ABSTRACT

LAGRANGIAN APPROACH TO MINIMIZE MAKESPAN OF NON-IDENTICAL PARALLEL BATCH PROCESSING MACHINES

Nurul Suhaimi, MS
Department of Industrial & Systems Engineering
Northern Illinois University, 2014
Purushothaman Damodaran, Director

Batch Processing Machines (BPMs) are commonly used in electronics manufacturing, semi-conductor manufacturing, and metal-working – to name a few. Scheduling these machines are not an easy task; practical considerations and the exponential number of decision variables involved impede schedulers (or decision makers) from making good decisions. This research focuses on minimizing the makespan of a set of non-identical parallel batch processing machines. In order to schedule jobs on these machines, two decisions are to be made. The first decision is to group jobs to form batches such that the machine capacity is not exceeded. The second decision is to sequence the batches formed on the machines such that the makespan is minimized. Both the decisions are intertwined as the processing time of the batch is determined by the composition of the jobs in the batch. The problem under study is shown to be NP-hard. A mathematical model from the literature is adopted to develop a solution approach which would help the decision maker to make meaningful decisions.

Lagrangian Relaxation approach has been shown to be very effective in solving scheduling problems. Using this decomposition approach, the mathematical model is decomposed and a sub-gradient approach was used to update the multipliers. Two sets of constraints were relaxed to consider two Lagrangian Relaxation models. Experiments were conducted with data sets from the literature. The solution quality of the proposed approach was compared with meta-heuristic (i.e. Particle Swarm Optimization (PSO) and Random Key Genetic Algorithm (RKGA)) published in the literature and a commercial solver (i.e. IBM ILOG CPLEX). On smaller instances (i.e. 10 and 20 jobs), the proposed approach outperformed PSO and RKGA. However, the proposed approach and CPLEX report the same results. On larger instances (i.e. 50, 100 and 200 job instances) with two and four-machines, the proposed approach was better than PSO whenever the variability in the processing times were smaller. The proposed approach generally outperformed RKGA and CPLEX on larger problem instances. Out of 200 experiments conducted, the proposed approach helped to find new improved solution on 34 instances and comparable on 105 instances when compared to PSO. The PSO approach was much faster than all other approaches on larger problem instances. The experimental study clearly identifies the problem instances on which the proposed approach can find a better solution. The proposed Lagrangian Relaxation solution approach helps the schedulers to make more informed decisions. Minor modifications can be made to use the proposed solution approach for other practical considerations (e.g. job ready times, tardiness objective, etc.) The main contribution of this research is the proposed solution approach which is effective in solving a class of non-identical batch processing machine problems with better solution quality when compared to existing meta-heuristic.

NORTHERN ILLINOIS UNIVERSITY
DEKALB, ILLINOIS

DECEMBER, 2014

LAGRANGIAN APPROACH TO MINIMIZE MAKESPAN OF NON-IDENTICAL PARALLEL BATCH PROCESSING MACHINES

BY

NURUL MUBINAH BINTI SUHAIMI

A THESIS SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF INDUSTRIAL & SYSTEMS ENGINEERING

Thesis Director:
Dr. Purushothaman Damodaran

ACKNOWLEDGEMENTS

Many individuals have contributed their expertise, time, and resources to various areas of this study, and to them I am most grateful. I would first like to express my deepest gratitude to my thesis advisor, Dr. Purushothaman Damodaran, from whom I have learned so much, and who has helped me shape and focus every aspect of this project. By sharing with me his exceptional knowledge of scheduling and optimization research, I have gained a deeper understanding of this research area which helped me building a better future path. As a mentor, he not only guided my research skill but also helped me to become a more confident student and scholar. I am indebted to my lecturer who was also the committee member for this thesis, Dr. Murali Krishnamurthi who always went above and beyond to help his students. Learning with him was indeed a true blessing. I am also indebted to Dr. Omar Ghrayeb and Dr. Christine Nguyen who have been a great support system in realizing this thesis. Their commitment to be the committee members was priceless. I would also like to thank my great friends who were there to help me in times of need throughout the period of this thesis. To Nishanth Yalam, learning with you has never been more fun and enjoyable. To Varun Kamarajan, Marianella Padilla, and Aziz Alomrani, your endless support and encouragement have been the pillars of my strength to keep moving on. To Maliha Afreen, Rabiah Anwar, Samar Rahman, and Tahanie Omar, all of you are my beacon of light when it is all dark in the United States. From all of you, I learned the art of being thankful.

DEDICATION

For the kindest, most generous spirit I have ever known,

and that what is best in me, I owe to her

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

The term “batch processing” is used to describe a system where data was collected together for a period of time before it was processed. Instead of processing every small job as it arrived, jobs were queued until the processer was ready to process them all at once. Similarly in manufacturing operations, the batch processing machines (BPM) act as the processor whereas a job remains as a job. BPM are capable of processing several jobs in a batch form concurrently as long as it does not exceed the machine capacity. Although processing multiple jobs simultaneously could be an advantage, the variability they have in terms of job size and processing time could be a drawback. Consequently, efficiently scheduling the BPM is vital in order to maximize the machine utilization (or minimize the makespan). Research in the area of scheduling batch processing machines has gained great attention in the past [1]. There are several exact and heuristic solution approaches proposed for different machine configurations with BPM. Generally, scheduling BPM are computationally difficult, even for simpler objectives such as makespan, and it is a common practice to propose heuristic and meta-heuristic solution approaches [1]. Heuristic and meta-heuristic solution approaches are known to be the best neighborhood search, but Lagrangian Relaxation method is able to find the optimal solution either locally or globally [3].

This research is motivated by a real life application in contract electronics manufacturing (i.e. an electronics manufacturer who assembles and tests printed circuit boards (PCBs) which are used in consumer products). In the company, environmental stress screening (ESS) chambers are used to test PCBs in order to detect early failures before they are used in the field. These chambers are capable of testing several PCBs concurrently. Hence, the chambers are equivalent to BPM. The PCBs are regarded as jobs in this research. The scheduler's responsibility is to form batches and schedule the batches on the chambers so that the completion time of the last batch is minimized (or maximize the utilization of the chambers). When the batches are formed, the composition of the batch determines the processing time of the batch. The batch processing time is equal to the job in the batch with the longest processing time. In electronics manufacturing, it is acceptable to test a job for longer than its prescribed testing time. However, testing a job for a longer time than its prescribed time will result in a waste of valuable chambers availability.

The objective of this research is to find the optimal schedule for non-identical parallel batch processing machines in order to minimize the makespan. Damodaran, et al. [4] showed that the problem under study is NP-hard. They proposed a Particle Swarm Optimization (PSO) algorithm. PSO is a meta-heuristic and does not guarantee an optimal solution. This research proposes a Lagrangian Relaxation (LR) approach for the problem studied in [4] and compares the effectiveness of the approach (in terms of solution quality and computational time) to PSO and other approaches found in the literature.

1.1 Problem Description

The problem considered in this research is described as follows. For a set J of n jobs, $J = \{J_1, \dots, J_n\}$, the jobs should be grouped to form the batches and the batches are then scheduled on a set M of q non-identical parallel batch processing machines, $M = \{M_1, \dots, M_q\}$. The processing time of each job (p_j), size of each job (s_j) and each machine's capacity (S_m) is known. The maximum number of batches required to process all the jobs is easy to determine. Assuming one job per batch will result in n batches, hence, the maximum number of batches needed is also n. The batch processing time is equal to the longest processing time of all the jobs in the batch. Eventually, the total size of all the jobs in the batch should not exceed the machine capacity in which it is processed. The objective function is to minimize the makespan (i.e. completion time of the last batch of jobs).

In order to schedule the batch processing machines, two decisions need to be made. First, jobs need to be grouped into batches and second, the batches formed need to be scheduled. Both decisions are considered dependent on each other as the formation of the batch determines the batch processing time which then affects the makespan.

1.2 Research Objective and Scope

The objective of this research is to develop LR approach for scheduling non-identical parallel batch processing machines to minimize the makespan. The effectiveness of the proposed approach will be evaluated by solving benchmark problems from the literature and comparing the solution with other meta-heuristic previously published in the literature. A comparative study will also be conducted to evaluate the solution quality with the solution obtained from a commercial solver (i.e. CPLEX). LR approach is capable of finding optimal or near-optimal solution for certain class of problems. This research study will explore the LR approach to verify if it is efficient to non-identical parallel batch processing machines with makespan objective. The following assumptions are made in this research and are consistent with the assumptions made in other relevant research work:

1. All jobs in a batch will begin and end processing at the same time.
2. All jobs are available at time 0.
3. The machines do not fail.
4. Once a machine begins processing a batch, new jobs cannot be added to the batch or removed from the batch.
5. When a machine completes processing a batch of jobs, the next batch can be immediately loaded with no time lost in set up or delay due to operator availability.

1.3 Organization

The thesis report is organized as follows: Chapter 2 reviews literatures on BPM and LR approach. Chapter 3 describes the mathematical model for the problem under study and Chapter 4 presents the LR approach to solve the problem. Chapter 5 presents the experimental study conducted. Finally, Chapter 6 concludes the research as well as proposes recommendations for future research.

CHAPTER 2

LITERATURE REVIEW

Research on scheduling problems started as early as in 1960 by Hanssman and Hess [5] when they implemented linear programming approach to production planning and employment scheduling. To date, the research in machine scheduling has grown into various machine environments, objectives, and methodologies. Machine environments in scheduling problems are broadly divided into two major groups: discrete processing machines and batch processing machines. Under machine environments, the scheduling problems can also be classified as single machine, parallel machine, flow shop, hybrid flow shop, job shop, and open shop problems.

The objectives considered varied from minimizing cost, tardiness, number of tardy jobs, completion times, and makespan. The solution approaches proposed also varied from exact methods (e.g. Branch-and-Bound, Dynamic Programming, Column Generation, etc.) to heuristic methods (e.g. Simulated Annealing, Genetic Algorithms, etc.). This research focuses on minimizing makespan on a set of batch processing machines (BPM) that can process several jobs in a batch simultaneously with the Lagrangian Relaxation (LR) approach. Table 1 lists the literature reviewed for this research.

Table 1: Literature Review Table

Source	Single BPM	Parallel BPM		Scheduling Objective		Methodology	
		Identical	Non-Identical	Other Objective	Makespan	Other Approach	Lagrangian Approach
[2]		✓			✓	✓	
[4]			✓		✓	✓	
[6]	✓			✓		✓	
[7]	✓				✓	✓	
[8]		✓			✓		✓
[10]	✓			✓		✓	
[11]		✓		✓		✓	
[12]	✓			✓		✓	
[13]		✓		✓		✓	
[15]		✓			✓		✓
[16]		✓			✓	✓	
[17]	✓			✓		✓	
[21]		✓			✓	✓	
[22]		✓			✓	✓	
[23]		✓			✓	✓	
[24]			✓		✓	✓	
[25]		✓			✓	✓	

(continued on following page)

Table 1 (continued)

Source	Single BPM	Parallel BPM		Scheduling Objective		Methodology	
		Identical	Non-Identical	Other Objective	Makespan	Other Approach	Lagrangian Approach
[26]		✓			✓		✓
[27]							✓
[31]				✓			✓
[32]				✓			✓
[33]				✓			✓
[34]				✓			✓
[35]					✓		✓
[36]					✓		✓
[38]		✓		✓		✓	
[39]	✓				✓	✓	
[40]		✓			✓	✓	
[41]		✓			✓	✓	
Suhaimi, 2014			✓		✓		✓

2.1 Batch Processing Machine

In BPM, jobs are grouped to form batches and processed at the same time simultaneously. Potts and Kovalyov [1] reviewed the literature on scheduling with batching decisions. The jobs are typically batched when they share the same setup on a machine or when the machine can process multiple jobs simultaneously [1]. The applications of BPM can be found in aeronautical industry [38], hospital sterilization services [2], electronic manufacturing [4] and many more.

2.1.1 Single Batch Processing Machine Environment

Uzsoy [6] shows that scheduling a single BPM with non-identical job sizes to minimize total completion times and makespan is NP-hard. Lee and Uzsoy [7] proposed polynomial and pseudo-polynomial time algorithm to minimize makespan of single BPM with dynamic job arrivals (i.e. job release times are not equal). The algorithm presented excellent results but took long computational time. Chandru et al. [12] studied Branch-and-Bound (B&B) algorithms to minimize total completion time or makespan in single BPM. The set of jobs to be scheduled are grouped into a number of families, where all jobs in the same family have the same processing time. In single BPM with non-identical job sizes, Perez et al. [10] proposed heuristic to minimize the total weighted tardiness in the diffusion step of semiconductor wafer fabrication process. Velez-Gallego et al. [39] studied constructive heuristics named as Modified Successive Knapsack Problem (MSKP) to minimize makespan in single BPM under the assumptions of non-identical job sizes and non-zero job ready times. The heuristic found to outperform other comparative approach for test instances with 50 or more jobs.

2.1.2 Parallel Batch Processing Machines Environment

Both parallel and single BPM required jobs to be formed in batches before they are processed. The formation resulted in a batch where the size of the batch is the sum of all the size of the jobs in the batch. In any circumstances, the batch size should not exceed the capacity of the machine that processes it [2]. In this research, literature on parallel BPM is divided into two categories: identical and non-identical parallel BPM. Identical BPM simply means that all the machines have the same capacity. As for non-identical, the capacities of the BPM are vary.

Identical parallel BPM scheduling problems have been widely researched. Ozturk et al. [16] applied classical bin packing heuristic to minimize makespan of washing medical devices operations. The washers used to wash the Reusable Medical Devices (RMD) were considered as identical parallel BPM. Damodaran et al. [41] considered Greedy Randomized Adaptive Search Procedure (GRASP) in minimizing makespan for parallel BPM in an electronics manufacturing company. The proposed GRASP approach outperformed other approaches mentioned in their paper and guaranteed optimal solution for 10 job problem instances. Damodaran et al. [15] applied Simulated Annealing (SA) algorithm to minimize the makespan on a group of identical parallel BPM and compared the results with GRASP in [41]. The proposed approach shows a comparable result to GRASP in terms of solution quality and computation time. Mokotoff [21] shows that minimizing makespan for identical parallel BPM can be done with a new approximation algorithm based on Linear Programming (LP). The MSKP heuristic in [39] was extended by Damodaran et al. [40] to identical parallel BPM and named it as Progressive Successive Knapsack Problem (PSKP). The heuristic aimed to minimize makespan with lesser computational time.

Another objective that was studied in identical parallel BPM is tardiness. Tardiness is equal to lateness where the completion time of the job is greater than its due date [17]. Genetic Algorithm (GA) approach was found to be able to minimize tardiness thus providing better results in larger size and difficult problems than the comparing neighborhood exchange search [11]. B&B algorithm was implemented in identical parallel BPM to minimize tardiness and was compared with traditional SA solution [13].

Non-identical parallel BPM offers a harder problem than identical BPM scenario as the machine capacity considerations are required for every batch formed. Common scheduling objective studied for non-identical BPM scenario is minimizing the total completion time or makespan [2]. One example of the research is done by Xu and Bean [24] where they presented Random Key Genetic Algorithm (RKGA) in minimizing makespan for non-identical parallel BPM. Following that, Damodaran et al. [4] proposed Particle Swarm Optimization (PSO) approach to minimize the makespan for non-identical parallel BPM. They adapted the mathematical formulation from [24] and simplified them with fewer binary variables. This research also considers non-identical parallel BPM. This research compares its result with RKGA method in [24] and PSO method in [4].

2.2 Minimize Makespan in Batch Processing Machines

Minimizing makespan has been one of the commonly studied objectives when it comes to scheduling problems. Majority of the literature considers the case of non-identical jobs and identical machines [20]. Earlier study on this objective may have been done by [2] where non-identical job size is considered. The methods used to minimize the makespan objective vary from exact approaches like B&B and Dynamic Programming (DP) to meta-heuristic approaches such as SA and PSO. Chang et al. [22] developed SA algorithm in identical parallel BPM to solve the makespan minimization problem. They compared the proposed approach with CPLEX and found the approach to outperform CPLEX in most of the instances. Kashan et al. [23] developed a Hybrid Genetic Heuristic (HGH) to minimize makespan with the same environment as [22] and compared the two approaches. From the computational experiments performed, HGH was shown to find optimal or near-optimal solutions faster than SA approach presented in [22].

Xu and Bean [24] solved their mathematical formulation using a standard programming software. But due to the difficulty of the problem, RKGA was proposed to schedule the non-identical parallel BPM. Shao et al. [25] used neutral networks approach to minimize the makespan with non-identical job sizes. The coding of neural networks approach is provided with a new method, Master Weight Matrix which was proven to be effective towards large-scale problem when compared with other heuristic. Li and Yuan [26] considered scheduling with an approximation algorithm called polynomial-time approximation schemes for both bounded and unbounded cases. In 2012, Damodaran et al. [4] presented PSO approach to minimize the makespan by grouping the PCBs into batches and schedule them with known testing times and non-identical PCB sizes. The PSO results were compared with a commercial solver and the RKGA approach presented in [24].

2.3 The Lagrangian Relaxation Approach

The origin of Lagrangian approach is dated back to the early 1970s when Held and arp [28] used a Lagrangian problem based on minimum spanning trees to formulate a successful algorithm for the Traveling Salesman Problem [29]. With the breakthrough, the encouragement to apply Lagrangian method in most general integer programming problem has gained ever since. Additionally, Fisher [29] also stated in the research that Lagrangian method is able to provide the best solution of any practical size for most scheduling or optimization problems. The observation viewed many hard optimization problems as easy problems but complicated by a relative set of side constraints [30]. By dualizing the side constraints, solution by Lagrangian methods can be produced where the optimal solution is either the lower bound (for minimization problems) or upper bound (for maximization problems) on the optimal value of the original problem [30]. Moreover, the Lagrangian problem can be used in linear programming relaxation to provide bounds in Branch-and-Bound algorithm.

2.3.1 Scheduling with Lagrangian Relaxation Approach

The first scheduling problem solved with Lagrangian Relaxation (LR) approach was as early as in 1975 by Muckstadt and Koenig [31] when they applied the method in scheduling a power generation system. They presented a mixed integer programming model to minimize the sum of unit commitment and economic dispatch costs subject to demand, reserve, generator capacity and generator schedule constraints. Lagrangian method is used to decompose the problem into single generator and sub-gradient method is used to update the Lagrange multipliers. In 1993, Luh [32] examined practical solution in three manufacturing scheduling problem. Each problem is formulated by adding and modifying constraints to increase the real world complexity. Again, LR is used to decompose the scheduling problems into job-level sub-problem. The sub-problem is easier to solve than the main problem and resulted in near-optimal schedule. To reduce computation time, Luh [32] used Lagrange multiplier from the last schedule to initialize the multiplier in the next instance.

Liu, Luh and Resch [33] obtained near-optimal schedule in critical stages of flow shop manufacturing with LR technique. Instead of using sub-gradient method to update the multiplier, they used a high level Reduced Complexity Bundle Method implying that the quality of the schedule is better than the common sub-gradient method. With the algorithm, the penalties on the production tardiness were minimized effectively. Irohara [34] proposed LR algorithms for Hybrid Flow-Shop (HFS) scheduling problems. They minimized the total weighted tardiness and earliness for the job. In this study, the usual relaxation of machine capacity is not only done, but also the precedence constraints. Perdomo et al. [35] applied Lagrangian technique in surgery room operations. They addressed the approach to minimize the completion time from assigning the patients to operating rooms and recovery beds.

2.3.2 Minimize Makespan with Lagrangian Approach

The Lagrangian approach has been used widely in the study of makespan minimization [8, 27, 36]. Velde [36] presented B&B algorithm for two-machine flow shop problem in order to minimize the sum of the job completion times. LR is used to provide the lower bounds for the problem. The algorithms were then compared with previous researches and were proven to outperform them. Chen, Chu, and Proth [27] used Lagrange multipliers to relax the capacity constraints on machines. Instead of using the basic decomposition method on the relaxed problem to break the problem into job level sub-problem, they proposed a pseudo-polynomial time Dynamic Programming (DP) algorithm to prevent oscillation in the solution. Through LR, the algorithm was able to find the optimal solution based on the "min-max" criteria for the job-shop scheduling. Tang, Xuan, and Liu [8] designed a DP algorithm for solving identical parallel machine sub-problems where jobs have negative weights. LR algorithm was used and decomposition was applied to decompose the master problem into several sub-problems so that the DP algorithm process is speeded up.

2.4 Summary

This research presents LR approach to solve makespan minimization problem in non-identical parallel BPM. Past literatures show that the problem under study is typically solved using heuristic and meta-heuristic. LR approach has worked for different types of scheduling problems but has not been tried to solve makespan minimization in non-identical parallel BPM environment.

CHAPTER 3
MATHEMATICAL FORMULATION

Xu and Bean [24] presented a Mixed Integer Linear Programming (MILP) model for minimizing the makespan of non-identical parallel batch processing machines with non-identical job sizes and equal release dates. Damodaran et al. [4] simplified the model in [24] with fewer binary variables. The notation used in the model proposed by Damodaran et al. [4] is presented below.

Sets

$\{j \in J\}$	Set of jobs
$\{k \in K\}$	Set of positions on each machine
$\{m \in M\}$	Set of machines

Parameters

p_j	Processing time of job j
s_j	Size of job j
S_m	Capacity of machine m

Decision variables

P_{km}	Processing time of kth batch scheduled on machine m
C_{max}	Makespan

$$X_{jkm} = \begin{cases} 1, \text{ if job } j \text{ is assigned to the } k^{\text{th}} \text{ batch processed on machine } m \\ 0, \text{ otherwise} \end{cases}$$

The MILP proposed by Damodaran et al. [4] is presented below. This model is referred to as model *P* from hereon.

Mixed Integer Linear Programming (P)

$$\text{Minimize } C_{max} \tag{1}$$

subject to:

$$\sum_{k\in K}\sum_{m\in M} X_{jkm} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j\in J} s_j X_{jkm} \le S_m \qquad \forall k \in K, m \in M \tag{3}$$

$$P_{km} \ge p_j X_{jkm} \qquad \forall j \in J, k \in K, m \in M \tag{4}$$

$$C_{max} \ge \sum_{k\in K} P_{km} \qquad \forall m \in M \tag{5}$$

$$P_{km} \ge 0 \qquad \forall k \in K, m \in M \tag{6}$$

$$C_{max} \ge 0 \qquad \forall m \in M \tag{7}$$

$$X_{jkm} \in \{0, 1\} \qquad \forall j \in J, k \in K, m \in M \tag{8}$$

The objective function (1) is to minimize the makespan. Constraint (2) ensures that each job is assigned to exactly one batch on one machine. Constraint (3) ensures that the total size of all the jobs assigned to the batch does not exceed the machine capacity. Constraint (4) ensures that the processing time of the k^{th} batch processed on each machine is at least equal to the longest processing time job in the batch. Constraint (5) determines the makespan. Makespan in this study is at least equal to the sum of all batch processing times on each machine. Constraints (6), (7) and (8) impose the non-negativity and binary restrictions on the decision variables.

In order to ensure a fair comparison with the previous approaches, the same data sets used in [4] were also used in this computational study. Table 2 describes the factors and levels used to generate the datasets for the problem under study. The number of jobs, n, considered in each problem instance is 10, 20, 50, 100, and 200. The processing times, p_j, were generated from a Discrete Uniform distribution with parameters from 1 to 10 and from 1 to 30. The sizes of the jobs, s_j, were generated from a Discrete Uniform distribution with parameters from 1 to 5 and from 1 to 20. Number of machines, m, considered were two and four. For the two machine instances, the capacity of the machines, S_m, were assumed to be 20 and 25, whereas for the four machines, the capacity of the machines were assumed to be 18, 20, 25 and 27. For each combination of the factors discussed above, five instances were generated. Altogether, 200 experiments were performed to evaluate the proposed solution approach. The solutions from the proposed approach were compared to solutions from RKGA by Xu and Bean [24] and PSO by Damodaran et al. [4]. In [4], the experimental study conducted clearly indicate that commercial mixed integer programming solvers (such as CPLEX) would take prohibitively long run time to solve modest size problems. Consequently, in this research LR approach is proposed. More details on the proposed solution approach is presented in Chapter 4.

Table 2: Factors and Levels

Factors	**Levels**
n	10, 20, 50, 100, 200
p_j	Discrete Uniform [1, 10] and Discrete Uniform [1, 30]
s_j	Discrete Uniform [1, 5] and Discrete Uniform [1, 20]

S_m	Two machines : {20, 25}
	Four machines : {18, 20, 25, 27}

CHAPTER 4
LAGRANGIAN RELAXATION APPROACH

The basic concept of Lagrangian Relaxation (LR) approach is to form a relaxed problem from the original problem by introducing the constraints into the objective function using a vector of Lagrangian multipliers. For a given set of Lagrangian multipliers, a Lagrangian Dual (LD) problem is solved. The Lagrangian multipliers are updated using a sub-gradient algorithm. Using the new multipliers, the LD is solved again to find a lower bound (for a minimization problem) or upper bound (for a maximization problem). This procedure is repeated until a termination criterion is met.

Figure 1 shows the process flow of the proposed LR approach. The linear programming relaxation is solved and the solution is noted as LP. Initially, the best known upper bound (UB_{best}) value is set to the sum of all job processing times. The best lower bound (LB_{best}) value is set to a large negative value. The lower bound (LB) value is obtained by solving the formulation obtained through LR. If the LB value is better (i.e. higher) than LB_{best}, the LB_{best} is set to LB. Else, then LB_{best} is not updated and a counter is increased by 1. When the counter reached a pre-determined limit, the scale parameter used in the Lagrangian approach is adjusted. The lower bound solution obtained need not guarantee integer feasible solution. Consequently, the infeasible solution is fed to an upper bound model after some data manipulation (through a heuristic) to determine an integer feasible solution. If the upper bound

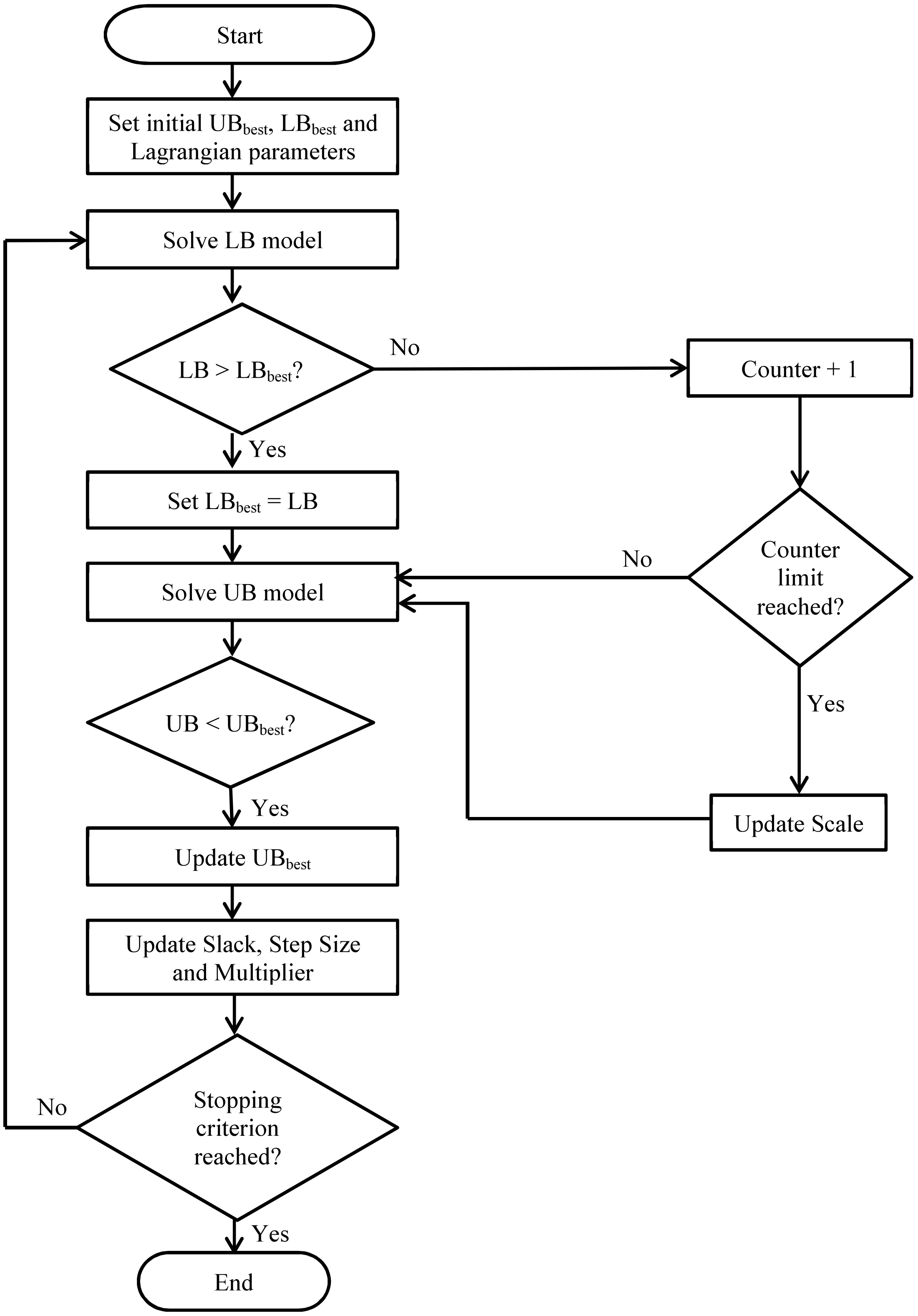


Figure 1: Flow Chart of Lagrangian Relaxation Approach

(UB) determined is better (i.e. less) than the UB$_{\text{best}}$, then the UB$_{\text{best}}$ is updated to UB. Slack is determined using the lower bound solution and then step size and multiplier is updated. With the revised multiplier, the LB model is solved again and the process continues until the stopping criterion is satisfied. In this study, two sets of constraints are relaxed to determine lower bounds for the problem under study. As the resulting solution from the relaxed problem may be infeasible for the original problem, a heuristic is applied to obtain a feasible schedule from the infeasible schedule.

For a given set of multipliers, the relaxed problem is solved. The set of multipliers is updated based on the degree to which the constraints of the relaxed solution are violated. With the new set of multipliers, an updated relaxed problem is then formulated and solved. This process continues until the termination criterion is reached. The solution quality is measured by the relative duality gap explained later in the chapter. The dual objective value provides a lower bound (Z_{LB}) for the optimal value of the original problem, while the objective value (Z_{UB}) of the feasible schedule is an upper bound for the problem. The details for the solution of the sub-problems, updating the Lagrangian multipliers, and construction of upper bound heuristic are presented in the following sections.

4.1 Lagrangian Relaxation

In this study, two sets of constraints are relaxed independently to get two Lagrangian Relaxation models (i.e. LR1 and LR2). In each proposed relaxation, the objective function can be formed by using a vector of the Lagrangian multipliers, λ_{km} (when constraint 3 from model *P* is relaxed) and λ_j (when constraint 2 from model *P* is relaxed).

LR1: Relaxation of size constraint (Equation 3)

$$Z_{LR1}(\lambda_{km}) = \min C_{max} - \sum_k \sum_m \lambda_{km}(S_m - \sum_j s_j X_{jkm}) \tag{9}$$

subject to: (2), (4), (5), (6), (7) and (8)

which leads to the Lagrangian Dual (LD1):

$$Z_{D1} \quad = \max Z_{LR1}(\lambda_{km}), \text{ where } \lambda_{km} \geq 0. \tag{10}$$

LR2: Relaxation of "one" constraint (Equation 2)

$$Z_{LR2}(\lambda_j) = \min C_{max} - \sum_j \lambda_j(1 - \sum_k \sum_m X_{jkm}) \tag{11}$$

subject to: (3), (4), (5), (6), (7) and (8)

which leads to the Lagrangian Dual (LD2):

$$Z_{D2} \quad = \max Z_{LR2}(\lambda_j), \text{ where } -\infty \leq \lambda_j \leq \infty. \tag{12}$$

Based on the duality theorem, the problem can be viewed in two perspectives which are the primal problem and the dual problem. In this study, the solution from the dual problem (Lagrangian Dual) provides a lower bound to the solution of the primal problem (i.e. presented in Chapter 3). The Lagrangian Dual (LD) is solved for different values of the multiplier. The multiplier value is updated using the classical sub-gradient approach. The multiplier update procedure is presented in section 4.2.

4.2 Updating Lagrangian Multipliers

In order to solve the dual problem, the classical sub-gradient method is adopted for updating the Lagrangian multipliers. Here, the vectors of Lagrangian multipliers, λ_{km} and λ_j are updated by (13) and (14).

$$\lambda_{km} = \max(0, \lambda_{km} + \varphi_{km}^{\lambda}\mu_{km}^{\lambda}) \tag{13}$$

$$\lambda_j = \lambda_j + \varphi_j^{\lambda}\mu_j^{\lambda} \tag{14}$$

where φ_{km}^{λ} and φ_j^{λ} are slacks (see equations 15 and 16), and μ_{km}^{λ} and μ_j^{λ} are the step sizes (see equations 17 and 18).

$$\varphi_{km}^{\lambda} = S_m - \sum_{j\in J} s_j X_{jkm}, \qquad \forall k \in K, m \in M \tag{15}$$

$$\varphi_j^{\lambda} = 1 - \sum_k \sum_m X_{jkm}, \qquad \forall j \in J \tag{16}$$

$$\mu_{km}^{\lambda} = \frac{\sigma(Z_{UB} - Z_{LB})}{\sum_{m\in M}\sum_{k\in K}(\varphi_{km}^{\lambda})^2} \tag{17}$$

$$\mu_j^{\lambda} = \frac{\sigma(Z_{UB} - Z_{LB})}{\sum_{j\in J}(\varphi_j^{\lambda})^2} \tag{18}$$

where Z_{UB} is the upper bound value of the best known feasible solution, Z_{LB} is the lower bound value found by solving the Lagrangian, and scale (σ) is between 0 and 2. Typically, the scale parameter is reduced by half whenever the solution of the Lagrangian problem fails to improve after a pre-determined number of iterations (N_{repeat}). The pseudo code for solving the LD1 is given below:

```
Begin
    $itercount \leftarrow 0$
```

```
Let the objective of the linear programming relaxation of $P$ be $Z_{LP}$
$LP \leftarrow Z_{LP}$
Set the initial parameter:
$\sigma \leftarrow 2, \lambda_{km} \leftarrow 0$
Initial Lower Bound (LB) is the solution of Lagrangian Relaxation model (LR1) as shown in equation (9) using the initial parameter:
$LB \leftarrow Z_{LR1}$
Initial Upper Bound (UB) is set to the total processing time for all jobs:
$UB \leftarrow \sum_{j \in J} p_j$
while $(LP - LB) \leq 0.00$
    $itercount \leftarrow itercount + 1$
    $LB \leftarrow Z_{LR1}$
    if $C_{max}(P) + 0.00001 < LB$, then
        $C_{max}(P) \leftarrow LB$
    else
        $repeat \leftarrow repeat + 1$
    end if
    if $repeat = N_{repeat}$, then
        $\sigma = \frac{\sigma}{2}$
        $repeat \leftarrow 0$
    end if
```

```
        Let $x$ be the current job to batch assignment from the current $LB$ and $H$ be the job
        assignment obtained from a heuristic (refer to Section 4.3)
        $H \leftarrow Heuristic(x)$
        Solve the Upper Bound model with the input from $H$
        $C_{max}(H) \leftarrow H$
        **If** $UB \geq C_{max}(H)$ **then**
            $UB \leftarrow C_{max}(H)$
        **end if**

        Compute the slack, $\varphi_{km}^{\lambda}$ as in equation (15)
        Compute the step size $\mu_{km}^{\lambda}$ as in equation (17)
        Update the Lagrangian multipliers, $\lambda_{km}$ as in equation (13)
        Solve $LR1$
        $LB \leftarrow Z_{LR1}$
    **end while**
**End**
```

The procedure to obtain a lower bound from LR2 is similar to the above pseudo-code, except that the Lagrangian model is as in equation (11), the step sizes are updated using equation (18), slack is computed using equation (16), and multipliers are updated by using equation (14).

4.3 Upper Bound Heuristic

Solving the Lagrangian Dual to optimality does not guarantee a feasible solution. The feasibility is retained by applying a simple heuristic. Some of the *X* decision variables from solving LR1 (or LR2) may take fractional values. With the heuristic, the integer feasibility is retained when solving the upper bound model. The pseudo code for the heuristic to determine the upper bound is shown below:

```
***Heuristic(x)***
**Begin**
    $LB \leftarrow Z_{LR1}$
    Let x be the job to batch assignment from the LB.
    For each x, let $H$ be the job to batch assignment obtained from the heuristic.
    **if** $x > 0$, **then**
        $H \leftarrow 1$
    **else**
        $H \leftarrow 0$
    **end if**
    For each H, let X be the job to batch assignment in UB model.
    **if** $H > 0$
        $X \leq H$
    **else**
        $X \leftarrow 0$
    end if
End
```

The heuristic begins after LB model is solved. The decision variable solution obtained from the LB model is used to set the upper bounds for all the decision variables in the UB model. For example, if $X_{jkm} = 0$ in the LB solution, then the corresponding X_{jkm} in the UB is bounded above by 0. In essence, this variable is fixed to zero. If X_{jkm} is a fractional value or 1 in the LB solution, then the corresponding X_{jkm} in the UB is bounded above by 1. After establishing these bounds, the UB model is solved to obtain the upper bound which would be a feasible solution for the problem under study as the X variables are defined as binary variables in the UB model.

The LR approach was implemented in IBM ILOG CPLEX. A computational study was conducted to evaluate its performance, in terms of its solution quality and computational time. Details of the experimental study are presented in Chapter 5.

CHAPTER 5
COMPUTATIONAL RESULTS

In Chapter 4, two constraints were relaxed to obtain two different Lagrangian Duals. Experiments were performed to evaluate the two Lagrangian Duals. The Lagrangian Relaxation (LR) approach presented in Chapter 4 was implemented in IBM ILOG CPLEX. The experiments were run on a personal computer with the following configuration: Intel® Core™ i5-2450M, 2.50 GHz, and 4 GB RAM. Data sets were generated as presented in Chapter 3. Overall, 200 experiments were conducted for each relaxation. Results from this study are compared with RKGA, PSO and CPLEX in terms of solution quality and computational time.

In comparing the solution quality, the average percentage of improvement (API) is used and calculated as follow:

$$API = \frac{C_{max}(Previous\ methods) - C_{max}(Lagrangian)}{C_{max}(Previous\ methods)} \times 100\% \tag{19}$$

From the equation, it can be inferred that the LR solution is equal or better than the other approaches with an API value equal to zero or more. On the other hand, the previous approaches are better than LR if the API value is less than zero.

A Design of Experiment (DOE) was conducted to choose the parameters used in the LR approach. The parameters that were decided through DOE are number of positions (k), same limit (N_{repeat}), and multiplier update (λ^n). Twenty job problem instances were used in the DOE. Through DOE, the effect of different parameters on the solution quality was determined and a main effects plot was developed (see Figure 2). The levels for the different factors were defined as follows: the number of positions was to be decided between 6 (33% of the number of jobs) or 10 (50% of the number of jobs). As the number of positions considered increased, the computational time was longer. However, fewer positions may lead to poorer solution with short run times. The same limit was between 5 or 15. This is the number of iterations allowed in the LR approach before the scale parameter is reduced by half. Multiplier update is the scale used to update the multiplier in case the multiplier update method explained in Chapter 4 resulted in a negative value. In this DOE, the multiplier update tested is 0%, 33%, or 50% from the current multiplier value. Figure 2 shows that the best parameter to achieve a better API when compared to PSO is if the number of positions is 50% of the number of jobs, same limit is 15, and multiplier update is 0. These parameters were used throughout the experimental study for 10, 20 and 50 job instances.

For 100 and 200 job instances, the LR parameters set earlier through DOE leads to an inferior solution even after a long computation run. Due to this reason, another DOE is performed. In this DOE, only the number of positions is considered. The same limit and multiplier update do not have any impact on the results for the larger instances. Number of positions considered in this DOE is 25%, 50% and 60% of the number of jobs. The result of the DOE is as shown in Figure 3 when the API is compared to PSO. Through the DOE, 60% of the number of jobs is chosen as the number of positions to consider in each 100 and 200 job instance. The number of positions considered increased with the number of jobs.

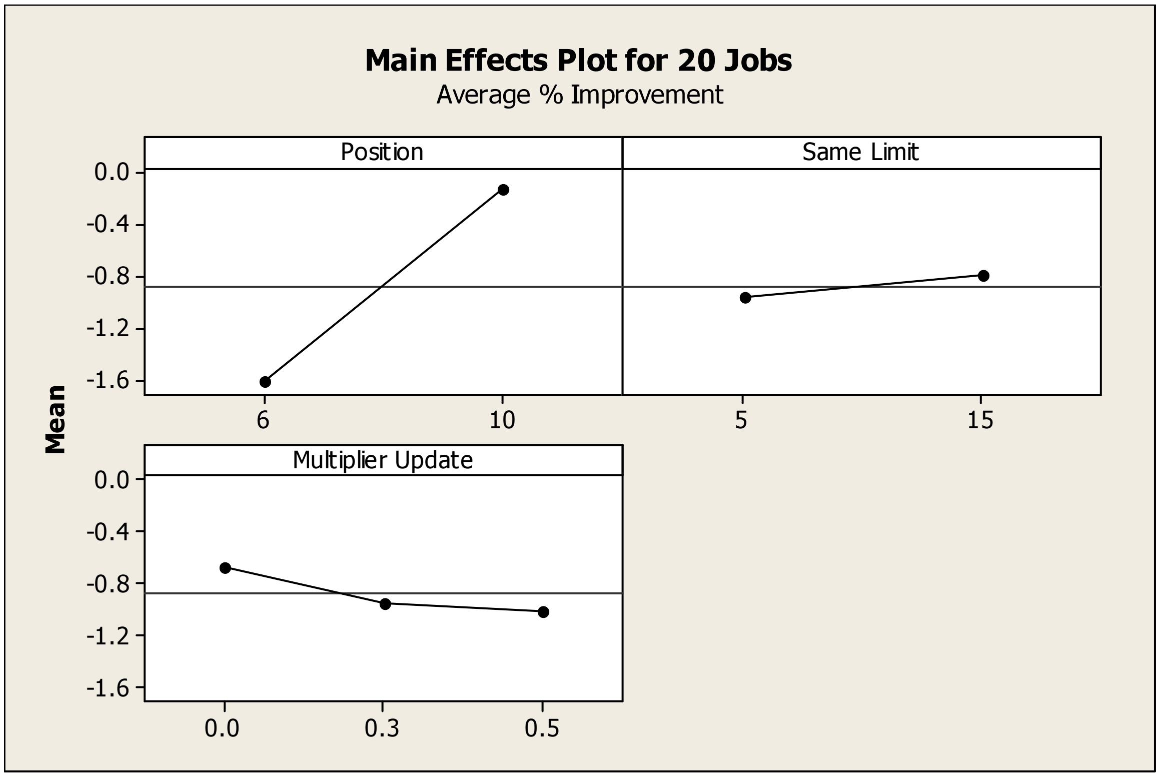


Figure 2: Main Effects Plot for 20 Jobs

Some other parameters were also set in the beginning of the experiments. These parameters do not have a significant effect on the results, therefore a formal DOE was not performed. The parameters are as follows:

initial multiplier (λ_{km}) = 0.5

scale (σ) = 2

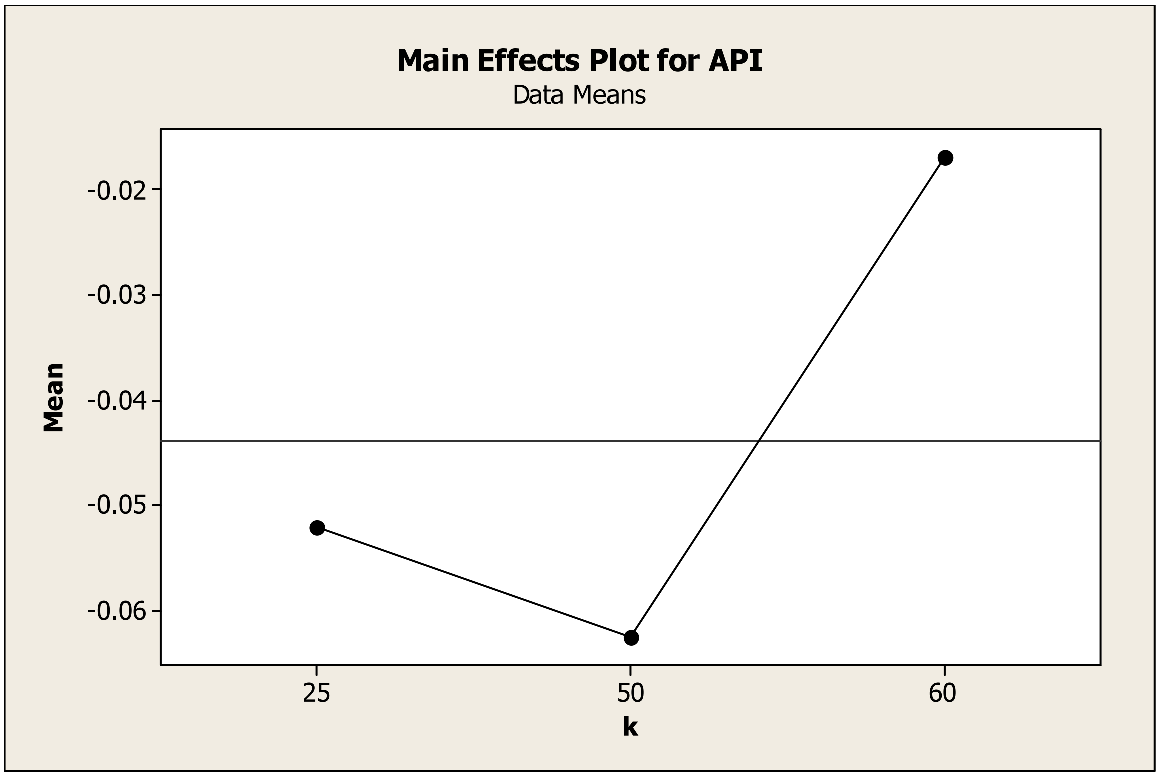


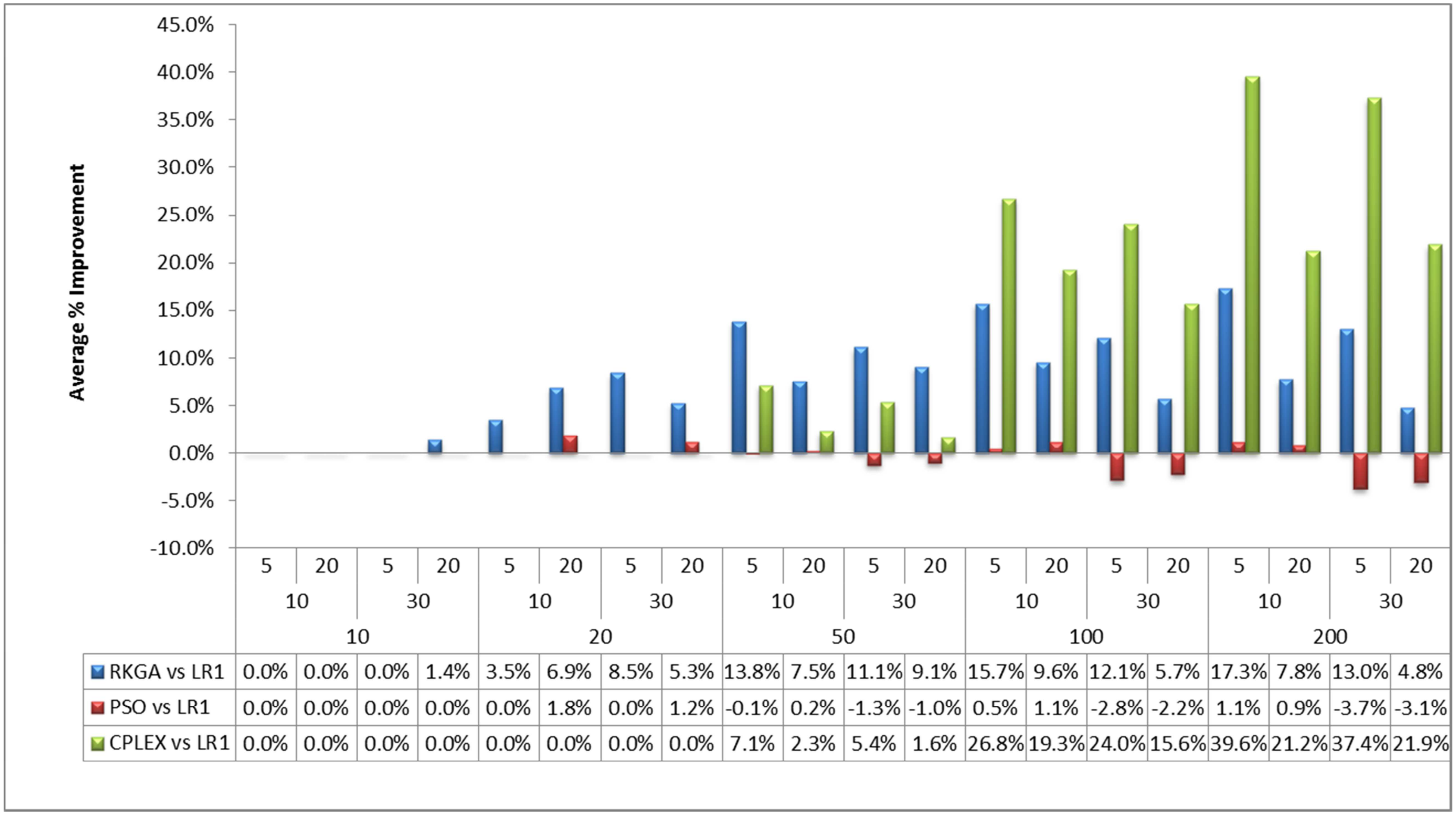
Figure 3: Main Effects Plot for 100 Jobs

5.1 Average Percentage of Improvement (API)

With all the parameters set, experiments on LR1 approach were conducted. As mentioned earlier, the higher (i.e. greater than zero) the API, the better the LR1 performance than other methods. The results for each experimental run is provided in individual tables (see Tables A-1 to A-5) included in Appendix A. The columns in the tables are numbered 1 through 9. Column 1 presents the run code. The run code is based on the number of jobs, processing time of the jobs, size of the jobs, and instances number. For example, J1p1s1#1 indicate a 10 job instance (J1) with processing times from Discrete Uniform 1 to 10 (p1), sizes from Discrete Uniform 1 to 5 (s1), and instance one (#1). The same run code is used for both two-machine and four-machine problems. Columns 2 through 5 report the results for instances with 2 machines. Columns 2, 3, 4, and 5 present the makespan from RKGA, PSO, CPLEX, and LR1 approaches, respectively. Columns 6 through 9 report the results for instances with 4 machines. Columns 6, 7, 8, and 9 present the makespan from RKGA, PSO, CPLEX and LR1 approaches, respectively.

5.1.1 Two-Machine Experiments

Figure 4 shows the API when LR1 is compared with the previous solution approaches (e.g. PSO, RKGA, and CPLEX) for two-machine problem instances. On 10 job instances with two machines, the API is mostly zero, indicating that the LR1 solution is equal to the other approaches. On two instances (J1p1s2#2 and J1p1s2#3), the API is positive when LR1 is compared with RKGA, indicating that LR1 is slightly better than RKGA on these instances. On average, LR1 is 0.35% better than RKGA. The API results for 10 job instances are shown in Appendix B (Table B-1). On 20 job instances with two machines, it is found that the solutions from LR1 and CPLEX were equal (API = 0). When compared with RKGA and PSO, LR1 resulted in either zero or positive API value. Out of 20 instances tested, LR1 reported better solution than RKGA on 15 instances and better than PSO on 5 instances. On an average, LR1 is 6.02% better than RKGA and 0.75% better than PSO. Table B-2 in Appendix B presents the API result for 20 job instances.



	5	20	5	20	5	20	5	20	5	20	5	20	5	20	5	20	5	20	5	20
	10		30		10		30		10		30		10		30		10		30	
	10				20				50				100				200			
🟦 RKGA vs LR1	0.0%	0.0%	0.0%	1.4%	3.5%	6.9%	8.5%	5.3%	13.8%	7.5%	11.1%	9.1%	15.7%	9.6%	12.1%	5.7%	17.3%	7.8%	13.0%	4.8%
🟥 PSO vs LR1	0.0%	0.0%	0.0%	0.0%	0.0%	1.8%	0.0%	1.2%	-0.1%	0.2%	-1.3%	-1.0%	0.5%	1.1%	-2.8%	-2.2%	1.1%	0.9%	-3.7%	-3.1%
🟩 CPLEX vs LR1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	7.1%	2.3%	5.4%	1.6%	26.8%	19.3%	24.0%	15.6%	39.6%	21.2%	37.4%	21.9%

Figure 4: API in Makespan for Two-Machine Instances

On 50 job instances, LR1 outperforms both RKGA and CPLEX across all experiments. A similar conclusion can be drawn on 100 and 200 job instances. However, LR1 is only better or equal than PSO on some instances and on other instances PSO is better than LR1. It is found that LR1 was able to outperform (or comparable) PSO on instances in which the processing times are less variable (when generated from Discrete Uniform distribution with parameters 1 and 10). For 100 and 200 job instances, CPLEX required prohibitively long computational time. Even after running for several hours, CPLEX did not converge to an optimum. Consequently, CPLEX was allowed to run for 1800 seconds and the best found feasible solution was used to determine the API. API results for 50, 100 and 200 job instances are presented accordingly in Appendix B (see Tables B-3 to B-5).

The API result for each instance is graphed and presented in Appendix C. Figure C-1 shows the API for two-machine problem by job instances. Figures C-2 to C-6 show the API result breakdown by the processing time and the size of the jobs in two-machine problem job instances.

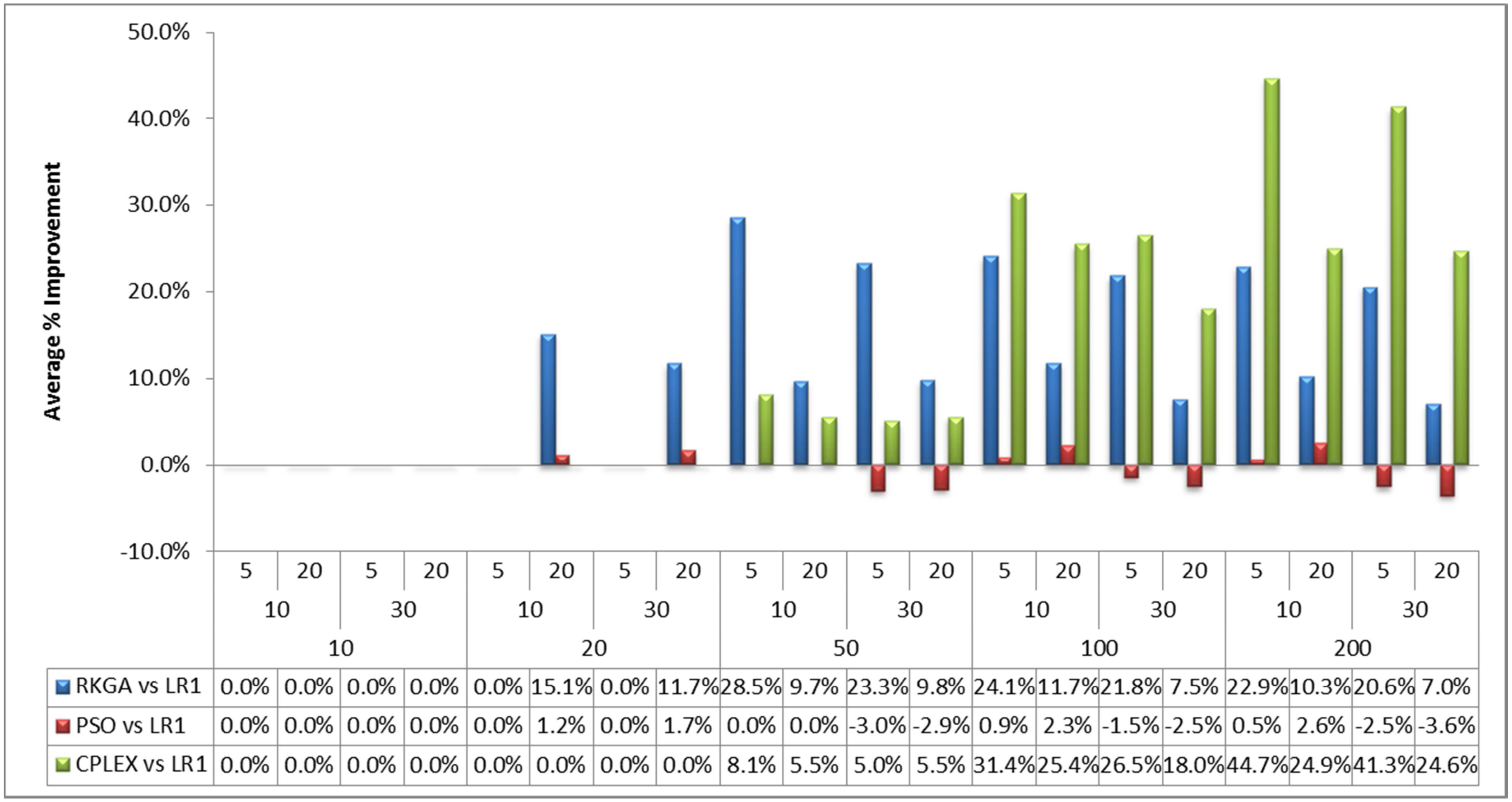
5.1.2 Four-Machine Experiments

Figure 5 shows the API when LR1 is compared with the previous solutions approaches (e.g. PSO, RKGA, and CPLEX) for four-machine problem instances. On 10 job instances with four machines, all the four approaches (RKGA, PSO, CPLEX and LR1) report the same solution. On 20 job instances, LR1 reported the same solution with CPLEX in all experiments. LR1 outperforms RKGA and is marginally better than PSO on instances with larger variability in job sizes (when generated from Discrete Uniform distribution with parameters 1 and 20). On other instances, LR1 reported the same solution with RKGA and PSO. On average, LR1 is 6.71% better than RKGA and 0.73% better than PSO. LR1 is able to outperform both RKGA and CPLEX on 50, 100, and 200 job instances. When compared with PSO, LR1 outperforms (or comparable) on instances where there is less variability in the processing times. However, PSO marginally outperforms LR1 on instances with high variability in the processing times. As in the two-machine instances, CPLEX was unable to converge to an optimum even after running for several hours. Consequently, CPLEX was allowed to run for 1800 seconds and the best known solution was used to compute the API.

The API result for each instance is graphed and presented in Appendix C. Figure C-7 shows the API for four-machine problem by jobs instances. Figure C-8 to C-12 show the API result breakdown by the processing time and the size of the jobs in four-machine problem.

5.2 Computational Time

Table 3 presents the average computation time required by each solution approach for various problem instances. CPLEX did not converge to optimum even after running for several hours on 50, 100, and 200 job instances. Consequently, CPLEX was allowed to run for 1800 seconds and the best feasible solution found was used for comparison. All the four approaches took relatively a short run time to solve all 10 and 20 job instances. PSO was by far the fastest solution approach for all instances. The run time required for RKGA and LR1 grew



	10				20				50				100				200			
	10		30		10		30		10		30		10		30		10		30	
	5	20	5	20	5	20	5	20	5	20	5	20	5	20	5	20	5	20	5	20
🟦 RKGA vs LR1	0.0%	0.0%	0.0%	0.0%	0.0%	15.1%	0.0%	11.7%	28.5%	9.7%	23.3%	9.8%	24.1%	11.7%	21.8%	7.5%	22.9%	10.3%	20.6%	7.0%
🟥 PSO vs LR1	0.0%	0.0%	0.0%	0.0%	0.0%	1.2%	0.0%	1.7%	0.0%	0.0%	-3.0%	-2.9%	0.9%	2.3%	-1.5%	-2.5%	0.5%	2.6%	-2.5%	-3.6%
🟩 CPLEX vs LR1	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	8.1%	5.5%	5.0%	5.5%	31.4%	25.4%	26.5%	18.0%	44.7%	24.9%	41.3%	24.6%

Figure 5: API in Makespan for Four-Machine Instances

exponentially with the size of the problem. In LR1, the longer run times are primarily in solving the upper bound model in CPLEX. An experimental study was conducted to set the time limit (i.e. 100, 200, or 300 seconds) in order to solve the upper bound model. The experimental study favored using 200 seconds as the time limit. As the upper bound model is solved each time a better lower bound is formed, but the computational time for LR1 gets longer for larger problem instances. Tables D-1 to D-5 in Appendix D show the computational results for all instances.

Table 3: Average Computational Times (in seconds) for All Job Instances

No. of Jobs	m = 2				m = 4			
	RKGA*	PSO*	CPLEX*	LR1#	RKGA*	PSO*	CPLEX*	LR1#
10	4.52	0.36	**0.18**	0.22	4.27	0.37	**0.19**	0.34
20	19.23	**0.74**	326.87	14.58	13.73	**0.75**	341.68	13.75
50	143.04	**2.83**	1800	1360.61	121.11	**2.85**	1712.08	2018.3
100	546.44	**9.53**	1800	1750.37	362.19	**9.6**	1709.71	2060.76
200	1780.31	**35.42**	1800	1529.87	1708.09	**35.63**	1805.21	2178.98

* indicates that the experiments were run on Pentium Core 2 Duo, T6400, 2.00 GHz computer with 3 GB RAM

# indicates that the experiments were run on Intel® Core™ i5-2450M, 2.50 GHz computer with 4 GB RAM

5.3 Constraint Considerations

Two different constraints were relaxed to obtain two LR procedures namely LR1 and LR2 as presented in Chapter 4. The API was determined by comparing LR1 with PSO and LR2 with PSO. Figure 6 compares the API's from this comparison. From this figure, it is evident that LR1 is better than LR2 on two machine problem instances. A similar comparison was also done for four machine problem instances (see Figure 7) and LR1 is found to outperform LR2. In both figures (Figures 6 and 7), the API results are comparable in 10 and 20 job instances. However, LR2 performs poorly as the number of jobs in an instance increases. LR1 is significantly better than LR2 on larger problem instances. Consequently, LR1 is recommended for solving the problem under study.

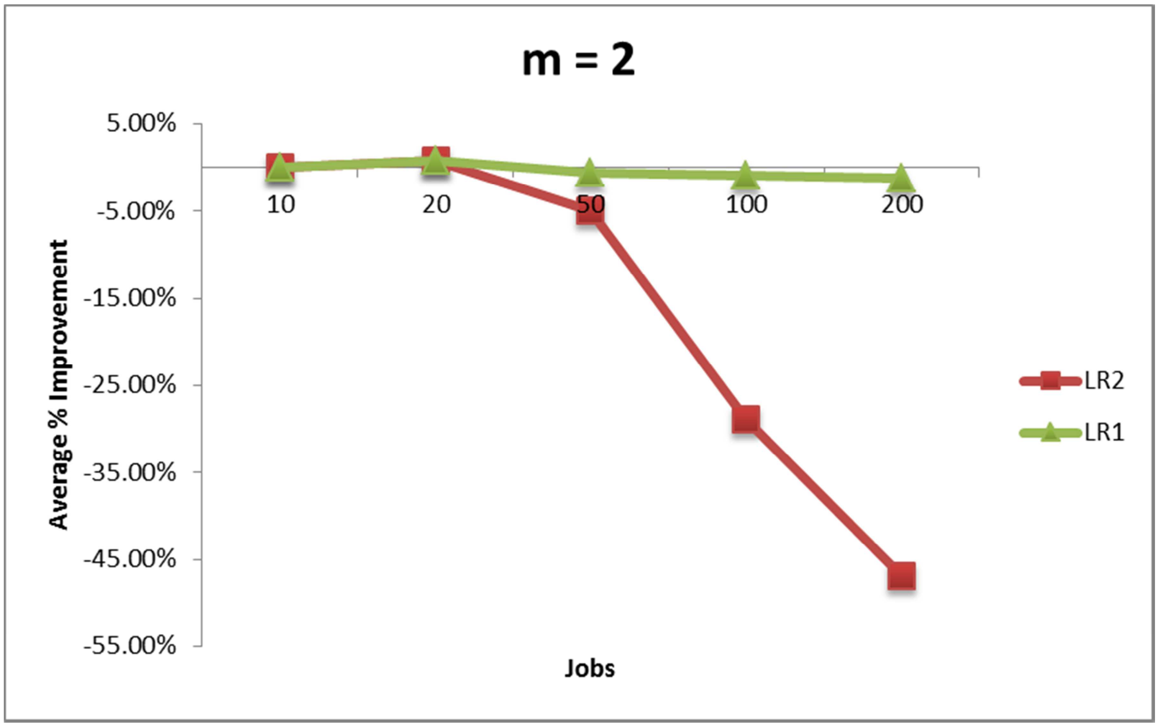


Figure 6: API Comparison between LR1 and LR2 for Two-Machine Problem

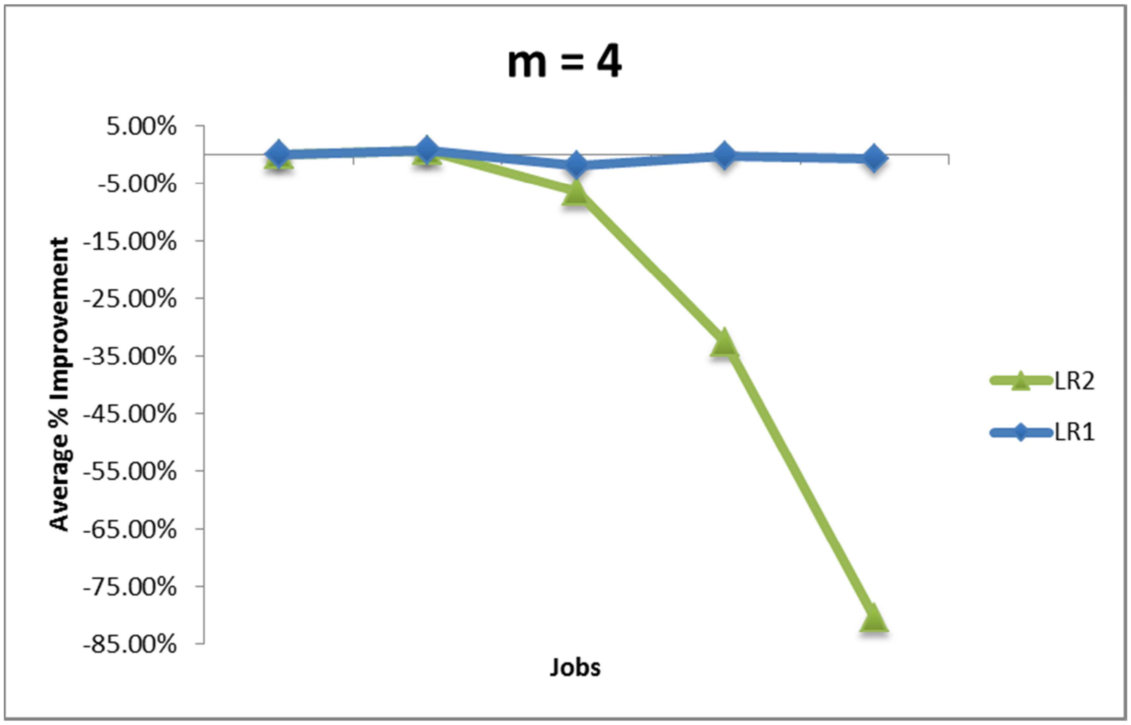


Figure 7: API Comparison between LR1 and LR2 for Four-Machine Problem

CHAPTER 6
CONCLUSION AND FUTURE WORK

Makespan minimization is a commonly used objective in the scheduling literature. Different studies are carried out under different machine environments and makespan objective. In this research, non-identical parallel batch processing machines are considered with the makespan objective. As the problem under study is NP-hard, researchers in the past proposed meta-heuristic such as Particle Swarm Optimization (PSO) [4] and Random Keys Genetic Algorithm (RKGA) [38]. Although meta-heuristic are useful and effective in finding a good solution, they may not guarantee an optimal solution. In this research, a Lagrangian Relaxation (LR) approach is proposed. A mathematical model for the problem under study is considered and two different constraint sets are relaxed using Lagrangian multipliers to obtain two Lagrangian Duals (i.e. LD1 and LD2). A sub-gradient approach was used to update the multipliers. A DOE was conducted to determine the best parameters for the LR.

An experimental study was conducted to evaluate the two Lagrangian models in terms of solution quality and run time. The solutions obtained from the two models are compared to PSO, RKGA, and CPLEX. Data sets from the literature were used to compare all the solution approaches. When comparing the solution from LR1 with LR2, it was noted that the solution from LR1 is almost always better than LR2. Consequently, the solution from LR1 was used to compare with other solution approaches reported in the literature.

Based on the experimental study conducted, it can be concluded that for small job instances (with 10 and 20 jobs) the solution from LR1 approach is similar to the solution obtained from PSO. However, the solution from LR1 approach was better than the RKGA and CPLEX on 50, 100, and 200 job instances with two and four machine problem instances. LR1 solution was better than PSO on larger problem instances (50, 100, and 200 jobs) when the processing times are less variable (when generated with Discrete Uniform 1 to 10). But on other instances where the processing times are highly variable, PSO solution was marginally better than LR1. On four machine problem instances, LR1 solution was better than RKGA and PSO on smaller problem instances. Similar to two-machine problem, LR1 was better than PSO on large problem instances with smaller processing time variability in four-machine problem instances. PSO was better than LR1 on larger processing time variability.

In terms of computational times, PSO approach was faster on all problem instances. RKGA and LR1 required longer computational times – especially as the problem size grew. CPLEX was unable to converge to optimum, hence, it was allowed to run for 1800 seconds and the best found solution was used for comparisons.

Out of the 200 experiments conducted on two and four machine problem instances, the proposed LR approach has resulted in finding 34 new improved solutions and 105 comparable solutions (or equal) when compared to the PSO approach. The LR approach proposed in this research can be improved by considering a Branch-and-Bound approach to retain feasibility instead of solving the upper bound model as explained in Chapter 4. Another possibility is to propose a better way to solve the upper bound model, perhaps a heuristic. These steps may help to reduce the run time required for the LR approach. However, the solution quality is to be evaluated if a heuristic is to be used.

This research can be extended to consider unrelated parallel machines easily. The only change to be made is to account for the different job processing times on different machines. The job ready times can also be considered by adding a couple of constraints. Other objectives such as tardiness and weighted tardiness can be useful to some manufacturers.

As mentioned in the previous literatures, most scheduling problems related to BPM are solved with heuristic and meta-heuristic approaches. This research demonstrates the successful application of LR approach to solve scheduling problems with non-identical parallel BPM – especially to minimize the makespan. The main contribution of this research is the solution approach based on sound Operations Research theory to solve a complex scheduling problem commonly observed in the industry. In addition, the experimental study demonstrates that there are several instances on which the LR approach outperforms several heuristics. This research and its findings can benefit practitioners and academics in that a new approach, in addition to the meta-heuristics already available, can be tried to solve BPM scheduling problems.

REFERENCES

[1] Potts, C. N., and Kovalyov, M. Y., 2000, “Scheduling with Batching: A Review,” European Journal of Operation Research, 120, pp. 228-249.

[2] Ozturk, O., Espinouse, M-L., Mascolo, M., and Gouin, A., 2012, “Makespan Minimization on Parallel Batch Processing Machines with Non-Identical Job Sizes and Release Dates,” *International Journal of Production Research*, 50(20), pp. 6022-6035.

[3] Raidl, G. R., and Gruber, M., 2008, “A Lagrangian Relax-and-Cut Approach for the Bounded Diameter Minimum Spanning Tree Problem,” *International Conference on Numerical Analysis and Applied Mathematics*, pp. 446-449.

[4] Damodaran, P., Diyadawagamage D. A., Ghrayeb, O., and Velez-Gallego M. C., 2012, “A Particle Swarm Optimization Algorithm for Minimizing Makespan of Nonidentical Parallel Batch Processing Machines,” *The International Journal of Advanced Manufacturing Technology*, 58, pp. 1131-1140.

[5] Hanssmann, F., and Hess, S. W., 1960, “A Linear Programming Approach to Production and Employment Scheduling,” *Management Technology*, 1(1), pp. 46-51.

[6] Uzsoy, R., 1994, “Scheduling a Single Batch Processing Machine with Non-Identical Job Sizes,” *International Journal of Production Research*, 32(7), pp. 1615-1635.

[7] Lee, C. Y., and Uzsoy, R., 1999, “Minimizing Makespan on a Single Batch Processing Machine with Dynamic Job Arrivals,” *International Journal of Production Research*, 37, pp. 219-236.

[8] Tang, L., Xuan, H., and Liu, J., 2006, “A New Lagrangian Relaxation Algorithm for Hybrid Flowshop Scheduling to Minimize Total Weighted Completion Time,” *Computers and Operations Research*, 33(11), pp. 3344-3359.

[9] Liu, L. L., Ng, C. T., and Cheng, T. C. E., 2008, “Scheduling Jobs with Agreeable Processing Times and Due Dates on a Single Batch Processing Machine,” *Theoretical Computer Science*, 374, pp. 159-169.

[10] Perez, I. C., Fowler, J. W., and Carlyle, W. M., 2005, “Minimizing total weighted tardiness on a single batch process machine with incompatible job-families,” *Computers and Operations Research*, 32, pp. 327-341.

[11] Sivrikaya-Serifoglu, F., and Ulusoy, G., 1999, "Parallel Machine Scheduling with Earliness and Tardiness Penalties," *Computers and Operations Research*, 26, pp. 773-787.

[12] Chandru, V., Lee, C. Y., and Uzsoy, R., 1993, "Minimizing total completion time on batch processing machine," *International Journal of Production Research*, 31, pp. 2097-2121.

[13] Shim, S-O., and Kim, Y-D., 2008, "A Branch and Bound Algorithm for an Identical Parallel Machine Scheduling Problem with Job-Splitting Property," *Computers and Operations Research*, 35(3), pp. 863-875.

[14] Lee, W-C., Wu C-C., and Chen, P., 2006, "A Simulated Annealing Approach to Makespan Minimization on Identical Parallel Machines," *The International Journal of Advanced Manufacturing Technology*, 31, pp. 328-334.

[15] Damodaran, P., and Velez-Gallego, M. C., 2012, "A Simulated Annealing Algorithm to Minimize Makespan of Parallel Batch Processing Machines with Unequal Job Ready Times," *Expert Systems with Applications: An International Journal*, 39(1), pp. 1451-1458.

[16] Ozturk, O., Espinouse, M-L., Di Mascolo, M., and Gouin, A., 2010, "Optimizing Makespan of Washing Operations of Medical Devices in Hospital Sterilization Services," *IEEE Workshop on Health Care Management (WHCM 2010)*, pp. 6.

[17] Al-Ghamdi, F., Al-Khaldi, M., Khouki, A., and Al-Slamah, M., 2012, "Minimizing the Tardiness in a Single Machine Batch Processing," *Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management*, pp. 121-128.

[18] Li, K., and Yang, S-L., 2009, "Non-Identical Parallel Machine Scheduling Research with Minimizing Total Weighted Completion Times: Models, Relaxations and Algorithms," *Applied Mathematical Modelling*, 33(4), pp. 2145-2158.

[19] Turkcan, A., Akturk M. S., and Storer, R. H., 2003, "Non-Identical Parallel CNC Machine Scheduling," *International Journal of Production Research*, 41(10), pp. 2143-2168.

[20] Dessouky, M. M., 1998, "Scheduling Identical Jobs with Unequal Ready Times on Uniform Parallel Machines to Minimize the Maximum Lateness," *Computers and Industrial Engineering*, 34(4), pp. 793-806.

[21] Mokotoff, E., 1999, "Scheduling to Minimize Makespan on Identical Parallel Machines: An LP-Based Algorithm," *Investigacion Operativa*, 8, pp. 97-107.

[22] Chang, P., Damodaran, P., Melouk, S., 2004, "Minimizing Makespan on Parallel Batch Processing Machines," *International Journal of Production Research*, 42, pp. 4211–4220.

[23] Kashan, A. H., Karimi, B., Jenabi, M., 2008, "A Hybrid Genetic Heuristic for Scheduling Parallel Batch Processing Machines with Arbitrary Job Sizes," *Computers and Operations Research*, 35, pp. 1084–1098.

[24] Xu. S., and Bean, J. C., 2007, "A Genetic Algorithm for Scheduling Parallel Non-Identical Batch Processing Machines," *IEEE Symposium on Computational Intelligence in Scheduling*, pp. 143-150.

[25] Shao, H., Chen, H-P., Huang, G. Q., Xu, R., Cheng, B-Y., Wang S-S., and Liu, B-W., 2008, "Minimizing Makespan for Parallel Batch Processing Machines with Non-Identical Job Sizes Using Neural Nets Approach," *IEEE Conference on Industrial Electronics and Applications*, pp. 1921-1924.

[26] Li, S., and Yuan, J., 2010, "Parallel-Machine Parallel-Batching Scheduling with Family Jobs and Release Dates to Minimize Makespan," *Journal of Combinatorial Optimization*, 19, pp. 84-93.

[27] Chen, H., Chu, C., and Proth, J-M., 1998, "An Improvement of the Lagrangian Relaxation Approach for Job Shop Scheduling: A Dynamic Programming Method," *IEEE Transactions on Robotics and Automation*, 14(5), pp. 786-795.

[28] Held. M., and Karp, R. M., 1970, "The Traveling-Salesman Problem and Minimum Spanning Trees," *Operations Research*, 18, pp. 1138-1162.

[29] Fisher, M. L., 1981, "The Lagrangian Relaxation Method for Solving Integer Programming Problems," *Management Science*, 27(1), pp. 1-18.

[30] Fisher, M. L., 1985, "An Applications Oriented Guide to Lagrangian Relaxation," *Interfaces*, 15(2), pp. 10-21.

[31] Muckstadt, J. A., and Koenig, S. A., 1975, "An Application of Lagrangian Relaxation to Scheduling in Power-Generation Systems," *Operations Research*, 25(3), pp. 387-403.

[32] Luh, P. B., and Hoitomt, D. J., 1993, "Scheduling of Manufacturing Systems Using the Lagrangian Relaxation Technique," *IEEE Transactions on Automatic Control*, 38, pp. 1066-1079.

[33] Liu, G., Luh, P. B., and Resch, R., 1997, "Scheduling Permutation Flow Shops Using the Lagrangian Relaxation Technique," *Annals of Operations Research*, 70, pp. 171-189.

[34] Irohara, T., 2008, "Lagrangian Relaxation Algorithms for Hybrid Flow-Shop Scheduling Problems with Limited Buffers," *Biomedical Soft Computing and Human Sciences*, 15(1), pp. 21-28.

[35] Perdomo, V., Augusto, V., and Xie X., 2006, "Operation Theatre Scheduling Using Lagrangian Relaxation," *International Conference on Service Systems and Service Management*, pp. 1234-1239.

[36] Velde, S. L., 1990, "Dual Decomposition of a Single-Machine Scheduling Problem," *Mathematical Programming*, 69, pp. 413-428.

[37] Balin, S., 2010, “Non-Identical Parallel Machine Scheduling with Fuzzy Processing Times Using Genetic Algorithm and Simulation,” *The International Journal of Advanced Manufacturing Technology,* 61(9), pp. 1115-1127.

[38] Malapert, A., Gueret, C., and Rousseau, L-M., “A Constraint Programming Approach for a Batch Processing Problem with Non-Identical Job Sizes,” European Journal of Operation Research, 221(3), pp. 533-545.

[39] Velez-Gallego, M. C., Damodaran, P., and Rodriguez, M., 2011, “Makespan Minimization on a Single Batch Processing Machine with Unequal Job Ready Times,” *International Journal of Industrial Engineering*, 18(10), pp. 536-546.

[40] Damodaran, P., and Velez-Gallego, M. C., 2010, “Heuristics for Makespan Minimization on Parallel Batch Processing Machines with Unequal Job Ready Times,” *International Journal of Advanced Manufacturing Technology*, 49, pp. 1119-1128.

[41] Damodaran, P., Velez-Gallego, M. C., and Maya, J., 2011, “A GRASP Approach for Makespan Minimization on Parallel Batch Processing Machines,” *Journal of Intelligent Manufacturing*, 22, pp. 767-777.

APPENDIX A: RKGA, PSO, CPLEX AND LR1 C_{max} RESULTS

Table A-1: RKGA, PSO, CPLEX and LR1 C_{max} results for 10 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J1p1s1#1	9	9	9	9	9	9	9	9
J1p1s1#2	10	10	10	10	10	10	10	10
J1p1s1#3	10	10	10	10	10	10	10	10
J1p1s1#4	10	10	10	10	10	10	10	10
J1p1s1#5	10	10	10	10	10	10	10	10
J1p1s2#1	17	17	17	17	10	10	10	10
J1p1s2#2	8	8	8	8	8	8	8	8
J1p1s2#3	12	12	12	12	9	9	9	9
J1p1s2#4	24	24	24	24	14	14	14	14
J1p1s2#5	14	14	14	14	9	9	9	9
J1p2s1#1	27	27	27	27	27	27	27	27
J1p2s1#2	30	30	30	30	30	30	30	30
J1p2s1#3	29	29	29	29	29	29	29	29
J1p2s1#4	30	30	30	30	30	30	30	30
J1p2s1#5	26	26	26	26	26	26	26	26
J1p2s2#1	39	38	38	38	28	28	28	28
J1p2s2#2	46	44	44	44	30	30	30	30
J1p2s2#3	40	40	40	40	30	30	30	30
J1p2s2#4	30	30	30	30	30	30	30	30
J1p2s2#5	35	35	35	35	29	29	29	29

Table A-2: RKGA, PSO, CPLEX and LR1 C_{max} results for 20 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J2p1s1#1	10	10	10	10	10	10	10	10
J2p1s1#2	12	11	11	11	10	10	10	10
J2p1s1#3	11	10	10	10	10	10	10	10
J2p1s1#4	10	10	10	10	10	10	10	10
J2p1s1#5	10	10	10	10	10	10	10	10
J2p1s2#1	22	21	21	21	14	11	11	11
J2p1s2#2	37	35	34	34	20	19	19	19
J2p1s2#3	33	33	32	32	19	17	16	16
J2p1s2#4	32	28	28	28	17	14	14	14
J2p1s2#5	32	31	30	30	19	16	16	16
J2p2s1#1	30	30	30	30	30	30	30	30
J2p2s1#2	34	30	30	30	30	30	30	30
J2p2s1#3	29	29	29	29	29	29	29	29
J2p2s1#4	36	30	30	30	30	30	30	30
J2p2s1#5	36	31	31	31	29	29	29	29
J2p2s2#1	125	117	117	117	64	61	58	58
J2p2s2#2	89	87	85	85	49	45	45	45
J2p2s2#3	90	84	81	81	46	41	41	41
J2p2s2#4	100	98	98	98	60	53	51	51
J2p2s2#5	59	57	57	57	33	28	28	28

Table A-3: RKGA, PSO, CPLEX and LR1 C_{max} results for 50 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J3p1s1#1	26	23	24	23	16	12	13	12
J3p1s1#2	23	20	22	19	16	10	10	10
J3p1s1#3	27	22	23	22	16	12	13	12
J3p1s1#4	22	19	21	19	14	10	12	10
J3p1s1#5	24	21	23	21	15	11	12	11
J3p1s2#1	77	74	76	74	40	36	40	37
J3p1s2#2	67	60	62	60	36	31	34	32
J3p1s2#3	72	67	71	67	38	34	37	34
J3p1s2#4	77	70	69	68	38	35	36	35
J3p1s2#5	65	61	61	61	35	30	32	31
J3p2s1#1	75	69	74	70	41	35	39	36
J3p2s1#2	65	56	62	57	40	29	32	30
J3p2s1#3	77	66	69	67	49	37	40	39
J3p2s1#4	71	62	67	62	46	31	33	32
J3p2s1#5	74	65	68	66	46	33	35	33
J3p2s2#1	190	177	184	178	100	88	99	90
J3p2s2#2	175	159	170	163	91	79	86	83
J3p2s2#3	237	218	219	219	119	108	115	111
J3p2s2#4	239	225	222	222	126	110	116	111
J3p2s2#5	217	174	180	179	102	87	97	90

Table A-4: RKGA, PSO, CPLEX and LR1 C_{max} results for 100 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J4p1s1#1	49	43	54	42	29	22	33	21
J4p1s1#2	47	40	56	40	26	20	30	20
J4p1s1#3	43	36	48	36	25	19	27	19
J4p1s1#4	41	34	50	34	23	18	26	18
J4p1s1#5	44	37	50	37	25	19	26	19
J4p1s2#1	135	126	154	126	71	62	83	60
J4p1s2#2	143	131	163	130	73	65	83	65
J4p1s2#3	158	142	171	141	79	72	96	70
J4p1s2#4	159	146	181	140	80	74	89	71
J4p1s2#5	149	135	164	135	73	67	95	66
J4p2s1#1	125	103	137	106	69	53	74	55
J4p2s1#2	146	121	173	123	81	63	83	63
J4p2s1#3	137	120	159	125	76	61	86	61
J4p2s1#4	125	108	146	112	73	56	79	57
J4p2s1#5	135	119	159	121	80	59	81	60
J4p2s2#1	391	360	440	373	202	181	231	188
J4p2s2#2	422	395	484	399	219	195	240	199
J4p2s2#3	424	393	471	403	212	196	251	200
J4p2s2#4	449	413	488	420	224	206	253	211
J4p2s2#5	427	389	478	397	221	195	241	199

Table A-5: RKGA, PSO, CPLEX and LR1 C_{max} results for 200 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J5p1s1#1	88	73	122	72	48	37	64	37
J5p1s1#2	89	74	120	73	47	37	66	37
J5p1s1#3	84	71	115	70	48	36	65	36
J5p1s1#4	84	71	115	70	46	36	67	36
J5p1s1#5	88	73	121	73	47	37	67	36
J5p1s2#1	276	255	346	250	138	128	165	122
J5p1s2#2	333	315	378	311	167	156	201	153
J5p1s2#3	302	280	349	281	152	140	174	144
J5p1s2#4	314	293	369	291	162	147	193	141
J5p1s2#5	307	283	351	281	153	140	191	133
J5p2s1#1	242	201	347	207	135	101	189	103
J5p2s1#2	282	241	400	250	153	121	203	120
J5p2s1#3	270	232	367	243	146	117	196	120
J5p2s1#4	262	219	356	228	147	112	195	117
J5p2s1#5	275	224	380	231	145	112	200	117
J5p2s2#1	808	739	994	773	405	365	481	374
J5p2s2#2	886	823	1125	851	451	411	550	426
J5p2s2#3	808	752	946	769	434	378	514	391
J5p2s2#4	833	765	1007	787	425	386	561	397
J5p2s2#5	879	814	1071	832	451	404	569	426

APPENDIX B: AVERAGE PERCENTAGE OF IMPROVEMENT TOWARDS LR1

Table B-1: Average percentage of improvement towards LR1 for 10 job instances

Runcode (1)	**m = 2**			**m = 4**		
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**RKGA (5)**	**PSO (6)**	**CPLEX (7)**
J1p1s1#1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s1#2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s1#3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s1#4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s1#5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s2#1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s2#2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s2#3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s2#4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p1s2#5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s1#1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s1#2	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s1#3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s1#4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s1#5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s2#1	2.56%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s2#2	4.35%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s2#3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s2#4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J1p2s2#5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Average	0.35%	0.00%	0.00%	0.00%	0.00%	0.00%

Table B-2: Average percentage of improvement towards LR1 for 20 job instances

Runcode (1)	**m = 2**			**m = 4**		
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**RKGA (5)**	**PSO (6)**	**CPLEX (7)**
J2p1s1#1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p1s1#2	8.33%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p1s1#3	9.09%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p1s1#4	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p1s1#5	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p1s2#1	4.55%	0.00%	0.00%	21.43%	0.00%	0.00%
J2p1s2#2	8.11%	2.86%	0.00%	5.00%	0.00%	0.00%
J2p1s2#3	3.03%	3.03%	0.00%	15.79%	5.88%	0.00%
J2p1s2#4	12.50%	0.00%	0.00%	17.65%	0.00%	0.00%
J2p1s2#5	6.25%	3.23%	0.00%	15.79%	0.00%	0.00%
J2p2s1#1	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p2s1#2	11.76%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p2s1#3	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p2s1#4	16.67%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p2s1#5	13.89%	0.00%	0.00%	0.00%	0.00%	0.00%
J2p2s2#1	6.40%	0.00%	0.00%	9.38%	4.92%	0.00%
J2p2s2#2	4.49%	2.30%	0.00%	8.16%	0.00%	0.00%
J2p2s2#3	10.00%	3.57%	0.00%	10.87%	0.00%	0.00%
J2p2s2#4	2.00%	0.00%	0.00%	15.00%	3.77%	0.00%
J2p2s2#5	0.00%	0.00%	0.00%	15.15%	0.00%	0.00%
Average	6.02%	0.75%	0.00%	6.71%	0.73%	0.00%

Table B-3: Average percentage of improvement towards LR1 for 50 job instances

Runcode (1)	m = 2			m = 4		
	RKGA (2)	PSO (3)	CPLEX (4)	RKGA (5)	PSO (6)	CPLEX (7)
J3p1s1#1	3.39%	0.00%	0.00%	25.00%	0.00%	7.69%
J3p1s1#2	11.54%	0.00%	4.17%	37.50%	0.00%	0.00%
J3p1s1#3	17.39%	5.00%	13.64%	25.00%	0.00%	7.69%
J3p1s1#4	18.52%	0.00%	4.35%	28.57%	0.00%	16.67%
J3p1s1#5	9.09%	-5.26%	4.76%	26.67%	0.00%	8.33%
J3p1s2#1	12.50%	0.00%	8.70%	7.50%	2.78%	7.50%
J3p1s2#2	3.90%	0.00%	2.63%	11.11%	0.00%	5.88%
J3p1s2#3	8.96%	-1.67%	1.61%	10.53%	0.00%	8.11%
J3p1s2#4	6.94%	0.00%	5.63%	7.89%	0.00%	2.78%
J3p1s2#5	11.69%	2.86%	1.45%	11.43%	0.00%	3.13%
J3p2s1#1	6.15%	0.00%	0.00%	12.20%	-2.86%	7.69%
J3p2s1#2	6.67%	-1.45%	5.41%	25.00%	-3.45%	6.25%
J3p2s1#3	12.31%	-1.79%	8.06%	20.41%	-5.41%	2.50%
J3p2s1#4	12.99%	-1.52%	2.90%	30.43%	-3.23%	3.03%
J3p2s1#5	12.68%	0.00%	7.46%	28.26%	0.00%	5.71%
J3p2s2#1	10.81%	-1.54%	2.94%	10.00%	-2.27%	9.09%
J3p2s2#2	6.32%	-0.56%	3.26%	8.79%	-5.06%	3.49%
J3p2s2#3	6.86%	-2.52%	4.12%	6.72%	-2.78%	3.48%
J3p2s2#4	7.59%	-0.46%	0.00%	11.90%	-0.91%	4.31%
J3p2s2#5	7.11%	1.33%	0.00%	11.76%	-3.45%	7.22%
Average	10.38%	-0.52%	4.08%	17.83%	-1.33%	6.03%

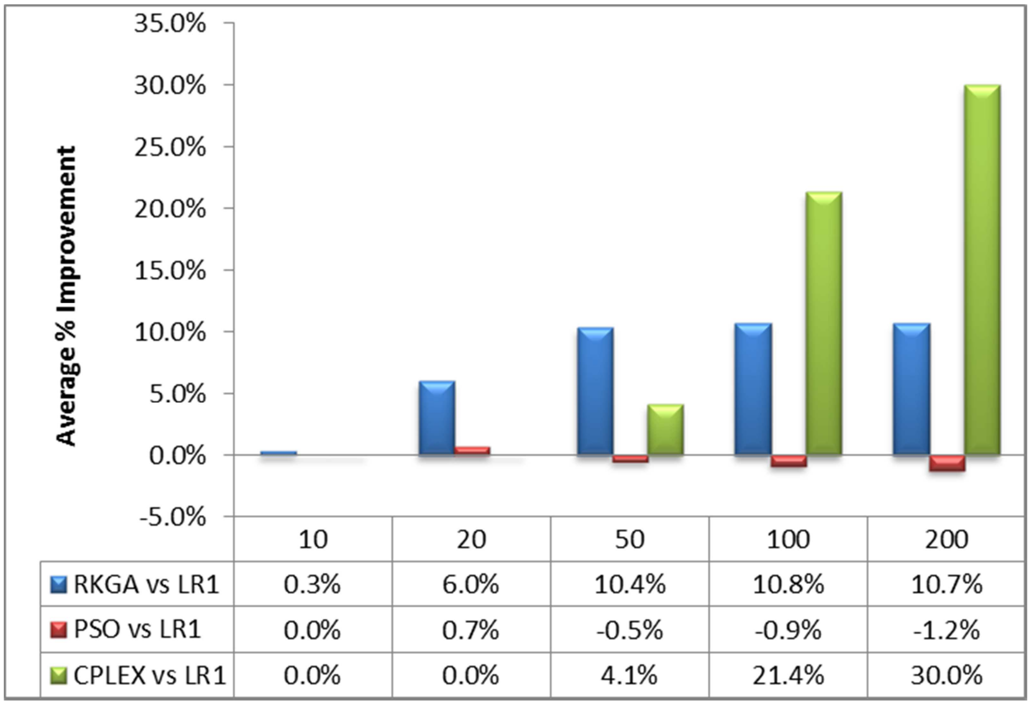
Table B-4: Average percentage of improvement towards LR1 for 100 job instances

Runcode (1)	m = 2			m = 4		
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**RKGA (5)**	**PSO (6)**	**CPLEX (7)**
J4p1s1#1	14.29%	2.33%	22.22%	27.59%	4.55%	36.36%
J4p1s1#2	14.89%	0.00%	28.57%	23.08%	0.00%	33.33%
J4p1s1#3	16.28%	0.00%	25.00%	24.00%	0.00%	29.63%
J4p1s1#4	17.07%	0.00%	32.00%	21.74%	0.00%	30.77%
J4p1s1#5	15.91%	0.00%	26.00%	24.00%	0.00%	26.92%
J4p1s2#1	6.67%	0.00%	18.18%	15.49%	3.23%	27.71%
J4p1s2#2	9.09%	0.76%	20.25%	10.96%	0.00%	21.69%
J4p1s2#3	10.76%	0.70%	17.54%	11.39%	2.78%	27.08%
J4p1s2#4	11.95%	4.11%	22.65%	11.25%	4.05%	20.22%
J4p1s2#5	9.40%	0.00%	17.68%	9.59%	1.49%	30.53%
J4p2s1#1	15.20%	-2.91%	22.63%	20.29%	-3.77%	25.68%
J4p2s1#2	15.75%	-1.65%	28.90%	22.22%	0.00%	24.10%
J4p2s1#3	8.76%	-4.17%	21.38%	19.74%	0.00%	29.07%
J4p2s1#4	10.40%	-3.70%	23.29%	21.92%	-1.79%	27.85%
J4p2s1#5	10.37%	-1.68%	23.90%	25.00%	-1.69%	25.93%
J4p2s2#1	4.60%	-3.61%	15.23%	6.93%	-3.87%	18.61%
J4p2s2#2	5.45%	-1.01%	17.56%	9.13%	-2.05%	17.08%
J4p2s2#3	4.95%	-2.54%	14.44%	5.66%	-2.04%	20.32%
J4p2s2#4	6.46%	-1.69%	13.93%	5.80%	-2.43%	16.60%
J4p2s2#5	7.03%	-2.06%	16.95%	9.95%	-2.05%	17.43%
Average	10.76%	-0.86%	21.42%	16.29%	-0.18%	25.35%

Table B-5: Average percentage of improvement towards LR1 for 200 job instances

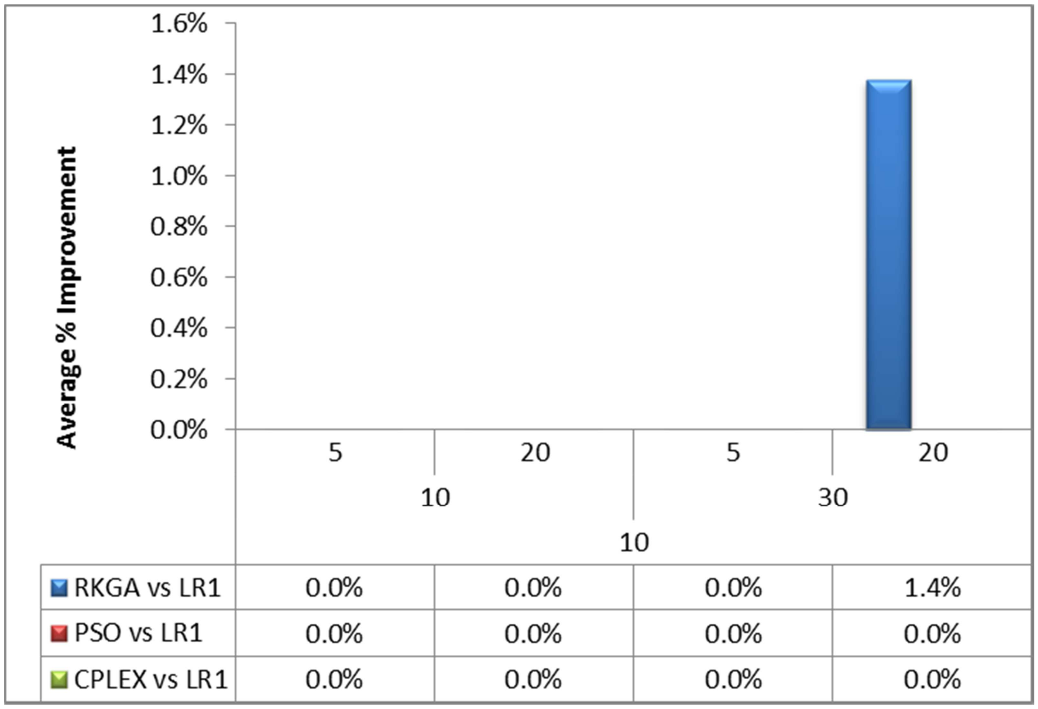
Runcode (1)	**m = 2**			**m = 4**		
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**RKGA (5)**	**PSO (6)**	**CPLEX (7)**
J5p1s1#1	18.18%	1.37%	40.98%	22.92%	0.00%	42.19%
J5p1s1#2	17.98%	1.35%	39.17%	21.28%	0.00%	43.94%
J5p1s1#3	16.67%	1.41%	39.13%	25.00%	0.00%	44.62%
J5p1s1#4	16.67%	1.41%	39.13%	21.74%	0.00%	46.27%
J5p1s1#5	17.05%	0.00%	39.67%	23.40%	2.70%	46.27%
J5p1s2#1	9.42%	1.96%	27.75%	11.59%	4.69%	26.06%
J5p1s2#2	6.61%	1.27%	17.72%	8.38%	1.92%	23.88%
J5p1s2#3	6.95%	-0.36%	19.48%	5.26%	-2.86%	17.24%
J5p1s2#4	7.32%	0.68%	21.14%	12.96%	4.08%	26.94%
J5p1s2#5	8.47%	0.71%	19.94%	13.07%	5.00%	30.37%
J5p2s1#1	14.46%	-2.99%	40.35%	23.70%	-1.98%	45.50%
J5p2s1#2	11.35%	-3.73%	37.50%	21.57%	0.83%	40.89%
J5p2s1#3	10.00%	-4.74%	33.79%	17.81%	-2.56%	38.78%
J5p2s1#4	12.98%	-4.11%	35.96%	20.41%	-4.46%	40.00%
J5p2s1#5	16.00%	-3.13%	39.21%	19.31%	-4.46%	41.50%
J5p2s2#1	4.33%	-4.60%	22.23%	7.65%	-2.47%	22.25%
J5p2s2#2	3.95%	-3.40%	24.36%	5.54%	-3.65%	22.55%
J5p2s2#3	4.83%	-2.26%	18.71%	9.91%	-3.44%	23.93%
J5p2s2#4	5.52%	-2.88%	21.85%	6.59%	-2.85%	29.23%
J5p2s2#5	5.35%	-2.21%	22.32%	5.54%	-5.45%	25.13%
Average	10.70%	-1.21%	30.02%	15.18%	-0.75%	33.88%

APPENDIX C: API RESULT FOR ALL JOBS



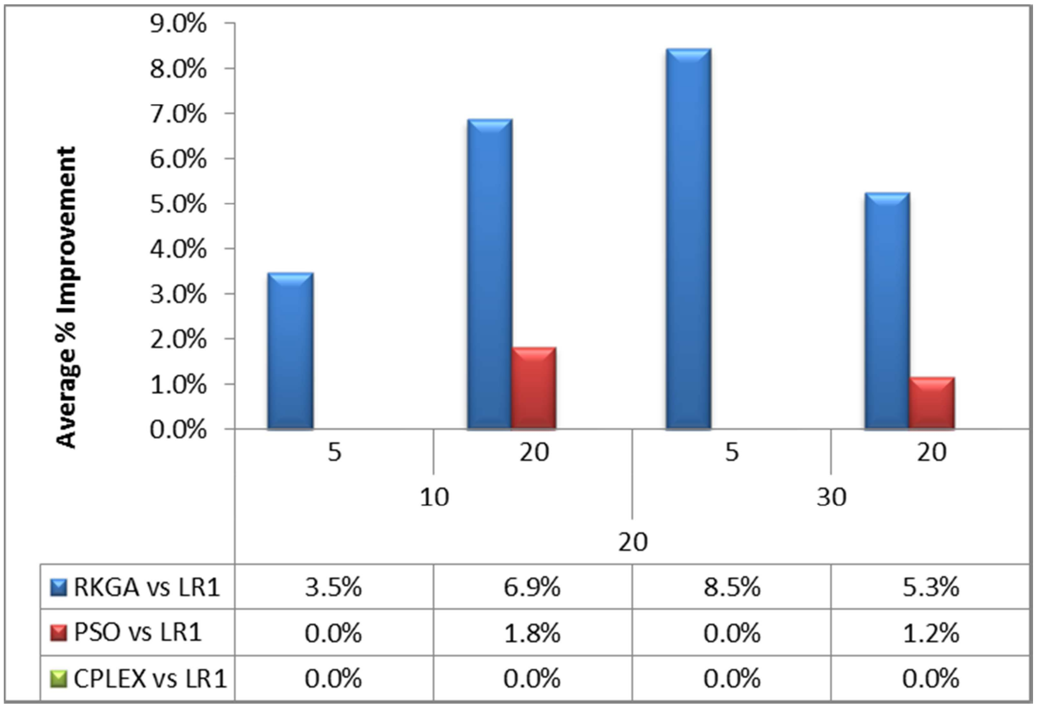
	10	20	50	100	200
🟦 RKGA vs LR1	0.3%	6.0%	10.4%	10.8%	10.7%
🟥 PSO vs LR1	0.0%	0.7%	-0.5%	-0.9%	-1.2%
🟩 CPLEX vs LR1	0.0%	0.0%	4.1%	21.4%	30.0%

Figure C-1: API Result for Two-Machine Problem



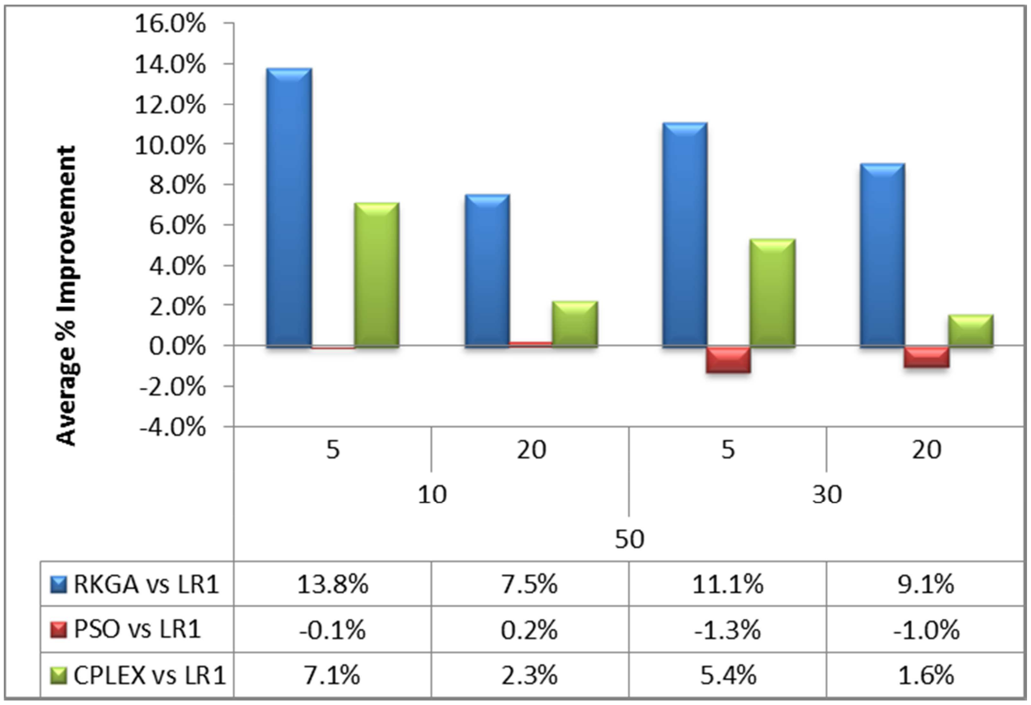
	10			
	10		30	
	5	20	5	20
🟦 RKGA vs LR1	0.0%	0.0%	0.0%	1.4%
🟥 PSO vs LR1	0.0%	0.0%	0.0%	0.0%
🟩 CPLEX vs LR1	0.0%	0.0%	0.0%	0.0%

Figure C-2: API Result for 10 Jobs in Two-Machine Problem



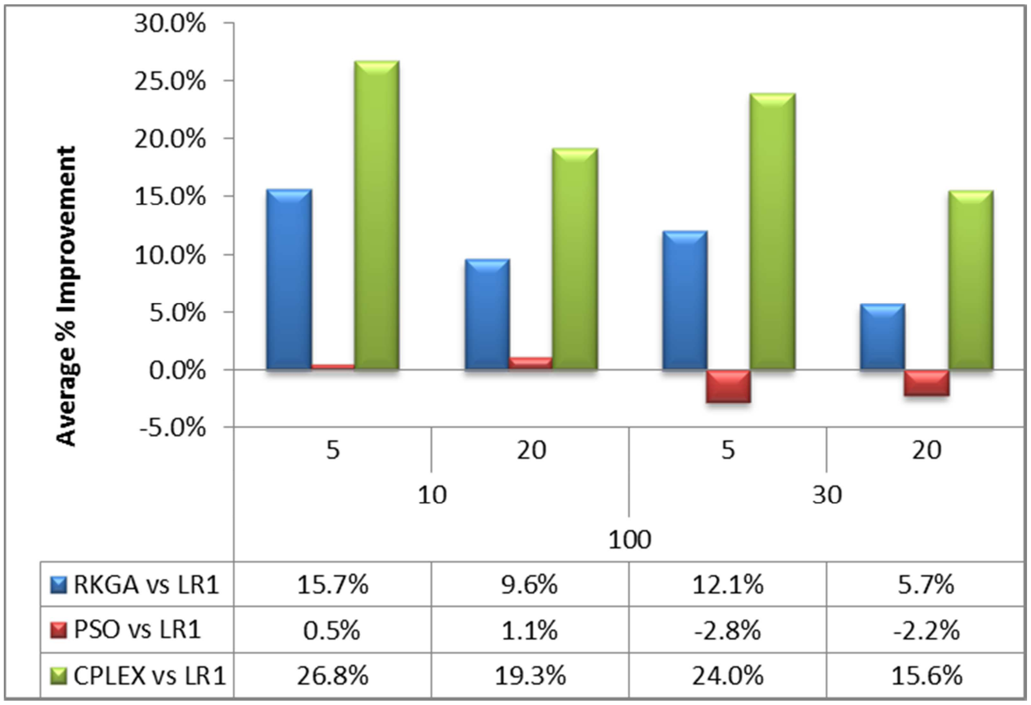
🟦 RKGA vs LR1	3.5%	6.9%	8.5%	5.3%
🟥 PSO vs LR1	0.0%	1.8%	0.0%	1.2%
🟩 CPLEX vs LR1	0.0%	0.0%	0.0%	0.0%

Figure C-3: API Result for 20 Jobs in Two-Machine Problem



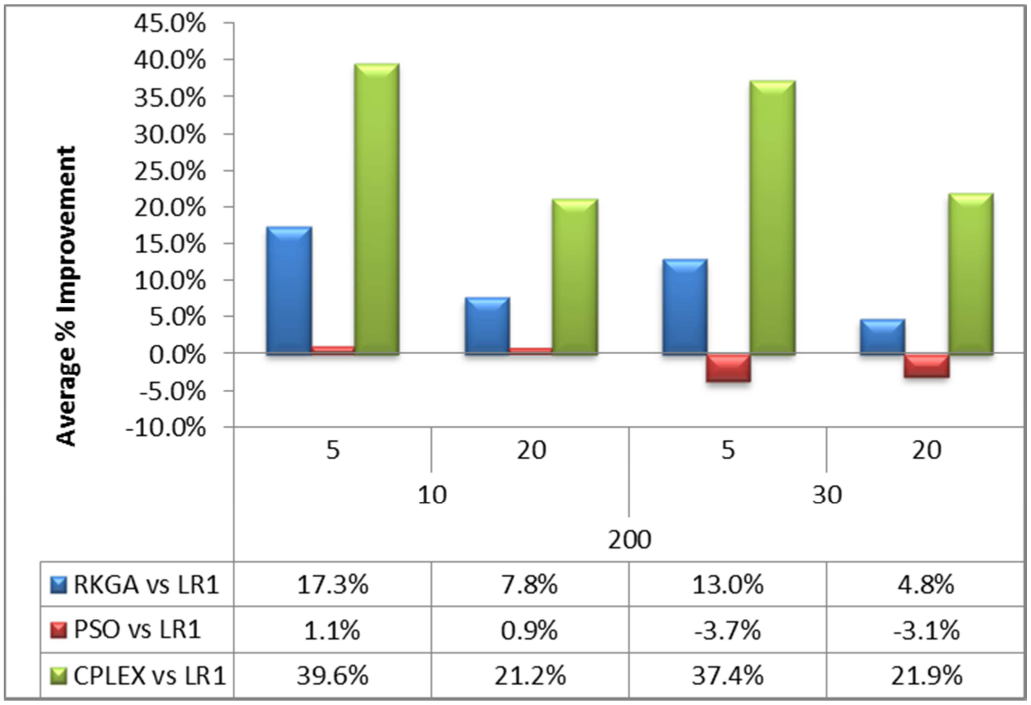
🟦 RKGA vs LR1	13.8%	7.5%	11.1%	9.1%
🟥 PSO vs LR1	-0.1%	0.2%	-1.3%	-1.0%
🟩 CPLEX vs LR1	7.1%	2.3%	5.4%	1.6%

Figure C-4: API Result for 50 Jobs in Two-Machine Problem



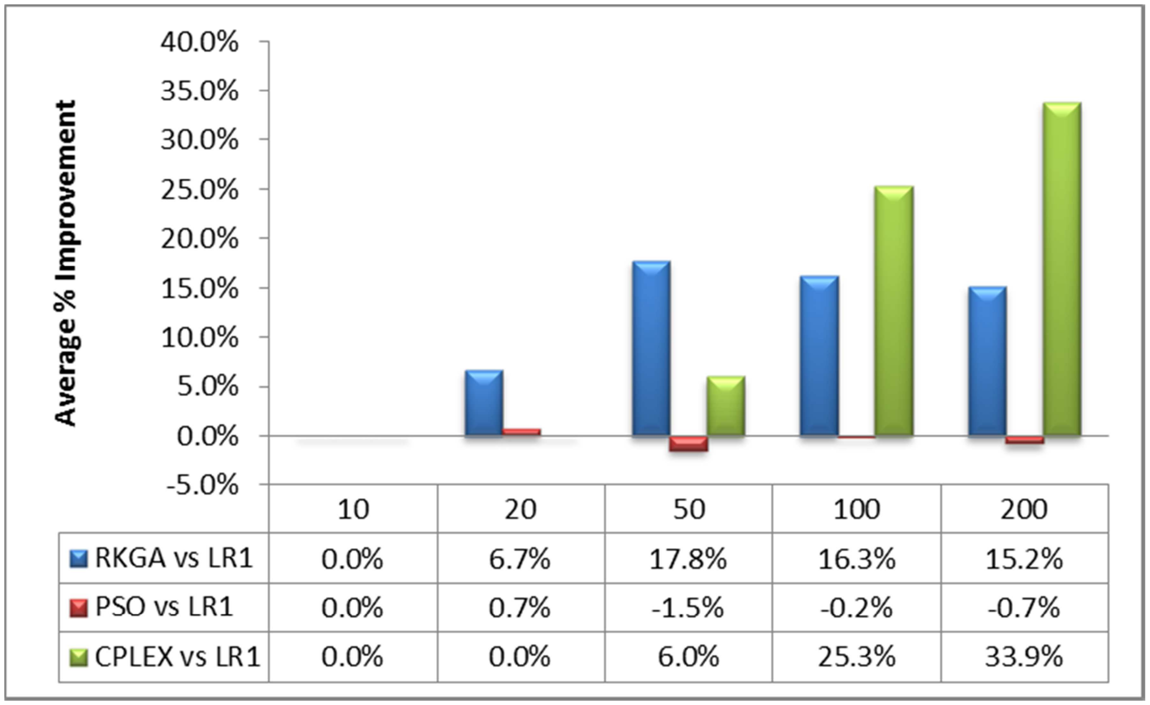
	5	20	5	20
🟦 RKGA vs LR1	15.7%	9.6%	12.1%	5.7%
🟥 PSO vs LR1	0.5%	1.1%	-2.8%	-2.2%
🟩 CPLEX vs LR1	26.8%	19.3%	24.0%	15.6%

Figure C-5: API Result for 100 Jobs in Two-Machine Problem



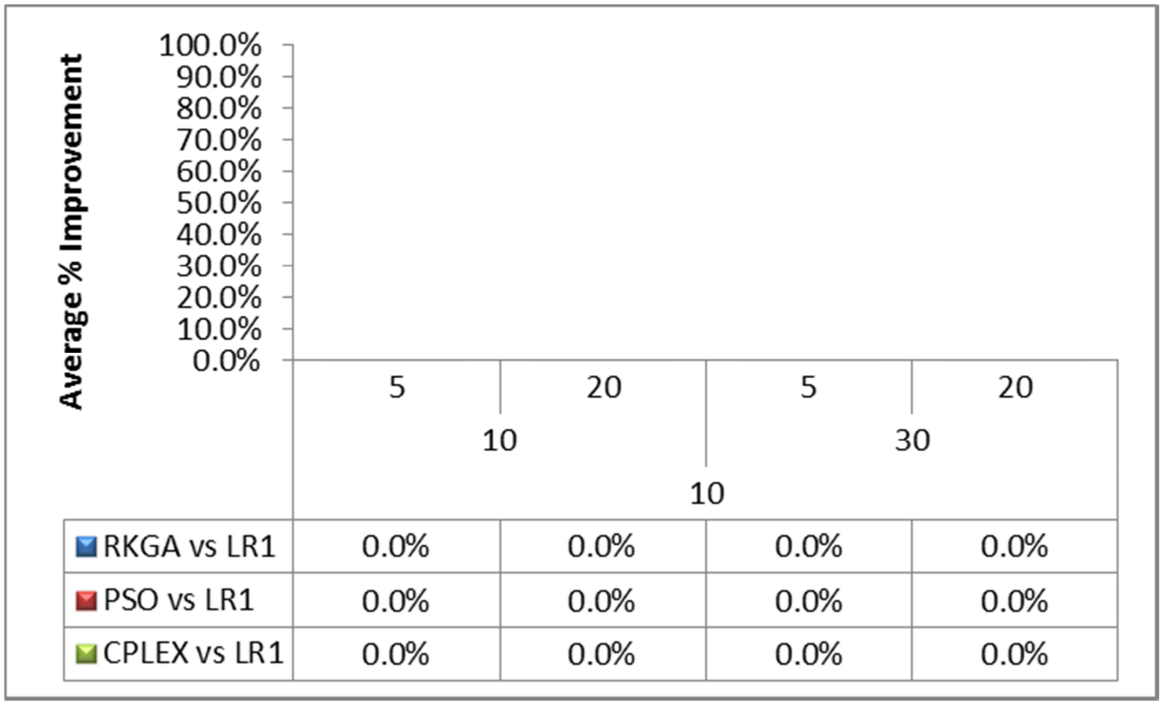
	5	20	5	20
🟦 RKGA vs LR1	17.3%	7.8%	13.0%	4.8%
🟥 PSO vs LR1	1.1%	0.9%	-3.7%	-3.1%
🟩 CPLEX vs LR1	39.6%	21.2%	37.4%	21.9%

Figure C-6: API Result for 200 Jobs in Two-Machine Problem



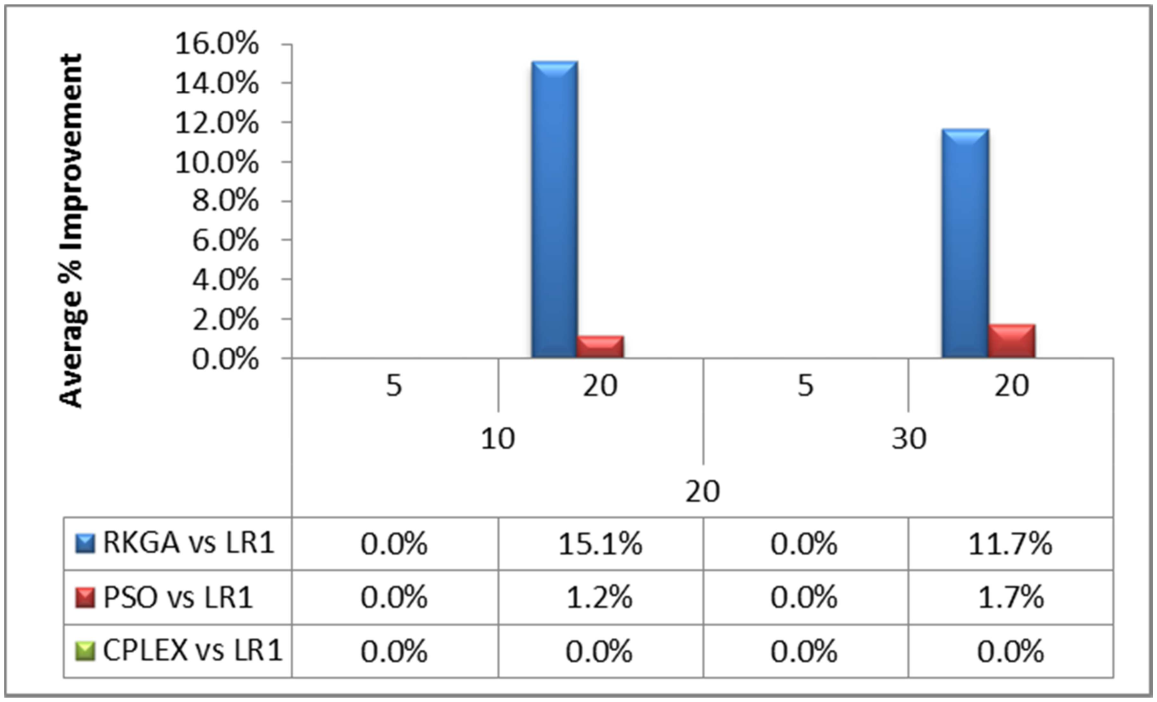
	10	20	50	100	200
🟦 RKGA vs LR1	0.0%	6.7%	17.8%	16.3%	15.2%
🟥 PSO vs LR1	0.0%	0.7%	-1.5%	-0.2%	-0.7%
🟩 CPLEX vs LR1	0.0%	0.0%	6.0%	25.3%	33.9%

Figure C-7: API Result for Four-Machine Problem



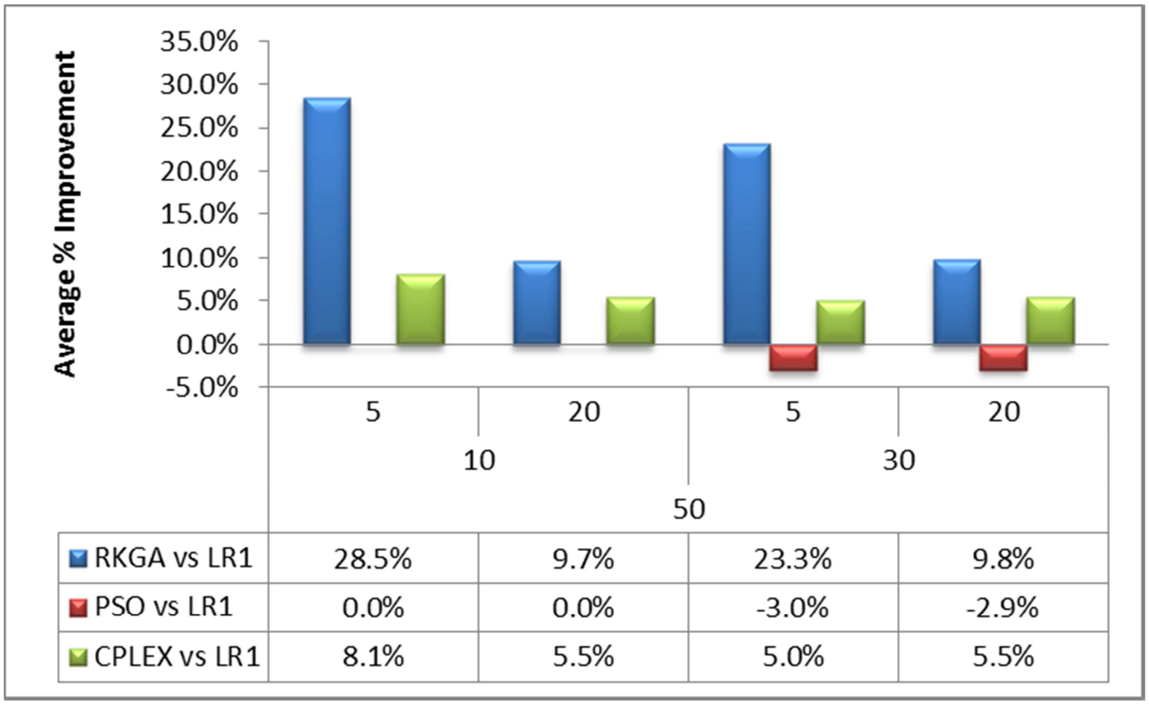
	10			
	10		30	
	5	20	5	20
🟦 RKGA vs LR1	0.0%	0.0%	0.0%	0.0%
🟥 PSO vs LR1	0.0%	0.0%	0.0%	0.0%
🟩 CPLEX vs LR1	0.0%	0.0%	0.0%	0.0%

Figure C-8: API Result for 10 Jobs in Four-Machine Problem



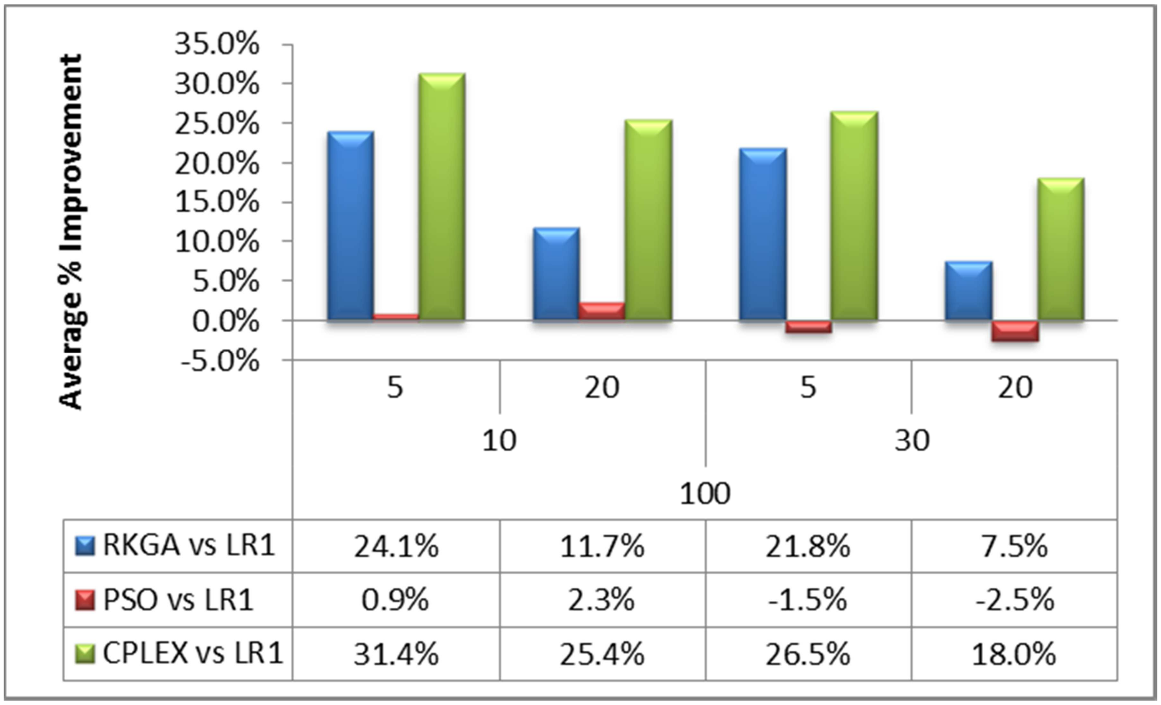
	20			
	10		30	
	5	20	5	20
🟦 RKGA vs LR1	0.0%	15.1%	0.0%	11.7%
🟥 PSO vs LR1	0.0%	1.2%	0.0%	1.7%
🟩 CPLEX vs LR1	0.0%	0.0%	0.0%	0.0%

Figure C-9: API Result for 20 Jobs in Four-Machine Problem



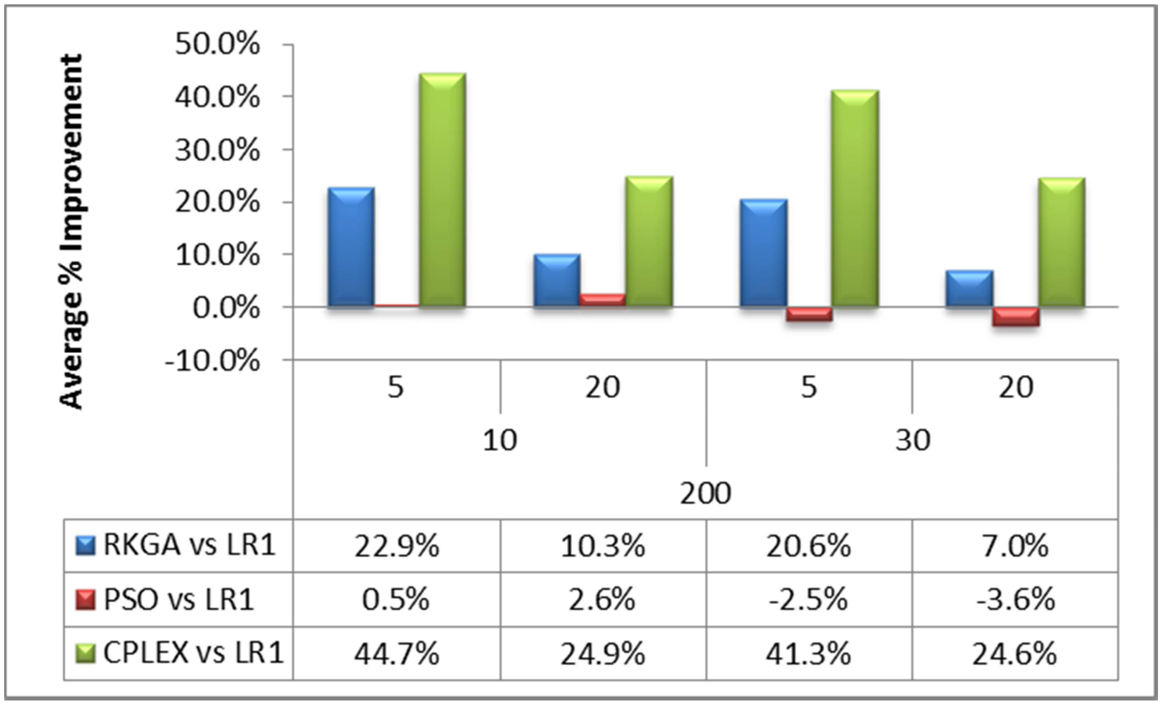
	50			
	10		30	
	5	20	5	20
🟦 RKGA vs LR1	28.5%	9.7%	23.3%	9.8%
🟥 PSO vs LR1	0.0%	0.0%	-3.0%	-2.9%
🟩 CPLEX vs LR1	8.1%	5.5%	5.0%	5.5%

Figure C-10: API Result for 50 Jobs in Four-Machine Problem



	5	20	5	20
	10		30	
	100			
🟦 RKGA vs LR1	24.1%	11.7%	21.8%	7.5%
🟥 PSO vs LR1	0.9%	2.3%	-1.5%	-2.5%
🟩 CPLEX vs LR1	31.4%	25.4%	26.5%	18.0%

Figure C-11: API Result for 100 Jobs in Four-Machine Problem



	5	20	5	20
	10		30	
	200			
🟦 RKGA vs LR1	22.9%	10.3%	20.6%	7.0%
🟥 PSO vs LR1	0.5%	2.6%	-2.5%	-3.6%
🟩 CPLEX vs LR1	44.7%	24.9%	41.3%	24.6%

Figure C-12: API Result for 200 Jobs in Four-Machine Problem

APPENDIX D: RKGA, PSO, CPLEX AND LR1 COMPUTATIONAL TIME

Table D-1: RKGA, PSO, CPLEX and LR1 computational time (in seconds) for 10 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J1p1s1#1	4.53	0.32	0.12	1.01	4.28	0.36	0.13	2.47
J1p1s1#2	4.15	0.30	0.08	0.98	4.10	0.30	0.09	2.34
J1p1s1#3	4.22	0.29	0.09	0.77	4.06	0.30	0.17	2.89
J1p1s1#4	4.47	0.29	0.09	0.27	4.07	0.24	0.17	0.89
J1p1s1#5	4.19	0.30	0.11	0.56	4.06	0.30	0.19	1.46
J1p1s2#1	4.23	0.41	0.14	64.40	4.13	0.40	0.25	10.32
J1p1s2#2	4.50	0.44	0.14	51.32	4.29	0.45	0.13	12.33
J1p1s2#3	4.58	0.40	0.25	59.13	4.12	0.41	0.19	8.32
J1p1s2#4	4.99	0.49	0.86	66.58	7.33	0.49	0.70	16.69
J1p1s2#5	5.98	0.47	0.34	65.41	4.11	0.49	0.25	14.65
J1p2s1#1	4.16	0.31	0.08	0.30	4.06	0.30	0.17	0.72
J1p2s1#2	4.46	0.30	0.11	1.06	4.06	0.29	0.17	0.45
J1p2s1#3	4.17	0.30	0.08	1.05	4.06	0.30	0.16	0.69
J1p2s1#4	4.17	0.29	0.11	0.89	4.06	0.29	0.16	0.55
J1p2s1#5	4.46	0.30	0.11	0.28	4.06	0.29	0.16	1.14
J1p2s2#1	4.19	0.41	0.17	117.10	4.11	0.42	0.14	10.44
J1p2s2#2	4.21	0.43	0.20	100.60	4.11	0.42	0.16	8.19
J1p2s2#3	5.62	0.43	0.27	106.76	4.10	0.45	0.17	2.78
J1p2s2#4	4.21	0.42	0.14	71.41	4.11	0.43	0.13	8.58
J1p2s2#5	4.95	0.38	0.19	94.64	4.11	0.38	0.16	1.06

Table D-2: RKGA, PSO, CPLEX and LR1 computational time (in seconds) for 20 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J2p1s1#1	11.14	0.52	0.39	114.075	10.53	0.48	0.83	6.188
J2p1s1#2	22.78	0.53	1.93	520.531	10.54	0.53	1.31	8.549
J2p1s1#3	11.75	0.54	0.69	33.322	10.53	0.53	1.00	5.134
J2p1s1#4	11.22	0.47	0.39	13.807	10.51	0.47	0.77	6.189
J2p1s1#5	22.64	0.53	0.44	15.025	10.53	0.53	0.81	6.225
J2p1s2#1	33.24	0.90	28.24	440.596	12.65	0.91	143.13	122.884
J2p1s2#2	15.01	1.00	139.68	389.546	18.56	1.01	1797.44	85.811
J2p1s2#3	18.12	0.95	593.41	362.093	12.63	0.95	386.79	70.341
J2p1s2#4	11.10	0.95	1800.00	96.004	13.66	0.95	87.19	122.65
J2p1s2#5	29.71	0.98	1800.00	89.763	15.59	1.02	546.86	19
J2p2s1#1	16.13	0.53	0.28	4.415	10.51	0.53	0.75	12.807
J2p2s1#2	19.77	0.53	0.48	0.967	10.52	0.54	0.89	6.7
J2p2s1#3	14.39	0.53	0.42	0.718	10.50	0.53	0.69	0.72
J2p2s1#4	21.15	0.53	6.27	1.155	10.78	0.53	1.26	2.786
J2p2s1#5	12.38	0.53	33.28	17.332	10.56	0.53	1.19	6.291
J2p2s2#1	12.65	1.10	29.91	99.401	27.55	1.13	120.18	97.889
J2p2s2#2	21.81	0.91	400.88	138.089	11.65	0.93	1800.00	65.71
J2p2s2#3	15.65	0.99	423.85	313.557	15.13	1.02	106.81	99.786
J2p2s2#4	34.64	0.94	1141.62	88.825	19.13	0.98	1800.00	67.124
J2p2s2#5	29.34	0.85	135.14	72.447	22.47	0.87	35.65	62.343

Table D-3: RKGA, PSO, CPLEX and LR1 computational time (in seconds) for 50 job instances

Runcode (1)	m = 2				m = 4			
	RKGA (2)	PSO (3)	CPLEX (4)	LR1 (5)	RKGA (6)	PSO (7)	CPLEX (8)	LR1 (9)
J3p1s1#1	100.81	2.17	1800.00	313.756	73.88	2.15	1799.88	313.756
J3p1s1#2	103.15	1.93	1800.00	172.067	55.55	1.99	42.87	1131.44
J3p1s1#3	87.91	2.02	1800.00	61.254	71.05	2.03	1800.00	1030.54
J3p1s1#4	137.98	2.01	1800.00	57.021	93.03	2.01	1799.88	1235.35
J3p1s1#5	137.26	2.02	1800.00	208.5	76.53	2.05	1799.96	2080.5
J3p1s2#1	179.49	3.90	1800.00	1033.13	163.71	3.88	1799.94	3304.22
J3p1s2#2	148.00	3.49	1800.00	1527.03	148.47	3.53	1799.91	2655.82
J3p1s2#3	165.33	3.53	1800.00	653.188	147.50	3.56	1800.00	2494.31
J3p1s2#4	157.86	3.68	1800.00	2170.78	152.11	3.69	1799.91	1965.04
J3p1s2#5	174.13	3.58	1800.00	824.977	144.75	3.62	1799.94	653.052
J3p2s1#1	143.18	2.12	1800.00	1309.86	116.81	2.13	1799.88	3342.34
J3p2s1#2	154.05	2.02	1800.00	1472.03	82.11	2.03	1799.91	1890.65
J3p2s1#3	108.16	2.01	1800.00	2934.09	84.83	2.02	1799.89	2934.09
J3p2s1#4	147.71	2.02	1800.00	1956.54	83.87	2.03	1799.91	1876.09
J3p2s1#5	129.08	2.09	1800.00	2417.59	93.08	2.11	1799.89	2417.59
J3p2s2#1	165.12	3.57	1800.00	1060.54	176.40	3.59	1799.97	1309.86
J3p2s2#2	201.85	3.37	1800.00	1546.24	143.84	3.40	1799.86	2423.51
J3p2s2#3	197.79	3.65	1800.00	1333.45	162.44	3.66	1799.96	2934.09
J3p2s2#4	171.95	3.91	1800.00	1902.34	182.63	3.88	1799.97	1956.54
J3p2s2#5	50.00	3.59	1800.00	1434.75	169.61	3.61	1799.97	2417.59

Table D-4: RKGA, PSO, CPLEX and LR1 computational time (in seconds) for 100 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J4p1s1#1	554.33	6.76	1800.00	3354.21	362.43	6.78	1799.77	2031.09
J4p1s1#2	446.51	6.78	1800.00	2103.45	401.75	6.82	1799.81	2616.968
J4p1s1#3	428.19	6.64	1800.00	2233.43	486.74	6.70	1799.72	2468.003
J4p1s1#4	492.46	6.73	1800.00	2097.54	364.61	6.79	1799.78	1220.956
J4p1s1#5	477.60	6.56	1800.00	3124.57	489.46	6.64	1799.77	1808.403
J4p1s2#1	535.44	12.15	1800.00	2341.052	566.48	12.23	1799.58	3102.431
J4p1s2#2	707.50	12.06	1800.00	3811.296	627.73	12.14	1799.69	3141.972
J4p1s2#3	588.59	12.18	1800.00	4190.459	682.78	12.32	1799.58	2743.377
J4p1s2#4	530.44	12.82	1800.00	2135.67	568.40	12.93	1799.72	2588.803
J4p1s2#5	585.41	11.99	1800.00	1831.77	838.79	12.02	1799.79	1334.962
J4p2s1#1	529.13	6.59	1800.00	1098.75	418.49	6.64	1799.69	1804.193
J4p2s1#2	599.85	6.79	1800.00	1132.43	470.49	6.90	1799.74	1726.076
J4p2s1#3	513.68	6.89	1800.00	1564.32	582.29	6.93	1799.75	1904.039
J4p2s1#4	462.02	6.69	1800.00	1980.54	461.25	6.69	1799.68	1505.297
J4p2s1#5	587.91	6.93	1800.00	2209.85	465.06	6.94	1799.77	2607.081
J4p2s2#1	549.31	12.31	1800.00	1615.662	660.34	12.50	1799.61	2119.655
J4p2s2#2	594.79	12.36	1800.00	2346.75	674.74	12.50	1799.92	1188.084
J4p2s2#3	550.85	12.68	1800.00	4320.76	849.61	12.75	1799.63	1349.823
J4p2s2#4	570.14	12.73	1800.00	1123.56	683.24	12.73	1799.68	2174.49
J4p2s2#5	624.57	12.02	1800.00	2464.643	589.10	12.14	1799.60	1779.494

Table D-5: RKGA, PSO, CPLEX and LR1 computational time (in seconds) for 200 job instances

Runcode (1)	**m = 2**				**m = 4**			
	RKGA (2)	**PSO (3)**	**CPLEX (4)**	**LR1 (5)**	**RKGA (6)**	**PSO (7)**	**CPLEX (8)**	**LR1 (9)**
J5p1s1#1	1718.45	24.55	1800.00	171.573	1698.09	24.67	1798.36	1673.162
J5p1s1#2	1759.82	24.15	1800.00	136.069	1694.01	24.38	1798.47	3165.128
J5p1s1#3	1756.97	24.16	1800.00	167.724	1598.89	24.34	1798.10	3242.3
J5p1s1#4	1753.38	24.42	1800.00	116.504	1696.59	24.53	1798.46	1970.229
J5p1s1#5	1754.88	24.53	1798.00	248.021	1696.87	24.74	1798.13	1816.126
J5p1s2#1	1797.00	46.42	1798.49	1705.421	1720.97	46.77	1807.61	1071.23
J5p1s2#2	1822.90	49.09	1798.10	1625.061	1722.34	49.22	1811.36	2610.976
J5p1s2#3	1813.89	44.93	1798.60	2545.125	1733.64	45.36	1809.14	1596.429
J5p1s2#4	1815.92	45.64	1798.19	2515.782	1727.90	45.73	1810.38	2427.649
J5p1s2#5	1816.54	45.59	1798.61	1569.421	1726.61	45.72	1815.48	2489.978
J5p2s1#1	1757.43	24.32	1798.50	117.922	1704.31	24.44	1798.95	2080.542
J5p2s1#2	1756.23	24.75	1798.71	1354.532	1702.06	24.86	1798.66	1039.68
J5p2s1#3	1765.60	24.47	1798.55	2132.919	1701.54	24.68	1799.59	1144.364
J5p2s1#4	1762.61	24.39	1798.80	1114.392	1700.65	24.77	1798.94	2765.4
J5p2s1#5	1763.56	24.59	1798.71	1117.875	1701.51	25.50	1807.29	2237.6
J5p2s2#1	1812.66	46.05	1810.24	3124.57	1729.24	46.48	1810.00	2176.53
J5p2s2#2	1820.05	46.78	1798.54	2341.052	1725.83	47.58	1810.25	1348.65
J5p2s2#3	1807.23	45.16	1811.28	3811.296	1725.87	45.43	1805.39	4365.76
J5p2s2#4	1737.01	47.56	1798.46	4190.459	1725.05	47.00	1809.55	2080.542
J5p2s2#5	1814.10	46.80	1798.47	2132.919	1729.86	46.40	1820.03	1039.68