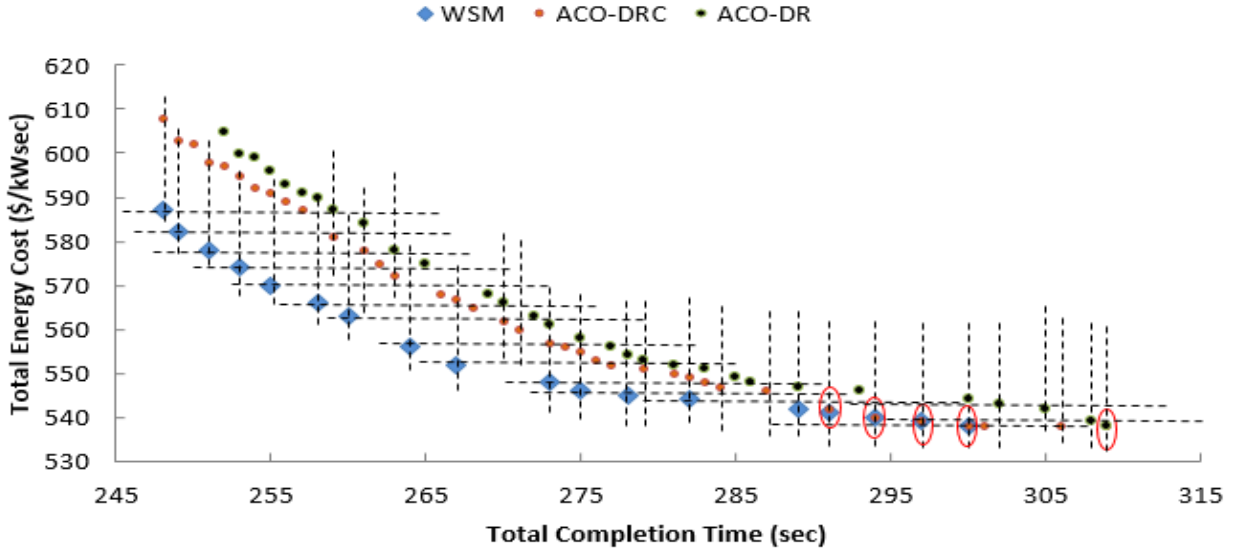


Figure 4.10: Pareto exact and approximation solutions obtained by WSM and applied algorithms, respectively, for instance 2 from Table 4.6

#### 4.9 Impact of Electricity Rate Structure on the Performance of Metaheuristics

Manufacturing sectors can reduce their energy cost by exploiting changes in electricity pricing under time-of-use electricity tariffs. Figure 4.8, in Section 4.6, shows time-of-use tariffs with different rates for electricity consumption during different time periods of the day. Some researchers such as Lei and Feng [38] propose daily and hourly electricity price forecasting in competitive energy market. However, shifting electricity usage from high electricity price periods to low or mid electricity price periods can reduce the total energy cost of the manufacturing sectors with trade-off of increased carbon dioxide emissions from electricity generation, assuming that the shift will not result to additional use of crude oil or coal-fired plants. Therefore, in this section, the energy cost and scheduling objectives under time-of-use tariffs is examined.

To study the effect of the changing electricity prices ( $E_t$ ), we apply the ACO-DRC algorithm on examples with 20, and 30 jobs. For each experiment, we generate three different electricity rate structures. Each electricity rate structure has different electricity prices over periods of time (i.e., one electricity rate structure has moderate, low, and high variation). Then, we run

each experiment three times while changing electricity rate structure every time. We also record the computational time required to obtain the Pareto fronts and compare the performance of the ACO-DRC algorithm using ESS and SSC performance metrics. Table 4.7 summarizes the results obtained with the ACO-DRC algorithm where all the metrics are calculated.

Table 4.7: Results of the ACO-DRC Algorithm

Instance	Number of Jobs	PHF	ACO Algorithm				
			Electricity Rate	Execution Time (sec)	Number of Non-dominated Solutions	ESS	SSC
1	20	0.75	Low – High – Mid	326	37	7.924	34407
2	20	0.5	Mid – High – low	342	37	7.937	34386
3	20	0.75	High – Mid – Low	337	35	7.759	34175
4	30	0.5	Low – High – Mid	376	39	6.836	34847
5	30	0.75	Mid – High – low	392	36	6.721	32594
6	30	0.5	High – Mid – Low	463	37	6.247	34342

From Table 4.7, it is clear to see that the algorithm takes slightly higher CPU time when the number of jobs is increased regardless the change of PHF and electricity rates. In the all cases, the algorithm is able to obtain more number of solutions when the time planning horizon starts with low or moderate electricity rates. A possible reason for this is that while constructing the schedule, heuristic information can be associated with the location of the electricity rates on the time planning horizon—in other words, this will minimize total energy costs. As a result, the algorithm may discover more solutions before hitting the stopping criterion by letting ants walk at the beginning of the time horizon.

The algorithm usually performs better in the ESS metric when the time planning horizon starts with high or moderate electricity rates. In contrast, the algorithm is able to obtain better SSC metric when the time planning horizon starts with low or moderate electricity rates. We can

conclude that the electricity rate structures have enormous significance than the length of planning horizon while solving the model.

The next section explains how the decision-maker can select the best solution among the approximate Pareto obtained by the ACO-DRC algorithm.

#### 4.10. Selecting a Solution from the Pareto Front for Implementation: A Case Study

After the performance of the proposed ACO-DRC algorithm is examined, we consider a case study to show how the decision-maker can select the best solution among the obtained approximate Pareto front. The algorithm uses the values obtained by parameter fine-tuning. In the case study, 40 jobs are available simultaneously and are sequenced on a preemptive single machine. In Figure 4.11, we can see the obtained Pareto fronts. The solutions on the Pareto fronts are well distributed and there is no cluster of solutions. Each solution represents sequence of jobs with their total completion time and total energy cost. This wide range of solutions allows the decision-maker to select the best solution.

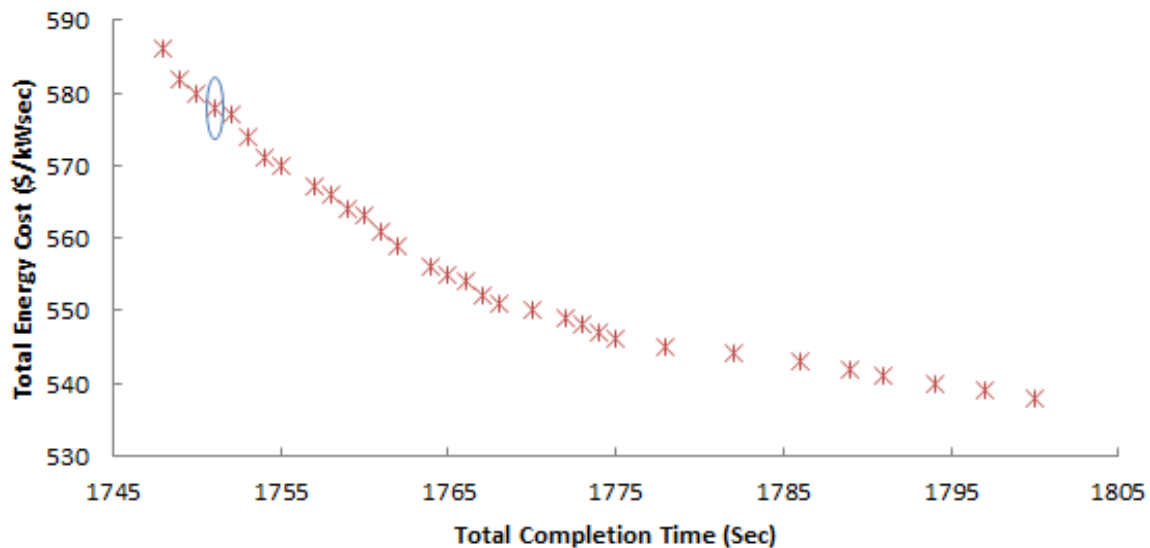


Figure 4.11: Pareto fronts for 40 random jobs with optimized parameter and optimal Pareto front with MOORA selection

MOORA is a method to solve different decision-making problems as frequently encountered in the real-time manufacturing system [39]. It is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. In a decision-making problem, the objectives (attributes) must be measurable and their outcomes can be measures for every decision alternative. Figure 4.12, shows the follow diagram of MOORA. Raw data form the basis of a decision matrix with objectives (attributes) as columns and alternative as rows. In our case, the attributes taken into the consideration are ranked as follows: 1) total energy cost; 2) total number of off-times on the planning horizon; 3) total completion time; 4) makespan (i.e., completion time of the last job) . The decision matrix ( $\Xi_{nb}$ ) showing the performance of different alternatives with respect to various attributes as shown in Table 4.8.

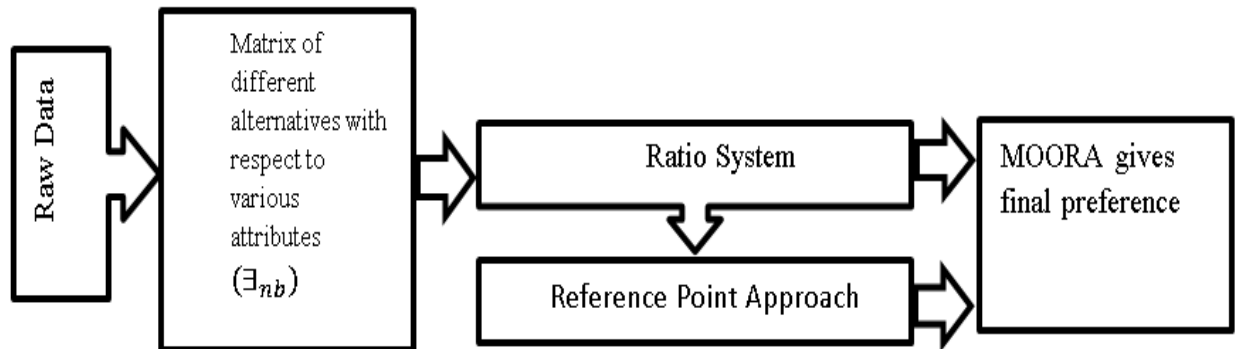


Figure 4.12: Diagram of MOORA

Table 4.8: Decision Matrix with Initial Values

Alternative ( <i>m</i> )	Attributes ( <i>b</i> )			
	Total Energy Cost	Total Number of off-times	Total Completion Time	Makespan
1	586	35	1748	66
2	582	35	1749	66
3	580	35	1750	66
4	578	35	1751	66
5	577	39	1752	70
6	574	39	1753	70
7	571	42	1754	73

Table 4.8 (continued)

Alternative ( $m$ )	Attributes ( $b$ )			
	Total Energy Cost	Total Number of off-times	Total Completion Time	Makespan
8	570	42	1755	73
9	567	42	1757	73
10	566	49	1758	80
11	564	51	1759	82
12	563	53	1760	84
13	561	53	1761	84
14	559	53	1762	84
15	556	58	1764	89
16	555	58	1765	89
17	554	62	1766	93
18	552	64	1767	95
19	551	66	1768	97
20	550	66	1770	97
21	549	68	1772	99
22	548	73	1773	104
23	547	75	1774	106
24	546	75	1775	106
25	545	80	1778	111
26	544	80	1782	111
27	543	85	1786	116
28	542	88	1789	119
29	541	90	1791	121
30	540	90	1794	121
31	539	95	1797	126
32	538	95	1800	126

Table 4.8 presents the performance measure of the  $m^{th}$  alternatives ( $\mathcal{M} = 32$ ) and  $b^{th}$  decision attributes ( $\mathcal{B} = 4$ ).

In the Ratio System each response of an alternative to the attribute is normalized as

$$\Xi_{mb}^* = \Xi_{mb} / \left[ \sum_{m=1}^{\mathcal{M}} \Xi_{mb}^2 \right]^{1/2} \quad \forall b$$

where  $\Xi_{mb}^*$  is a dimensionless number which belongs to the intervals  $[0,1]$  representing the normalized performance of  $m^{th}$  alternative on  $b^{th}$  attribute.

For optimization, in a case of maximization the response (weighted normalized attributes) are added and, in a case of minimization, weighted normalized attributes are subtracted, respectively:

$$\bar{y}_m = \sum_{b=1}^g v_b \Xi_{mb}^* - \sum_{b=g+1}^B v_b \Xi_{mb}^* \quad \forall b$$

where  $g$  is the number of attributes to be maximized,  $(B - g)$  is the number of attributes to be minimized,  $\bar{y}_m$  is the normalized assessment value of  $m^{th}$  alternative with respect to all the attributes, and  $v_b$  is the weight of the  $b^{th}$  attribute, which can be determined applying entropy method as follows:

First step, from Table 4.8, the Ideal and Non-ideal solutions ( $I$  and  $N$ ) are computed as

$$I = (\max \bar{y}_m ; \forall m)$$

$$N = (\min \bar{y}_m ; \forall m)$$

The next step is to compute  $r_{mb}$  matrix which is the normalized value of the  $b^{th}$  attribute on the  $m^{th}$  alternative as

$$r_{mb} = (\Xi_{mb} - \min_b \Xi_b) / (\max_b \Xi_b - \min_b \Xi_b)$$

Now, the entropy can be defined as

$$H_b = -k \sum_{m=1}^{\mathcal{M}} \epsilon_{mb} \ln \epsilon_{mb} \quad \forall m$$

where,

$$\epsilon_{mb} = r_{mb} / \sum_{m=1}^{\mathcal{M}} r_{mb}, \quad k = \frac{1}{\ln(\mathcal{M})}, \quad \text{when } \epsilon_{mb} = 0, \text{ so } \epsilon_{mb} \ln \epsilon_{mb} = 0$$

After the entropy of the  $m^{th}$  alternative is defined, the definition of the entropy weight of the  $b^{th}$  attribute is

$$v_b = (1 - H_b) / \left( \mathcal{B} - \sum_{b=1}^{\mathcal{B}} H_b \right), \quad 0 \leq v_b \leq 1, \quad \sum_{m=b}^{\mathcal{B}} v_b = 1$$

The matrix of the entropies of the  $b^{th}$  attribute is

$$v_b = [0.23, 0.26, 0.24, 0.27]$$

Note that, the  $\bar{y}_m$  value can be positive or negative depending on the totals of its maxima and minima in the decision matrix. An ordinal ranking of  $\bar{y}_m$  shows the final preference. Thus, the best alternative has the highest  $\bar{y}_m$  value, while the worst alternative has the lowest  $\bar{y}_m$  value.

Table 4.9 represents the obtained  $\bar{y}_m$  values where the decision-maker is able to rank them in decreasing order and the preferred solution is the one with maximum value of  $\bar{y}_m$ .

Table 4.9: Ranking of Alternatives Applying Ratio System

Alternative ( $m$ )	$\bar{y}_m$	Rank
1	-0.14235	4
2	-0.14248	3
3	-0.14260	2
4	-0.14288	1
5	-0.14698	6
6	-0.14718	5
7	-0.15021	9
8	-0.15039	8
9	-0.15044	7
10	-0.15870	10
11	-0.16101	11
12	-0.16315	14
13	-0.16327	13
14	-0.16340	12
15	-0.16902	16
16	-0.16907	15
17	-0.17384	17
18	-0.17616	18
19	-0.17852	20
20	-0.17854	19
21	-0.18093	21
22	-0.18697	22
23	-0.18931	24
24	-0.18936	23

Table 4.9 (continued)

Alternative ( $m$ )	$\bar{y}_m$	Rank
25	-0.19540	25
26	-0.19542	26
27	-0.20154	27
28	-0.20519	28
29	-0.20760	30
30	-0.20760	29
31	-0.21369	32
32	-0.21369	31

Using the results from Table 4.9, the decision-maker can select the best solution among all non-dominated solutions based on his/her preference. According to the decision-maker's preference and MOORA, the best solution is a number four (see Table 4.9 and highlighted solution in Figure 4.11).

#### 4.11. Conclusion

Given dynamic energy prices over the last ten years, energy-efficiency scheduling which involves saving energy and reducing environmental impacts provided an immediate opportunity to decrease energy intensity. This chapter is focused on sustainability in manufacturing and energy saving without any major equipment investment, focusing efforts on operations. While evaluating the energy-efficiency of production planning and scheduling at the manufacturing system level, we proposed a multiobjective mixed-integer mathematical model to minimize the total completion time and total energy cost by inserting machine on/off time on a single machine setting (i.e. by simply changing the state of the machines between on/off mode), where preemptions are allowed.

The WSM was used to illustrate the mathematical model and gain insight into the multiobjective problem. Due to the computational complexity involved in solving the mathematical model, an efficient multiobjective ACO-DR and ACO-DRC algorithms to solve larger-sized problems in a reasonable amount of CPU time were proposed in order to obtain an

approximate set of non-dominated solutions. The results have shown that the proposed ACO-DRC algorithm outperforms ACO-DR algorithm. In addition, the ACO-DRC algorithm was able to find a good Pareto approximation of the WSM's Pareto-optimal front while maintaining a diverse solutions. In addition, ACO-DRC also outperformed the WSM with respect to computational CPU time for obtaining the approximate Pareto front.

Given an artificial ant's tour, one has information on the on/off time over planning horizon. When this information is known, the original model of MMIP-MEC-TCT was reduced to the multiobjective mixed-integer mathematical model MMIP-MEC-TCT-R in which the number of variables and constraints were reduced to  $|N| + |N||T'|$  and  $2|N| + |T'| + |N||T'|$ , respectively. This reduction accelerates heuristic solutions.

We discussed the results of computational experiments to test the performance and effectiveness of the developed algorithm by varying several. It was observed that selecting of good parameters can play vital role in the running time of CPU. After fine-tuning the algorithm (i.e. using the best value for each parameter of  $t_{max}$ ,  $\alpha$ ,  $\beta$ ,  $\rho_l$ ,  $\rho_g$ ,  $k_{max}$  and  $q_0$ ), the algorithm was illustrated on a case study. The MOORA method to select the most appropriate solution based on different criteria was utilized.

A direct extension of this research would be to investigate other exact solution techniques to solve the MMIP-MEC-TCT model. For example, utilizing the eps-constraint method will generate supported and non-supported Pareto-optimal solutions, since the WSM can only generate the supported Pareto-optimal solutions (i.e., solutions that lie on the convex envelop of the Pareto frontier). Another research direction would be to study other problems with conflicting objectives on various operating environments (i.e., with different scheduling objectives, having sequence-dependent setup times, or on multi-machine settings). For example, the formulation may change

significantly when other scheduling objectives are considered on a single machine setting, but the proposed solution approach will be quite similar, except for the mixed-integer linear programming, which is used to generate the approximate Pareto front and may change in ACO algorithm based on the type of scheduling objective that is employed. Amore detailed experiment to determine if there is a clear pattern between the parameters of the experimental design and the objective could be designed.

#### 4.12. References

- [1] U.S. Energy Information Agency, "Annual Energy Outlook 2013," [Online: Annual Report], Nov. 2013; [cited: May 2013], available from World Wide Web: <http://www.eia.gov/forecasts/aeo/>.
- [2] M. Schipper, "Energy-related carbon dioxide emissions in US manufacturing," in *US Energy Information Administration*, Oct. 2006, pp. 8-10.
- [3] K. Fang, N. Uhan, F. Zhao, and J.W. Sutherland, "A new shop scheduling approach in support of sustainable manufacturing," in *Glocalized Solutions for Sustainability in Manufacturing*, Springer, March 2011, pp. 305-310.
- [4] M. Garetti and M. Taisch, "Sustainable manufacturing: trends and research challenges," *Production Planning & Control*, vol. 23, no. 2-3, Nov. 2012, pp. 83-104.
- [5] Y. He, B. Liu, X. Zhang, H. Gao, and X. Liu, "A modeling method of task-oriented energy consumption for machining manufacturing system," *Journal of Cleaner Production*, vol. 23, no. 1, March 2012, pp. 167-174.
- [6] G. Chen, L. Zhang, J. Arinez, and S. Biller, "Energy-efficient production systems through schedule-based operations," *Automation Science and Engineering, IEEE Transactions on*, vol. 10, no. 1, April 2013, pp. 27-37.
- [7] M.A. Salido, J. Escamilla, F. Barber, A. Giret, D. Tang, and M. Dai. "Energy-aware parameters in job-shop scheduling problems," in *GREEN-COPLAS 2013: IJCAI 2013 Workshop on Constraint Reasoning, Planning and Scheduling Problems for a Sustainable Future*, vol. 1, May 2013, pp. 44-53.
- [8] R. Drake, M.B. Yildirim, J.M. Twomey, L.E. Whitman, J.S. Ahmad, and P. Lodhia. "Data collection framework on energy consumption in manufacturing," in *Wichita State University Libraries SOAR: Shocker Open Access Repository*, Wichita State University, May 2006.

- [9] T. Gutowski, C. Murphy, D. Allen, D. Bauer, B. Bras, T. Piwonka, P. Sheng, J. Sutherland, D. Thurston, and E. Wolff, "Environmentally benign manufacturing: observations from Japan, Europe and the United States," *Journal of Cleaner Production*, vol. 13, no. 1, Feb. 2005, pp. 1-17.
- [10] M.B. Yildirim and G. Mouzon, "Single machine sustainable production planning to minimize total energy consumption and total completion time using a multiple objective genetic algorithm," *Engineering Management, IEEE Transactions on*, vol. 59, no. 4, July 2012, pp. 585-597.
- [11] V.V. Prabhu, H.W. Jeon, and M. Taisch. "Modeling green factory physics—An analytical approach," in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, vol.: IEEE, Sept. 2012, pp. 46-51.
- [12] Q. Chang, G. Xiao, S. Biller, and L. Li, "Energy Saving Opportunity Analysis of Automotive Serial Production Systems," *Automation Science and Engineering, IEEE Transactions on*, vol. 10, no. 2, March 2013, pp. 334-342.
- [13] G. Mouzon, M.B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *International Journal of Production Research*, vol. 45, no. 18-19, July 2007, pp. 4247-4271.
- [14] V. Swaminathan and K. Chakrabarty, "Energy-conscious, deterministic I/O device scheduling in hard real-time systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 7, Oct. 2003, pp. 847-858.
- [15] J.J. Kanet and V. Sridharan, "Scheduling with inserted idle time: problem taxonomy and literature review," *Operations Research*, vol. 48, no. 1, Feb. 2000, pp. 99-110.
- [16] M. Pinedo, *Planning and Scheduling in Manufacturing and Services*, vol. 3: Springer, Aug. 2005, pp 96-157.
- [17] M.S. Akturk, J.B. Ghosh, and E.D. Gunes, "Scheduling with tool changes to minimize total completion time: a study of heuristics and their performance," *Naval Research Logistics (NRL)*, vol. 50, no. 1, Oct. 2003, pp. 15-30.
- [18] B. Hölldobler, *The Ants*, Harvard University Press, Sept. 1990, pp 64-112.
- [19] M. Dorigo and L.M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, July 1997, pp. 53-66.
- [20] A. Coloni, M. Dorigo, and V. Maniezzo. "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142: Paris, France, Aug. 1991, pp. 134-142.
- [21] M. Dorigo and T. Stützle, "The ant colony optimization metaheuristic: Algorithms, applications, and advances," in *Handbook of Metaheuristics*, Springer, Dec. 2003, pp. 250-285.

- [22] A. Bauer, B. Bullnheimer, R.F. Hartl, and C. Strauss. "An ant colony optimization approach for the single machine total tardiness problem," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2: IEEE, Oct. 1999
- [23] P.R. McMullen, "An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives," *Artificial Intelligence in Engineering*, vol. 15, no. 3, June 2001, pp. 309-317.
- [24] G. Leguizamón and C.A.C. Coello, "Multi-objective ant colony optimization: A taxonomy and review of approaches," *Integration of Swarm Intelligence and Artificial Neural Network*, July 2011, pp. 67-94.
- [25] V. T'kindt, H. Scott, and J.C. Billaut, *Multicriteria Scheduling: Theory, Models And Algorithms*, Springer Science & Business Media, Dec. 2006, pp 62-267.
- [26] M. Caramia and P. Dell'Olmo, *Multi-Objective Management In Freight Logistics: Increasing Capacity, Service Level And Safety With Optimization Algorithms*, Springer Science & Business Media, Oct. 2008, pp. 68-208.
- [27] A. Jaskiewicz, *Multiple objective metaheuristic algorithms for combinatorial optimization*, Poznan University of Technology, Dec. 2001, pp. 68-147.
- [28] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, *Multiobjective optimization: Interactive and evolutionary approaches*, Springer Science & Business Media, Oct. 2008, pp. 9-45.
- [29] O. Grodzevich and O. Romanko. "Normalization and other topics in multi-objective optimization," Oct. 2006, [cited: March 2013]; available from Word Web: <http://www.maths-in-industry.org/miis/233/1/fmipw1-6.pdf>.
- [30] I.Y. Kim and O. De Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Structural and multidisciplinary optimization*, vol. 29, no. 2, Sept. 2005, pp. 149-158.
- [31] C.M. Fonseca and P.J. Fleming. "Genetic algorithms for multiobjective optimization: formulation discussion and generalization," in *ICGA*, vol. 93, Nov. 1993, pp. 416-423.
- [32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, Dec. 2002, pp. 182-197.
- [33] H. Pohlheim, the GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab," June 1994 – March 2006.
- [34] I. ILOG, "Inc. CPLEX 12.5 User Manual," Somers, NY, USA, May 2012.
- [35] D.A. Van Veldhuizen and G.B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis (report)," T.R. TR-98-03, Editor, Citeseer: Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Dec. 1998.

- [36] J.R. Schott, Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, Department of Aeronautics and Astronautics (technical report), Massachusetts Institute of Technology, May 1995.
- [37] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, March 1999, pp. 257-271.
- [38] M. Lei and Z. Feng, "A proposed grey model for short-term electricity price forecasting in competitive power markets," *International Journal of Electrical Power & Energy Systems*, vol. 43, no. 1, Nov. 2012, pp. 531-538.
- [39] S. Chakraborty, "Applications of the MOORA method for decision-making in manufacturing environment," *The International Journal of Advanced Manufacturing Technology*, vol. 54, no. 9-12, Oct. 2011, pp. 1155-1166.

## CHAPTER 5

### MULTIOBJECTIVE OPTIMIZATION MODEL FOR PARALLEL MACHINE SCHEDULING WITH LOAD BALANCING, TOTAL ENERGY COST, AND TOTAL COMPLETION TIME

#### Abstract

Improving the energy-efficiency in a manufacturing environment with incorporating real-time energy pricing via scheduling problem is a challenging problem. The purpose of this chapter is to study a production planning and scheduling problem on a non-preemptive parallel machine to minimize the total energy cost, total completion time and the load balance (MIP-MO-TE-TC-LB-PM) under time-of-use electricity tariffs with varying energy prices, which is a multiobjective mixed-integer mathematical programming model. The proposed model is solved via several methods including the  $\varepsilon$ -constraint method, multiobjective GRASP, and multiobjective GA based on the dominance ranking procedure and crowding distance comparison to determine its population composition (GA-DRC). All metaheuristics solve the model and obtain an approximate Pareto front in a reasonable amount of time. We provide detail experimental results evaluating the performance of the proposed algorithms. Finally, we provide an analysis, and we detail experimental results evaluating the performance of the algorithms, which greatly maintain the quality of solutions.

**Keywords:** Green manufacturing,  $\varepsilon$ -constraint method, Multiobjective optimization, Energy-aware scheduling, Time-of-use electricity tariffs

#### 5.1 Introduction

Energy-efficiency is an indispensable productivity objective in manufacturing systems due to raising energy costs and increasing pressure to reduce energy consumption costs and environmental impacts (i.e., CO<sub>2</sub> emissions). Therefore, many countries have urged manufacturers

to reduce their energy consumption and improve their environmental performance. Energy cost is no longer considered as an additional cost but as valuable and manageable resource.

An improved manufacturing process is limited because resources like knowledge, time, and transparency are lacking in relation to energy consumption costs. Therefore, in this chapter, we focus on sustainability and green manufacturing systems by accomplishing savings in energy consumption costs without any major equipment investment. Our chapter focuses its efforts on operations instead of focusing solely on process efficiency by developing and designing more energy-efficient machines to reduce the power and energy demands of machines and tools. As such, we investigate shifting electricity usage from high-electricity price periods to low- or mid-electricity price periods and how we can reduce the total energy consumption costs of the manufacturing systems. In this case, the tradeoff is increased carbon dioxide emissions from electricity generation, assuming that the shift will not result in additional use of crude oil or coal-fired plants. Therefore, in this chapter, the main goal is to minimize the total energy costs via scheduling problems in a parallel machine setting under time-of-use tariffs. The result of the problem is a multiobjective mixed-integer optimization model used to minimize the total energy cost, total completion time, and deviation from load balancing on a parallel machine (MIP-MO-TE-TC-LB-PM), where preemption is not allowed, (i.e., the processing of a job cannot be interrupted). By solving this model, a Pareto front that contains the non-dominated solutions is obtained. Therefore, we utilize an  $\varepsilon$ -constraint method to obtain the exact Pareto front, and we also develop a multiobjective GRASP and multiobjective GA based on the dominance ranking procedure and crowding distance comparison to determine its population composition (GA-DRC) metaheuristics to solve the model and obtain an approximate Pareto front. We also use the exact Pareto front as a benchmark to evaluate the performance of the approximate Pareto front. Finally,

we analyze the performance of the GRASP and the GA-DRC using performance measures such as the Pareto-optimal solutions that are obtained from the  $\varepsilon$ -constraint method.

Production planning and management approaches are potential tools that increase energy-efficiency and allow manufacturing to achieve proper output with less energy consumption [1]. For this reason, many articles studied approaches to increase energy-efficiency in various level of manufacturing. Schmidt et al. [2] present a methodology for the reliable prediction of energy consumption of arbitrary manufacturing processes, using consumption models to help to calculate the product carbon footprint and validate measures to improve energy-efficiency in production. Mustafaraj et al. [3] develop a methodology that accurately and flexibly determines the auxiliary and value-added electricity in manufacturing operations. Shrouf and Miragliotta [4] present a framework to support gather energy data integration into a company's information technology in order to improve efficiency. Mousavi et al. [5] develop an integrated conceptual framework to model the energy consumption of a production system, where the outcome potentially leads to a more streamlined process plan and a more energy-efficient production system. May et al. [6] provide a seven-step method to support manufacturing companies in order to develop energy-based performance indicators. They enable energy-related information to assess the ability of manufacturing companies to reach their energy-efficiency goals. Fysikopoulos et al. [7] propose a generic energy-efficiency approach into four manufacturing levels—process, machine, production line, and factory. Based on aforementioned publications, the research studies aim to incorporate the energy consumption costs in different measures and methodologies. In brief, they found significant improvement in the potential of energy-efficiency in their energy consumption costs models and methods.

Recently, time-varying electricity prices have become an important factor in defining energy consumption costs. Thus, because electricity prices vary hourly, they represent a huge opportunity to minimize energy costs by shifting electricity usage from high-peak hours to low-peak or mid-peak hours. For example, the wholesale electricity market is open to anyone who connects to the grid, and anyone who is willing to receive electricity prices at different altitudes, depending on the time of day (i.e., retail electricity prices that vary hourly to reflect dynamics in the market). Such time-of-use tariffs provide real-time electricity pricing over time that represents an interesting challenge to minimize the total energy cost in a planning and scheduling problem. Therefore, the most recent research has focused on reducing the total electricity consumption costs via time-of-use tariffs. For example, Zhang et al. [8] develop a mathematical model to minimize the electricity cost and the carbon footprint under time-of-use tariffs without compromising production throughput. In further research, Zhang et al. [9] use a distributed optimization approach to minimize the total electricity cost as a function of the manufacturing schedule, where energy-efficient scheduling is subject to real-time electricity pricing. Cheng et al. [10] investigate a new bi-objective single machine batch scheduling problem with a time-of-use policy to improve the productivity and minimize the total electricity cost, whereas Fang et al. [11] consider the problem of scheduling jobs on a single machine to minimize the total electricity cost of processing these jobs under time-of-use electricity tariffs.

In a manufacturing environment, there is significant interest in reducing the total energy consumption costs in a scheduling problem. Mouzon et al. [12] present a scheduling methodology to reduce the energy consumption of manufacturing equipment, discovering that if non-bottleneck machines are turned off instead of kept idle until the arrival of the next job, up to 80% of the total energy consumption could be saved. In further research, Yildirim and Mouzon [13] develop a

multiobjective GA to minimize energy consumption and reduce the total completion time of a single machine scheduling problem. In order to expedite the computational time required, they also integrate a heuristic to obtain Pareto optimal solutions for a sequence of jobs, instead of solving a multiobjective linear programming subproblem. Moon et al. [14] propose a hybrid generic algorithm to solve the unrelated parallel machine problem with the same due date by inserting idling time to reduce the total electricity costs. Furthermore, Escamilla et al. [15] develop a GA to solve an extended version of the job-shop scheduling problem wherein machines consume different amounts of energy to process jobs at different rates. Shrouf et al. [16] consider variable energy prices during one day to develop a mathematical model to minimize energy consumption costs for single machine during production scheduling. Dai et al. [17] present a multiobjective total energy consumption and makespan job-shop problem to describe an energy-aware integrated process plan and schedule. Xu et al., [18] focus on power-peak related energy-aware scheduling problem and present a mixed-integer programming model and simulation to ensure a global optimum. Duerden et al. [19] present a methodology for modifying a manufacturing production schedule with the goal to minimize variance in production line energy consumption. Ding et al. [20] consider a permutation flow shop scheduling problem with the objective of minimizing the total carbon emission and the total length of the schedule (makespan). May et al. [21] focus on developing different policies in order to control the behavior of machine components: therefore, the energetic states at which the machines should be operated are dependent upon the period duration and power requirements of the various states.

In a manufacturing environment, parallel machines are added to the assembly lines to increase capacity, to add flexibility, to reduce the work in process, and to remove bottlenecks. In the scheduling problem, sometimes the machines are identical and have the same specifications.

However, most of the time, the machines are installed at different times and thus have different specifications as a result of a brand new machine or a new technology. Thus, the processing speed can vary, so that each job has a different processing time on each of the machines. Therefore, it is important to minimize the total completion time of a set of jobs on the machines. The total completion time correlates to the jobs' cycle time and it is a good measure of the productivity objective in a manufacturing system. However, minimizing the total completion time could result in different loads being processed on the machines. Therefore, load balancing assists in accelerating the shifting of jobs on the machines by removing bottlenecks. This technique is used to distribute the workload on the machines in order to reduce the job's waiting time. Load balancing tends to equal the utilization of each machine. As such, it is also important to minimize the total deviation so that all machines balancing. Finally, in order to minimize the total energy costs and maintain sustainability in a manufacturing environment, we utilize the time-of-use tariffs as a key factor in the proposed model. The solution to the problem will be a set of non-dominated solutions representing different compromises among the three objectives.

The resulting problem is a multiobjective mixed-integer optimization problem, where the objective of the total completion time is an NP-hard problem [22]. As a result, the mathematical model in this chapter is also an NP-hard problem since one of the objectives solved is NP-hard [23]. Thus, this problem cannot be solved optimally in a reasonable amount of central processing unit (CPU) time. The problem is solved globally in one step using either the  $\epsilon$ -constraint method or the GRASP and GA-DRC metaheuristic algorithms.

In this chapter our key contributions will include (1) considering the energy cost with a scheduling objective while planning jobs on a parallel non-identical machine; (2) designing a multiobjective mixed-integer mathematical problem to minimize the total energy cost, total

completion time, and deviation from load balancing on machines; (3) utilizing the  $\varepsilon$ -constraint method as an exact method for generating the Pareto-optimal front for the MIP-MO-TE-TC-LB-PM model; (4) developing a multiobjective GRASP and GA-DRC metaheuristic algorithms to solve the model and obtain the approximate Pareto front in a reasonable amount of time; (5) providing analysis and detailed experimental results evaluating the performance of the algorithms, which greatly maintains the quality of solutions.

The organization of this chapter is as follows: in section 5.2 and 5.3, the multiobjective model is defined and an  $\varepsilon$ -constraint method is utilized to generate the Pareto-optimal front. In section 5.4, GRASP and GA-DRC metaheuristic algorithms are developed to solve the multiobjective problem. Section 5.5 defines the experimental design. Section 5.6 presents the results of the  $\varepsilon$ -constraint method. Sections 5.7 and 5.8 design the parameter fine-tune and discuss the results of the GRASP, GA-DRC algorithms and the  $\varepsilon$ -constraint method. Finally, in section 5.9, we present our conclusion and the future work of the research.

## **5.2 Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost, Total Completion Time, and Deviation from Load Balancing on Machines MMIP-MO-TE-TC-LB-PM**

The MMIP-MO-TE-TC-LB-PM problem is a parallel non-identical machine scheduling problem with deterministic processing times ( $p_j$ ) jobs, which are available at the same time (i.e.  $r_j = 0$ ). Under time-of-use electricity tariffs, the electricity prices vary over time. A machine can process one job at a time, and once the process of a job is started, it cannot be interrupted, i.e., jobs are non-preemptive. The objectives are to minimize the total energy cost, the total completion time, and the deviation from load balancing. The MMIP-MO-TE-TC-LB-PM model is formulated as a mixed-integer programming (MIP) problem; however, we utilize the time-index variables formulation in which the planning horizon is discretized into intervals of one unit length.

### 5.2.1 Notation and Formulation

The notation used in the problem statement and through the chapter is as follows:

#### Sets

- N Jobs,  $\{1, \dots, n\}$   
T Planning horizon,  $\{1, \dots, |T|\}$   
K Machines,  $\{1, \dots, |K|\}$

#### Indices

- $j, i$  Jobs,  $j, i \in N$   
 $t, t'$  Time interval,  $t, t' \in T$   
 $k$  Machines,  $k \in K$

#### Parameters

- M A large number  
 $E_{tk}$  Electric price signal at  $t$   
 $p_j$  Process time of job  $j$

#### Variables

- $C_{jk}$  Completion time of job  $j$   
 $L_k$  Load balancing of machine  $k$   
 $x_{jtk} = \begin{cases} 1, & \text{if job } j \text{ starts at time } t \text{ on machine } k \\ 0, & \text{otherwise} \end{cases}$   
 $\delta_{jik} = \begin{cases} 1, & \text{if job } j \text{ is processed before job } i \text{ on machine } k \\ 0, & \text{otherwise} \end{cases}$   
 $Y_{jtk} = \begin{cases} 1, & \text{if } x_{jtk} \text{ equals to 0 on machine } k \\ 0, & \text{otherwise} \end{cases}$   
 $h_{jt'k} = \begin{cases} 1, & \text{if job } j \text{ is processed in interval } t \text{ on machine } k \\ 0, & \text{otherwise} \end{cases}$

The proposed time-indexed mixed-integer programming model is a mathematical program with multiple objectives and several constraints (Note: that these constraints could also be written in terms of the start-time variables).

$$\text{Minimize } \sum_{k=1}^k \sum_{j=1}^n C_{jk} \quad (5.1)$$

$$\text{Minimize } \sum_{k=1}^k \sum_{j=1}^n \sum_{t \in T} E_t h_{jtk} \quad (5.2)$$

$$\text{Minimize } \sum_{k=1}^k L_k \quad (5.3)$$

$$\sum_{k=1}^k \sum_{t=1}^{|T|-p_j+1} x_{jtk} = 1 \quad \forall j \in N \quad (5.4)$$

$$\sum_{j=1}^n \sum_{t'=\max(0,t-p_j+1)}^t x_{jtk} \leq 1 \quad \forall t \in T; k \in K \quad (5.5)$$

$$C_{jk} \geq \sum_{t=1}^{|T|-p_j+1} (t-1+p_j)x_{jtk} \quad \forall j \in N; k \in K \quad (5.6)$$

$$x_{jtk} \leq M(1-\gamma_{jtk}) \quad \forall j \in N; \forall t \in T; k \in K \quad (5.7)$$

$$-(h_{jt'k} - 1) \leq M\gamma_{jtk} \quad \forall t' \geq t \text{ and } t' \leq (t+p_j-1) \quad (5.8)$$

$$C_{jk} + p_i \leq C_{ik} + M(1-\delta_{jik}) \quad \forall j, i \in N; k \in K \text{ and } j < i; k < 1 \quad (5.9)$$

$$C_{ik} + p_j \leq C_{jk} + M\delta_{jik} \quad \forall j, i \in N; k \in K \text{ and } j < i; k < 1 \quad (5.10)$$

$$L_k \geq \sum_{j=1}^n C_{jk} - \frac{\sum_{k=1}^k \sum_{j=1}^n C_{kj}}{|K|} \quad \forall k \in K \quad (5.11)$$

$$-L_k \leq \sum_{j=1}^n C_{jk} - \frac{\sum_{k=1}^k \sum_{j=1}^n C_{kj}}{|K|} \quad \forall k \in K \quad (5.12)$$

$$C_{jk} \leq \max(t) \quad \forall j \in N; k \in K \quad (5.13)$$

$$C_{jk} \geq 0 \quad \forall j \in N; k \in K \quad (5.14)$$

$$x_{jtk} \in \{0, 1\} \quad \forall j \in N; k \in K; t \in T \quad (5.15)$$

$$\gamma_{jtk} \in \{0, 1\} \quad \forall j \in N; k \in K; t \in T \quad (5.16)$$

$$h_{jt'k} \in \{0, 1\} \quad \forall j \in N; k \in K; t \in T \quad (5.17)$$

$$\delta_{jik} \in \{0, 1\} \quad \forall j, i \in N; k \in K \quad (5.18)$$

Equations (5.1), (5.2), and (5.3) are the objectives of the optimization problem. The first objective is to minimize the total completion time, the second objective is to minimize the energy cost, and the third objective is to minimize the load balance in a parallel machine setting. The assignment constraints (5.4) ensure that each job is started exactly once on a machine. The capacity constraint (5.5) states that each machine can handle at most only one job during that time interval.

Constraint (5.6) defines the completion time of each job. Constraint set (5.7) states that if the job  $j$  starts at exactly one particular time  $t$ , then constraint set (5.8) enforces that the time ( $t'$ ) between the start time ( $t$ ) and the end time of job  $j$  ( $t + p_j - 1$ ) can be processed at any time over the planning horizon (i.e. the whole processing of job  $j$  is satisfied). Constraint sets (5.9) and (5.10) are disjunctive constraints which enforce that either job  $j$  is processed before job  $i$  or job  $i$  is processed before job  $j$  for any pair of jobs on a given machine. Constraint sets (5.11) and (5.12) define the deviation from a balance line as the gap between the total completion time on a machine and the average total completion time on all machines ( $\sum_j C_{jk} - ((\sum_k \sum_j C_{jk})/|K|)$ ). Constraint set (5.13) defines the boundary of completion time over the planning horizon  $T$ . Furthermore, constraint set (5.14) is the non-negativity constraint and sets (5.15 – 5.18) are integrality constraints. In this formulation, the value of big  $M$  is a large number that is equal to the sum of the processing times of all jobs.

The MMIP-MO-TE-TC-LB-PM has  $4|N||K| + |T||K| + 3|K| + |N| + |T|$  variables and  $|N| + 2|T| + |K||T| + 4|N||K| + |N||K||T| + |K| \sum_{j=1}^N (p_j(|T| - p_j + 1) + p_j(\frac{p_j-1}{2}))$  constraints for a problem of size  $|T|$ ,  $|N|$  and  $|K|$ , where  $|T| \geq \sum_j^n p_j$ . Therefore if either  $|T|$  or the number of jobs increases, then the size of the problem increases significantly. This multiple-objective scheduling and pricing problem can be solved exactly using multiobjective programming solution techniques such as WSM,  $\epsilon$ -constraints method, goal programming, multi-level programming, or metaheuristic methods such as GA, GRASP, simulating annealing, and Tabu search [24].

In the next section, we will utilize the  $\epsilon$ -constraint method to illustrate the mathematical model and gain some insight into the problem. Then we will propose two efficient multiobjective metaheuristics to solve larger-sized problems in a reasonable amount of time.

### 5.3 Using $\varepsilon$ -constraint Method to Solve MIP-MO-TE-TC-LB-PM Problem

In practical, the  $\varepsilon$ -constraint method is the most widely used in terms of generation (a posteriori) optimization. In this method, it optimizes one of the objective functions while transforming the other objective function into constraints with an upper bound of  $\varepsilon$ , where  $\varepsilon$  is a parameter that varies over iterations as shown below [25]:

$$\begin{array}{ll} \text{Min} & f_1(x) \\ \text{st} & \\ & f_2(x) \leq \varepsilon_2 \\ & f_3(x) \leq \varepsilon_3 \\ & x \in S \end{array}$$

where  $x$  is the vector of the decision variables,  $f_1(x), f_2(x), f_3(x)$  are the objective functions,  $\varepsilon_2, \varepsilon_3$  are on the right hand sides of the constrained objective functions, and  $S$  is the feasible region for the objectives. By solving the problem for different values of  $\varepsilon$ , the Pareto front (i.e., efficient solutions) of the problem are obtained. Mavrotas [25] proposes a novel version of the  $\varepsilon$ -constraint method that avoids the production of weak Pareto optimal solutions and accelerates the solution process by avoiding redundant iterations. In further research, Mavrotas and Florios [26] use the  $\varepsilon$ -constraint method for the generation of the Pareto-optimal solutions (i.e., produce the exact and complete Pareto set, which means all possible Pareto-optimal solutions) in multiobjective mathematical programming problems with discrete variables. Using the  $\varepsilon$ -constraint method, Salari et al. [27] develop a box algorithm that sequentially generates weak Pareto-optimal points using the  $\varepsilon$ -constraint method. Brown et al. [28] propose a multiobjective security game model and develop iterative- $\varepsilon$ -constraints for generating the Pareto front. As a result, in our research, we use the iterative- $\varepsilon$ -constraints for generating the exact Pareto front.

In our implementation, the exact Pareto-optimal front can be found by solving a sequence of constrained single-objective optimization problems. First, the payoff table is calculated by using lexicographic optimization of the objective functions. In this approach, the first objective function is optimized, and then, using a lexicographic optimization of the objective functions, the second possible alternative objective function is optimized, and so on. The  $f_1(x)$  function is selected to be optimized, obtaining  $\min f_1(x) = z_1^*$ . Then,  $f_2(x)$  function is selected to be optimized by adding the constraint  $f_1(x) = z_1^*$  in order to keep the optimal solution of the first optimization. Subsequently,  $f_3(x)$  function is selected to be optimized by adding the constraints  $f_1(x) = z_1^*$  and  $f_2(x) = z_2^*$  in order to keep the previous optimal solutions. In conclusion, by using lexicographical optimization, results will be more meaningful and more adequately describe the Pareto-optimal front [26].

After calculating the payoff table, we discretize the objective space to equal intervals utilized as values of  $\varepsilon$  in the  $\varepsilon$ -constraint method. As a result, the multiobjective mathematical problem is transferred to the  $\varepsilon$ -constraint equivalent, which is on the right hand side ( $\varepsilon_2, \varepsilon_3$ ) providing the Pareto-optimal solutions by varying  $\varepsilon_2$  and  $\varepsilon_3$ . Also, from the payoff table, the upper bounds (extreme objective vectors) are used to constitute an  $M$ -dimensional linear hyper-plane (see Figure 5.1). Finally, the Pareto-optimal solutions are simply mapped onto the above-constrained hyper-plane using a parametric problem on  $\varepsilon_2$  and  $\varepsilon_3$ , and so on. For example, solving the problem for different value of  $\varepsilon$  will generate a finite number of Pareto-optimal solutions as target.

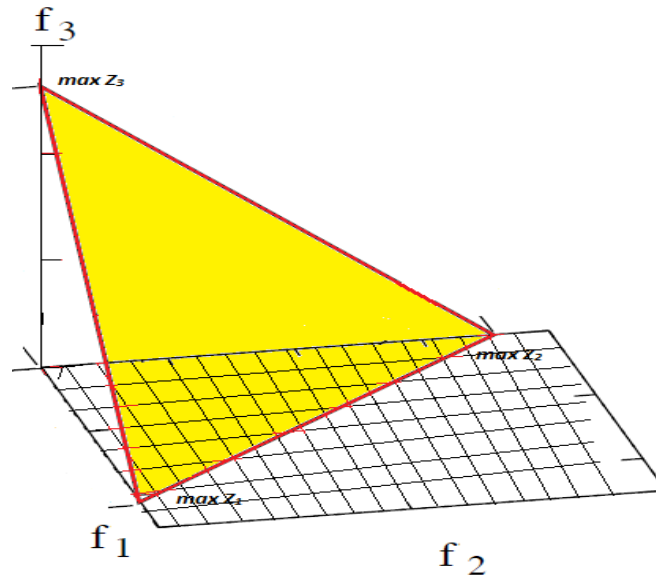


Figure 5.1: Illustration of extreme points for three-objective problem

In the next section we propose the multiobjective metaheuristic approaches (i.e., GRASP and GA-DRC) to determine an approximate Pareto front for a large-sized problem in a reasonable amount of time in order to schedule and reschedule jobs when there are disruptions on the shop floor or changes in the manufacturing environment. Therefore, metaheuristics must be designed instead of employing exact algorithms. The goal of metaheuristics in a multiobjective optimization is to find an approximation of the Pareto-optimal front in a timely fashion (i.e., the non-dominated solutions to the problem).

#### 5.4 Solution to MIP Model Using a Multiobjective GRASP and GA-DRC Metaheuristics

After modeling the MMIP-MO-TE-TC-LB-PM problem, the multiobjective GRASP and GA-DRC metaheuristic algorithms are proposed in order to obtain an approximate Pareto front. In these metaheuristics, a specific neighborhood solution is defined and searched in order to obtain either the optimal solution in the neighborhood solution or a solution that satisfies all three objectives. The algorithms are also very efficient when rescheduling is needed due to changes in the manufacturing environment.

### 5.4.1 The Proposed Multiobjective GRASP Algorithm

GRASP is a powerful search technique in metaheuristic approaches [29], being successfully implemented for solving different single-objective combinatorial problems [30]. In GRASP, all iterations consist of two different phases: construction and local search. In the construction phase, a feasible solution is constructed using a greedy randomized algorithm. The local search phase is searched locally to improve the solution obtained in the first phase. This is accomplished by investigating its neighborhood until a local minimum is found (i.e., find a better solution according to the objective in a defined neighborhood). Those two phases are repeated until the algorithm reaches a stopping criterion. The neighborhood definition is very important to design GRASP algorithm in order to ensure the convergence of the local search to a local optimum. The best overall solution is kept as the final solution.

Arroyo and Souza [31] present a multiobjective GRASP for solving the permutation flow shop scheduling problem in order to minimize two and three objectives simultaneously. GRASP is also used to solve the multiobjective knapsack problem to minimize number of objectives [32]. In another application, GRASP is used in airline scheduling where a solution is needed quickly to avoid and minimize delay costs or flight cancellations due to schedule disruptions [33]. Also, GRASP is used to find a quick upper bound utilizes a branch-and-bound algorithm [34]. Binato et al. [35] use GRASP for solving the job-shop scheduling problem in order to minimize the total completion time, and they have shown that the algorithm is competitive compared to other heuristic solving the same problem.

In order to solve the MMIP-MO-TE-TC-LB-PM problem, the proposed GRASP algorithm can be seen in Figure 5.2. The first phase is used to construct solutions of one element (i.e., job) at a time. At all iterations, the candidate's job is selected from a restricted candidates list (RCL), and

is then added to the current solution. The RCL is created, consisting of a list of non-dominated jobs, and then some dominated jobs provide randomness to the construction phase.

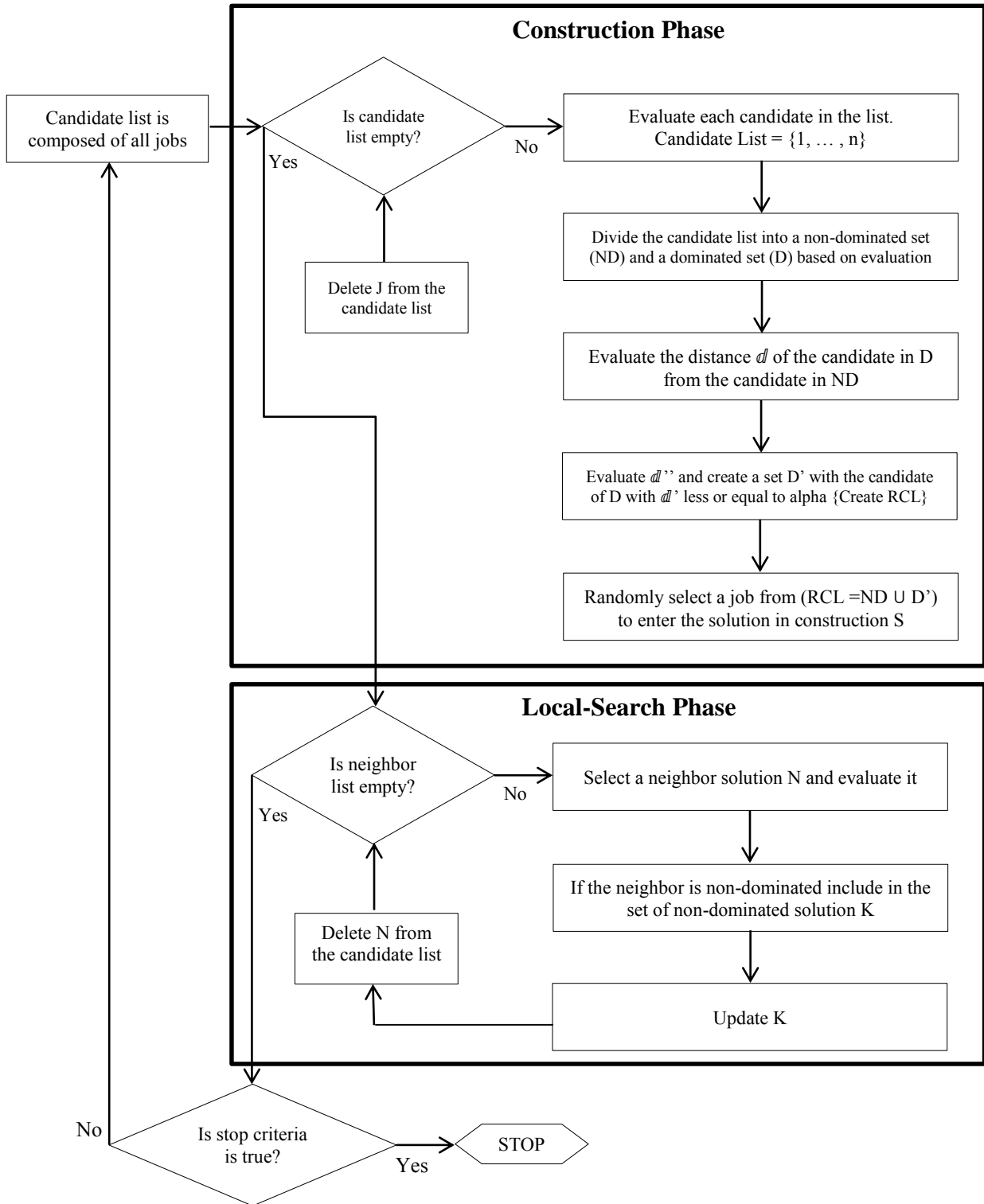


Figure 5.2: Multiobjective GRASP algorithm

### 5.4.1.1 Construction Phase

In the typical implementation of the construction phase, each candidate job must be evaluated by adding this job to the sequence of the jobs which is a partial solution that provides the order of jobs. Now, when a partial of a solution is given, the order of jobs is known. As a result, MMIP-MO-TE-TC-LB-PM reduces to the following multiobjective mixed-integer mathematical model (MMIP-MO-TE-TC-LB-PM-R) where  $R$  refers to reduce model. In the reduced model, the disjunctive constraints (5.9 and 5.10) are removed, and the constraint set (5.27) is added to ensure that two jobs cannot be processed at the same machine at the same time.

$$\text{Minimize } \sum_{k=1}^k \sum_{j=1}^n C_{jk} \quad (5.19)$$

$$\text{Minimize } \sum_{k=1}^k \sum_{j=1}^n \sum_{t \in T} E_t h_{jtk} \quad (5.20)$$

$$\text{Minimize } \sum_{k=1}^k L_k \quad (5.21)$$

$$\sum_{k=1}^k \sum_{t=1}^{|T|-p_j+1} x_{jtk} = 1 \quad \forall j \in N \quad (5.22)$$

$$\sum_{j=1}^n \sum_{t'=\max(0,t-p_j+1)}^t x_{jtk} \leq 1 \quad \forall t \in T; k \in K \quad (5.23)$$

$$C_{jk} \geq \sum_{t=1}^{|T|-p_j+1} (t-1+p_j) x_{jtk} \quad \forall j \in N; k \in K \quad (5.24)$$

$$x_{jtk} \leq M(1-\gamma_{jtk}) \quad \forall j \in N; \forall t \in T; k \in K \quad (5.25)$$

$$-(h_{j't'k} - 1) \leq M\gamma_{jtk} \quad \forall t' \geq t \text{ and } t' \leq (t+p_j-1) \quad (5.26)$$

$$C_{ik} + p_i \geq C_{jk} \quad \forall j, i \in N; k \in K \text{ and } j < i; k < 1 \quad (5.27)$$

$$L_k \geq \sum_{j=1}^n C_{jk} - \frac{\sum_{k=1}^k \sum_{j=1}^n C_{kj}}{|K|} \quad \forall k \in K \text{ and} \quad (5.28)$$

$$-L_k \leq \sum_{j=1}^n C_{jk} - \frac{\sum_{k=1}^k \sum_{j=1}^n C_{kj}}{|K|} \quad \forall k \in K \text{ and} \quad (5.29)$$

$$C_{jk} \leq \max(t) \quad \forall j \in N; k \in K \quad (5.30)$$

$$C_{jk} \geq 0 \quad \forall j \in N; k \in K \quad (5.31)$$

$$x_{jtk} \in \{0, 1\} \quad \forall j \in N; k \in K; t \in T \quad (5.32)$$

$$\gamma_{jtk} \in \{0, 1\} \quad \forall j \in N; k \in K; t \in T \quad (5.33)$$

$$h_{j't'k} \in \{0, 1\} \quad \forall j \in N; k \in K; t \in T \quad (5.34)$$

$$\delta_{jik} \in \{0, 1\} \quad \forall j, i \in N; k \in K \quad (5.35)$$

MMIP-MO-TE-TC-LB-PM-R has  $2|N||K| + |T||K| + 3|K| + 2|N| + |T|$  variables and  $2|N| + 2|T| + |K||T| + 2|N||K| + |N||K||T| + |K| \sum_{j=1}^N (p_j(|T| - p_j + 1) + p_j(\frac{p_j-1}{2}))$  constraints for a problem of size  $T$  and  $|N|$ , where  $T \geq \sum_j^n p_j$ . Solving the MMIP-MO-TE-TC-LB-PM-R will provide the set of non-dominated solutions corresponding to the known order of jobs. When the order of jobs is fixed, the number of variables and constraints are reduced by  $2|N||K|$ , respectively. Note that in this formulation,  $C_{jk}$  and  $p_j$  indicate completion time and process time of the job in position  $j$ , respectively, in the sequence defining the order of the jobs. Note that, the MMIP-MO-TE-TC-LB-PM-R can be solved using the  $\varepsilon$ -constraint method as well.

For any partial solution, a set of non-dominated solutions can be obtained. The solutions are compared, and the non-dominated solutions are separated from the dominated solutions. The smallest distance between non-dominated solutions and a new (candidate) solution is used to do the comparison and separation from the dominated solutions. For example, in Figure 5.3, the distance,  $d$ , between a candidate solution  $D$  and the non-dominated solutions  $(A, B, C)$  is the distance,  $d$ , between  $D$  and  $B$ . Note that the relative information distance of each dominated candidate solution is divided by the maximum distance in order to normalize the distance.

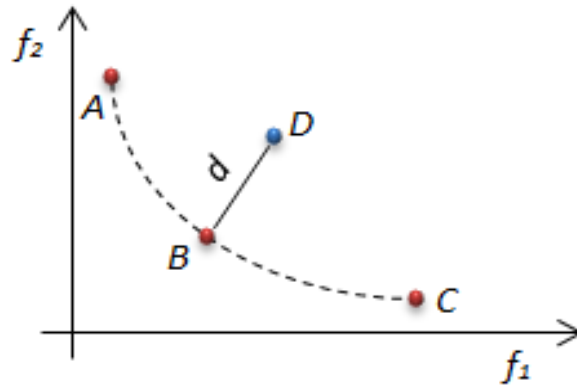


Figure 5.3: The distance between a solution and the non-dominated solutions

In greedy algorithm, a candidate job resulting to a non-dominated solution is chosen to be added to the current solution. However, there exists the need to add some randomness into the algorithm; therefore we add a candidate job from the dominated solution which is " $\alpha$  percent" distant from the non-dominated solutions. The goal of the parameter  $\alpha$  is to adjust the randomness of the GRASP algorithm: if  $\alpha$  is equal to one, then the algorithm is totally random, and if  $\alpha$  is equal to zero, then it is totally greedy.

The RCL consists of a list of the candidates resulting from a non-dominated solution and some candidates resulting from dominated solutions that are at a maximum relative distance of  $\alpha$  from the the non-dominated solutions, which provides some randomness in the construction phase. Next, a random candidate job is selected from the RCL to enter the current solution. The process is repeated until the current solution obtains a complete solution with the sequence of the jobs.

#### **5.4.1.2 Local Search Procedure**

In the second phase of the GRASP algorithm, a local search procedure starts with the best solution obtained by the construction phase. All of the solutions in the defined neighborhood of the constructed solution  $D$  are evaluated. As a result, if the new solution is a non-dominated solution, then it is added to the Pareto-optimal set. Here, all iterations obtain an approximate Pareto front.

The local search procedure for the multiobjective problem is similar to the single-objective problem. The only difference is that instead of keeping the best solution, a set of non-dominated solutions is obtained. Armentano and de Araujo [36] propose a GRASP algorithm to solve a single-objective total tardiness problem and use two different definitions of neighborhoods: (1) the exchange of two jobs and (2) the insertion of a job between consecutive jobs. They found that the size of each neighborhood can be large, especially if the size of problem is large. In this case, each

neighbor is evaluated using a series of linear programs which are too time-consuming. In order to avoid this problem and solve the problem in a timely fashion, a neighborhood with a similar size has to be defined: new solutions are obtained by swapping the position of the two consecutive jobs, which results in  $(n - 1)$  neighbours.

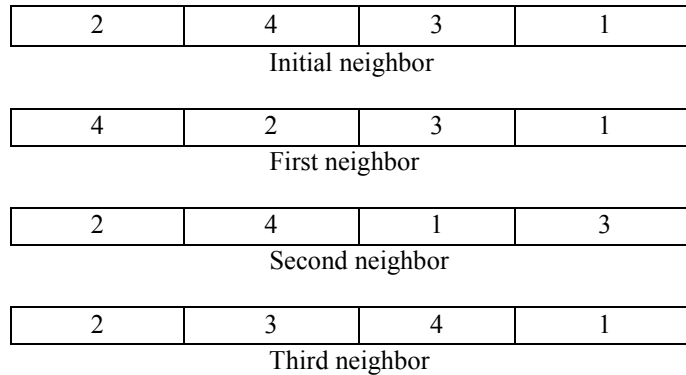


Figure 5.4: Illustration of job neighborhood

In Figure 5.4, the initial solution has three neighbors which consist of swapping two consecutive jobs such as job 2 with job 4, job 3 with job 1, and job 4 with job 3.

### 5.4.2 The Proposed Multiobjective GA-DRC Algorithm

GA-DRC is inspired from the NSGA-II which was developed by Deb et al. [37] and has been proven to produce high quality results on test problems. NSGA-II is a popular metaheuristic algorithm for solving the multiobjective optimization problems. NSGA-II is an elitist algorithm, meaning that it obtains high quality individual solutions that survive from generation to generation (i.e., it keeps the best possible individuals from each population). The difference between the NSGA-II and traditional GAs is that NSGA-II does not require the choosing of a sharing parameter whereas it uses the Pareto-dominance concept of solutions to guide the search process, and returns the Pareto-optimal set as the best results.

#### 5.4.2.1 GA-DRC Overview

GA-DRC has three main operators: crossover, mutation and selection operators. In the crossover operator, two chromosomes (parents) are selected to generate new chromosomes, called

offspring. Mutation operator results in establishing a new solution from the current chromosome by changing the order of the two jobs. Crossover and mutation operators are used to generate new solutions (i.e., generate additional child chromosome) within the search space based on the variation of existing ones. Selection operator is where the algorithm decides which solutions will carry over the next generation. The complete cycle of these operators corresponds to one iteration.

To evaluate the quality of the solutions (fitness function) in the current population, GA-DRC uses two main procedures: the non-dominated sorting procedure and the crowding-distance sorting procedure. The non-dominated sorting procedure ranks the solution in different Pareto fronts, while the crowding-sorting procedure measures the density of individuals in the solution space (i.e., it calculates the dispersion of the solution in each front and preserves the diversification of the algorithm). At each generation, these two operators form the Pareto front. The flow chart of the GA-DRC is shown in Figure 5.5.

#### **5.4.2.2 Representation (Coding)**

In our implementation, a chromosome provides information about the order of jobs: a problem with  $n$  jobs is represented by a chromosome of  $n$  genes (i.e., an array with  $n$  cells). For example, a chromosome represents a four-job problem [4 2 1 3] will process all jobs in the order of 4, 2, 1 and 3.

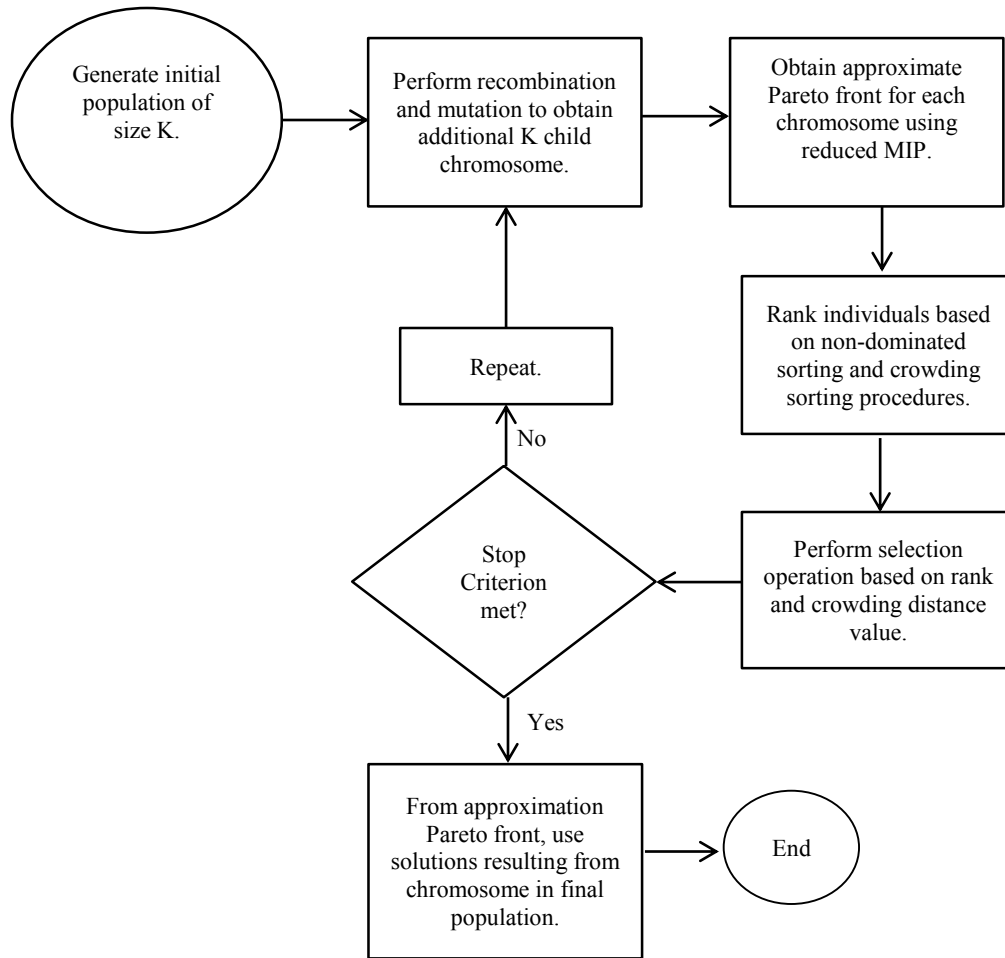


Figure 5.5: Flow chart of GA-DRC

### 5.4.2.3 Crossover

At the beginning of the GA-DRC, an initial population size is randomly generated. Then, crossover operator occurs and mutation operator is used to generate new chromosomes, called offspring. In the crossover operator, two chromosomes are selected as input to generate two child chromosomes by randomly selecting a crossover point (see Figure 5.6).

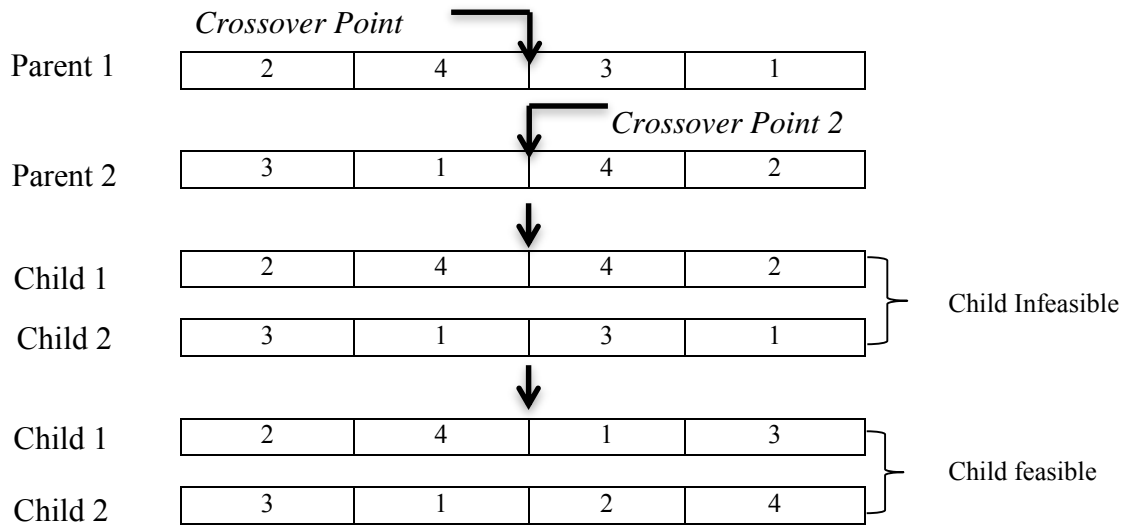


Figure 5.6: Illustration of the crossover operator

In Figure 5.6, the crossover operation is utilized on two chromosomes: [2 4 3 1] and [3 1 4 2] by having the crossover points between the second and third genes. Each child receives the first part of its chromosome from one of the parents and the second part of the chromosome from the other parent. Sometimes, the crossover operation generates an infeasible child. For example, in Figure 5.6, child [3 1 3 1] shows that job 3 and job 1 are scheduled twice, whereas job 2 and job 4 are not yet scheduled. In this case, when there is an infeasible child, the repeated jobs are replaced with jobs that have not appeared in the chromosome in a lexicographical order (i.e., the repeated jobs are replaced in lexicographical order by jobs that have not yet appeared in the chromosomes). As a result, the infeasible child [3 1 3 1] corrects as [3 1 2 4].

#### 5.4.2.4 Mutation

After the crossover operator is performed, the mutation operator selects a single chromosome as input and generates a new one by randomly exchanging the order of two jobs (see Figure 5.7). The goal of the mutation operation is to avoid the population chromosomes from being too similar to each other. In Figure 5.7, the mutation operator exchanges the location of jobs 2 and 3.

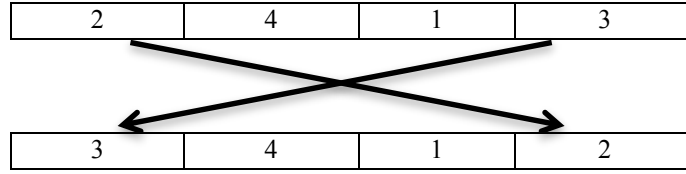


Figure 5.7: Illustration of the mutation operator

#### 5.4.2.5 Evaluation of Chromosome: Obtaining Approximate Pareto Front

After the crossover and mutation operations are performed, one chromosome has the information about the sequence of the jobs. In studying such problems, one needs to determine the starting time of each job and the assignment of the jobs to the machines. When the sequence of the jobs is known, the MMIP-MO-TE-TC-LB-PM-R is used to evaluate the chromosome. As a result, the MMIP-MO-TE-TC-LB-PM-R generates an approximate Pareto front for each chromosome in the current population. Note that the MMIP-MO-TE-TC-LB-PM-R can be solved using the  $\varepsilon$ -constraint method.

#### 5.4.2.6 Evolution of Fitness Function

After the approximate Pareto front (non-dominated solutions) is obtained, then all individuals of certain population are sorted into each front based on non-domination and crowding distance (see Figure 5.8). The non-dominated sorting procedure, level 1 in Figure 5.8, contains all the dominant individuals within the population. If the individuals in level 1 are not considered, then the second set of dominant individuals creates level 2, and so on. These levels represent the rank, which is the most important factor of its fitness. As a result, the individual with the lower rank is preferable because it represents the strong non-dominated solutions.

Next, the crowding distance is assigned after the non-dominated sorting procedure is performed. In Figure 5.8, the perimeter of the cuboid is estimated by using the nearest neighbor as the vertices [37]. The crowding distance is a measure of how close an individual is to its neighbors, and it guarantees the diversity of the population. Note that, for an individual, the crowding distance

is defined as the sum of the normalized distance between its right and left neighbors for each objective function. For the first and last individuals (extreme solutions) there is a crowding-distance equal to infinity.

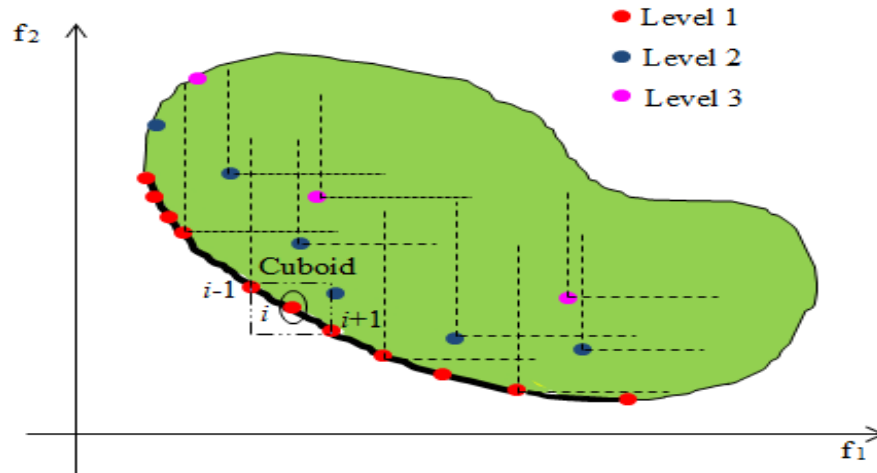


Figure 5.8: Non-dominated levels and computation of crowding distance

#### 5.4.2.7 Selection Operator

After all the individuals of the initial population are sorted using the non-dominated sorting and crowding distance sorting procedures, the algorithm then applies selection, crossover, and mutation operators to create the first offspring set. Here, the selection operator is based on two comparison rules: (1) the lower the rank to which the individual belongs, the better the solution, and (2) if the individuals have the same ranking value, then an individual with greater distance has a better solution because the area to which that individual belongs is less crowded. Note that the algorithm is repeated until it reaches a stopping criterion. Here, when the algorithm hits the maximum number of generations, it stops. In this stage, the approximate Pareto-optimal front is obtained. This process can be described as shown in Figure 5.9. Note that the chromosomes with lower rank have better solutions, if they have the same Pareto front (i.e., rank), then the

chromosome with greater crowding distance has a better solution and surviving in the next generation.

*Step 1:* Sort individuals based on non-domination rank  $rank_{\iota}$  and with crowding distance assigned  $Front_{\#}(Dis_{\cdot\iota})$ , where  $\iota$  corresponds to the distance  $Dis_{\cdot\iota}$  of the  $\iota^{th}$  solution in  $Front_{\#}$ . Note that each solution  $\iota$  belongs to one chromosome  $\psi$ .

*Step 2:* Utilize a *crowded-comparison-operator* ( $<_{\#_{\iota\xi}}$ ), where  $\#_{\iota\xi}$  is the number of solutions  $\iota$  in generation  $\xi$ , based on: (1)  $rank_{\iota}$  and (2)  $Front_{\#}(Dis_{\cdot\iota})$ .

For example,  $\iota_{\#4} <_{\#_{\iota\xi}} \iota_{\#5}$  if

- $rank_{\iota_{\#4}} < rank_{\iota_{\#5}}$
- Or if  $\iota_{\#4}$  and  $\iota_{\#5}$  belong to the same  $Front_{\#}$  then  $Front_{\#}(Dis_{\cdot\iota_{\#4}}) > Front_{\#}(Dis_{\cdot\iota_{\#5}})$  i.e. the crowding distance should be more.

*Step 3:* Utilize a binary tournament selection with *crowded-comparison-operator* from step 2 until chromosomes are selected.

Figure 5.9: Process of selection of chromosomes for generating the next population

## 5.5 Experiment Design and Evaluation

The GAMS is used to model the MMIP-MO-TE-TC-LB-PM problem, and the CPLEX 12.5 [38] is utilized to solve the problem. To evaluate the performance and effectiveness of the proposed GRASP and GA-DRC algorithms, the MATLAB “R2010a” language is used to code the algorithms and IBM ILOG CPLEX 12.5 via MATLAB is utilized to solve the problem. The computational experiments are performed on an Intel i5 2.27GHz machine with 4GB of memory and a Windows 7 operation system.

### 5.5.1 Generation of Experiments Data

The experiment’s problems are constructed by using varying combinations of the four parameters: number of jobs ( $n$ ), planning horizon ( $T$ ), processing times ( $p_j$ ), and electricity prices (i.e., time-of-use tariffs) ( $E_t$ ). During our analysis, we observe that the number of jobs  $|N|$  and the size of the planning horizon  $|T|$  have a major influence on the performance of the algorithm since

the larger the size of the problem the more CPU time is required. For this reason, we test the quality of algorithmic parameters as well.

In the experimentation, problems with 5, 10, 20, 30, 40, and 50 jobs are generated. For each job, an integer processing time is obtained randomly from a uniform distribution between 1 and 10. An integer planning horizon  $|T|$  is computed as  $|T| = \sum_j^n p_j / \text{PHF}$ , where PHF is the planning horizon factor, which takes two values: 0.50 for a longer planning horizon and 0.75 for a shorter planning horizon. The electricity prices,  $E_t$ , are generated from three uniform distributions [1, 5], [6, 10], [11, 16] over three different periods of time to create low, moderate, and high variations (see Figure 5.10).

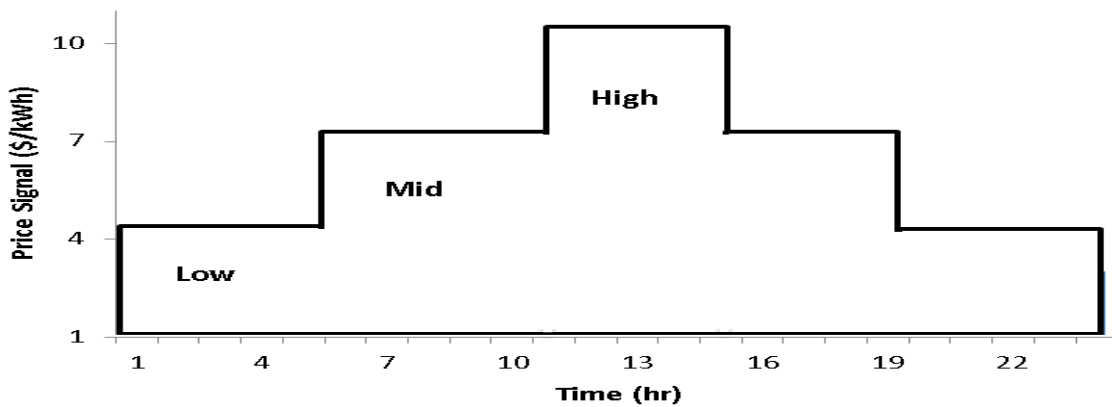


Figure 5.10: Prices signal (\$/kWh) vs. time (hr.) (i.e., example of time-of-use tariffs)

## 5.6 Solving MMIP-MO-TE-TC-LB-PM Using $\epsilon$ -constraint Method

A set of 12 test problems are randomly generated to gain insight into the MMIP-MO-TE-TC-LB-PM multiobjective optimization problem (see Table 5.1). We generate 36 instances (i.e., three different instances for each test), solve every instance individually, and measure the average time in minutes.

Using the  $\epsilon$ -constraint method, the MMIP-MO-TE-TC-LB-PM problem translates into a sequence of constrained single-objective optimization problems. One objective is selected as the

primary objective to be minimized while the upper-bound constraints are added for the other secondary objectives as

$$\begin{aligned}
 & \text{Min} \quad \sum_{k=1}^k \sum_{j=1}^n C_{jk} \\
 & \text{st} \\
 & \quad \sum_{k=1}^k \sum_{j=1}^n \sum_{t \in T} E_t h_{jtk} \leq \varepsilon_2 \\
 & \quad \sum_{k=1}^k L_k \leq \varepsilon_3 \\
 & \quad x \in \mathcal{S}
 \end{aligned}$$

By varying the constraints (i.e., parameters on the right hand side of the constrained objective functions  $\varepsilon$ ), the non-dominated solutions of the problem are obtained. Note that we use the  $\varepsilon$ -constraint method to solve the subproblem in the GRASP and GA-DRC algorithms in order to obtain the approximate Pareto-optimal front as well.

Table 5.1 shows the results of the  $\varepsilon$ -constraint method for each problem with the amount of execution time and the number of non-dominated solutions obtained. It is certain that the larger the size of the problem the more CPU time it takes to solve, i.e., when the number of jobs or the planning horizon increases, the  $\varepsilon$ -constraint method requires a significant amount of CPU time to determine the Pareto-optimal front, as the search space for the problem increases in terms of variables and constraints. For example, for  $n = 30$  with  $PHF = 0.75$  the Pareto-optimal front is solved in less than 1,152 minutes, whereas for  $n = 30$  with  $PHF = 0.5$  it is exponentially increased to more than 1,440 minutes, i.e., more than one day to solve the problem. In conclusion, we see that there is a need to study and apply the metaheuristics in order to obtain an approximate Pareto front in a timely fashion.

Table 5.1: Features of Multiobjective Optimization Instances and Results of  $\varepsilon$ -constraint Method

$n$	$PHF$	Number of			
		Execution Time (minutes)		Non-dominated Points	
		Min	Max	Min	Max
5	0.75	180	246	9	13
5	0.5	228	282	13	16
10	0.75	270	357	16	19
10	0.5	498	593	21	23
20	0.75	648	837	18	25
20	0.5	693	961	23	24
30	0.75	815	1,152	17	22
30	0.5	855	>1,440	19	26
40	0.75	991	>1,440	23	29
40	0.5	1,055	>1,440	28	31
50	0.75	1,112	>1,440	22	29
50	0.5	1,220	>1,440	27	31

### 5.7 Parameter Fine-Tuning

In this section, extensive numerical experiments are performed to fine-tune the GRASP and GA-DRC algorithms and to analyze and observe on the different parameter effects and performance measure in comparing alternative Pareto fronts. In practical, each test is run with the same parameters five times, and the average measure of performance is taken. The comparison between the different Pareto fronts is based on the performance measure of the CPU time and MID, which determines the nearness among the Pareto solutions and ideal point (0,0). Since minimization objectives are considered in a result, the Pareto front with a lower value has a better performance. MID is defined as

$$MID = \frac{\sum_{q=1}^Q \sqrt{x_q^2 + y_q^2 + z_q^2}}{Q}$$

Where  $x, y,$  and  $z$  are the values of the three objectives for a non-dominated solution in a Pareto front, and  $Q$  is the number of solutions in the front.

In the GRASP algorithm, the goal is to fine-tune the  $\alpha$  parameter. Note that the  $\alpha$  parameter allows for the inclusion of randomness into the algorithm; however, when  $\alpha$  is equal to one, then the algorithm is totally random, and when  $\alpha$  is equal to zero, then the algorithm is totally greedy. As a result, the  $\alpha$  parameter takes values between 0 and 1 to compare between a totally greedy and a totally random algorithm. The algorithm is tested with increasing  $\alpha$  values from 0 to 1 in increments of 0.25, *PHF* values of 0.75 and 0.5, and problem sizes of 10, 20, 30 jobs. Note that the amount of time each experiment runs depends upon the number of iterations that the GRASP algorithm runs.

Table 5.2 shows that the  $\alpha$  value is fine-tune at 0.25 when *PHF* = 0.75, whereas Table 5.3 shows that the  $\alpha$  value is fine-tune at 0 when *PHF* = 0.5. As a result, we can conclude that, as the *PHF* increases, the optimal value for the  $\alpha$  parameter decreases from 0.25 to 0. In conclusion, a totally greedy GRASP algorithm will perform a better solution when the planning horizon is longer.

Table 5.2: Performance Depending on  $\alpha$  Parameter and *PHF* = 0.75

$\alpha$	$n = 10$	CPU (min)	$n = 20$	CPU (min)	$n = 30$	CPU (min)
<b>0</b>	43.7	21.4	55.3	26.9	64.3	30.3
<b>0.25</b>	39.2	21.7	51.7	27.3	63.8	31.2
<b>0.5</b>	41.7	22.4	53.1	27.9	66.4	31.7
<b>0.75</b>	42.5	22.9	53.7	28.4	65.7	32.4
<b>1</b>	43.1	23.5	52.3	28.7	65.3	32.9

Table 5.3: Performance Depending on  $\alpha$  Parameter and *PHF* = 0.50

$\alpha$	$n = 10$	CPU (min)	$n = 20$	CPU (min)	$n = 30$	CPU (min)
<b>0</b>	46.3	23.3	74.7	29	68.7	34.3
<b>0.25</b>	46.8	23.8	78.2	29.7	69.3	35.1
<b>0.5</b>	47.6	24.4	78.7	30.2	69.9	35.8
<b>0.75</b>	48.2	25.3	77.1	30.6	72.6	36.4
<b>1</b>	48.6	25.9	76.4	31.2	70.1	36.9

In the GA-DRC algorithms, the performance is affected by the factors such as crossover rate, generation size, and running time (i.e., the number of the generation performed). Therefore, the goal is to fine-tune four parameters: the crossover rate, mutation rate, generation size, and number of generations. Selecting good parameters to run the GA-DRC can play a vital role in determining an approximate Pareto front in a timely fashion [24]. In the following test problems, we will apply the GA-DRC to examples with 10, 20, and 30 jobs. Initially, the population size is assumed to be 20, the number of generation is 20, and the crossover and mutation rates are 75% and 25% respectively. Each test is run twice, and the average is taken as the measure of performance. Based on these values, we fine-tune the parameters of the GA one at a time, because considering all possible combinations of parameters may take significant amount of time.

Using the measure of performance MID, results are compared using different rates of crossover from 0 to 1 in increments of 0.25, *PHF* values of 0.75 and 0.5, and problem sizes of 10, 20, 30 jobs, since the fine-tune rates of crossover may vary with the size of time and the number of jobs considered. Note that a Pareto front with a lower MID value performs better. Tables 5.4 and 5.5 show that a crossover rate of 0.75 leads to good results for the problems with different jobs and with different *PHF* values. From the results, we can observe that there is a high convergence between the solutions when the crossover rate is between 0.75 and 1. In other words, a high probability crossover will achieve a better solution when the problem size is increased.

Table 5.4: Performance Depending on Crossover Rates and *PHF* = 0.75

<b>Crossover rate</b>	<b><i>n</i> = 10</b>	<b>CPU (min)</b>	<b><i>n</i> = 20</b>	<b>CPU (min)</b>	<b><i>n</i> = 30</b>	<b>CPU (min)</b>
<b>0</b>	46.2	21.3	53.2	25.6	65.4	26.8
<b>0.25</b>	44.2	20.6	51.6	25.3	63.5	26.3
<b>0.5</b>	43.8	20.1	51.2	24.5	61.3	25.7
<b>0.75</b>	41.6	20.4	48.6	24.9	58.4	25.3
<b>1</b>	42.8	20.9	49.7	24.5	60.1	26.3

Table 5.5: Performance Depending on Crossover Rates and PHF = 0.5

Crossover rate	$n = 10$	CPU (min)	$n = 20$	CPU (min)	$n = 30$	CPU (min)
<b>0</b>	46.3	24.7	64.8	29.2	73.2	34.6
<b>0.25</b>	46.8	25.3	63.7	29.8	71.4	34.9
<b>0.5</b>	44.7	25.9	63.4	30.4	66.5	35.1
<b>0.75</b>	38.5	26.5	58.5	31.1	59.7	35.6
<b>1</b>	42.5	27.2	60.2	31.6	63.8	36.1

Using the fine-tuned crossover rate of 0.75 while keeping all other parameters constant, the same type of experiment can be run to determine the fine-tune number of individuals in a generation as a function of the number of jobs to be processed. From the results, we observe that there is a trade-off between the generation size and the CPU time of an individual iteration. Tables 5.6 and 5.7 show that the number of generations of 20 yields reasonable quality results for the problems with different jobs and with different *PHF* values.

Table 5.6: Performance Depending on Number of Individuals in a Generation and PHF = 0.75

Number of individuals	$n = 10$	CPU (min)	$n = 20$	CPU (min)	$n = 30$	CPU (min)
<b>10</b>	77.3	14.2	73.6	15.4	85.5	15.8
<b>20</b>	76.4	19.6	88.7	24.3	90.7	25.7
<b>30</b>	83.9	36.1	84.4	41.6	80.4	42.1
<b>40</b>	85.8	52.9	71.3	57.4	92.3	59
<b>50</b>	88.2	70.1	91.9	74.3	95.6	76.2

Table 5.7: Performance Depending on Number of Individuals in a Generation and PHF = 0.5

Number of individuals	$n = 10$	CPU (min)	$n = 20$	CPU (min)	$n = 30$	CPU (min)
<b>10</b>	81.2	16.7	91.4	19.4	96.6	24.3
<b>20</b>	80.6	22.6	94.7	27.6	98.1	32.6
<b>30</b>	86.8	39.5	90.6	44.7	97.2	48.4
<b>40</b>	89.4	57.3	97.4	61.4	94.3	67.6
<b>50</b>	90.1	73.4	98.3	78.6	98.7	83.7

While keeping the crossover rate and the number of individuals in a generation at their fine-tune values, the number of generations varies in each run from 10 to 50 in increments of 10.

*PHF* values of 0.75 and 0.5. Tables 5.8 and 5.9 show that the best number of generations is 50, which is the maximum number allowable. In other words, if we keep increasing the number of generations, the better the solution becomes. A possible reason for this is the logic behind the GA, which states the more generations, the better the solution. However, since we are interested in the CPU time, the number of generation is limited to 20 generations.

Table 5.8: Performance Depending on Number of Generation and  $PHF = 0.75$

Number of generations	$n = 10$	CPU (min)	$n = 20$	CPU (min)	$n = 30$	CPU (min)
10	78.9	14.8	81.1	16.5	85.6	15.3
20	75.1	19.6	79.9	24.3	82.3	25.7
30	74.7	45.2	78.1	51	80.2	55
40	71.1	71.5	75.3	79.9	76.9	83.3
50	69.3	98.9	72.4	110.3	74.7	115.7

Table 5.9: Performance Depending on Number of Generations and  $PHF = 0.5$

Number of generations	$n = 10$	CPU (min)	$n = 20$	CPU (min)	$n = 30$	CPU (min)
10	92.8	16.8	98.2	18.7	99.6	19.3
20	92.2	23.4	97.9	27.3	98.1	33.4
30	90.3	53.6	96.3	57.6	97.7	63.1
40	88.7	83.4	93.4	87.9	94.6	96.7
50	86.9	118.1	88.8	123.7	92.4	136

Now, because considering all possible combinations of a parameter may take significant amount of time, we fine-tune the parameters of the GRASP and the GA-DRC one at a time. In the next section, we are going to use the optimized parameters and compare the performance of the proposed algorithms.

## 5.8 Comparison of GRASP, GA-DRC Algorithms and $\epsilon$ -constraint Method

Given the fine-tune set of parameters for the GRASP and GA-DRC algorithms, we compare the performance and quality of the proposed algorithms using the same parameters while varying the number of jobs. As a test, in this section, we discuss the state-of-the-art algorithms

based on problem sizes of 10, 20, 30, 40, 50 jobs and *PHF* values of 0.75 and 0.5 (see Table 5.10). Note that we apply the proposed algorithms on the same instances used to characterize the performance of the  $\varepsilon$ -constraint method. In addition, the comparison of the algorithms is based on the following three performance measurement tools:

- Number of Solutions on Pareto Front (No. Pareto): This tool counts the total number of non-dominated solutions that are achieved by an algorithm. The algorithm with a higher total number is preferred.
- Quality Metric (QM): this tool builds a new Pareto front after a set of non-dominated solutions from each algorithm is obtained for the same problem. Then a pairwise comparison is performed (i.e., the performance of one algorithm is the percentage of solutions in the new Pareto front). The algorithm with a higher percentage performs better.
- CPU time: this tool measures the computational time required to obtain the Pareto front (i.e., the time spent on the processor running the program's code). The algorithm with less time is desirable.

Aforementioned, the goal of the GRASP and GA-DRC is to obtain an approximate solution in a reasonable amount of computational time without any guarantee of optimality. Note that since the GRASP and GA-DRC are based on randomized iterations that use different random candidate jobs and chromosomes, they may not return the same results twice. Therefore, the best way to evaluate the quality of the GRASP and GA-DRC algorithms is by comparing their Pareto front with the Pareto-optimal front obtained by the  $\varepsilon$ -constraint method. Each test is run with the same parameters three times, and the average measure of performance is taken. Table 5.10 summarizes the results obtained by the applied algorithms and  $\varepsilon$ -constraint method.

Table 5.10: Pairwise Comparison of GRASP, GA-DRC Algorithms and  $\epsilon$ -constraint Method

<i>n</i>	<i>PHF</i>	GRASP			GA-DRC			$\epsilon$ -constraint		
		No. Pareto	QM (%)	CPU (min)	No. Pareto	QM (%)	CPU (min)	No. Pareto	QM (%)	CPU (min)
10	0.75	27	17	21	29	25	19.2	19	58	357
10	0.5	31	23	23.4	33	31	20.4	23	46	593
20	0.75	34	16	27	34	19	22.2	25	65	837
20	0.5	36	21	29.4	37	28	23.4	24	52	961
30	0.75	31	17	31.8	33	28	25.2	22	55	1,152
30	0.5	34	13	34.8	36	26	27.6	26	61	>1,440
40	0.75	34	18	39	36	27	29.4	29	55	>1,440
40	0.5	36	20	41.4	37	26	31.2	31	54	>1,440
50	0.75	32	22	46.8	34	31	33.6	29	47	>1,440
50	0.5	35	21	49.2	38	35	35.4	31	44	>1,440

Table 5.10 shows that in all 10 test problems, the  $\epsilon$ -constraint method is able to perform better solutions than GRASP and GA-DRC in terms of QM. A reason for this is that the  $\epsilon$ -constraint method characteristics that always generate the Pareto-optimal solutions while solving the multiobjective mathematical problems. Another possible reason for the  $\epsilon$ -constraint method performs the best is that the GRASP and GA-DRC are based on randomized iterations that use different random candidate jobs and chromosomes without any guarantee of optimality. In contrast, GRASP and GA-DRC have a better number of non-dominated solutions and CPU time compared with the  $\epsilon$ -constraint method. For this reason, we use GRASP and GA-DRC to determine an approximate Pareto front for large-sized problems in a good quality solution and a reasonable amount of time in order to schedule or reschedule jobs when there are disruptions on the shop floor or changes in the manufacturing environment.

The GA-DRC performs the best in all 10 test problems compared with GRASP. The GA-DRC performs the best because the GA-DRC has characteristics that allow it to maintain enough non-dominated solutions in the final GA population where the total number of non-dominated

solutions is slightly higher compared to the non-dominated solutions obtained in the final iteration of GRASP.

Figure 5.11 shows the combined Pareto-optimal front from the GRASP algorithm, the GA-DRC algorithm, and the  $\varepsilon$ -constraint method for the problem where  $n = 30$  and  $PHF = 0.5$ . It is clear from Figure 5.11 that the GA-DRC and GRASP have an identical local Pareto-optimal front from the exact solutions found in the  $\varepsilon$ -constraint method, as well as identical solutions from one another. For the same problem, the numbers of non-dominated solutions are similar, on average, 61% of the combined Pareto-optimal solutions are from the  $\varepsilon$ -constraint method, 26% of the solutions are from the GA-DRC, and 13% of the solutions are from the GRASP.

Table 5.10 shows that in all 10 test problems, GA-DRC is able to perform slightly better computationally in terms of CPU time for obtaining the approximate Pareto front in comparison to the GRASP. In conclusion, the GA-DRC outperforms the performance of the GRASP with respect to solution quality and also outperforms the  $\varepsilon$ -constraint method with respect to computational CPU time for obtaining the approximate Pareto front. More importantly, the GA-DRC and GRASP have been able to come closer (i.e., match solutions) to the  $\varepsilon$ -constraint method Pareto-optimal front (see Figure 5.11).

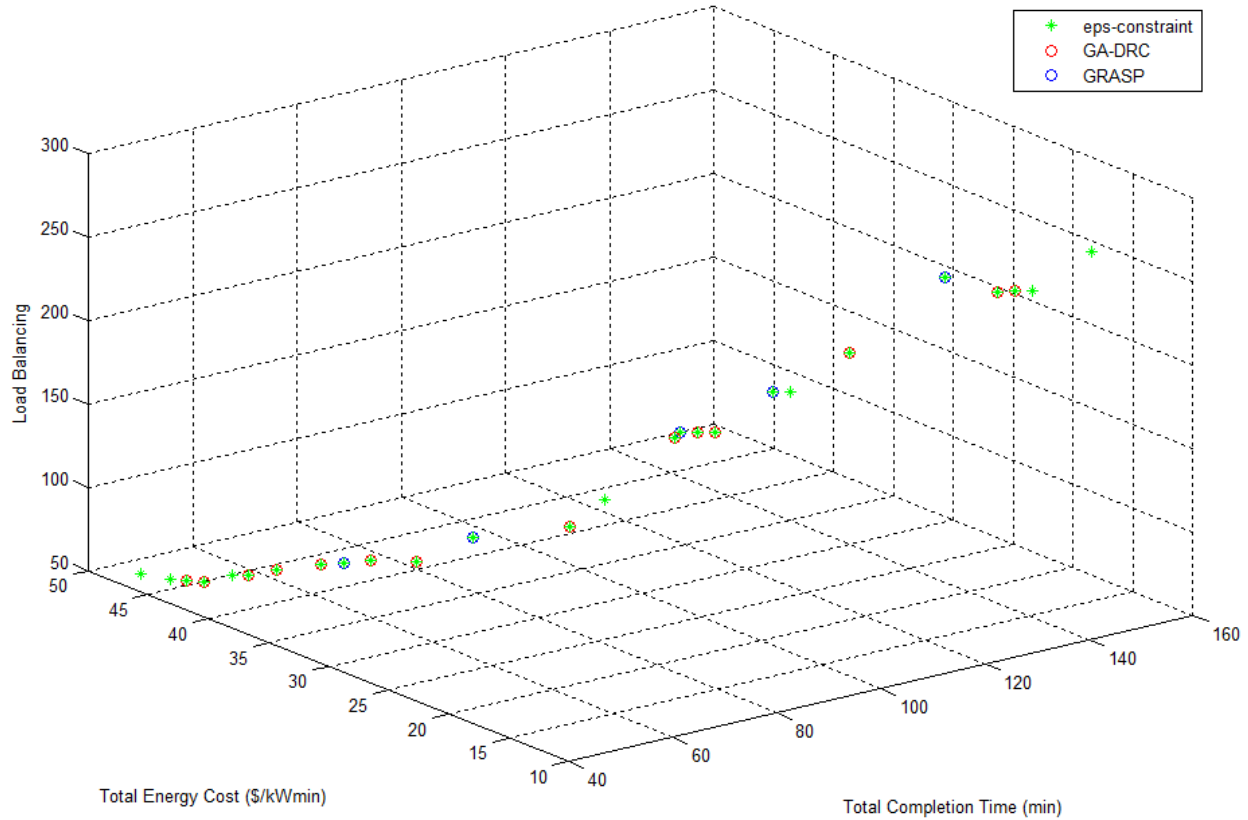


Figure 5.11: Pareto front for problem  $n = 30$  and  $PHF = 0.5$  using  $\epsilon$ -constraint method, GA-DRC, and GRASP

## 5.9 Conclusion

This study provides a tool for sustainable manufacturing via implementing greener production planning under time-of-use tariffs. This tool is focused on energy saving without any major equipment investment, instead focusing its efforts on operations. Thus, electricity prices vary hourly and represent a huge opportunity to minimize energy costs by shifting electricity usage from high-peak hours to low-peak or mid-peak hours.

In this chapter, we have considered the energy cost with a scheduling objective while planning jobs on parallel non-identical machine. We have proposed a multiobjective mixed-integer mathematical problem to minimize the total energy cost, the total completion time, and the deviation from load balancing on machines. To solve this model, we have developed the  $\epsilon$ -constraint method as an exact method for generating the Pareto-optimal front and multiobjective

GRASP and GA-DRC metaheuristic algorithms for obtaining a good approximate Pareto front in a reasonable amount of time in order to solve the model.

The  $\varepsilon$ -constraint method takes a considerable amount of time to produce the exact and complete Pareto set. To overcome this problem, the GRASP and GA-DRC were used to provide an approximate Pareto front in a reasonable amount of time without the guarantee of optimality. After fine-tuning the GRASP and GA-DRC metaheuristic algorithms, we have provided analysis and detail experimental results to evaluate the performance of the algorithms, which maintain the quality of solutions. From the results, we observed that GA-DRC outperforms the GRASP algorithm with respect to solution quality and also outperforms the  $\varepsilon$ -constraint method with respect to computational CPU time for obtaining the approximate Pareto front. Also, the GA-DRC and GRASP have been able to come closer (i.e., match solutions) to approximating the  $\varepsilon$ -constraint method for Pareto-optimal front

A direct extension of this research is to investigate methods that accelerate the GRASP and GA-DRC algorithms, such as dispatch rules for scheduling. Another research direction involves studying other problems with conflicting objectives on various operating environments (i.e., with different scheduling objectives that have sequence-dependent setup times, or on multi-machine settings). These will not affect the results in changing the solution approach, except that the multiobjective mixed-integer mathematical problem will change based on the type of scheduling objective that is employed. Finally, more detailed experiments could be designed to determine if there is a clear pattern between the parameters of the experimental design and the objective.

## 5.10 Reference

- [1] N. Weinert, S. Chiotellis, and G. Seliger, "Methodology for planning and operating energy-efficient production systems," *CIRP Annals-Manufacturing Technology*, vol. 60, no. 1, 2011, pp. 41-44.
- [2] C. Schmidt, W. Li, S. Thiede, S. Kara, and C. Herrmann, "A methodology for customized prediction of energy consumption in manufacturing industries," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 2, no. 2, Oct. 2015, pp. 163-172.
- [3] G. Mustafaraj, J. Cosgrove, M.J. Rivas-Duarte, F. Hardiman, and J. Harrington, "A methodology for determining auxiliary and value-added electricity in manufacturing machines," *International Journal of Production Research*, vol. 53, no. 17, Dec 2015, pp. 1-13.
- [4] F. Shrouf and G. Miragliotta, "Energy management based on Internet of things: practices and framework for adoption in production management," *Journal of Cleaner Production*, vol. 100, no. 1, Aug. 2015, pp. 235-246.
- [5] S. Mousavi, S. Thiede, W. Li, S. Kara, and C. Herrmann, "An integrated approach for improving energy efficiency of manufacturing process chains," *International Journal of Sustainable Engineering*, Jan. 2015, pp. 1-14.
- [6] G. May, I. Barletta, B. Stahl, and M. Taisch, "Energy management in production: A novel method to develop key performance indicators for improving energy efficiency," *Applied Energy*, vol. 149, no. 1, July 2015, pp. 46-61.
- [7] A. Fysikopoulos, G. Pastras, T. Alexopoulos, and G. Chryssolouris, "On a generalized approach to manufacturing energy efficiency," *The International Journal of Advanced Manufacturing Technology*, vol. 73, no. 9-12, 2014, pp. 1437-1452.
- [8] H. Zhang, F. Zhao, K. Fang, and J.W. Sutherland, "Energy-conscious flow shop scheduling under time-of-use electricity tariffs," *CIRP Annals-Manufacturing Technology*, vol. 63, no. 1, March 2014, pp. 37-40.
- [9] H. Zhang, F. Zhao, and J.W. Sutherland, "Energy-efficient scheduling of multiple manufacturing factories under real-time electricity pricing," *CIRP Annals-Manufacturing Technology*, vol. 64, no.1, July 2015, pp. 41-44.
- [10] J. Cheng, F. Chu, W. Xia, J. Ding, and X. Ling. "Bi-objective optimization for single machine batch scheduling considering energy cost," in *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on IEEE*, Nov. 2014, pp. 236-241.
- [11] K. Fang, N.A. Uhan, F. Zhao, and J.W. Sutherland. "Scheduling on a single machine under time-of-use electricity tariffs," May 2014, [cited: Feb. 2015]; available from World Wide Web: [http://www.optimization-online.org/DB\\_FILE/2014/04/4313.pdf](http://www.optimization-online.org/DB_FILE/2014/04/4313.pdf).

- [12] G. Mouzon, M.B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *International Journal of Production Research*, vol. 45, no. 18-19, July 2007, pp. 4247-4271.
- [13] M.B. Yildirim and G. Mouzon, "Single machine sustainable production planning to minimize total energy consumption and total completion time using a multiple objective genetic algorithm," *Engineering Management, IEEE Transactions on*, vol. 59, no. 4, July 2012, pp. 585-597.
- [14] J.Y. Moon, K. Shin, and J. Park, "Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency," *The International Journal of Advanced Manufacturing Technology*, vol. 68, no. 1-4, March 2013, pp. 523-535.
- [15] J. Escamilla, M.A. Salido, A. Giret, and F. Barber, "A Metaheuristic Technique for Energy-Efficiency in Job-Shop Scheduling," *Constraint Satisfaction Techniques for Planning and Scheduling*, Nov. 2014, pp. 42-52.
- [16] F F. Shrouf, J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *Journal of Cleaner Production*, vol. 67, no. 15, March. 2014, pp. 197-207.
- [17] M. Dai, D. Tang, Y. Xu, and W. Li, "Energy-aware integrated process planning and scheduling for job shops," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 229, no. 1, Oct. 2014, pp. 13-26.
- [18] F. Xu, W. Weng, and S. Fujimura. "Energy-efficient scheduling for flexible flow shops by using MIP," in *IIE Annual Conference. Proceedings*, Institute of Industrial Engineers-Publisher, Nov. 2014, pp. 1040.
- [19] C. Duerden, L. Shark, G. Hall, and J. Howe. "Genetic algorithm based modification of production schedule for variance minimisation of energy consumption," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, Oct. 2014
- [20] J.-Y. Ding, S. Song, and C. Wu, "Carbon-efficient scheduling of flow shops by multi-objective optimization," *European Journal of Operational Research*, vol. 248, no. 3, Feb. 2015, pp. 758-771.
- [21] G. May, B. Stahl, M. Taisch, and V. Prabhu, "Multi-objective genetic algorithm for energy-efficient job shop scheduling," *International Journal of Production Research*, vol. 45, no. 18-19, Jan 2015, pp. 1-19.
- [22] C. Ng, T.E. Cheng, and J. Yuan, "Strong NP-hardness of the single machine multi-operation jobs total completion time scheduling problem," *Information processing letters*, vol. 82, no. 4, Aug. 2002, pp. 187-191.
- [23] V. T'kindt, H. Scott, and J.C. Billaut, *Multicriteria Scheduling: Theory, Models And Algorithms*, Springer Science & Business Media, Dec. 2006, pp 62-267.

- [24] J. Branke, K. Deb, K. Miettinen, and R. Slowinski, Multiobjective optimization: Interactive and evolutionary approaches, Springer Science & Business Media, Oct. 2008, pp. 9-45.
- [25] G. Mavrotas, "Effective implementation of the  $\epsilon$ -constraint method in Multi-Objective Mathematical Programming problems," *Applied Mathematics and Computation*, vol. 213, no. 2, July 2009, pp. 455-465.
- [26] G. Mavrotas and K. Florios " A novel version of the  $\epsilon$ -constraint method for finding the exact Pareto set in Multi-Objective Integer Programming problems," in *GAMS Optimization Software*, Dec. 2011.
- [27] E. Salari, J. Wala, and D. Craft, "Exploring trade-offs between VMAT dose quality and delivery efficiency using a network optimization approach," *Physics in medicine and biology*, vol. 57, no. 17, July 2012, pp. 5587.
- [28] M. Brown, B. An, C. Kiekintveld, F. Ordóñez, and M. Tambe, "An extended study on multi-objective security games," *Autonomous agents and multi-agent systems*, vol. 28, no. 1, Oct. 2014, pp. 31-71.
- [29] T.A. Feo and M.G. Resende, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, no. 2, Sept. 1995, pp. 109-133.
- [30] P. Festa and M.G. Resende, "An annotated bibliography of GRASP–Part II: Applications," *International Transactions in Operational Research*, vol. 16, no. 2, Oct. 2009, pp. 131-172.
- [31] J.E.C. Arroyo and A.A. de Souza Pereira, "A GRASP heuristic for the multi-objective permutation flowshop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 55, no. 5-8, May 2011, pp. 741-753.
- [32] D. Soares and J. Arroyo. "A GRASP algorithm for the multi-objective knapsack problem," in *IEEE International Conference of the Chilean Computer Science Society, Arica, Chile*, Sept. 2004, pp. 69-75.
- [33] M.F. Argüello, J.F. Bard, and G. Yu, "A GRASP for aircraft routing in response to groundings and delays," *Journal of Combinatorial Optimization*, vol. 1, no. 3, June 1997, pp. 211-228.
- [34] P.L. Rocha, M.G. Ravetti, and G.R. Mateus. "The metaheuristic GRASP as an upper bound for a branch and bound algorithm in a scheduling problem with non-related parallel machines and sequence-dependent setup times," in *Proceedings of the 4th EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-Heuristics*, vol. 1, Oct. 2004, pp. 62-67.
- [35] S. Binato, W. Hery, D. Loewenstern, and M. Resende, "A GRASP for job shop scheduling," in *Essays and surveys in metaheuristics*, Springer, Oct 2002, pp. 59-79.
- [36] V.A. Armentano and O.C.B. De Araujo, "Grasp with memory-based mechanisms for minimizing total tardiness in single machine scheduling with setup times," *Journal of Heuristics*, vol. 12, no. 6, Oct. 2006, pp. 427-446.

- [37] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, Dec. 2002, pp. 182-197.
- [38] I. ILOG, "Inc. CPLEX 12.5 User Manual," Somers, NY, USA, May 2012.

## CHAPTER 6

### MINIMIZING TOTAL COMPLETION TIME AND ENERGY COST ON A PREEMPTIVE SINGLE MACHINE WITH SEQUENCE-INDEPENDENT SETUP TIMES

#### Abstract

An energy-aware scheduling methodology may improve energy-efficiency in manufacturing environments. This research presents a preemptive scheduling problem on a single machine to minimize the total completion time and total energy cost under time-of-use electricity tariffs with varying energy prices, which is multiobjective mixed-integer mathematical programming model. Each job has non-preemptive sequence-independent setup time(s) which is performed only one time before the job being processed on the machine for the first time. The proposed model solved via several methods including  $\varepsilon$ -constraint method, multiobjective GA based on the dominance ranking procedure (GADR), and dominance ranking procedure and crowding distance comparison (GARC). All multiobjective GA solve the model and obtain an approximate Pareto front in a reasonable amount of time. We provide a detailed experimental results evaluating the performance of the proposed GA algorithms.

**Keywords:** Energy-aware scheduling, Sequence-dependent setup time, Time-of-use electricity tariffs, Multiobjective genetic algorithm optimization,  $\varepsilon$ -constraint method

#### 6.1 Introduction

Most countries have limited energy sources. Therefore, it is important to reduce energy consumption. Since the last decade, energy demand and prices have been increasing steadily due to the scarcity of energy resources paired with increasing population size. Due to growing demand and instabilities in manufacturing environments, this trend is expected to increase. As a result,

energy-aware practices are gaining momentum in manufacturing environments since they improve product performance and optimize the production process— saving energy.

Because of increasing energy prices over the last decade and due to the efforts for sustainability in manufacturing environment, it is necessary to focus efforts on developing and designing more energy-efficient machines via operations planning and scheduling in order to reduce the power and energy demands on machines. In this chapter, we focus our attention in a single machine problem, where a machine can response to insert machine “off/on” time over planning horizon and in time-of-use electricity tariffs, where energy prices vary hourly over planning horizon, which is typically announced a day ahead. This is one of the strategies that energy savings could be accomplished without any major equipment investment. In the result, this strategy integrates two different scheduling decisions: one control request switches the machine “off” wherein inserting machine off time is occurring, and the second request switches the machine “on” wherein the job (or part of it) is resuming.

Because of the time-of-use electricity tariffs, shifting electricity usage from periods of high electricity demand to low electricity demand can reduce the total energy cost of the manufacturing sectors with trade-off of increased carbon dioxide emissions from electricity generation, assuming that the shift will not result in the additional use of crude oil or coal-fired plants. As a result of time-of-use electricity tariffs, there is opportunity to reduce total energy cost and potentially carbon dioxide emissions. In practice, the smart grid devices transpose electricity in order to make two-way communication technology (i.e., between utilities and manufacturers) to save energy and increase levels of sustainability and transparency in manufacturing environments. The type of real (actual) time-of-use tariff also mitigates price spikes. As a result, the manufacturer response refers to changes in the price of electricity depending on the time. For example, the Korea Power

Exchange enables a day-ahead market by sending electricity prices via messages (i.e., electronic data) to industries and thus allowing them shift their electricity usage [1]. Consequently, manufacturing industries set their production schedule to save energy after incorporating time-of-use tariff (i.e., time-dependent electricity cost).

Several articles studied time-dependent industrial planning in order to minimize total cost of energy production. Nilsson and Söderström [2] investigate the influence of time-dependent electricity costs on industrial production planning, and they point out that it is profitable to shift their electricity demand from high-rate periods to low-rate periods. In another study, Nilsson [3] shows that electricity generation also benefits when peaks in the electricity demand shifts to time periods with a lower electricity price. Castro et al. [4] present discrete and continuous time formulation, where the scheduling of continues plants subject to energy constraints related to time-independent electricity cost and availability. Ghobeity and Mitsos [5] optimize the operation of seawater reverse osmosis in order to minimize the total energy cost, and their results showed significant electricity and production cost-saving potentials. Cheng et al. [6] study a new bi-objective single machine batch scheduling problem with a time-dependent electricity cost policy to improve the productivity and minimize the total electricity cost of production. Zhang et al. [7] propose a mathematical model to minimize the electricity cost and carbon footprint under time-dependent electricity cost without compromising production throughput.

In an industrial environment, there has been significant interest in reducing the total energy consumption costs via a production planning and scheduling problem. Escamilla et al. [8] propose a GA to solve an extended version of the job-shop scheduling problem in which machines can consume different amounts of energy to process jobs at different rates. Shrouf et al. [9] study a mathematical model in a single machine problem to minimize total energy consumption of

production scheduling. Xu et al. [10] focus on power peak related energy-aware scheduling problems and present a mixed-integer programming model and simulation to ensure a global optimum. Dai et al. [11] study a multiobjective total energy consumption and makespan job-shop problem to describe an energy-aware integrated process planning and scheduling problem. Ding et al. [12] study a permutation flow shop scheduling problem in order to minimize total carbon emission and makespan.

A few research efforts focused on the problem of inserting machine on/off time on a single machine setting with setup time, where a multiobjective mathematical model is used. Such problems arise in computer application and network routing and battery-operated systems such as laptops or mobile phone. For example, Swaminathan and Chakrabarty [13] study a control system to reduce energy consumption and extend battery life. They show that by changing the state of the machines (e.g., on/off, etc.), energy consumption can be reduced significantly. Inserting machine on/off time occurs when a machine is kept waiting (i.e. the machine is turned off for a predetermined amount of time) for the arrival of a job (or a part of it) that is being processed on a machine. This may be desirable during the delay of a job where there are energy costs for early completion (i.e. a significant amount of energy will be saved when idle machines are turned off for a certain amount of time). Kanet and Sridharan [14] provide a taxonomy of environments in which inserting idle-time scheduling is necessary.

In order to improve manufacturing sustainability, it is necessary to reduce the energy consumption during the product's life cycle [15]. However, each manufacturing environment should establish a method to reduce energy costs. Such a method enables industries to have access to time-of-use electricity tariffs in order to shift their electricity demand from high rates to periods with mid or low rates. This method could be also paired with a production planning and scheduling

problem. Therefore, this chapter considers the following problem, in which a set of jobs  $n$  needs to be processed on a single machine setting, where non-preemption setup times are sequence-independent and preemption processing times are allowed with no additional setup time, (i.e., the setup times depend only on job  $j$ , not on the jobs that precede or follow it. When the processing time of a job is started, it can be interrupted at any time and restarted at a later time in favor of another job with no additional setup time). Each job  $j$  has an initial setup time  $s_j$  representing the setup (loaded) time needed before job  $j$  is processed on the machine. The setup time is assumed to occur only one time before job  $j$  is processed, and when a job is interrupted, it is resumed without a setup time. Note that once a setup time is started, it is processed without interruption and it can be allocated at any time before its job starts. In other words, during setup, a machine cannot process or set up any other job (or part of it). Such problems arise in industrial applications; for example, a single machine (i.e., a battery station) schedules to charge two batteries (two jobs) where each job has a different design or specification that needs a given non-negative setup time to be connected (loaded) before a job is started.

According to the problem above, in this chapter, we propose a multiobjective mixed-integer mathematical model to minimize the total energy cost and total completion time on a single machine with sequence-independent setup times (MMIP-MO-TE-TC-SM-SIST). Note that the setup time is non-preemptive whereas preemption processing times are allowed with no additional setup time. By solving this model, a Pareto front that contains the non-dominated solutions is obtained. Therefore, we utilize an  $\epsilon$ -constraint method in order to obtain the exact Pareto front and we also develop a multiobjective GA based on the dominance ranking procedure (GADR) and the dominance ranking procedure and crowding distance comparison (GARC) metaheuristics to solve the model and obtain an approximate Pareto front. Then we use the exact Pareto front as a

benchmark to evaluate the performance of the approximate Pareto front. Finally, we analyze the performance of the GADR and GARC using some performance measures including the Pareto-optimal solutions that were obtained from the  $\varepsilon$ -constraint method.

In scheduling problems, the minimization of total completion time is widely discussed. By definition, the completion time of job  $j$  ( $C_j$ ) is when a machine completes processing that job. Total completion time is a measure of the total holding cost for a given schedule [16] and it is also a measure of the total cycle time. When all release dates are equal (i.e. all the jobs are available at the beginning) and the processing time is not interrupted (i.e. preemptions are not allowed), the SPT rule minimizes the total completion time [17]. When one of these assumptions is omitted (i.e. processing time is interrupted due to inserting machine on/off time), the total completion time problem is NP-hard [18] i.e. no known algorithm can solve this problem in polynomial time.

In summary, our contributions are as follows: (1) while planning jobs on a single machine where non-preemption setup times are sequenced-independent and preemption processing times are allowed with no additional setup time, energy costs should be considered, (2) utilize time-of-use electricity tariffs, where energy prices vary hourly, (typically announced a day ahead), (3) consider sustainability practices (i.e., incorporating energy-aware via production scheduling ) to balance the energy demands and generating capacity (i.e., reducing the demand when energy prices are high and increasing the demand when energy prices are low), (4) design a multiobjective mixed-integer mathematical problem to minimize total energy cost and total completion time of a scheduling problem, (5) utilize the  $\varepsilon$ -constraint method as an exact method for generating the Pareto-optimal front for a multiobjective mixed-integer mathematical model, and (6) develop a multiobjective GADR and GARC algorithm to obtain a near-optimal Pareto front in a timely fashion.

The organization of this chapter is as follows: in sections 6.2 and 6.3, the multiobjective model is defined, and an  $\varepsilon$ -constraint method is utilized to generate the Pareto-optimal front. In section 6.4, GADR and GARC metaheuristic algorithms are developed to solve the multiobjective problem. Section 6.5 defines the experimental design and presents the results of the  $\varepsilon$ -constraint method. Section 6.6 defines the parameter. Section 6.7 discusses the comparison of the GADR, GARC algorithms and the  $\varepsilon$ -constraint method. Finally, in section 6.8, conclusions and future work of the research is presented.

## **6.2 Multiobjective Mixed-Integer Programming Model to Minimize Total Energy Cost and Total Completion Time MMIP-MO-TE-TC-SM-SIST**

The MMIP-MO-TE-TC-SM-SIST problem is a single machine scheduling problem with sequence-independent setup times scheduling problem with deterministic processing times ( $p_j$ ) jobs, which are available at the beginning (i.e.  $r_j = 0$ ). Under time-of-use electricity tariffs, the electricity prices vary over time. A machine can process one job at a time and once the process of a job is started, it can be interrupted and restarted at a later time without additional setup times (i.e., the jobs are preemptive). Each job  $j$  has an initial non-preemptive setup time ( $s_j$ ), which occurs only one time before the job is executed (i.e., once the process of a job is interrupted, it can be restarted at a later time without additional setup times). The objectives are to minimize the total energy cost and total completion time. The MMIP-MO-TE-TC-SM-SIST model is formulated as a mixed-integer programming (MIP) problem; however, we utilize the time-index variable formulation, in which the planning horizon is discretized into intervals of one unit of length.

### **6.2.1 Notation and Formulation**

The notation used in the problem statement and through the chapter is as follows:

#### **Sets**

N Jobs,  $\{1, \dots, n\}$

$T$  Planning horizon,  $\{1, \dots, t\}$   
 $T'$  Subset of planning horizon,  $T' \subseteq T$

### **Indices**

$j, i$  Jobs,  $j, i \in N$   
 $t, t', t''$  Time interval,  $t \in T$  and  $t', t'' \in T'$

### **Parameters**

$E_t$  Electric price signal at  $t$   
 $p_j$  Processing time of job  $j$

### **Variables**

$C_j$  Completion time of job  $j$   
 $x_{jt} = \begin{cases} 1, & \text{if the job (or part of job) is assigned to time unit } t \\ 0, & \text{otherwise} \end{cases}$   
 $y_{jt} = \begin{cases} 1, & \text{if setup time starts at time } t \\ 0, & \text{otherwise} \end{cases}$   
 $\sigma_{jt} = \begin{cases} 1, & \text{if binary variable } x_{jt} \text{ equals to } 0 \\ 0, & \text{otherwise} \end{cases}$   
 $\hbar_{jt} = \begin{cases} 1, & \text{if } y_{jt} \text{ is equal to } 1 \\ 0, & \text{otherwise} \end{cases}$   
 $\delta_{jt} = \begin{cases} 1, & \text{if binary variable } y_{jt} \text{ equals to } 0 \\ 0, & \text{otherwise} \end{cases}$

The proposed time indexed MIP model is a mathematical program with multiple objectives and several constraints: Note that the following constraints could also be written in terms of the start-time variables.

$$\text{Minimize } \sum_{j=1}^n C_j \quad (6.1)$$

$$\text{Minimize } \sum_{j=1}^n \sum_{t=1}^{|T|} (E_t x_{jt} + E_t \hbar_{jt}) \quad (6.2)$$

$$\sum_{t=1+s_j}^{|T|} x_{jt} = p_j \quad \forall j \in N \quad (6.3)$$

$$\sum_{j=1}^n x_{j(t=t+s_j)} \leq 1 \quad \forall t \in T \quad (6.4)$$

$$\sum_{t=1}^{|T|} y_{jt} = 1 \quad \forall j \in N \quad (6.5)$$

$$\sum_{j=1}^n \sum_{t'=\max(0,t-p_j)}^t y_{jt'} \leq 1 \quad \forall t, t' \in T \quad (6.6)$$

$$x_{j(t=t+s_j)} \leq M(1 - \sigma_{j(t=t+s_j)}) \quad \forall j \in N; t \in T \quad (6.7)$$

$$\sum_{t'}^T -(y_{j(t'=t-s_j)} - 1) \leq M \sigma_{jt} \quad \forall j \in N; t, t' \in T \text{ and } t' \leq t \quad (6.8)$$

$$y_{jt} \leq M(1 - \delta_{jt}) \quad \forall j \in N; t \in T \quad (6.9)$$

$$-(\hbar_{jt'} - 1) \leq M \delta_{jt} \quad \forall j \in N; t, t' \in T \text{ and } t' \geq t, t' \leq t + s_j - 1 \quad (6.10)$$

$$\sum_j^n (x_{jt} + y_{jt}) \leq 1 \quad \forall t \in T \quad (6.11)$$

$$\sum_j^n (x_{jt} + \hbar_{jt}) \leq 1 \quad \forall t \in T \quad (6.12)$$

$$\sum_{t=1}^{|T|} \hbar_{jt} = s_j \quad \forall j \in N \quad (6.13)$$

$$C_j \geq t x_{jt} + 1 \quad \forall j \in N; t \in T \quad (6.14)$$

$$C_j \leq |T| \quad \forall j \in N \quad (6.15)$$

$$C_j \geq 0 \quad \forall j \in N \quad (6.16)$$

$$x_{jt} \in \{0, 1\} \quad \forall j \in N; t \in T \quad (6.17)$$

$$y_{jt} \in \{0, 1\} \quad \forall j \in N; t \in T \quad (6.18)$$

$$\hbar_{jt} \in \{0, 1\} \quad \forall j \in N; t \in T \quad (6.19)$$

$$\sigma_{jt} \in \{0, 1\} \quad \forall j \in N; t \in T \quad (6.20)$$

$$\delta_{jt} \in \{0, 1\} \quad \forall j \in N; t \in T \quad (6.21)$$

The first objective (6.1) minimizes the total completion time and the second objective (6.2) minimizes total energy cost. The satisfaction constraint set (6.3) states that the whole processing time of every job is satisfied; the capacity constraint (6.4) ensures that the machine can handle at most one job during each time interval. Using the time-index variables, the assignment constraint

set (6.5) states that each setup time must start exactly once, whereas the capacity constraint (6.6) ensures that the machine can handle, at most, one setup time during each time interval. Constraints (6.7) and (6.8) ensure that if job  $j$  assigns at time  $t$ , then the machine starts the setup time for job  $j$  at any time  $t'$  before executing job  $j$  (i.e.,  $t' < t$ ). Constraints (6.9) and (6.10) ensure that if the setup time of job  $j$  starts at time  $t'$ , then the machine processes the setup time of job  $j$  from the start time  $t'$  until the completion time. This information will be used to calculate the energy cost of setup time job  $j$ . Constraint (6.11) ensures that the setup times and processing times do not overlap. Constraint (6.12) ensures that the energy setup cost and energy processing times cost do not overlap. Constraint (6.13) states that the whole setup time of every job is satisfied. Constraint (6.14) provides the completion time of each job. Constraint (6.15) defines the boundary of completion time over the planning horizon  $T$ . Finally, constraint (6.16) is non-negativity, and constraint sets (6.17) to (6.21) are integrality constraints.

MMIP-MO-TE-TC-SM-SIST has  $|N| + 4|N||T| + 5|N|^2|T| + |N| \sum_{j=1}^N (|T| - s_j) + |N| \sum_{j=1}^N (s_j(|T| - s_j + 1) + s_j(\frac{s_j-1}{2}))$  variables and  $4|N| + 4|T| + 3|N||T| + \sum_{j=1}^N (|T| - s_j) + \sum_{j=1}^N (s_j(|T| - s_j + 1) + s_j(\frac{s_j-1}{2}))$  constraints for a problem of size  $|T|$  and  $|N|$  where  $|T| \geq \sum_{j=1}^n p_j$ . Obviously, if either  $|T|$  or the number of jobs  $|N|$  increases, the size of the problem also increases significantly. This multiple objective scheduling and energy cost problem can be solved by using multiple objective programming solution techniques such as WSM,  $\epsilon$ -constraints method, goal programming, and multi-level programming [19] or by using metaheuristic methods such as GA, simulating annealing, and tabu search [20].

In the next section, we will utilize the  $\epsilon$ -constraint method to illustrate the mathematical model and gain some insight into the problem. Then we will propose two efficient multiobjective metaheuristics to solve larger-sized problems in a reasonable amount of time.

### 6.3 Using $\varepsilon$ -constraint Method to Solve MIP-MO-TE-TC-LB-PM Problem

The  $\varepsilon$ -constraint method is one of the most widely used multiobjective solution method in terms of generation (a posteriori) optimization. This method optimizes one of the objective functions while transforming the other objective function into constraints with an upper bound of  $\varepsilon$ , where  $\varepsilon$  is a parameter that varies over iterations as shown below [21]:

$$\begin{array}{ll} \text{Min} & f_1(x) \\ \text{st} & \\ & f_2(x) \leq \varepsilon_2 \\ & x \in S \end{array}$$

In the above mathematical program,  $x$  is the vector of the decision variables  $f_1(x)$  and  $f_2(x)$  are the objective functions.  $\varepsilon_2$  is on the right-hand side of the constrained objective function and  $S$  is the feasible region for a two objective problem. By solving the problem for a different value of  $\varepsilon$ , the Pareto front (efficient solutions) of the problem is obtained. Mavrotas [21] proposes a novel version of the  $\varepsilon$ -constraint method that avoids the production of a weak Pareto-optimal solution and accelerates the process by avoiding redundant iterations. In further research, Mavrotas and Florios [22] use the  $\varepsilon$ -constraint method to generate Pareto-optimal solutions (i.e., to produce the exact and complete Pareto set, which includes all of the possible Pareto-optimal solutions) in multiobjective mathematical programming problems with discrete variables. Using the  $\varepsilon$ -constraint method, Salari et al. [23] develop a box algorithm that sequentially generates weak Pareto-optimal points using the  $\varepsilon$ -constraint method. Brown et al. [24] propose a multiobjective security game model and develop iterative- $\varepsilon$ -constraints in order to generate the Pareto front.

In our implementation, the exact Pareto-optimal front for the proposed model can be found by solving a sequence of constrained single-objective optimization problems. First, the payoff table is calculated by using lexicographic optimization of the objective functions. In this approach,

the first objective function is optimized and then in a lexicographic optimization of the objective functions, the second possible alternative objective function is optimized and so on. The  $f_1(x)$  is selected to be optimized, obtaining  $\min f_1(x) = z_1^*$ . Then, the  $f_2(x)$  is selected to be optimized by adding the constraint  $f_1(x) = z_1^*$  in order to keep the optimal solution of the first optimization. In conclusion, using the lexicographic optimization, the results will be more meaningful and will adequately describe the Pareto-optimal front [19].

After calculating the payoff table, we discretize the objective space to equal intervals which are used as the values of  $\varepsilon_p$ , where  $P$  is the number of objective functions, in the  $\varepsilon$ -constraint method. As a result, the multiobjective mathematical problem is transferred to the  $\varepsilon$ -constraint equivalent, which is at the right-hand side on the  $\varepsilon_p$  to obtain the Pareto-optimal solutions by varying the parameter  $\varepsilon_p$ . Finally, the Pareto-optimal solutions are mapped onto the above-constrained using a parametric problem on  $\varepsilon_p$ , and so on. For example, solving the problem for a different value of  $\varepsilon_p$  will generate a finite number of Pareto-optimal solutions as a target.

In the next sections, we propose the GADR and GARC multiobjective metaheuristic approaches to determine an approximate Pareto front for solving a large-sized problems in a reasonable amount of time. Also, we provide a detailed experimental results evaluating the performance of the proposed GA algorithms. A high-quality solution in a reasonable amount of time is necessary to schedule/reschedule jobs when there are disruptions on the shop floor or changes in the manufacturing environment. Therefore, metaheuristics must be designed instead of exact algorithms. The goal of a metaheuristic in multiobjective optimization is to find an approximation of the Pareto-optimal front in a timely fashion (i.e., the non-dominated solutions to the problem).

#### **6.4 Solution to MIP Model Using a Multiobjective GADR and GARC Algorithms**

GAs are inspired from evolution theory: a population generates new offspring (chromosomes) via crossovers and mutations, and new generations (individuals) are formed via the survival of the fittest principle. GAs are largely used to solve combinatorial problems and have also been implemented to solve problems with multiple objectives. Van and Lamont [25] provide a detailed literature review on multiobjective GAs. Meza et al. [26] implement a multiobjective GA to the power generation expansion problem with a special fitness function to obtain a well-distributed near-optimal Pareto front.

In this chapter, after modeling the MMIP-MO-TE-TC-SM-SIST problem, a multiobjective GADR and GARC metaheuristic algorithm propose to obtain an approximate Pareto front. In those, a specific neighborhood solution is defined and searched for in order to obtain either the closest optimal solution in the neighborhood solution or a solution that satisfies the two objectives. They are also very efficient when rescheduling is needed due to changes in the manufacturing environment.

GARD and GARC algorithms have three main operators: crossover, mutation, and selection operators. The crossover operator chooses two chromosomes (parents) to generate new chromosomes, called offspring. The mutation operator establishes a new solution from the current chromosome by changing the order of two genes (i.e., time). Crossover and mutation operations are used to generate new solutions within the search space based on the variation of existing chromosomes. The selection operator decides which solutions will carry over to the next generation. The complete cycle of these operators corresponds to one iteration.

In our implementation, GADR and GARC have the same chromosome representation, crossover and mutation procedures. Also, GARD and GARC are solving the same MMIP-MO-

TE-TC-SM-SIST subproblem. The only differences are the logic of defining the neighborhood solution and searching around the neighborhood solution in order to obtain either the optimal solution or a solution that satisfies the two objectives. In the next subsection, we discuss different components of the algorithms in more details.

#### 6.4.1 Representation (Coding)

In our implementation, a chromosome provides information about the fixed time (on/off time) over the planning horizon, where the machine is turned on/off for a certain amount of time. In construction of a chromosome, the length of a chromosome is equal to  $|T|$ , where  $|T| > \sum_j p_j + \sum_j s_j$ . On a chromosome, the goal is to randomly having the machine “on” during  $T' \subseteq T$  where  $|T'| = \sum_j p_j + \sum_j s_j$ . In other words,  $|T'|$  genes have a “1” value representing “on” time, while  $|T| - |T'|$  genes have a value of “0” representing “off” time. For example,  $[1, 0, 0, 0, 1, 1, 1, 0, 1, 1]$  represents a problem with a time horizon of 10 units and a total of setup time and processing time of 6 time units. Note that during time periods 2, 3, 4, and 8 the machine is off (i.e., the off times are represented by genes having a value of “0”).

Since the proposed problem has non-preemptive setup time(s) for each job  $j$ , in iterative steps we fix consecutive  $\beta$  genes equal to the length of the maximum setup time (i.e.,  $\beta = \max_j s_j$ ). Note that the fixing operation can be placed at any time between start-time 1 and end-time  $|T|$ . To illustrate the construction of the chromosome by means of a numerical example, consider the data on Table 6.1. The length of the chromosome is equal to the length of  $T$ , where  $|T| > 6$  (e.g.  $|T| = 10$ ). The length of  $|T'| = 6$  and the length of fixed consecutive genes  $\beta = 2$ . In iterative steps, the  $\beta$  genes are inserted at the machine’s “on time” (i.e.,  $t' \in |T'|$ ) until we reach the length of  $|T'| = 6$ , where the machine is executing the setup and processing times. In any iteration, for example, a chromosome  $[1, 1, 0, 1, 1, 0, 1, 1, 0, 0]$  will process all setup times and jobs on fixed times  $|T'|$  (i.e.,

time periods 1,2,4,5,7 and 8), where off times are represented by values of “0”. Note that, in a chromosome, we have information on the on/off times only. Also, if the total length of fixed time is less than  $|T'|$ , then we fix a number of random genes until we obtain the exact value of  $|T'|$ .

Table 6.1: Numerical Example

$(n)$ Jobs	$j_1$	$j_2$
$p_j$	2	1
$s_j$	1	2

### 6.4.2 Crossover

The crossover operator inputs two chromosomes to generate two feasible child chromosomes by randomly selecting a crossover point. To illustrate the operation of the crossover operator, consider the example on Table 6.1. In step (a) at Figure 6.1, the crossover operator inputs two chromosomes (Parent 1 and Parent 2) to generate two chromosomes (Child 1 and Child 2) by randomly selecting a crossover point between genes 5 and 6. Each child receives the first part of its chromosome from one of the parents and the second part of the chromosome from the other parent as shown in step (b) at Figure 6.1. Note that, for example, the first child in our example is  $[1, 1, 0, 0, 1, 0, 0, 1, 1, 0]$ , which is infeasible since the total length of  $|T'|$  is less than what it should be (i.e.,  $|T'| = 6$ ). In addition, the second child  $[1, 1, 1, 1, 0, 1, 1, 1, 0, 0]$  is also infeasible since the total length of  $|T'|$  is greater than what it should be.

To correct the infeasibility, we utilize the following procedure: in the child chromosome, we count the total length of fixed time and identify all possible genes that cause infeasibility (i.e., we find a number of genes that need to be exchanged to “off” or “on” time in order to obtain the exact length of fixed time). If the infeasible child has less fixed time (i.e., “on” time), then we exchange the genes that are unfixed (“off” time) with genes that are fixed (“on” time) that have not appeared in the chromosome in an ascending lexicographic order. As a result, the first

infeasible child results in [1, 1, 1, 0, 1, 0, 0, 1, 1, 0] when this procedure is utilized (see step (c) at Figure 6.1). If the infeasible child has a large number of fixed time, then we exchange the genes that are fixed (“on” time) with genes that are unfixed (“off” time) that have appeared in the chromosome in a descending lexicographic order. Therefore, when this procedure is utilized, the second infeasible child results in [1, 1, 1, 1, 0, 1, 1, 0, 0, 0] (see step (c) at Figure 6.1).

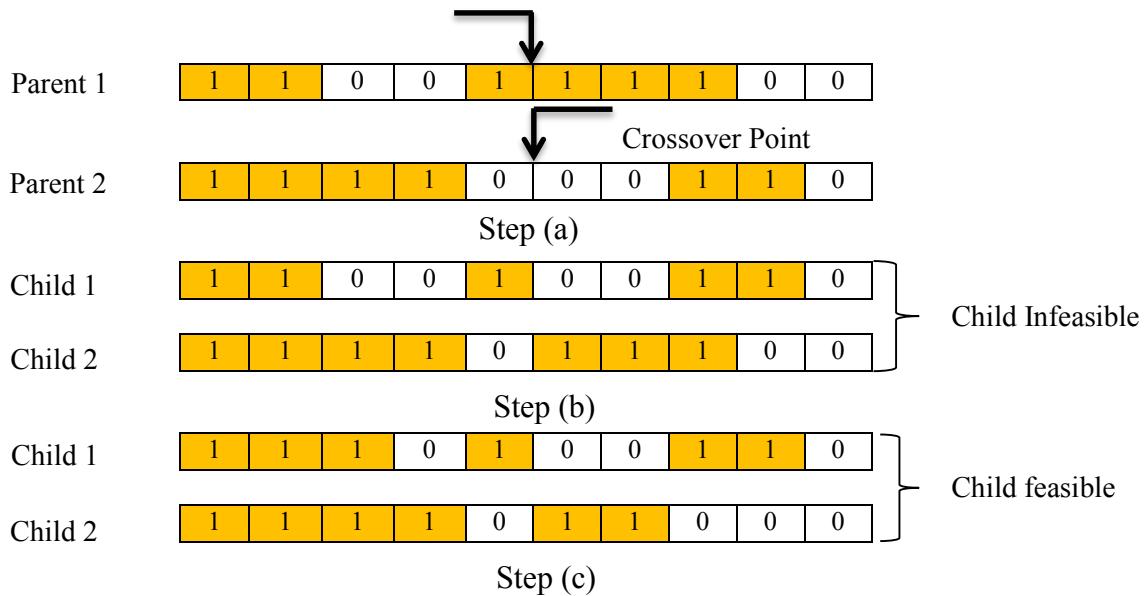


Figure 6.1: Crossover operator

### 6.4.3 Mutation

The mutation operator selects a single chromosome as input and generates one new chromosome by randomly exchanging the last order of  $\beta$  fixed time (i.e.,  $\beta = 2$ ) to another order from the unfixed times (“off” times). The mutation operation is used to avoid the population of chromosomes from becoming too similar to each other and slowing or even stopping the evolution process. Figure 6.2 illustrates the operation of the mutation operator, where the locations of fixed times and unfixed times are changed.

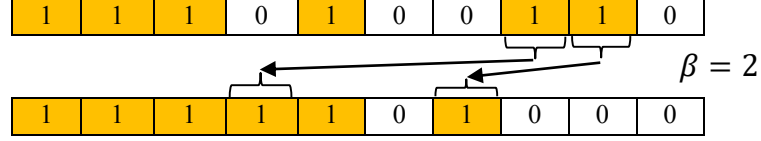


Figure 6.2: Mutation operator

#### 6.4.4 Evaluation of Chromosome: Obtaining Approximate Pareto Front of Chromosome Using Reduced MIP

In a given chromosome, feasible schedules of start on/off times for the machine are fixed. Therefore, the fixed start time “on” is a subset of the time horizon (i.e., a chromosome represents the subset of time horizon,  $T' \subseteq T$ ). When this information is known, the original model of MMIP-MO-TE-TC-SM-SIST is reduced to the following multiobjective mixed-integer mathematical model (MMIP-MO-TE-TC-SM-SIST-R) where R means reduced, in which the number of variable and constraints are reduced to  $|N| + 3|N||T'| + 5|N|^2|T'| + |N| \sum_{j=1}^N (|T'| - s_j) + |N| \sum_{j=1}^N (s_j(|T'| - s_j + 1) + s_j(\frac{s_j-1}{2}))$  and  $4|N| + 3|T'| + 3|N||T'| + \sum_{j=1}^N (|T'| - s_j) + \sum_{j=1}^N (s_j(|T'| - s_j + 1) + s_j(\frac{s_j-1}{2}))$  respectively. Note that in this formulation,  $C_j$ ,  $p_j$  and  $s_j$  indicate completion time, process time and setup time of job in position  $j$  in the sequence defining the orders of the jobs.

$$\text{Minimize } \sum_{j=1}^n C_j \quad (6.22)$$

$$\text{Minimize } \sum_{j=1}^n \sum_{t' \in T'} (E_{t'} x_{jt'} + E_{t'} \hbar_{jt'}) \quad (6.23)$$

$$\sum_{t'=1+s_j}^{|T'|} x_{jt'} = p_j \quad \forall j \in N \quad (6.24)$$

$$\sum_{t' \in T'} y_{jt'} = 1 \quad \forall j \in N \quad (6.25)$$

$$\sum_j^n \sum_{t''=\max(0, t'-s_j)}^{|T'|} y_{jt''} \leq 1 \quad \forall t' \in T' \quad (6.26)$$

$$x_{j(t'=t'+s_j)} \leq M(1 - \sigma_{j(t'=t'+s_j)}) \quad \forall j \in N; t' \in T' \quad (6.27)$$

$$\sum_{t' \in T'} -(y_{j(t'=t'-s_j)} - 1) \leq M \sigma_{jt'} \quad \forall j \in N; t', t'' \in T' \text{ and } t'' \leq t' \quad (6.28)$$

$$y_{jt'} \leq M(1 - \delta_{jt'}) \quad \forall j \in N; t' \in T' \quad (6.29)$$

$$-(\hbar_{jt''} - 1) \leq M \delta_{jt'} \quad \forall j \in N; t', t'' \in T' \text{ and } t'' = t' \quad (6.30)$$

$$\sum_j^n (x_{jt'} + y_{jt'}) \leq 1 \quad \forall t' \in T' \quad (6.31)$$

$$\sum_j^n (x_{jt'} + \hbar_{jt'}) = 1 \quad \forall t' \in T' \quad (6.32)$$

$$\sum_{t' \in T'} \hbar_{jt'} = s_j \quad \forall j \in N \quad (6.33)$$

$$C_j \geq t x_{jt'} + 1 \quad \forall j \in N; t' \in T' \quad (6.34)$$

$$C_j \leq |T'| \quad \forall j \in N \quad (6.35)$$

$$C_j \geq 0 \quad \forall j \in N \quad (6.36)$$

$$x_{jt'} \in \{0, 1\} \quad \forall j \in N; t' \in T' \quad (6.37)$$

$$y_{jt'} \in \{0, 1\} \quad \forall j \in N; t' \in T' \quad (6.38)$$

$$\hbar_{jt'} \in \{0, 1\} \quad \forall j \in N; t' \in T' \quad (6.39)$$

$$\sigma_{jt'} \in \{0, 1\} \quad \forall j \in N; t' \in T' \quad (6.40)$$

$$\delta_{jt'} \in \{0, 1\} \quad \forall j \in N; t' \in T' \quad (6.41)$$

For a given a random feasible chromosome, the subset of time horizon has a fixed energy cost. Below, we provide a theoretical proof showing the local optimality of the total energy cost function when given a random chromosome. Note that the total energy cost for a chromosome is constant, i.e., fixed. The following proposition is used to prove this proposition.

**Proposition 1:** For a given a feasible chromosome (i.e., a chromosome that represents the subset of time horizon,  $T' \subseteq T$ ), the total energy cost is constant.

**Proof:** The total energy cost function (equation 6.23) is:

$$\sum_j^n \sum_{t' \in T'} (E_t x_{jt'} + E_t \hbar_{jt'}) = \sum_{t' \in T'} E_t \sum_{j=1}^n (x_{jt'} + \hbar_{jt'}) \quad (6.42)$$

The constraint set (6.32) ensures the machine can handle only one job or one setup time at any given time interval (i.e.,  $t' \in T'$ ):

$$\sum_{j=1}^n x_{jt'} + \sum_{j=1}^n \hbar_{jt'} = 1 \quad (6.43)$$

Therefore, from equations (6.42) and (6.43), we can prove that the total energy cost function is fixed as follows:

$$\sum_j^n \sum_{t' \in T'} (E_t x_{jt'} + E_t \hbar_{jt'}) = \sum_{t' \in T'} E_t \sum_{j=1}^n (x_{jt'} + \hbar_{jt'}) = \sum_{t' \in T'} E_t \quad (6.44)$$

Now, the MMIP-MO-TE-TC-SM-SIST-R is reduced to a single-objective function (i.e., total completion time function). As a result, for each feasible chromosome, we are solving only the total completion time to optimality since the total energy cost is fixed.

For a given chromosome, a set of non-dominated solutions is obtained. Therefore, fitness function needs to be utilized to determine the quality of the solutions in the current population. In the next sub-section, we present the evaluation of fitness function for chromosome in GADR and GARC algorithms. Note that, the solution with best measure of the fitness function over the set of solutions is kept to evaluate the chromosome.

#### 6.4.5 Fitness Function and Value for Chromosomes in GARD Algorithm

After a set of non-dominated solutions for a chromosome is obtained, a fitness function is used to evaluate the quality of the solutions in the current population. Each individual has a fitness value, which is assigned according to its dominance. Therefore, a solution determines the order of

the jobs and setup times, the completion time of all jobs, and the total energy cost. Note that the chromosomes with higher fitness function values have a higher probability of surviving in the next generation.

Since the multiobjective optimization optimizes the components of a vector values cost function, and the solution to the problem is a family of points known as the Pareto optimal set, the fitness function based on its dominance ranking is used to evaluate the quality of the solutions in the current population [27]. In this fitness function, let  $\Pi_{\alpha\varphi}$  be the fitness function value for a solution  $\alpha$  in generation  $\varphi$ . The rank of the solution is equal to  $1 + \#_{\alpha\varphi}$ , where  $\#_{\alpha\varphi}$  is the number of solutions that determines a solution  $\alpha$  in generation  $\varphi$ . If the solution is non-dominated, then its rank is 1. Note that the higher the rank, the poorer the solution. This can be written as

$$Fitness_1(\Pi_{\alpha\varphi}) = \frac{1}{rank(\Pi_{\alpha\varphi})} = \frac{1}{1 + \#_{\alpha\varphi}}$$

Note that the fitness function value for a chromosome  $\Psi$ ,  $Fitness_{\Psi}$  is the sum of the fitness function value for each solution obtained from a chromosome  $\Psi$ .

#### 6.4.5.1 Selection Process in GARD Algorithm

The selection process aims to reflect nature's survival of the fittest. As mentioned previously, in a GA, a chromosome with a better fitness function will earn more chances to survive in the next generations. In this type of evaluation of a fitness function, we intend to utilize the roulette wheel system, which selects parents for the next generation. This process can be described as shown in Figure 6.3. Note that the chromosomes with higher fitness function values have a higher probability of surviving in the next generation.

*Step 1:* Order the chromosomes in descending fitness function value.

*Step 2:* Compute the probability function  $Pr_\psi$  for a chromosome  $\psi$  as follows:

$$Pr_\psi = \frac{Fitness_\psi}{\sum_{l=1} Fitness_l}$$

*Step 3:* Utilize the roulette wheel method using the probability values from step 2 until chromosomes are selected.

Figure 6.3: Process of selection of chromosomes for generating the next population

#### 6.4.6 Fitness Function and Value for Chromosomes in GARC Algorithm

The fitness function in GARC is inspired from the NSGA-II [28], which is proven to generate high-quality results on test problems. In addition, NSGA-II is a popular metaheuristic algorithm with a better sorting methodology, ensures elitism and does not require that a sharing parameter is chosen. NSGA-II uses the Pareto-dominance concept of solutions to guide the search process and it returns the Pareto-optimal set as the best results. In our implementation, we use the non-dominated sorting procedure and crowding distance sorting procedure to evaluate the quality of solutions in the current GARC population.

The non-dominated sorting procedure ranks the solution in different Pareto fronts. Then, the crowding distance sorting procedure measures the density of individuals in solution space (i.e., calculates the dispersion of the solution in each front and preserves the diversification of the algorithm). At each generation, these two operators form the Pareto front.

After a set of non-dominated solutions for a chromosome is obtained, each solution is associated with a rank equal to its non-dominance level. For example, in Figure 6.4, level 1 (rank 1) contains all the dominant individuals within the population. If the individuals in level 1 are not considered, then the second set of dominant individuals creates level 2 (rank 2), and so on. As a

result, the individual with the lower rank is preferable because it represents the sorting non-dominated solutions.

After the non-dominated solutions is ranked, the crowding distance is used to measure how close an individual is to its neighbors in order to guarantee the diversity of the population. In Figure 6.4, the parameter of the cuboid is estimated by using the nearest neighbor as the vertices [28]. Note that, for an individual, the crowding distance is computed as the sum of the normalized distance between its right and left neighbors for each objective function. For the first and last individuals (extreme solutions), there is a crowding distance equal to infinity.

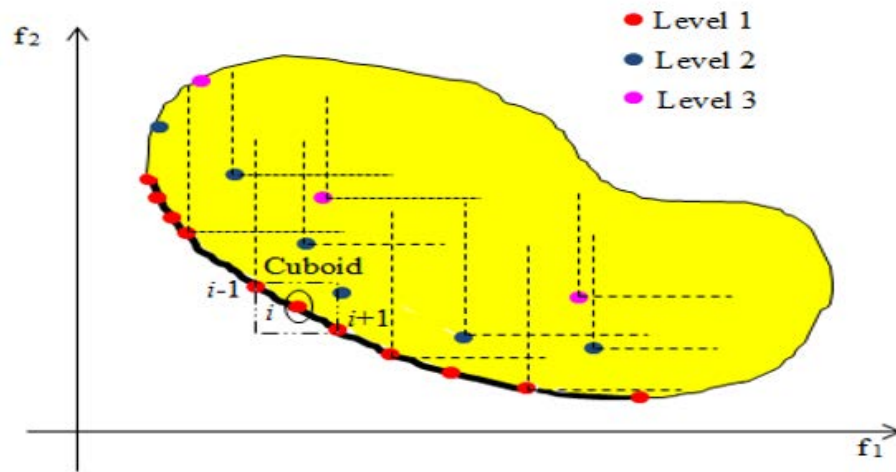


Figure 6.4: Non-dominated levels and computation of the crowding distance

#### 6.4.6.1 Selection Process in GARC Algorithm

After all individuals of the initial population are sorted using the non-dominated sorting and crowding distance sorting procedure, the algorithm assigns the selection operator to create the first offspring set— the chromosome with a better selection operator will have a better chance of surviving in the next generation. The selection is based on two comparison rules: (1) the lower rank the individual belongs to, the better solution is; (2) if two individuals have the same rank, the individual with greater crowding distance has a better solution because the area it belongs to is less

crowded. This process can be described as shown in Figure 6.5. Note that the chromosomes with lower rank have better solutions, if they are from the same Pareto front, then the chromosome with greater crowding distance has better solution and surviving in the next generation.

In the next section, we will present the experimental design and evaluation of performance and effectiveness of the proposed algorithms. Also, the comparison of the proposed algorithm is discussed in order to solve the MMIP-MEC-TT problem.

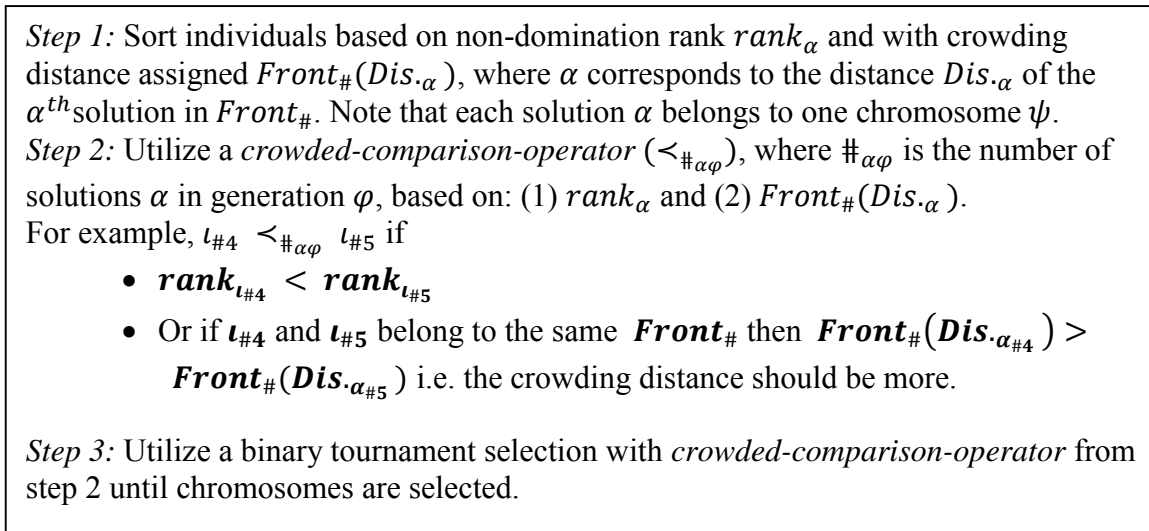


Figure 6.5: Process of selection of chromosomes for generating the next population

## 6.5 Experimental Design and Evaluation

The GAMS is used to model the MMIP-MO-TE-TC-SM-SIST problem, and the CPLEX 12.5 [29] is utilized to solve the problem. To evaluate the performance and effectiveness of the proposed GADR and GARC algorithms, the MATLAB “R2010a” language is used to code the algorithms and IBM ILOG CPLEX 12.5 via MATLAB is utilized to solve the problem. The computational experiments are performed on an Intel i5 2.27GHz machine with 4GB of memory and a Windows 7 operation system.

### 6.5.1 Generation of Experiments Data

The experiment's problems are obtained through-combining the five parameters in different ways: number of jobs ( $n$ ), planning horizon ( $T$ ), processing times ( $p_j$ ), setup times ( $s_j$ ), and electricity prices (i.e., time-of-use tariffs) ( $E_t$ ). During our analysis, we observe that the number of jobs  $|N|$  and the size of the planning horizon  $|T|$  have a major influence on the performance of the algorithm since the problem size and CPU time are directly related (i.e., when problem size increases, CPU time increases). For this reason, we tested the quality of algorithmic parameters as well.

In the experimentation, problems with 10, 20, 30, 40, and 50 jobs are generated. For each job, an integer processing time and setup time is obtained randomly from a uniform distribution between  $[1,10]$  and  $[1,3]$  respectively. An integer planning horizon  $|T|$  is computed as  $|T| > \left( \sum_j p_j + (\max_j s_j * n) \right) / PHF$ , where  $PHF$  is the planning horizon factor, which takes two values 0.5 for a longer planning horizon or 0.75 for a shorter planning horizon. The electricity prices,  $E_t$ , are generated from three uniform distributions  $[1, 5]$ ,  $[6, 10]$ ,  $[11, 16]$  over three different periods of time to create low, moderate, and high variations (see Figure 6.6).

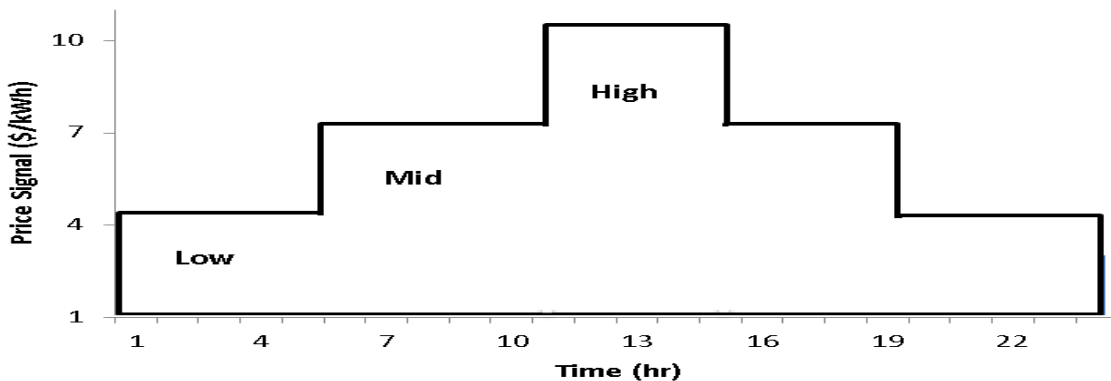


Figure 6.6: Prices signal (\$/kWh) vs. time (hr.) (i.e., example of time-of-use tariffs)

### 6.5.2 Solving MMIP-MO-TE-TC-SM-SIST Using $\varepsilon$ -constraint Method

A set of 6 problems is randomly generated to gain insight into the MMIP-MO-TE-TC-SM-SIST multiobjective optimization problem (see Table 6.2). We generate 30 instances (i.e., five different instances for each test), solve every instance individually, and measure the average time of execution (CPU) in minutes.

Using the  $\varepsilon$ -constraint method, the MMIP-MO-TE-TC-SM-SIST problem translates into a sequence of constrained single-objective optimization problems. One objective is selected as the primary objective to be minimized while the upper-bound constraints are added for the other secondary objectives as

$$\begin{aligned}
 & \text{Min} \quad \sum_{j=1}^n C_j \\
 & \text{st} \\
 & \quad \sum_{j=1}^n \sum_{t=1}^{|T|} (E_t x_{jt} + E_t \hat{h}_{jt}) \leq \varepsilon_2 \\
 & \quad x \in S
 \end{aligned}$$

By varying the constraints (i.e., the parameters on the right-hand side of the constrained objective functions  $\varepsilon$ ), the non-dominated solutions of the problem are obtained. Note that we use the  $\varepsilon$ -constraint method to solve the subproblem in the GADR and GARC algorithms in order to obtain the approximate Pareto-optimal front.

Table 6.2 illustrates the results of the  $\varepsilon$ -constraint method for each instance and shows the minimum and maximum amount of execution time and the number of non-dominated solutions obtained. For example, for  $n = 10$  there is two instances for each  $PHF = 0.75$  and  $0.5$ , which  $\varepsilon$ -constraint method has failed to solve optimally within its time limit; for  $n = 30$  there are already 6 timeouts in total; and for  $n = 50$  there are 7 timeouts since CPLEX is running out of memory while solving the model. It is clear to note that when the number of jobs increases, the CPU time

also increases since the search space for the problem increases in terms of both variables and constraints. In addition, the CPU time can be decreased if fewer number of varying  $\varepsilon_P$  parameter are used (i.e., we will not need as many single-objective mixed-integer problems to be solved). However, this will have a significant impact on the quality of the Pareto front. In conclusion, we see that there is a need to study and apply the metaheuristics in order to obtain an approximate Pareto front in a timely fashion.

Table 6.2: Features of Multiobjective Optimization Instances and Results of  $\varepsilon$ -constraint Method

$n$	$PHF$	Timeouts	Execution Time (minutes)		Number of Non-dominated Points	
			Min	Max	Min	Max
10	0.75	2	675	>1,440	14	19
10	0.5	2	821	>1,440	16	21
30	0.75	3	953	>1,440	19	27
30	0.5	3	1034	>1,440	26	34
50	0.75	3	1249	>1,440	28	35
50	0.5	4	1286	>1,440	32	38

## 6.6 Parameter Fine-Tuning

Selecting good parameters to run GAs can play a vital role in determining a good approximate Pareto front in a timely fashion. For this goal, extensive numerical experiments are performed to fine-tune the GARD and GARC algorithms and to analyze and observe the different parameter effects and performance measures in comparing alternative Pareto fronts. Each test is run with same parameters twice, and the average measure of performance is taken. The comparison between the Pareto fronts is based on the performance measure of CPU time and the measure equal to the area of functional space covered by the non-dominated solutions vector [30]. For example, each dominated vector represents a rectangle defined by the points  $(0,0)$  and  $(f_1(x_i), f_2(x_i))$ , where  $f_1(x_i)$  and  $f_2(x_i)$  represent a non-dominated solution (see Figure 6.7). The area covered by

three solutions in Figure 6.7 is the sum of the area of rectangles a, b and c. This metric estimates the size of the global dominated set in objective space. Note that a solution with low value of the union of the area of all the rectangles may have a high convergence performance. In contrast, the higher value of the union of the area of all the rectangles may result in better diversity performance (i.e., non-dominated solutions are distributed in a Pareto optimal set over the non-dominated region). In our implementation, since we solve minimization objectives, the performance of an algorithm is better with a lower value of the union of the area of all the rectangles.

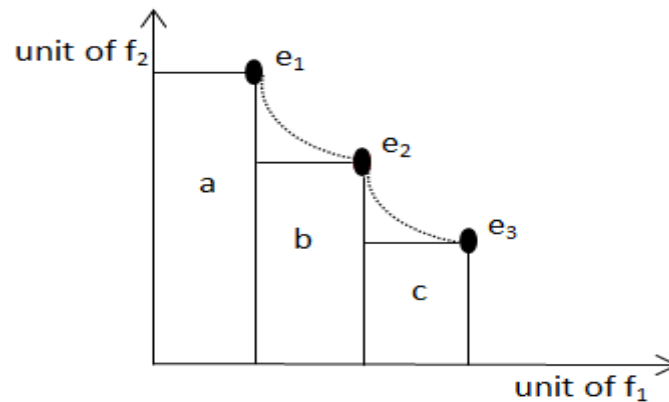


Figure 6.7: Measure of performance of a set of non-dominated solutions

To obtain the Pareto front for the MMIP-MO-TE-TC-SM-SIST efficiently, optimization of the parameters of the GARD and GARC algorithms, such as crossover and mutation rates, generation size, number of generations should be performed. In our implementation, the population size (the individuals) is assumed to be 15, the number of generation is 20, the rate for crossover is 70% and the mutation rate is 30%. Using the measure of performance equal to the union of the area of all the rectangles, results are compared using different rates of crossover from 0 to 1 in increments of 0.2, with *PHF* values of 0.75 and 0.5, and a problem size of 10, 30 and 50 jobs since the optimal rates of crossover may vary with the size of time and the number of jobs considered. Note that a Pareto front with a lower value of the union of the area of all the rectangles

performs better than a Pareto front with a higher value. Table 6.3 shows that a crossover rate of 0.7 leads to better results for the problems with different jobs and with different *PHF* values. From the results, we can observe that there is a high convergence between the solutions when the crossover rate is between 0.75 and 1. In other words, a high probability crossover will achieve a better solution when the problem size is increased.

Using the fine-tuned crossover rate of 0.7 while keeping all other parameters constant, the same type of experiments can be run to determine the fine-tuned number of individuals in a generation as a function of the number of jobs to be processed. By increasing the number of individuals in a generation from 10 to 60 in increments of 10, we can observe how the performance measure change as a function of population size. Note that, we observe a trade-off between the generation size and CPU time of an individual iteration. As a result, we fine-tuned number of individuals for the problems around 20, which yields reasonable quality solutions (see Table 6.4).

While keeping the crossover rate and the number of individuals in a generation at their fine-tuned values, the same type of experiments can be run with the number of generation varying in each run from 10 to 60 in increments of 10, with *PHF* values of 0.75 and 0.5. Table 6.5 shows that the good number of generations is 60, which is the maximum number allowed. In other words, if we keep increasing the number of generations, the better the solution becomes. A possible reason for this is from the GA, which states the more generations, the better the solution, but the longer the CPU time. However, since we are interested in the CPU time, the number of generations is limited to 20.

Table 6.3: Performance Depending on Crossover Rates and PHF = 0.75 and 0.5

Crossover rate	$n = 10$		$n = 10$		$n = 30$		$n = 30$		$n = 50$		$n = 50$	
	PHF = 0.7	CPU (min)	PHF = 0.50	CPU (min)	PHF = 0.75	CPU (min)	PHF = 0.50	CPU (min)	PHF = 0.75	CPU (min)	PHF = 0.50	CPU (min)
0	4917	24.2	5832	25.4	6714	32.8	7264	34.2	7782	42.2	8124	44.8
0.2	5274	23.6	6447	24.9	7591	32.3	8061	33.7	7924	41.9	8792	44.3
0.4	5138	22.8	6219	24.6	8439	31.8	7889	33.4	8017	41.3	8761	43.8
0.6	4786	22.3	6319	23.8	8116	31.3	7325	32.8	7813	40.3	7928	43.2
0.7	3995	22	5043	23.5	6473	30.8	6837	32.5	7582	40	7219	42.7
1	4281	21.6	5549	22.8	7295	29.6	6918	31.8	7754	39.4	7594	41.4

Table 6.4: Performance Depending on Number of Individuals in a Generation and PHF = 0.75 and 0.5

Number of individuals	$n = 10$		$n = 10$		$n = 30$		$n = 30$		$n = 50$		$n = 50$	
	PHF = 0.75	CPU (min)	PHF = 0.5	CPU (min)	PHF = 0.7	CPU (min)	PHF = 0.5	CPU (min)	PHF = 0.7	CPU (min)	PHF = 0.5	CPU (min)
10	5047	11.5	6547	12.4	5478	14.3	6254	14.9	5937	16.3	6432	17.3
20	4736	22	5249	23.5	5361	30.8	5274	32.5	5749	40	6015	42.7
30	4973	32.3	5295	33.1	6736	41.1	5343	41.8	5432	52.4	6154	54.2
40	5642	42.8	5978	43.8	5215	50.9	5847	53.2	6651	66.2	7813	66.4
50	5814	54.8	6351	54.4	7497	61.3	6421	63.5	7149	77.4	5965	79.6
60	5883	66.5	6875	65.8	7791	72.6	7426	73.3	7631	87.9	9418	93.2

Table 6.5: Performance Depending on Number of Generations and PHF = 0.75 and 0.5

Number of generation	$n = 10$		$n = 10$		$n = 30$		$n = 30$		$n = 50$		$n = 50$	
	PHF = 0.7	CPU (min)	PHF = 0.5	CPU (min)	PHF = 0.7	CPU (min)	PHF = 0.5	CPU (min)	PHF = 0.7	CPU (min)	PHF = 0.5	CPU (min)
10	5954	16.3	6025	16.9	6371	17.5	6582	17.9	7248	20.6	7384	21.3
20	5713	22	5873	23.5	6234	30.8	6495	32.5	7193	40	7335	42.7
30	5584	51.5	5849	51.9	6189	59.2	6438	60.1	7024	68.5	7157	72.3
40	5451	80.9	5542	80.9	5946	88	6195	89.6	6936	97.9	7122	102.5
50	5208	109.6	5456	109.4	5831	117	6144	119.8	6817	126.3	7064	132
60	5124	135.2	5319	136.8	5763	144.7	6025	147.3	6736	152.8	6943	160.6

In conclusion, because considering all possible combinations of a parameter may take significant amount of time, we optimize the parameter of the GARD and GARC one at a time. In section 6.7, good parameters are used to compare the performance of the proposed algorithms.

## 6.7 Comparison of GADR, GARC Algorithms and $\epsilon$ -constraint Method

Given the fine-tuned parameters for each GA, we compare the performance and quality of the proposed algorithms using the same number of jobs. In other words, both GARD and GARC algorithms have the same representation (e.g., chromosome, mixed-integer, etc.), operators (e.g., crossover and mutation), but different parameters (e.g., rate of the crossover and mutation, size of a generation, number of generations, etc.) on all problems to be tested. However, in this section, we investigate the proposed algorithms based on problem sizes 10, 20, 30, 40, 50 and *PHF* values of 0.75 and 0.5 (see Table 6.6). Note that we apply the proposed algorithms on the same instances used to characterize the performance of the  $\epsilon$ -constraint method. In addition, the comparison of the algorithms is based on the following six performance measurement tools:

- Number of Solutions on Pareto Front (No. Pareto):

This metric counts the total number of non-dominated solutions that are achieved by the algorithm. The algorithm with a higher total number is preferred.

- Generational Distance (GD):

This metric evaluates the convergence performance of the algorithm [31]. This metric estimates how “far” the elements (i.e., non-dominated solutions) are from the global Pareto front.

GD is computed as

$$GD = \frac{\sqrt{\sum_{q=1}^Q d_q^2}}{Q}$$

where  $Q$  is the total number of non-dominated solutions in the Pareto-optimal front (i.e., the global Pareto front) and  $d_q$  is the Euclidean distance that is measured in objective space between each of these and the nearest member of Pareto-optimal front. If a value of  $GD = 0$ , then all elements generated are in the Pareto-optimal front. Any value given indicates how “far” we are from the global Pareto front of our problem. Note that the lower value of  $GD$  is desirable.

- Size of the Area Covered (SAC):

This metric estimates the size of the global dominated set in an objective area [32]. It is similar to the performance measure used in parameter fine-tuning. SAC can evaluate the diversity and convergence performance of algorithms. If an algorithm resulting in low value of SAC may have a high convergence performance, in contrast, the higher values of SAC may result in higher diversity performance. In this section, we need to measure the diversity performance (i.e., non-dominated solutions are distributed in Pareto-optimal set over the non-dominated region), therefore an algorithm resulting in high value of SAC is desirable.

- Error Rate (ER):

This metric measures the percentage of the obtained Pareto front and not the Pareto-optimal front [33]. In our implementation, the  $\varepsilon$ -constraint method always generates the Pareto-optimal front (i.e. global Pareto front). ER is computed as

$$ER = \frac{\sum_{r=1}^R E_r}{R}$$

where  $R$  is the number of vectors in the obtained Pareto front,  $E_r = 0$  if vector  $r$  is a member of the Pareto-optimal front, and  $E_r = 1$  otherwise. If  $ER = 0$ , then all the vectors generated by an algorithm belongs to the Pareto-optimal front. Note that this metric requires knowing the number of elements of the Pareto-optimal front set, which are the solutions of the  $\varepsilon$ -constraint method. Therefore, in this metric we use the exact Pareto-optimal front from the  $\varepsilon$ -constraint method as a

benchmark to evaluate the performance of the approximate Pareto fronts that obtained by GARD and GARC algorithms.

- Quality Metric (QM):

This metric builds a new Pareto front after a set of non-dominated solutions from each algorithm is obtained for the same problem. Then a pairwise comparison is performed (i.e., the performance of one algorithm is the percentage of solutions in the new Pareto front). The algorithm with a higher percentage performs is desirable.

- CPU time:

This metric measures the computational time required to obtain the Pareto front (i.e., the time spent on the processor running the program's code). The algorithm with less time is desirable.

In our implementation, each test is run with the same parameters three times, and the average measure of performance is taken. Note that since the GARD and GARC are based on randomized iterations that use different random chromosomes, the same results may not be returned twice. Therefore, the best way to compare the quality of the GARD and GARC algorithms is by comparing their Pareto front with the Pareto-optimal front (i.e., the global Pareto front) obtained by the  $\varepsilon$ -constraint method. Table 6.6 summarizes the results obtained by the applied algorithms and the  $\varepsilon$ -constraint method.

The number of Pareto solutions shows that GARC has a better performance than the GARD algorithm and  $\varepsilon$ -constraint method. Also, GARC and GARD algorithms have higher performance in the number of Pareto solutions compared with the  $\varepsilon$ -constraint method. A possible reason for this is that the proposed GAs solve many iterations that use different random chromosomes without any guarantee of optimality (i.e., approximate Pareto solutions), whereas the  $\varepsilon$ -constraint method characteristics always generate the Pareto-optimal solutions (i.e., exact solutions).

The value of GD estimates how far the elements are in approximate Pareto fronts from those in Pareto-optimal fronts of the  $\varepsilon$ -constraint method. All the elements generated by the  $\varepsilon$ -constraint method are Pareto-optimal front, therefore it should be clear that the  $\varepsilon$ -constraint method values in  $GD = 0$  for all instances. As a result, the GD values for GARC and GARD indicate how far the solutions are from the Pareto-optimal front of the  $\varepsilon$ -constraint method. From Table 6.6, we can see that the GARC performs closer than GARD to the solutions of  $\varepsilon$ -constraint method for all instances. A possible reason for this is that the GARC has characteristics (e.g., sorting methods) that allow it to achieve close non-dominated solutions to the Pareto-optimal front. That means GARC performs better than GARD for all instances.

The values of SAC show that GARC has a higher diversity performance than both the GARD and  $\varepsilon$ -constraint methods have for all instances. A possible reason for the diversity performance is higher in GARC is that GARC has better sorting methods (i.e., non-dominated sorting and crowding distance sorting) that allow it to maintain enough non-dominated solutions in the final GA population. In contrast, GARD finds a better converged set of non-dominated solution for all instances compared to GARC, but the distribution in solutions is better with GARC.

The ER comparison indicates that, on average, GARC has 54% of solutions that are not members of the Pareto-optimal front of the  $\varepsilon$ -constraint method whereas GARD has 83% for all instances. That means GARC is better than GARD for all instances. Note that the  $\varepsilon$ -constraint method has  $ER = 0$  since it determines the Pareto-optimal front for all instances.

Figure 6.8 shows the Pareto fronts—for instance  $n = 30$  and  $PHF = 0.75$  using the  $\varepsilon$ -constraint method, GARC and GARD algorithms. If we combine all the Pareto fronts in a new Pareto front as shown in QM metric, Figure 6.8 shows that GARC and GARD have an identical local Pareto-optimal from the exact solutions found in the  $\varepsilon$ -constraint method, as well as identical

solutions from one another. In the same instance, the number of non-dominated solutions are similar, on average, 45% of the combined Pareto-optimal solutions are from the  $\varepsilon$ -constraint method, 21% of the solution are from the GARD, and 34% of the solutions are from the GARC algorithm.

Table 6.6 shows that, in all instances, GARC obtains the Pareto front at a faster CPU time than the GARD algorithm can. A possible reason for this is that GARD uses a fitness function based on dominance ranking, which means, once a set of non-dominated solutions for a chromosome is obtained, fitness function is used to evaluate the quality of the solutions in the current population. In one result, there is a computational CPU time consumed to assign each individual to its dominance based on its fitness value. In contrast, GARC uses a very efficient sorting procedures to maintain enough non-dominated solutions in the current population at a faster CPU time. In all instances, the  $\varepsilon$ -constraint method requires more than 1440 minutes (more than one day) to be solved exactly. Consequently, the goal of the proposed GAs is to determine an approximate Pareto front for a large-sized problem in a timely fashion in order to schedule and reschedule jobs when there are disruptions on the shop floor or changes in the manufacturing environment.

In conclusion, the GARC outperforms the performance of the GARD with respect to solution quality and also outperforms the  $\varepsilon$ -constraint method with respect to computational CPU time for obtaining the approximate Pareto front. More importantly, the GARC and GARD have come closer (i.e., match solutions) to the  $\varepsilon$ -constraint method Pareto-optimal front.

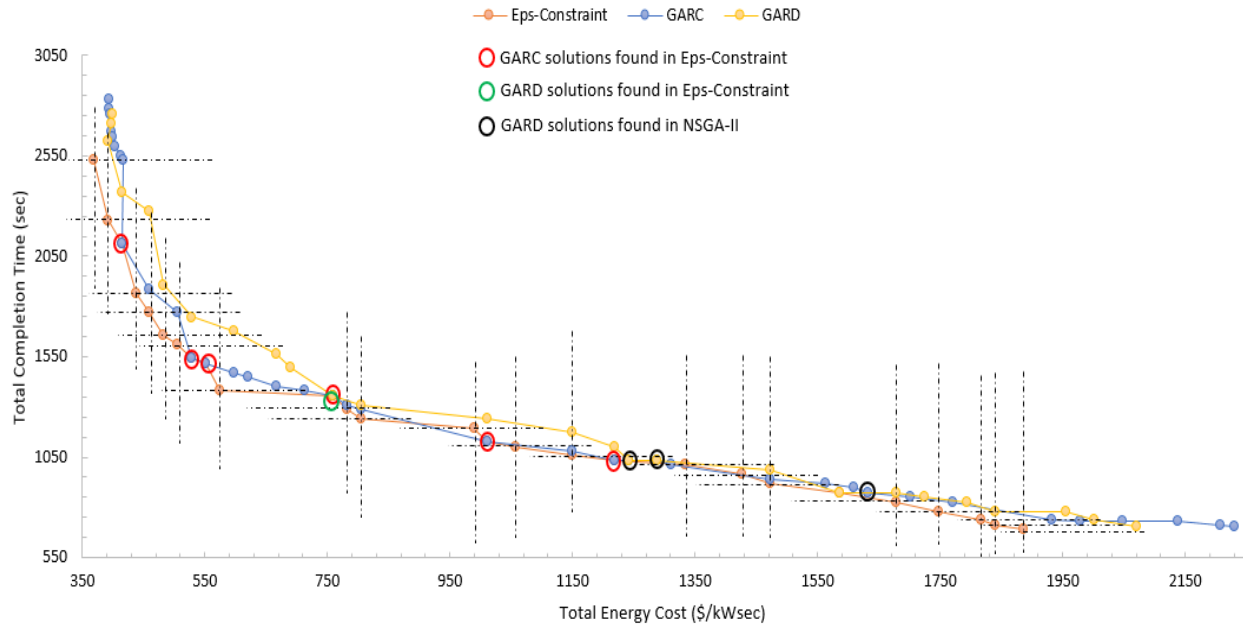


Figure 6.8: Pareto fronts for problem  $n = 30$  and  $PHF = 0.75$  using  $\epsilon$ -constraint method, GARC, and GARD

Table 6.6: Pairwise Comparison of GARD, GARC Algorithms and  $\epsilon$ -constraint Method

$N$	$PHF$	GADR						GARC						$\epsilon$ -constraint					
		No. Pareto	GD	SAC	ER (%)	QM (%)	CPU (min)	No. Pareto	GD	SAC	ER (%)	QM (%)	CPU (min)	No. Pareto	GD	SAC	ER (%)	QM (%)	CPU (min)
10	0.75	24	382	5214	0.74	24	25.6	27	189	5768	0.26	32	22	19	0	4865	0	44	>1440
10	0.5	27	398	5320	0.63	21	27.1	31	194	6247	0.38	38	23.5	21	0	5187	0	42	>1440
20	0.75	26	243	5715	0.86	15	29	33	202	6459	0.39	26	25	22	0	5351	0	59	>1440
20	0.5	31	247	6141	0.92	10	33.7	37	208	6673	0.63	26	26.3	27	0	5943	0	65	>1440
30	0.75	26	240	5864	0.96	21	36.3	38	217	6682	0.78	34	30.8	27	0	6039	0	45	>1440
30	0.5	32	274	5731	0.85	14	39.4	42	258	7164	0.47	24	32.5	34	0	6630	0	62	>1440
40	0.75	30	297	5717	0.84	19	41.8	39	281	6947	0.66	22	35.2	31	0	6072	0	58	>1440
40	0.5	34	437	6129	0.86	16	42.6	45	406	7328	0.73	19	38.5	34	0	6658	0	65	>1440
50	0.75	34	484	6203	0.81	13	44.7	41	397	7106	0.42	26	40	35	0	6731	0	61	>1440
50	0.5	36	446	6346	0.85	12	47.5	49	424	7740	0.67	21	42.7	38	0	6697	0	67	>1440
<b>Average</b>		30	345	5838	0.83	16	36	38	277	6811	0.54	27	32	29	0	6017	0	57	>1440

## 6.8 Conclusion

This study mainly focuses on energy-aware scheduling problems in manufacturing environments since electricity is one of the most important forms of energy, and the energy costs of many manufacturing operations are closely related to the peak power value of that company's operating system. In this chapter, we provided a method to allow manufacturers to have sustainable manufacturing processes by saving energy via operations scheduling without investing in new machines. Using real-time pricing of electricity (i.e., time-of-use electricity tariffs), we have shown that there is a huge opportunity to minimize energy costs by shifting electricity usage from high-peak hours to low-peak or mid-peak hours.

In this chapter, we considered the energy costs of a scheduling objective on a single machine setting where non-preemption setup times, sequenced-independent, and preemption processing times were allowed with no additional setup time. We proposed a multiobjective mixed-integer mathematical problem to minimize the total energy cost, and total completion time objectives of single machine scheduling problem. To solve this model, we developed the  $\varepsilon$ -constraint method as an exact method for generating the Pareto-optimal front and multiobjective GARD and GARC algorithms for obtaining a good approximate Pareto front in a reasonable amount of time.

The  $\varepsilon$ -constraint method took a considerable amount of time to produce the exact and complete Pareto set. To overcome this problem, the GARD and GARC were used to provide an approximate Pareto front in a reasonable amount of time without guaranteeing optimality. After fine-tuning the GARD and GARC algorithms, we provided an analysis and detailed experimental results to evaluate the performance of the algorithms, which maintain the quality of solutions. From the results, we observed that GARC outperforms the GARD algorithm with respect to

solution quality in all test problems. In addition, GARC outperforms the  $\varepsilon$ -constraint method with respect to computational CPU time for obtaining the approximate Pareto front. Moreover, GARC and GARD have been able to come closer to approximating the  $\varepsilon$ -constraint method for the Pareto-optimal front.

A direct extension of this research is to investigate methods that accelerate the GARD and GARC algorithms, such as dispatching rules for scheduling. Another research direction involves studying other problems with conflicting objectives on various operating environments (i.e., with different scheduling objectives that have multi-machine settings). These will not affect the results in changing the solution approach, except that the multiobjective mixed-integer mathematical problem will change based on the type of scheduling objective that is employed. Finally, more detailed experiments could be designed to determine if there is a clear pattern between the parameters of the experimental design and the objective.

## 6.9 Reference

- [1] KPX, *The Korea Power Exchange And Electricity Market*, March 2011, Document <http://www.kpx.or.kr/eng/contents.do?key=299>.
- [2] K. Nilsson and M. Söderström, "Industrial applications of production planning with optimal electricity demand," *Applied energy*, vol. 46, no. 2, Sept. 1993, pp. 181-192.
- [3] K. Nilsson, "Industrial production planning with optimal electricity cost," *Energy conversion and management*, vol. 34, no. 3, Nov. 1993, pp. 153-158.
- [4] P.M. Castro, I. Harjunoski, and I.E. Grossmann, "New continuous-time scheduling formulation for continuous plants under variable electricity cost," *Industrial & engineering chemistry research*, vol. 48, no. 14, June 2009, pp. 6701-6714.
- [5] A. Ghobeity and A. Mitsos, "Optimal time-dependent operation of seawater reverse osmosis," *Desalination*, vol. 263, no. 1, Oct. 2010, pp. 76-88.
- [6] J. Cheng, F. Chu, W. Xia, J. Ding, and X. Ling. "Bi-objective optimization for single machine batch scheduling considering energy cost," in *Control, Decision and Information Technologies (CoDIT), 2014 International Conference on IEEE*, Nov. 2014, pp. 236-241.

- [7] H. Zhang, F. Zhao, and J.W. Sutherland, "Energy-efficient scheduling of multiple manufacturing factories under real-time electricity pricing," *CIRP Annals-Manufacturing Technology*, vol. 64, no.1, July 2015, pp. 41-44.
- [8] J. Escamilla, M.A. Salido, A. Giret, and F. Barber, "A Metaheuristic Technique for Energy-Efficiency in Job-Shop Scheduling," *Constraint Satisfaction Techniques for Planning and Scheduling*, Nov. 2014, pp. 42-52.
- [9] F F. Shrouf, J. Ordieres-Meré, A. García-Sánchez, and M. Ortega-Mier, "Optimizing the production scheduling of a single machine to minimize total energy consumption costs," *Journal of Cleaner Production*, vol. 67, no. 15, March. 2014, pp. 197-207.
- [10] F. Xu, W. Weng, and S. Fujimura. "Energy-efficient scheduling for flexible flow shops by using MIP," in *IIE Annual Conference. Proceedings*, Institute of Industrial Engineers-Publisher, Nov. 2014, pp. 1040.
- [11] M. Dai, D. Tang, Y. Xu, and W. Li, "Energy-aware integrated process planning and scheduling for job shops," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 229, no. 1, Oct. 2014, pp. 13-26.
- [12] J.-Y. Ding, S. Song, and C. Wu, "Carbon-efficient scheduling of flow shops by multi-objective optimization," *European Journal of Operational Research*, vol. 248, no. 3, Feb. 2015, pp. 758-771.
- [13] V. Swaminathan and K. Chakrabarty, "Energy-conscious, deterministic I/O device scheduling in hard real-time systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 22, no. 7, Oct. 2003, pp. 847-858.
- [14] J.J. Kanet and V. Sridharan, "Scheduling with inserted idle time: problem taxonomy and literature review," *Operations Research*, vol. 48, no. 1, Feb. 2000, pp. 99-110.
- [15] Y. He, B. Liu, X. Zhang, H. Gao, and X. Liu, "A modeling method of task-oriented energy consumption for machining manufacturing system," *Journal of Cleaner Production*, vol. 23, no. 1, March 2012, pp. 167-174.
- [16] P.I. Cowling and P. Merz, *Evolutionary Computation in Combinatorial Optimization: 10th European Conference, EvoCOP 2010, Istanbul, Turkey, Proceedings*, vol. 6022: Springer Science & Business Media, April 2010, pp 52-73.
- [17] M. Pinedo, *Planning and Scheduling in Manufacturing and Services*, vol. 3: Springer, Aug. 2005, pp 96-157.
- [18] M.S. Akturk, J.B. Ghosh, and E.D. Gunes, "Scheduling with tool changes to minimize total completion time: a study of heuristics and their performance," *Naval Research Logistics (NRL)*, vol. 50, no. 1, Oct. 2003, pp. 15-30.
- [19] M. Caramia and P. Dell'Olmo, *Multi-Objective Management In Freight Logistics: Increasing Capacity, Service Level And Safety With Optimization Algorithms*, Springer Science & Business Media, Oct. 2008, pp. 68-208.

- [20] A. Jaskiewicz, *Multiple objective metaheuristic algorithms for combinatorial optimization*, Poznan University of Technology, Dec. 2001, pp. 68-147.
- [21] G. Mavrotas, "Effective implementation of the  $\epsilon$ -constraint method in Multi-Objective Mathematical Programming problems," *Applied Mathematics and Computation*, vol. 213, no. 2, July 2009, pp. 455-465.
- [22] G. Mavrotas and K. Florios " A novel version of the  $\epsilon$ -constraint method for finding the exact Pareto set in Multi-Objective Integer Programming problems," in *GAMS Optimization Software*, Dec. 2011.
- [23] E. Salari, J. Wala, and D. Craft, "Exploring trade-offs between VMAT dose quality and delivery efficiency using a network optimization approach," *Physics in medicine and biology*, vol. 57, no. 17, July 2012, pp. 5587.
- [24] M. Brown, B. An, C. Kiekintveld, F. Ordóñez, and M. Tambe, "An extended study on multi-objective security games," *Autonomous agents and multi-agent systems*, vol. 28, no. 1, Oct. 2014, pp. 31-71.
- [25] D.A. Van Veldhuizen and G.B. Lamont. "On measuring multiobjective evolutionary algorithm performance," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 1, Oct. 2000, pp. 204-211.
- [26] J.L.C. Meza, M.B. Yildirim, and A.S.M. Masud, "A model for the multiperiod multiobjective power generation expansion problem," *Power Systems, IEEE Transactions on*, vol. 22, no. 2, Sept. 2007, pp. 871-878.
- [27] C.M. Fonseca and P.J. Fleming. "Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization," in *ICGA*, vol. 93: Citeseer, Sept. 1993, pp. 416-423.
- [28] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, Dec. 2002, pp. 182-197.
- [29] I. ILOG, "Inc. CPLEX 12.5 User Manual," Somers, NY, USA, May 2012.
- [30] E. Zitzler, *Evolutionary Algorithms For Multiobjective Optimization: Methods And Applications*, Swiss Federal Institute of Technology Zurich for the degree of Doctor of Technical Sciences, Citeseer, Dec. 1999.
- [31] D.A. Van Veldhuizen and G.B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis (report)," T.R. TR-98-03, Editor, Citeseer: Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, Dec. 1998.
- [32] J.R. Schott, *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*, Department of Aeronautics and Astronautics (technical report), Massachusetts Institute of Technology, May 1995.

- [33] R. Sarker and C.A.C. Coello, "Assessment methodologies for multiobjective evolutionary algorithms," in *Evolutionary Optimization*, Springer US, vol. 48, Oct. 2002, pp. 177-195.

## CHAPTER 7

### CONCLUSIONS AND FUTURE RESEARCH

#### 7.1 Conclusions of This Dissertation

This dissertation focused on energy-aware scheduling problems on single and parallel machine industrial environments. Various models and algorithms that would provide solutions to significant energy savings or energy reduction in energy-efficient machines in the presence of real-time energy pricing (time-of-use electricity tariffs) were proposed. Multiobjective mixed-integer mathematical models were proposed to solve scheduling problems on single or parallel machines in which one objective is to minimize the total energy cost in an industrial environment and another objective is usually related to scheduling preferences. In addition, for each mathematical model, methods were developed to obtain a set of non-dominated solutions belonging to the global Pareto-optimal front or near-optimal approximate Pareto front. The proposed models are solved to the global Pareto-optimal (complete and exact solutions) via the WSM or  $\varepsilon$ -constraint method. Also, the proposed models are solved based on either multiobjective GAs, multiobjective ant colony algorithms, or GRASP in order to obtain the near-optimal approximate Pareto front. Moreover, a framework based on multiobjective problems such as TOPSIS and MOORA was designed to select the best solution among all of non-dominated solutions.

The frameworks proposed in this dissertation can be utilized to solve large-sized scheduling problems with different objectives. The following sections provide a short summary and conclusions for each problem presented in this dissertation.

##### 7.1.1 Conclusion in Chapter 3

Chapter 3 proposed a multiobjective mixed-integer mathematical programming model on a non-preemptive single machine to minimize the total tardiness and total energy cost under time-

of-use tariffs with varying energy prices. The mathematical model is solved via several methods including the WSM, and multiobjective GAs based on the dominance ranking procedure (GA-1), weighted sum aggregation (GA-2), and the dominance ranking procedure and crowding distance comparison (GA-3). Results show that finding the global Pareto-optimal front via WSM took a considerable amount of time. As a result, it was necessary to find an approximate Pareto front in a reasonable amount of time in order to schedule and reschedule jobs when there are disruptions on the shop floor or changes in the manufacturing environment. A detailed description of the proposed algorithms was presented, including the definition of all steps such as crossovers, mutations, fitness functions, selection processes, etc. In addition, the proposed algorithms illustrate how to obtain a set of non-dominated solutions and how the algorithms differ. The proposed GAs generate a chromosome that can represent more than one solution. The parameters of the proposed GAs (i.e., rate of crossover, rate of mutation, size of a generation, and parameter sharing) were fine-tuned depending on detailed experimental results. Each GA was compared against each other and the WSM in order to gain more insight into the algorithms' dynamics. In addition, the GAs performance including a random number of generations was discussed and compared in order to explore the quality of solutions and find a way to expedite the solution process.

Results show that the GA-3 algorithm found more diverse solutions, a better spread, and more non-dominated solutions in the Pareto-optimal region than the other algorithms. Finally, in a case study, GA-3 was used to illustrate the TOPSIS method in order to assist the decision-maker in choosing the most efficient schedule with an appropriate energy-cost level.

### **7.1.2 Conclusion in Chapter 4**

Chapter 4 presented a scheduling problem on a preemptive single machine to minimize the total completion time and total energy cost under time-of-use electricity tariffs, where energy

prices varied hourly and were announced a day ahead. This problem was modeled using a multiobjective mixed-integer mathematical programming and was solved using either the WSM or a multiobjective ant colony algorithm based on the dominance ranking procedure or based on the dominance-ranking procedure and crowding distance comparison. Results showed that the WSM took a considerable time to obtain the global Pareto-optimal set. Therefore, the proposed ant colony algorithms were developed to provide a set of non-dominated solutions belonging to an approximate Pareto-optimal front in a timely fashion. The proposed ant colony algorithms generated an ant tour that can represent more than one solution. The parameters of the proposed ant colony algorithms (i.e., size of ant population, and appropriate  $\alpha$ ,  $\beta$ ,  $q_0$ ,  $\rho_l$ , and  $\rho_g$  parameters) were fine-tuned based on detailed experimental results with a varying number of jobs using a specific performance measure in the experimental setup. In addition, the performance of the proposed ant colony algorithms were compared to each other using several performance measures in order to clearly explain how Pareto fronts can be characterized.

Results show that when incorporated with theorem 1 (heuristic method), ant colony algorithms perform better than the CPLEX solver (solving MIP model) and have a faster CPU time. Also, results show that ACO-DRC found more diverse solutions, a better spread, and more non-dominated solutions in the Pareto-optimal region than the other algorithms. Finally, the ACO-DRC was illustrated in a case study, and the MOORA method was used to assist the decision-maker in choosing the most appropriate, efficient, and cost-effective schedule.

### **7.1.3 Conclusion in Chapter 5**

Chapter 5 focused on improving the energy-efficiency in a manufacturing environment by incorporating real-time energy pricing via the scheduling problem. The purpose of Chapter 5 was to study a production planning and scheduling problem on a non-preemptive parallel machine to

minimize the total completion time, load balance, and total energy cost under time-of-use tariffs. This problem was modeled using a multiobjective mixed-integer mathematical programming model and was solved via several methods including the  $\varepsilon$ -constraint method, multiobjective GRASP, and multiobjective GA based on the dominance ranking procedure and crowding distance comparison. The  $\varepsilon$ -constraint method took a considerable amount of time to obtain the exact and complete Pareto set. To overcome this problem, the proposed metaheuristics were used to obtain a timely and high-quality approximate Pareto front for large-sized problems, and to schedule or reschedule jobs whenever disruptions on the shop floor or changes in the manufacturing environment occur. After fine-tuning the proposed metaheuristic algorithms, an analysis and detailed experimental results were provided to evaluate the performance of the algorithms, which maintain the quality of solutions.

From the results, it was found that the proposed metaheuristics discovered more diverse solutions than the  $\varepsilon$ -constraint method, although at the expense of higher CPU times. Additionally, the GA-DRC algorithm outperformed GRASP in the quality of solutions and number of solutions in the Pareto-optimal front.

#### **7.1.4 Conclusion in Chapter 6**

Chapter 6 presented a preemptive scheduling problem on a single machine to minimize the total completion time and total energy cost of time-of-use electricity tariffs with varying energy prices. Each job had a non-preemptive sequence-independent setup time, which was performed only once before the job was first processed on the machine. The proposed model was solved via several methods, including the  $\varepsilon$ -constraint method, the multiobjective GA based on the dominance ranking procedure, and the dominance ranking procedure and crowding distance comparison. Results showed that finding the complete Pareto-optimal front via the  $\varepsilon$ -constraint

method took a considerable amount of time. As a result, the GADR and GARC algorithms were proposed in order to obtain an approximate Pareto front in a timely fashion. In those GAs, a specific neighborhood solution was defined and searched in order to obtain either the Pareto-optimal solution (near or exact) in the neighborhood solution or a solution that satisfied the two objectives. Results showed that the proposed GAs are very efficient when rescheduling is needed due to changes in the manufacturing environment. In addition, the GAs were compared to each other and to the  $\epsilon$ -constraint method in order to gain insight into the algorithm's performances.

Both GARC and GARD obtained an identical local Pareto-optimal solution from the exact solutions found in the  $\epsilon$ -constraint method, as well as identical solutions as one another. Additionally, results of the performance measures showed that GARC had a better solution quality than GARD.

### **7.1.5 Summary of Contributions of This Dissertation**

The problems and models explored in this dissertation examine energy-aware manufacturing environments while incorporating real-time energy pricing via production planning and scheduling problems. One of the objectives was to minimize the total energy cost. The contributions of this dissertation can be summarized as follows:

- New multiobjective mixed-integer mathematical programming models were proposed for single- and parallel machine scheduling problems in order to reduce the total energy cost.
- In order to reduce total energy cost, real-time energy pricing was used to design energy-efficient scheduling problems that could provide significant energy savings or an energy reduction in energy-efficient machines.
- Exact methods and metaheuristics were developed to solve each model presented.

- Based on the multiobjective optimization problems, frameworks and decision-maker preferences were designed to select the best solution among all non-dominated solutions.
- Several performance measures were used to analyze and explore the proposed metaheuristics dynamics, in order to greatly evaluate the quality of solutions.
- Different fitness functions were proposed in order to evaluate the quality of solutions in each metaheuristic iteration developed.

The metaheuristics proposed in this dissertation may be used to solve a large-sized scheduling problem with different objectives. However, section 7.2 explores the conditions and possible extensions that will define future research.

## **7.2 Future Research**

In this dissertation, the developed methods and mathematical models were defined to minimize energy cost and other scheduling objectives on a single or parallel machine. A direct extension of the proposed mathematical models could be developed to more complex multi-machine environments, such as a complete operations on the assembly line. Additionally, when maintenance planning is required, a scheduling program could be developed to minimize the total energy cost. A study of the reliability of machines under sporadic on/off cycles (times) could be performed to prevent a machine failure resulting from mechanical shocks. Models and methods could be designed for machines with sequence-dependent and sequence-independent setup times. Finally, simple heuristics or methods can be helpful to obtain an approximate Pareto front without complex calculations (i.e., dispatch rules) for the scheduling problem.

### **7.2.1 Maintenance Planning and Scheduling with Energy-Aware Scheduling**

Maintenance planning and scheduling is vital to the efficiency and life of a machine. If a machine begins to wear, it can consume more energy and perform poorly. If wear begins to occur,

the machine needs to be serviced before failure. Usually either a maintenance request (i.e., machine is repaired when a breakdown happens) or preventive maintenance is required. Compared to the cost of preventative maintenance, a major machine breakdown is significantly more expensive. Therefore, an energy-aware scheduling model would be needed to handle this specific maintenance problem.

Energy cost, which is an expected outcome of machine maintenance, is not considered in the total cost. Mills and Rosenfeld [1] define maintenance activity as one of the energy-efficiency measurements needed to achieve a reduction in the energy cost of the manufacturing environments. Lung et al. [2] examine the importance of energy savings resulting from the adoption of maintenance management in manufacturing environments through the use of a supply conservation curve, which is an analytical tool that captures both the engineering and the economic perspectives of energy conservation. Tam et al. [3] provide a detailed study of a maintenance scheduling model with the objective of minimizing the total cost associated with production loss due to a machine breakdown, material replacement and maintenance costs on a single machine. Celen and Djurdjanovic [4] develop an integrated scheduling operation that takes into account the product target. Their metaheuristic method takes into account machine conditions, such as operations executed on the machine, in order to find an outcome of a preventive maintenance schedule. Cassidy and Kutanoglu [5] propose a single-objective model to minimize total weighted expected completion time on a single machine based on the probability of failure and effect of preventive maintenance on the failure probability. Huene and Kiesmüller [6] study a scheduling problem on a single machine, which is subjected to stochastic machine failure, in order to avoid long downtime and properly plan the preventive maintenance.

A scheduling preventative maintenance team along with an energy-aware model to take into account the total energy cost under time-of-use electricity tariffs is important. For example, a model could schedule a maintenance team to perform preventive maintenance during high-peak times of high energy use when the machine is turned off for a certain amount of time. A systems approach would consider the total cost (i.e., energy cost, productivity cost, maintenance cost).

### **7.2.2 Reliability Studies in Machine Inserting On/Off Time with Energy-Aware Scheduling**

Repeated on/off cycles increase the wear and tear and decrease the reliability (lifetime) of machines, thus increasing maintenance and replacement costs. Chen et al. [7] investigate the energy-consumption reduction in production systems through effective scheduling of machine on/off cycles. In their model, they consider serial production lines that have finite buffers and machines that incorporate a Bernoulli reliability model, where the downtime is relatively short compared to the machine's cycle time. Their recommendation is to schedule the machine on and off based on the status of the production line. In similar work, Nicholson [8] proposes a dynamic programming approach to schedule the shutdown of machines in an automotive production line to reach a given end state problem in the manufacturing system. Nicholson's key approach is to make use of the capacity and ordering constraints of the line to dramatically simplify the feasible decisions of the dynamic program.

To continue this research, a study could be developed to determine the tradeoff between machine wear and tear, and energy savings due to repeated on/off cycles. A model could be designed and incorporated into the energy-usage minimization framework in order to determine the effect of off/on time on a machine's reliability.

### 7.2.3 Multiobjective Scheduling with Sequence-Dependent Setup Times

Setup times are defined as the work needed to prepare the machine to perform a job. Chapter 6 discusses a non-preemptive sequence-independent setup time(s) performed only once before the first job proposal is processed. Further research could be conducted to study more scheduling problems like single or parallel machines with sequence-dependent setup times. Setup times can be sequence-dependent if they are scheduled in accordance with the processing times of the current and immediately preceding jobs. Each setup time would have a different processing time and energy cost, which could be included in the solution framework in order to efficiently schedule these setup times. Usually, scheduling setup times are calculated into the processing time, as a result, there is negligible cost. Liu and Chang [9] show that setup times in production scheduling consume more than 20% of an available machine's capacity. For example, in a printed circuit board assembly, Trovinger and Bohn [10] point out that about 50% of the effective capacity can be lost due to setup times. To the best of this author's knowledge, energy cost, which is an increasing function of processing setup times or unexpected activities of the machine, is usually not considered in the total cost. Usually the cost function used in these problems is associated with lost production cost and/or the scheduling performance of setup times. Therefore, developing an energy-aware scheduling model with multiple sequence-dependent setup times under time-of-use electricity tariffs for the machine may result in significant energy savings. In this environment, the energy-aware scheduling model depends on the electricity rate structure in order to minimize the load in on-peak periods so that energy is saved for the rest of the machine scheduling problem.

Another problem is scheduling sequence-dependent or independent setup times in other machine environments, including parallel machines and a multi-machine, in order to minimize the total energy cost and associated cost function. However, setup times would be performed

according to the model, in order to minimize the total cost (i.e., energy cost, productivity cost, and maintenance cost).

#### **7.2.4 Simple Heuristic for Reducing Number of Variables and Constraints, and Generating Approximate Pareto Front**

Dispatching rules are heuristics, which create a queue of jobs for the machine, and which are easy to perform and calculate. Dispatching rules are not as time consuming as exact methods or metaheuristics, and their use is very common in real time in order to schedule jobs. Usually, dispatching rules are used to provide a quick solution by assessing the available jobs and selecting the next job to process. For example, when one job is completed, dispatch decides which job is going to be scheduled and processed next. Mouzon et al. [11] use a dispatching rule to decide whether a machine should be turned off for a certain amount of time before the next job arrives. This method saves energy consumption by minimizing the machine's expected energy output.

In Chapter 2, an ant colony algorithm was used to solve the model at different times, and scheduling problems can be improved by incorporating a dispatching rule such as SRPT. Further dispatching rules research could be used to overcome complex calculations in metaheuristics. In addition, the performance of these dispatching rules must be analyzed against the performance of the metaheuristics.

Other research could be conducted to study dispatching rules for single- and parallel machine problems and other types of scheduling problems, including energy cost under time-of-use electricity tariffs as an objective to minimizing job energy cost and other scheduling objectives. Consequently, these dispatching rules can quickly obtain a Pareto front in real time for multiobjective scheduling problems.

### 7.2.5 Consideration of Multi-Machine, Multiobjective Scheduling with Energy-Aware Scheduling

In Chapter 5, a parallel machine model to minimize total completion time, total energy cost, and deviation from load balancing was developed. Another scheduling model needs to be developed to solve scheduling problems with multiple machines in series. A specific framework for minimizing energy cost usage via production planning and scheduling problems by incorporating real-time energy costs needs to be designed for this problem. Further research should be focused on more energy savings by considering the flow of jobs from machine-to-machine and considering the flow of a job as a system. Here, cases involving scheduling in the whole system will be a chance for more energy and other cost savings.

### 7.3 References

- [1] E. Mills and A. Rosenfeld, "Consumer non-energy benefits as a motivation for making energy-efficiency improvements," *Energy*, vol. 21, no. 7, Oct. 1996, pp. 707-720.
- [2] R.B. Lung, A. McKane, R. Leach, and D. Marsh, "Ancillary savings and production benefits in the evaluation of industrial energy efficiency measures," *Proceedings of the 2005 American Council for an Energy-Efficient Economy Summer Study on Energy Efficiency in Industry*, Washington, DC, Oct. 2005.
- [3] A.S. Tam, W.M. Chan, and J.W. Price, "Maintenance scheduling to support the operation of manufacturing and production assets," *The International Journal of Advanced Manufacturing Technology*, vol. 34, no. 3-4, Nov. 2007, pp. 399-405.
- [4] M. Celen and D. Djurdjanovic, "Operation-dependent maintenance scheduling in flexible manufacturing systems," *CIRP Journal of Manufacturing Science and Technology*, vol. 5, no. 4, Dec. 2012, pp. 296-308.
- [5] C.R. Cassady and E. Kutanoglu, "Integrating preventive maintenance planning and production scheduling for a single machine," *IEEE Transactions on Reliability*, vol. 54, no. 2, May 2005, pp. 304-309.
- [6] W. Huene and G.P. Kiesmüller, "Maintenance and Production Scheduling on a Single Machine with Stochastic Failures," Technical Report in Arbeitspapiere des Instituts für Betriebswirtschaftslehre, CAU Kiel, May 2015.
- [7] G. Chen, L. Zhang, J. Arinez, and S. Biller, "Energy-efficient production systems through schedule-based operations," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 1, April 2013, pp. 27-37.

- [8] B.E. Nicholson, *Scheduling Shutdowns for Manufacturing Systems with an Application to Automotive Production Lines: Optimization Models and Computation*, ProQuest LLC, Ann Arbor, Michigan, Oct. 2008, pp 43-86.
- [9] C.Y. Liu and S.C. Chang, "Scheduling flexible flow shops with sequence-dependent setup effects," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 4, July 2000, pp. 408-419.
- [10] S.C. Trovinger and R.E. Bohn, "Setup time reduction for electronics assembly: Combining simple (SMED) and IT-based methods," *Production and Operations Management*, vol. 14, no. 2, Feb. 2005, pp. 205-217.
- [11] G. Mouzon, M.B. Yildirim, and J. Twomey, "Operational methods for minimization of energy consumption of manufacturing equipment," *International Journal of Production Research*, vol. 45, no. 18-19, July 2007, pp. 4247-4271.