COMPARISON OF GENETIC ALGORITHM AND PARTICLE SWARM OPTIMIZATION ALGORITHM FOR PERMUTATION FLOW SHOP SEQUENCING PROBLEM WITH CRITERION OF NUMBER OF TARDY JOBS

by

Hatice UÇAR

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering

May 2005
Istanbul, Turkey

APPROVAL PAGE

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

____
Prof. Dr. Mazhar ÜNSAL
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

____
Assist. Prof. M. Fatih TAŞGETİREN
Supervisor

Examining Committee Members

Assist. Prof. M. Fatih TAŞGETİREN	____________________
Prof. Dr. Mazhar ÜNSAL	____________________
Assis.Prof. Nurullah ARSLAN	____________________

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

____
Assis.Prof. Nurullah ARSLAN
Director

Date
May 2005

COMPARISON OF GENETIC ALGORITHM AND PARTICLE SWARM OPTIMIZATION ALGORITHM FOR PERMUTATION FLOW SHOP SEQUENCING PROBLEM WITH CRITERION OF NUMBER OF TARDY JOBS

Hatice UÇAR

M. S. Thesis - Industrial Engineering
May 2005

Supervisor Assist. Prof. M. Fatih TAŞGETİREN

ABSTRACT

In this study, we aimed to minimize the number of tardy jobs in permutation flow shops. The number of tardy jobs criterion is a measure for the number dissatisfied customers. In other words, it monitors the performance of the managers. In order to achieve the minimum number of tardy objective, we developed two algorithms; one is a traditional genetic algorithm and the latter is a discrete particle swarm optimization algorithm. The algorithms are implemented using the due date configurations of Demirkol et al.'s data sets. The statistical tests are done to measure the fitness and cpu values for each algorithm. The performances of both algorithms to find the optimal processing sequence for the jobs through m machines are compared. It is concluded from the experiments that the particle swarm optimization algorithm gives promising solutions by means of the proposed SPV (Smallest Position Value) heuristic rule.

Keywords: Scheduling, Number of Tardy Jobs, Genetic Algorithm, Particle Swarm Optimization, Smallest Position Value Rule, Permutation Flow Shop

POZİTİF GECİKMELİ İŞ SAYISI KRİTERLİ PERMÜTASYON AKIŞ TİPLİ ATÖLYE ÇİZELGELEME PROBLEMİ ÜZERİNDE GENETİK ALGORİTMA VE PARTİKÜL SÜRÜ OPTİMİZASYONU YÖNTEMLERİNİN MUKAYESESİ

Hatice UÇAR

Yüksek Lisans Tezi – Endüstri Mühendisliği
Mayıs 2005

Tez Yöneticisi: Yrd. Doç. M. Fatih TAŞGETİREN

ÖZ

Bu çalışmada permütasyon akış tipli atölyelerdeki pozitif gecikmeli iş sayısını minimuma indirmeyi amaçladık. Pozitif gecikmeli iş sayısı kriteri memnun olmayan müşterilerin sayısına ait bir ölçüdür. Diğer bir deyişle, bu kriter yöneticilerin performansına ait bir göstergedir. Minimum sayıda gecikmeli iş sayısı hedefini sağlayabilmek için iki algoritma geliştirdik; biri geleneksel bir genetik algoritma diğeri ise kesikli partikül sürü optimizasyonu algoritması. Algoritmalarda Demirkol ve diğerleri'nin data setlerindeki teslim tarihleri kullanıldı. Herbir algoritmanın gecikmeli iş sayısına ait değerleri ve işlemci süreleri için istatistiksel ölçümler yapıldı. Her iki algoritmanın m makinadaki en optimal iş sıralamasını bulmadaki performansları karşılaştırıldı. Yapılan deneylerde partikül sürü optimizasyonu algoritmasının En Küçük Konum (EKK) sezgisel yöntemi sayesinde umut verici neticeler verdiği sonucuna varıldı.

Anahtar Kelimeler: Çizelgeleme, Pozitif Gecikmeli İş Sayısı, Genetik Algoritma, Partikül Sürü Optimizasyonu, En Küçük Konum Kuralı, Permutasyon Akış Tipli Atölye

To my family

ACKNOWLEDGEMENT

I would like to thank my thesis supervisor Assist. Prof. Fatih TAŞGETİREN for his guidance throughout the research.

I express my sincere appreciation to the faculty member Mehmet ŞEVKLİ for his invaluable help during the implementation phase of the thesis.

Thanks go to the jury members, Prof. Dr. Mazhar ÜNSAL and Assist. Prof. Nurullah ARSLAN for their valuable suggestions and comments.

I express my thanks and appreciation to my family for their understanding, and patience. I also thank my friends for their motivation. Lastly, but in no sense the least, I am thankful to my elders who have encouraged and supported me during my education.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYSMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION

d_j	: Due date for Job j
C_{max}	: Completion time of all jobs in the flow shop
T_j	: Tardiness of Job j
$\sum U_i$	: Total Number of Tardy Jobs
PFSP	: Permutation Flow Shop Sequencing Problem
$Fm\| \|\sum U_i$	: Flowshop Sequencing Problem with Criterion of Number of Tardy Jobs
GA	: Genetic Algorithm
PSO	: Particle Swarm Optimization Algorithm
SPV	: Smallest Position Value Heuristic Rule
CPU	: Central Processing Unit
JxM	: j Number of Jobs through m Number of Machines

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Scheduling is of great importance in both manufacturing and service industries. It significantly improves the productivity and utilization of resources and profitability of production lines. It has many applications ranging from distribution networks to machine environment. In this study, we deal with the machine scheduling in permutation flow shops.

Flow-shop scheduling is one of the most well-known problems in the area of scheduling. Various dispatching rules, exact and heuristic methods have been proposed since the pioneering work of Johnson (1954) for the job sequencing problems on single or two machines, whereas the examples of flow shop scheduling are too few; and the available ones are mostly concerned with the makespan and/or maximum lateness minimization. (Hariri & Potts, 1989) developed the first exact algorithm for the NP-hard number of tardy jobs problem for flow shops, $Fm| \,|\sum U_i$. But, there is no metaheuristics developed for flow shops with the number of tardy jobs criterion.

We will be the first to apply the metaheuristics to the $Fm| \,|\sum U_i$ problem. Two algorithms to be used in the study are the genetic algorithms and the particle swarm optimization algorithm. GA is mostly used in scheduling problems and recently (Tasgetiren et al., 2004 a, b, c, d) applied PSO to sequencing problems. We preferred to use the benchmark suites of (Demirkol, 1998) instead of (Taillard, 1993)'s instances; since Taillard's instances don't include the due dates which are necessary for determining the number of tardy jobs.

1.2 PERMUTATION FLOWSHOP SCHEDULING

In a flow shop manufacturing system, different machines are set up in series and each task is performed on those machines. There are n jobs available to be processed on m number of machines. If the flow shop is a permutation type, a sequence of n jobs is determined and the jobs undergo through operation on the m machine without changing their sequence and skipping none of the machines. i.e., the operating sequence of tasks on each machine doesn't change.

Given the processing times p_{jk} for job j on machine k, and a job permutation $\pi=\{\pi_1,\pi_2, ...,\pi_n\}$, n jobs (j = 1, 2, ..., n) will be sequenced through m machines (k=1,2,...,m) using the same permutation.

The assumptions for the permutation flow shop problem are:

- Each task is to be processed once on each machine from machine 1 to m.
- Each task is done on at most one machine at a time.
- Each machine can process only one task at a time.
- Preemption is not allowed.
- Set-up times are negligible.
- All tasks release at work center at time zero.
- The operating sequence of tasks is the same on every machine.

1.3 NUMBER OF TARDY JOBS

Minimizing the number of tardy jobs is important for the manufacturer to avoid from the loss of good will of the customer. When the customer is dissatisfied, he will prefer another company. Therefore, it is of great interest in industry to minimize the number of tardy jobs. Studies are mostly done on single machine or two machine environments. In our study, we aim to determine a sequence that will minimize the number of tardy jobs in a permutation flow shop.

In order to define a job as tardy or early, we have to compute the completion time and the tardiness of each job. We compute the completion time, C_j, the tardiness, T_j and the total number of tardy jobs, $\sum U_j$ from the processing order of jobs. So we calculate the tardiness of jobs j using the formula, $Tj = \max\{0, Cj - dj\}$. Let's say c is the total number of tardy jobs ($\sum Uj$) counter,

```
$c_0 = 0,$
$for(j = 1, j <= n, j++)$
$if\ Uj = 1\ (job\ j\ is\ tardy) \Rightarrow c = c + 1$
```

And if we formulate the number of tardy job minimization we obtain,

$$\text{minimize} \sum_{j=1}^{n} U_j$$

subject to

$$\sum_{j=1}^{n} p_{ij} \le d_j \quad \text{for all i,} \quad \text{i} = 1,\ldots,\text{m} \tag{1.1}$$

$$U_j = \begin{cases} 0, & \text{if } C_j \le d_j \\ 1, & \text{if } C_j > d_j \end{cases}$$

$$p_{ij}, d_j \ge 0$$

1.3.1 Moore/Hodgson Algorithm

Moore's algorithm constitutes a basis for the number of tardy job minimization problems. The model was developed for the single machine case; however the model can be adapted to the m-machine case. As a matter of fact Hariri & Potts solved the permutation flow shop problem by dividing m-machine into one-machine subgroups and then applied Moore's algorithm to the single machine subgroups. Now let's mention a little bit about the algorithm and solve an example related to the solution method.

Step 1. Order the jobs according to the earliest due date rule (EDD). Set $c = 0$.

Step 2. Compute the tardiness of each job. Find the first tardy job, i. If none of the jobs is tardy, go to step 4.

Step 3. Find the longest job beginning from the 1st to ith position and remove it from the sequence and set $c = c + 1$. Then go to step 2.

Step 4. Form the schedule where the removed jobs are placed at the end of the scheduled jobs. The removed jobs are sequenced among themselves in any order.

Let's illustrate the rule with an example. In the given table the processing time, p_j and the due date, d_j of 5 jobs are given.

Table 1.1 An Example for Moore's Algorithm

Jobs, j	p_j	d_j
1	5	7
2	8	15
3	3	10
4	4	21
5	6	18

The jobs are ordered in EDD sequence and the tardiness, T_j of each job is calculated. In this schedule, the total number of tardy jobs is 3. Job 2 is found to be the first tardy job. Among the jobs 1, 3 and 2, job two has the longest processing time. It is seen that none of the jobs until job 2 is tardy.

Table 1.2.a Implementation of Moore's Algorithm

Jobs, j	p_j	C_j	d_j	T_j
1	5	5	7	0
3	3	8	10	0
2	8	16	15	1
5	6	22	18	4
4	4	26	21	5

Job 2 is removed from the schedule and placed to the end of the jobs. The tardiness of jobs is recalculated. Only job 2 is found to be tardy. This is the last schedule, since we can't do any modification on the new table. The total number of tardy jobs is reduced to 1. The last sequence is as given in Table 1.2.b.

Table 1.2.b Implementation of Moore's Algorithm

Jobs, j	p_j	C_j	d_j	T_j
1	5	5	7	0
3	3	8	10	0
5	6	14	18	0
4	4	18	21	0
2	8	26	15	11

1.3.2 Hariri & Potts' Implementation

The permutation flow shop problem with the objective of minimizing the completion time or the maximum lateness is of much interest in literature. But permutation flow shop problems with other objectives have received little attention. Hariri & Potts decided to apply the Branch & Bound Algorithm (B&B) to the NP-hard number of tardy job minimization problem in permutation flow shops.

In their study, the permutation flow shop is divided into separate single machine subproblems. In each subproblem, an initial partial sequence is developed for the early jobs and it is denoted as σ. When σ is not empty, C(σ, j) shows the earliest completion time for the jobs in σ on machine j (j =1, …, m). If σ is empty, then $C(\sigma, j) = \min_{i \in S} (\sum_{k=1}^{j-1} p_{ik})$ where S denotes the set of jobs not sequenced in σ. Now the permutation flow shop problem is equivalent to single machine subproblems in which job i (iϵS) becomes available for processing at time zero, requires a processing time p_{ij} on the machine for its completion and has a due date $d_i - C(\sigma, j) - \sum_{k=j+1}^{m} p_{ik}$. Moore's algorithm is now applicable to each single machine subproblem. E_j and L_j give the sets of early and tardy jobs. A lower bound can be obtained through max{|L_1|, …, | L_m|}. Two other lower bounds are also obtained. An upper bound for the number of tardy jobs is generated with a heuristic method. Then, using these lower and upper bounds a B&B algorithm is developed for the problem.

The algorithms were tested on problems having 2, 4, 6 and 8 machines with 10 jobs, 2, 3, 4 and 5 machines with 15 jobs, having 2, 3 and 4 machines with 20 jobs and 2 and 3 machines with 25 jobs. As m or n increases problems become rapidly harder. Optimal solution is obtained for the problems with m = 4 and n = 15 case.

1.4 EXACT METHODS, HEURISTICS, METAHEURISTICS

Performance of a solution method is determined through the execution time of the algorithm and the quality of the results. In order to be qualified, a method should give logical outputs and should achieve the objective of the study optimally.

Minimizing the number of tardy jobs on a single machine is an easy problem which can be solved polynomially within $O(n \log n)$ time and this kind of problem can be solved by mathematical (exact) models such as dynamic programming, branch &

bound or integer linear programming algorithms. If the complexity of the problem is a little bit increased, it is still possible to solve the problem polynomially by making some assumptions.

Exact methods are able to solve only small sized problems. As a matter of fact Hariri & Potts's B&B algorithm is useful when the number of machines and jobs are less. If we attempt to solve complex problems with exact methods, even we are sure that we will get the optimal results at last, our lives may not allow completing the runs; since enumerating the problem takes very long times.

Dispatching rules also work as well for simple problems, but they are not so efficient for more complex problems. The Shortest Processing Time (SPT) rule used to be the most effective method for small sized problems until Lodree et al. (2004) proposed a new sequencing rule named Earliest Adjusted Due Date (EADD) for the $Fm|r_j|\sum U_i$ problems up to 50 jobs. They derived it by the same way as the way of Hariri & Potts, dividing the problem into one machine subproblems. The EADD rule is computationally more expensive than SPT.

Heuristics reveal invaluable solutions. Known heuristic methods start with a single solution and try to develop better solutions in the next generations from the solution currently at hand. Worse solutions are not accepted. Therefore, the execution terminates at the first local minimum being trapped.

Note that, a landscape of an objective function does not have a continual increase or decrease. Namely, let's say in a minimization problem, two succeeding minima may have a maximum point in between. If our solution is closer to the local minimum, then we have the risk to be trapped at this local minimum. Not accepting the worse solutions will hinder to overcome the hill and reach the global minimum. Thus, we can say that, heuristics do not always offer us optimal solutions.

Heuristics were developed for the minimizing the number of tardy jobs in single machine problems or they were used for initializing the solution or the solution set of metaheuristics.

Recently metaheuristics are of the greatest interest; since they give the optimal or near-optimal solutions in a shorter time than exact algorithms do. Mostly used metaheuristics are the Simulated Annealing, Tabu Search, Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization etc.

In Simulated Annealing and Tabu Search, solutions are obtained from an initially formed solution; whereas in Genetic Algorithms, Particle Swarm Optimization or Ant Colony Optimization methods, the solutions are obtained from an initially constructed population. The general usage of metaheuristics for flow shop problems consists of minimizing the makespan or maximum tardiness/earliness of jobs.

(Bertel & Billaut, 2004) developed a GA to minimize the weighted number of tardy jobs in small sized flow shops.

(Tasgetiren et al., 2004d) applied PSO to single machine problems to minimize the total weighted tardiness. The algorithm is modified for permutation flowshop sequencing problems with the makespan or tardiness criterion in (Tasgetiren et al., 2004 a, b, c).

The solutions obtained from the metaheuristic methods can be improved by hybridizing the algorithm with local search. The algorithm starts with a complete solution and tries to find a better solution by using the neighborhood of the current solution. We call the solutions as neighbors if the latter solution can be obtained by modifying the current one.

Metaheuristics are able to solve difficult problems in few minutes. Since in metaheuristics, worse solutions are given an opportunity, being trapped at local optima is prevented. Therefore, the solution quality will be increased if metaheuristic methods are used.

Our study is organized as follows: Chapter 2 gives extensive information about the literature on permutation flow shop problems with different objectives, with the subject of number of tardy minimization, genetic algorithm and particle swarm optimization algorithm usage. In Chapter 3 and 4, the Genetic and Particle Swarm Optimization Algorithms and their methodology are introduced respectively. The experimental design and computational results of the study are presented in Chapter 5. Finally, in Chapter 6 we draw conclusions from results.

CHAPTER 2

LITERATURE REVIEW

2.1 PERMUTATION FLOWSHOP SCHEDULING WITH THE CRITERION OF NUMBER OF TARDY JOBS

There are many exact, heuristic and metaheuristic methods developed for machine scheduling. In this study, we are going to focus on the literature about the number of tardy jobs on a single machine, two-machine and m-machine environment. The goal in the articles is generally to minimize the number of tardy jobs, or maximize the total number of on time jobs.

Single machine scheduling constitutes the widest part of the literature. Scheduling n jobs on one machine is fundamental to scheduling theory and it constitutes a basis for n job problems with more than one machine. There are also examples of two machine scheduling; but examples of m-machine environment in the literature are too few.

(Moore, 1968) found that minimizing the number of tardy jobs (∑Ui) is an easy problem and requires $O(n \log n)$ time to be solved; whereas (Karp, 1972) demonstrated that the problem of minimizing the weighted number of tardy jobs (∑wiUi) on a single machine, to be NP-hard. (Lawler, 1976) generalized Moore's algorithm to the case of agreeable weights. i.e., $p_i < p_j$ implies $w_i \geq w_j$

(Lawler & Moore, 1969) presented a functional equation and applied this function to resource allocation and sequencing problems including the criterion of weighted number of tardy jobs. The function is similar to the knapsack problem formulation when common due date is used. They derived a pseudo polynomial dynamic programming (DP) algorithm which requires $O\left(n \min\left\{\sum_j p_j, \max_j \{d_j\}\right\}\right)$ time.

(Sahni, 1976) presented an algorithm which requires $O\left(n \min\left\{\sum_j p_j, \sum_j w_j \max_j \{d_j\}\right\}\right)$ time and is a generalized version of Lawler & Moore's. In this algorithm, weights are set to be integer.

The DP sequencing method of Lawler & Moore is more efficient than the methods of (Held & Karp, 1962). It carries out a search 2^n whereas the latter has n! possibilities. (Hariri & Potts, 1994) showed a DP algorithm that solves the weighted number of tardy jobs problem with up to 300 jobs. They used another term as "deadline" besides the "due date" term. A job must be completed before its deadline. Time complexity of the algorithm is $O(n2^n)$. DP state-space relaxation technique is used to derive the lower bounds for the problem.

(Kise et al., 1978) proposed a $O(n^2)$ time algorithm for minimizing the number of tardy jobs with the assumption of $r_i < r_j \Rightarrow d_i \leq d_j$ whereas Lawler developed a $O(n \log n)$ for this problem.

Branch & Bound (B&B) algorithm is successful in solving the knapsack-like problem. (Villareal & Bulfin, 1983) proposed a B&B algorithm with 2 lower bounding procedures and a dominance theorem. Their method could solve at most 50-job problems. The algorithm of (Potts & Wassenhove, 1998) requires $O(n \log n)$ time. Their method is successful in solving problems with up to 100 jobs. Besides, they also described some lower bounding schemes. (Hallah & Bulfin, 2003) used the B&B algorithm. Till that time there was not exist any exact algorithm for the problems with more than 1000 jobs. They developed a heuristic and an exact algorithm for 2500-job problems. (Lawler & Moore, 1969)'s dynamic programming approach used to solve problems with up to 1000 jobs.

(Yoo & Martin-Vega, 2001) presented many heuristics and developed a general algorithmic approach from the Moore/Hodgson algorithm to minimize number of tardy jobs. They compared their results with other algorithms, GAA obtained rather good results. They showed that GAA is a viable approach in dynamic problems.

(Kethley & Alidaee, 2002) evaluated the performance of various scheduling rules, heuristics and algorithms on the problem of the minimization of number of tardy jobs on a single machine.

When precedence constraints are added to the number of tardy jobs on a single machine problem, it becomes NP-hard. (Steiner, 1997) worked on the weighted version of the problem and proposed polynomial solutions.

In the relocation problem of (Lin & Cheng, 1999), a reconstruction sequence for a set of old buildings is determined under a limited budget. They solved the problem with due date constraints as a resource constrained scheduling problem. They studied on two problems. In the first problem, they showed that weighted number of tardy jobs on a single machine problem is NP-hard when common due date is used. In the second problem, the tardiness issue is studied. When each jobs has individual due dates, the problem is to be strongly NP-hard.

(Rote & Woeginger, 1998) studied on the $1|s_f|\sum U_j$ problem. Setup is needed whenever a switch occurs between different family batches. However the problem was found to be NP-hard, they showed that the problem with uniform family due dates is solvable in polynomial time.

(Dauzére-Pérés, 1995) proposes a lower bound and a heuristic for the $1|r_j|\sum u_j$ problem. He relaxed the MILP formulation in order to obtain the lower bound. It is found to be the first heuristic to minimize the number of late jobs in the general single machine scheduling problem. The NP-hard problem can be solved in $O(n \log n)$ time (Moore, 1968), when release dates, r_j are equal. He proposes a B&B algorithm where the release dates are non-equal in another article (Dauzere-Peres and Sevaux). The algorithm solves most of the instances up to 100 jobs in reasonable CPU times. (Baptiste et al., 2003) solve problems with 200 jobs with a B&B algorithm.

(Dauzére-Pérés & Sevaux, 1999) was first to propose an algorithm for the NP hard $1|r_j|\sum w_j U_j$ problem and to introduce the “master sequence” notion. The algorithm was able to solve problems of more than 100 jobs. They used lagrangean relaxation algorithm to find both the upper and lower bounds and introduced a new

MILP formulation for the problem. In (Sevaux and Dauzere-Peres, 2003), they improved the lagrangean relaxation algorithm for the $1|r_j|\sum w_jU_j$ problem. The paper is the first to apply metaheuristics on a single machine scheduling problem.

(Peridy et al., 2001) used a short-term scheduling memory approach to find lower bounds and then applied a B&B algorithm to the $1|r_j|\sum w_jU_j$ problem. The short-term memory stores some of the last jobs processed to avoid repetitions. The method can solve optimally up to 100 jobs.

(Güner et al., 1998) considered a dual objective in the one machine problem such as the minimization of the maximum earliness and the minimization of the number of tardy jobs. B&B algorithm was presented by connecting Moore's algorithm. The algorithm is not practical for problems with more than 25 jobs because of the combinatorial nature of the problem.

(Chang & Su, 2001) considered another bicriteria scheduling problem with the aim of minimizing the maximum lateness and the number of tardy jobs. A simple procedure is introduced to identify 2 critical jobs. They used a B&B algorithm which can solve problems with 50 jobs.

(Köksalan & Keha, 2003) considered the two single machine bicriteria scheduling problems which are the minimization of flow time & number of tardy jobs and the minimization of flow time & maximum earliness. They used a hybrid GA to solve the problems. The algorithm is successful in solving bicriteria scheduling problems.

The examples given above are about scheduling of one machine with non-preemption. Preemption case is allowing other jobs to interrupt the processing of a job, so that job is resumed at a later date. (Lawler, 1990) studied on that kind of problem. He described a dynamic programming algorithm for $1|pmtn,r_j|\sum w_ju_j$ problem. (Gordon & Kubiak, 1998) examined the complexity of the $1|r_j|\sum w_jU_j$ problem for both the preemptive and non-preemptive case and he showed that both problems are NP-hard.

Recently (Mosheiov & Sidney, 2004) presented a paper and studied the one machine scheduling problem from a different perspective. They studied the effect of learning on the $1\| |\sum U_j$ problem and as a result they found that Moore's algorithm doesn't guarantee optimal schedule when learning curves are assumed.

There are a few articles written about minimizing the number of tardy jobs on two machines. The $F2\|\Sigma U_i$ problem complexity is first settled by (Lenstra et al., 1977) to be strongly NP-hard. (Gupta & Hariri, 1997) presents four special cases which can be solved polynomially and several heuristics by which problems can be solved near optimal. The problem with up to 60 jobs can be solved by the algorithms.

(Croce et al., 2000) proposes a B & B algorithm for the two-machine flow-shop sequencing problem. In the problem, the jobs have to be completed at a common due date. This problem with a common due date is also proven to be NP-hard. The proposed algorithm is capable of finding optimal solutions for problems up to 900 jobs. Bounds are derived from the continuous relaxation of the multidimensional KP formulation.

In (Lin, 2001), Lin proposes a polynomial time algorithm for some special cases for two machine flowshop with two objectives. The objectives are the minimization of the maximum tardiness, T_{max} and the number of tardy jobs, N_t.

(Bulfin and Hallah, 2003) applied a heuristic and a B&B algorithm for the weighted and unweighted number of tardy jobs on two machines problem. KP formulation was relaxed in order to find the bounds. The algorithm is successful in solving problems with 100 jobs for both the weighted and unweighted cases. The previous exact algorithms could solve the unweighted problems up to 25 jobs (Hariri and Potts, 1989).

Scheduling n jobs on permutation flowshop problems are mainly concerned with the minimization of the completion time (Iyer et al, 2004), (Reeves et al, 1998), (Yamada and Reeves), (Bertel and Billaut, 2004). (Taillard, 1990) compared the best heuristic methods with his Tabu Search (TS) algorithm that he developed for FSSP. (Ying & Liao, 2004) presented the Ant Colony System (ACS) heuristic to the minimization of completion time in PFSP. This is the first metaheuristic to apply ACS to this kind of problem. (Tasgetiren et al., 2004 a, b and c) applied Particle Swarm Optimization (PSO) algorithm to permutation flowshop sequencing problems. Among these articles, the first one has a single criterion and the last two have two criteria. (Leu & Hwang, 2002) solved a mixed precast production problem as a resource constrained flowshop scheduling problem using Genetic Algorithms (GA). (Ruiz et al., 2004) hybridized GA in order to solve the PFSP with sequence dependent setup times.

To our knowledge, (Hariri & Potts, 1989) presented the first exact algorithm for the NP-hard $Fm| |\sum U_i$ problem. They aimed to find a sequence that minimizes the number of tardy jobs in flowshop problems. They derived a lower bound by dividing m machines into several one machines. According to the paper, the m-machine problem is equivalent to a single machine problem with due dates. Moore's algorithm is applied to the single machine subproblems to solve the flowshop problem.

The Shortest Processing Time (SPT) rule used to be the most effective method until (Lodree et al., 2004) proposed a sequencing rule for $Fm|r_j|\sum U_i$ problems up to 50 jobs. The new rule is named as the Earliest Adjusted Due Date (EADD). They derived it by the same way as the way of Hariri & Potts, dividing the problem into one machine subproblems. The EADD rule is computationally more expensive than SPT.

(Bertel & Billaut, 2004) developed a GA for an industrial multiprocessor FSSP with recirculation. The problem is to perform jobs between a release date and a due date, in order to minimize the weighted number of tardy jobs.

2.2 GENETIC ALGORITHMS

Genetic algorithm (GA) is a well-known and mostly used evolutionary computation technique, which was developed by John Holland and his PhD students (Holland, 1975). The idea was inspired from Darwin's natural selection theorem which is based on the idea of the survival of the fittest.

Genetic algorithms have an initial population composed of randomly generated solutions. There are three stochastic operators such as selection, crossover and mutation which are applied to the set of solutions iteratively to produce hopefully better solutions. In selection, most fit members survive and the least fit are eliminated. Differentiation is attained through crossover and mutation. There is a probability for crossover and mutation. (Beasley et al., 1993 a and b) give detailed information about the fundamental and advanced aspects of GAs in both articles respectively. The concepts like selection, crossover, mutation, and their techniques are described and some terms like, deception, epistatis, restricted mating, speciation etc are given. Starkweather et al. compared six crossover operators over Traveling Salesman Problem and a warehouse scheduling problem found that the effectiveness of the sequencing operators changes depending on the problem domain.

Whitley presents the strengths and weaknesses of evolutionary algorithms covering genetic algorithms, evolution strategies, genetic programming and evolutionary programming. He gives more experimental forms of GAs including the parallel island models and parallel cellular genetic algorithms in his articles. He reviews the theoretical foundations of GA.

Genetic algorithms have a wide range of applications ranging from optimization, design and machine learning problems to scheduling problems and etc.

The success in solving scheduling applications mostly depends on the choice of the search algorithm. Choosing an appropriate technique can be possible in two ways: the generality of the algorithm can be examined or a comparison can be done after applying many algorithms to the scheduling problem. (Rana et al.) opted to compare the performance of heuristic, evolutionary and local search approaches on solving a warehouse scheduling problem.

In this century, industrial robots perform many tasks. The aim of using them is to reduce the cycle time and to obtain high productivity. An industrial robot has constant finishing time for each operation, but it is possible to reduce the makespan with an optimal sequence of tasks. Zacharia and Aspragathos applied GA to cope with this problem (Zacharia and Aspragathos, 2005)

It is significant to arrange an optimal curriculum schedule for every school. This is not only required for educational goals, but also for effectively utilizing the faculty resources. This is a rather difficult task and GA has a great reputation in solving this problem. There are many articles written over this topic (Wang, 2005).

(Chan and Chung, 2004) emphasizes the trade-off between the earliness on time and the tardiness situations for a distribution network. They develop a multi-criterion genetic optimization methodology. The proposed algorithm combines analytical hierarchy process, a multi-criterion decision making tool, with GAs.

Whitley used GAs for setting weights in neural networks (NN). The training data were used to estimate the output behavior of NN. It is understood that combining GA with NN gives promising results.

GA has a wide usage in scheduling job shops and flowshops and they give rather good results for both kinds of problems. There are many articles available in literature. Generally objective function is set to minimize the make span or minimize the earliness/tardiness in these articles. In (Leu and Hwang, 2002), a resource constrained flowshop scheduling model is used to solve a mixed precast production problem. Since precast production has many tasks which are done in the same order through all tasks, it is as flowshop process.

Spending less time when executing the optimization algorithms is substantial. In order to produce high performance for the application's execution, a genetic algorithm was proposed to create a partition and schedule in the study of Moore and Rodrigues.

Good properties of search methods are generally integrated so as to find better solutions in a reasonable amount of time. This kind of algorithms is called as hybrid algorithms. In (Kim, 2003), a hybrid GA was combined with fuzzy logic for solving resource-constrained project scheduling. Nearchou added some features of GA and local search to his Simulated Annealing (SA) algorithm for finding an optimal scheduling for a flowshop problem with the makespan criterion (Nearchou, 2004). The hybrid GA was successfully applied to permutation flowshops for solving sequence-dependent set-up times (Ruiz et al., 2004). The parameters were fine tuned by using design of experiments approach. Hino et al. combined the best characteristics of genetic algorithms and tabu search to solve the earliness/tardiness problem in a single machine environment (Hino et al., 2005). Whitley et al. compared several heuristic methods for scheduling the shipment of customer orders for a warehouse. They aimed to reduce the shipping time and minimize the inventory. Hybrid GA was appeared to be the best algorithm for solving the problem.

2.3 PARTICLE SWARM OPTIMIZATION

The Particle Swarm Optimization (PSO) algorithm is a new population-based evolutionary computation technique, which was originally developed by Kennedy & Eberhart in 1995. It was inspired from the social behavior of animals such as fish shoaling and bird flocking. It has some tuning parameters which influence the performance of the algorithm; the exploration and exploitation tradeoff. In the work of Eberhart, Simpson & Dobbins, it was realized that some of the parameters were redundant, and they removed from the original algorithm, (Eberhart, 1996). The mathematical equations of the original version of PSO is as,

$$v_{id}^{k} = v_{id}^{k-1} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id}) \tag{2.1.a}$$

$$x_{id} = x_{id} + v_{id} \tag{2.1.b}$$

c_1: cognition learning rate
c_2: social learning rate
(c_1, c_2 are taken to be 2.05 in general)

(Trelea, 2003) gives some insights about parameter selection in PSO. According to the article, some parameters can be discarded; since they add no value to the algorithm. Trelea analyzes the deterministic PSO algorithm for its dynamic behavior and convergence property.

The velocities of particles' on each dimension are restricted to V_{max}. A larger V_{max} facilitates global exploration, while smaller V_{max} facilitates local exploitation. Shi and Eberhart added the inertia weight as a constant to the velocity in order to control the exploration and exploitation (Shi and Eberhart, 1998 a and b). The use of inertia weight improved the performance of the algorithm in many applications.

$$v_{id}^{k} = w v_{id}^{k-1} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id}) \tag{2.2.a}$$

$$x_{id} = x_{id} + v_{id} \tag{2.2.b}$$

w: inertia weight

Chatterjee & Siarry proposed a new variation of PSO which introduced a nonlinear inertia weight for the particle's old velocity in PSO equations and it improved the convergence as well. It also presented a way for parameter selection and compared the results with other parameter selection methods

(Clerc, 1999) introduced the constriction factor (K) to PSO. It controls, constrains velocities and thus insures convergence. The constriction factor negated the need for V_{max}.

$$v_{id}^{k} = \kappa\left[v_{id}^{k-1} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id})\right] \tag{2.3.a}$$

$$\kappa = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \text{ where } \varphi = c_1 + c_2, \quad \varphi > 4 \tag{2.3.b}$$

(Eberhart and Shi, 2001d) demonstrated that although previous evolutionary paradigms can generally solve static problems, PSO can successfully optimize dynamic systems. It can not be known when a larger or a smaller inertia weight is needed. Therefore, that value is set to a dynamic value which starts from 0.9 and descends linearly till 0.4.

Five or six years after the introduction of PSO Eberhart and Shi reviewed the development, applications and the written books and articles (Eberhart and Shi, 2001b)

We can see many applications of PSO algorithm in the literature. The first application was about the network architecture and is available in the articles of (Shi and Eberhart, 1998), (Kennedy et al., 2001). (Eberhart & Hu, 1999) evolved the neural network with PSO in order to diagnose the human tremor.

When numerically controlled machines emerged, the productivity was far from being optimal. As an example to optimize these systems, the metal removal operation was investigated by Tandon (Tandon, 2000).

(Pavlidis et al., 2004) compared PSO with other computational intelligence methods in finding the Nash equilibrium in game theory.

Voss and Howland introduce a new method called social programming, which is a logical extension of PSO, the Group Method of Data Handling and Cartesian Programming. They used the method for predicting closing stock prices.

The (Zhang et al., 2004) research constitutes an alternative solution solving scheme for resource-constrained project scheduling problem.

(Ghoshal, 2004) compared some metaheuristic techniques such as PSO, a hybrid PSO and a hybrid GA-SA to optimize the proportional-integral derivative gains, which are used in multi-area thermal generating plants. He found PSO to be more optimal and it is achieved in least time.

Recently (Tasgetiren et al., 2004 a, b, c and d) applied PSO to sequencing problems. In (Tasgetiren et al., 2004d) they used the Smallest Position Value (SPV) heuristic rule to apply continuous PSO algorithm to all classes of sequencing problems which are to be NP-hard. The study is the first to apply PSO to single machine problems. Variable Neighborhood local search technique was embedded in the algorithm. This had a great impact on the solution quality. The algorithm is modified for permutation flowshop sequencing problems in (Tasgetiren et al., 2004 a, b and c). While the first article has a single criterion, the last two ones include two criteria.

Other applications include, power and voltage control, ingredient mix optimization, system design, multi-objective optimization, pattern recognition, biological system modeling, signal processing, robotic applications, decision-making, simulation,…etc.

The evolutionary computation paradigms and PSO algorithm were compared in many articles (Eberhart and Shi, 2001a), (Angeline, 1998) and (Kennedy and Spears, 1998). In the study of Eberhart and Shi, the operators of each paradigm are reviewed. The objective of comparison is not to determine which algorithm is better; but to demonstrate how each of them works, and how can they be combined to improve the performance. Some of the features of GA were incorporated to PSO. (Penev and Littlefair, 2004) introduced a noval population-based optimization method, called Free Search, and compared it to different optimization methods including PSO. They say that PSO has a good convergence speed.

In the work (Carlisle & Dozier, 2001) of Carlisle & Dozier some parameter settings are recommended such as;

Population size $= 30$
$V_{\max} = 0;$ if $X_{\max}$ is enforced
$c_1 = 2.8$ *and* $c_2 = 1.3$
$\varphi = 4.1$

They wanted to define a general purpose a PSO swarm, to be used as a base swarm description. In another study, they showed a method for adapting the particle swarm optimizer to dynamic environments. The particle resets its previous best record whenever the environment changes and forgets about its experience to that point. The type of resetting changes based on the iteration count or the magnitude of change in the environment. It is concluded that a more gradual reset throughout the population might provide better convergence.

(Løbjerg et al., 2001) combined PSO with genetic algorithm concepts and evaluated if it was competitive on function optimization. They employed the concepts of breeding and subpopulation for velocity and position updates. The method was heavy computationally due to the additional burden of breeding and subpopulation.

(He et al., 2004) presented a new hybrid particle swarm optimizer with passive congregation. Passive congregation is a biological term. It insures the integrity in the swarm by social forces. By means of passive congregation, information can be shared between particles and this will help avoiding being trapped at local optima.

Like many other evolutionary and classical minimization methods, PSO suffers from being trapped at local optima. Another new technique to alleviate the local optima problem was introduced by Parsopoulos et al. The method is called Stretching Technique. It consists of two-stage objective function. The method is applied after a local minimum has been found. The local minima were eliminated while the global minimum is preserved in the method.

Finally, we can say that PSO is a powerful method for optimizing continuous functions. However, it is not sufficient for solving discrete cases. (Deroussi et al., 2004) showed the Discrete Particle Swarm Optimizer (DPSO) to solve the combinatorial optimization problems. He combines local search and path relinking to DPSO and applies it to the well-known Traveling Salesman Problem. The proposed algorithm competes with the best iterated local search methods.

CHAPTER 3

GENETIC ALGORITHMS

3.1 AN OVERVIEW

Genetic algorithms (GA) belong to the class of metaheuristics[1]. It was firstly introduced by John Holland in 1960. The idea was inspired from a biological issue which is known to be Darwin's evolution theorem. The evolutionary ideas of the natural selection and genetics constitute the basics of GA. The concepts used in the algorithm are same as the ones which are used in biology, e.g., the genes on each chromosome correspond to variables of each solution. In the algorithm, the survival of the fittest among individuals over consecutive generations is simulated when a problem is being solved.

GAs are good at solving continuous and discrete combinatorial problems. The probability of getting 'stuck' at local optima is less than the gradient search methods. But GAs are computationally expensive. It is simple to deal with them; since very good results can be obtained for different kind of problems, even when a little change is done on the existing algorithm.

Whereas most stochastic search methods start with a single solution, genetic algorithms start with a population of solutions. An initial population is formed randomly or by means of a heuristic algorithm. Solutions are encoded in a form, which are called chromosomes. Each chromosome shows a complete solution to a problem. They are each assigned to a fitness score which represents the ability of chromosomes to compete for mating and staying alive.

[1] The other well-known metaheuristics are Simulated Annealing and Tabu Search. There are also some recently used metaheuristics available; the Ant Colony Optimization, Differential Evolution and the Particle Swarm Optimization.

Parents are picked up to mate according to their fitness values. The fitter chromosomes produce more offspring than the less fit chromosomes. The solution set is then imposed to crossover, mutation and inversion. These stochastic operators are required for diversifying the solution pool and especially getting better solutions. Since the size of the population should be maintained statically, some weak individuals in the population die, and better solutions thrive to stay alive. The cycle continues until some certain number of iterations is executed or once the population converges[2].

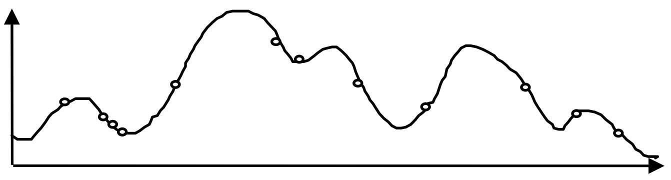
Figure 3-1.a Distribution of Individuals in Generation 0

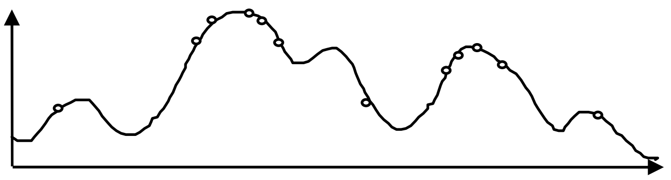
Figure 3-1.b Distribution of Individuals in Generation N

3.2 THE PSEUDOCODE OF THE ALGORITHM

```
Genetic Algorithm ( )
{
Initialize population P of size λ            /* a randomly generated population*/
Evaluate λ individuals in P                  /*check the fitness of each chromosome*/
While termination criteria not satisfied do
        { Select 2* μ individuals from P
        Crossover individuals to produce μ offspring
        Mutate some individuals in μ
        Add μ offspring to λ individuals in P
        Evaluate (λ + μ) individuals in P
        Select λ individuals from (λ + μ) individuals in P }
End Genetic Algorithm ( )}
```

[2] Convergence is defined as the progression of solutions towards uniformity. Similarity among fitness values increases as the population converges to the best fitness value obtained so far.

3.3 REPRESENTATION

Any representation can be used for chromosomes such as; strings of bits, arrays, trees, lists, or any other object. Mutation and crossover operators are defined according to the representation used. For example, for permutation flow shop sequencing problem, each gene on a chromosome is represented as a list of the job numbers; e.g., in chromosome {9,2,4,1,5,7,3,6,10,8}, the numbers indicate the operating sequence of jobs on each n machines from 1 to n. In many application, string representation is used; e.g., {0,1,0,0,1,1,1}

3.3.1 Schema Theorem

A schema helps determining the similarities among chromosomes. The similar section of the chromosome is written neatly and the rest part is denoted with *. e.g., the sequences of genes on those two chromosomes are similar, which are to be {1, 4, 6, 3, 2, 5} and {5, 1, 6, 3, 2, 4}

In these two chromosomes, it is obviously seen that there is a similarity of genes at certain locations. The schema of can be represented as {*,*, 6, 3, 2, *}

As the number of schemas increases, the solution pool moves to uniformity; namely it converges. This means that the fitness of the chromosomes begins to stabilize, which helps the algorithm stop running.

Schema Theorem has some formulations. With the calculation of the formulas, it helps to provide information about how GA works and to calculate the effect of selection, crossover and mutation.

3.4 SELECTION

Selection method is used for two objectives; for determining the mates to reproduce and for determining the fitter chromosomes which will be maintained in the next generation.

This method has a magnificent effect on the results. If the selector picks only the best individual, then the population will quickly converge to that best value. The selector should also pick individuals that are not so good, but have hopefully good genetic material to avoid from early convergence.

Selection is done according to the fitness scores. By using fitness scores, fitter chromosomes are chosen to reproduce and weaker ones are eliminated and hence the population is differentiated and diversified. There are many selection methods available.

3.4.1 Roulette Wheel Selection

Individuals have $\frac{f(i)}{\sum_i f(i)}$ probability to be chosen whereas f(i) denotes the fitness of that certain chromosome and ∑f(i) denotes the sum of the fitness of each chromosome in the population. The proportion is compared with a randomly generated number and the chromosomes are selected, whose fitness proportion is close to the generated value.

3.4.2 Ranking Selection

The fitness of the chromosomes is calculated and the values are sorted in descending ordered. Then, the selection is done downward.

3.4.3 Elitist Selection

Few of the best individuals are directly inserted to the mating pool. Another selection method is used for the rest of the pool.

3.4.4 Tournament Selection

In this selection method, the best being solution is picked up among k number of selected individuals and it is inserted to the mating pool. The best results are attained when k equals 2.

3.4.5 Steady-State Selection

Different selection strategies can be followed for both mating and replacement. e.g., Fitness of parents can be taken into account during mating whereas replacement can be done randomly or the selection can be done according to fitness for both stages, etc.

3.4.6 Stochastic Universal Selection

All individuals have the same probability to be selected.

3.5 CROSSOVER

In crossover, two individuals, called parents combine to produce two more individuals which are called the children. One chromosome exchanges its subpart with the latter, which is a mimicking of a biological recombination. But there are also asexual and single-child type crossovers. Crossover enables to move to promising regions of the solution space.

The main objective of crossover is to transfer the good characteristics of previous generation to the subsequent generation. Therefore, it matches generally good parents to produce better solutions.

3.5.1 Single Point Crossover

Parent chromosomes are broken from the same point and the alleles after that point are swapped between parents, e.g., let's say parent {a1,a2,a3,a4,a5,a6,a7} and {b1,b2,b3,b4,b5,b6,b7} chromosomes are broken after the third point. Then the produced chromosomes will be {a1,a2,a3,b4,b5,b6,b7} and {b1,b2,b3,a4,a5,a6,a7}

Goldberg [103] describes another single point crossover which performs well in flow shop sequencing problems. Both parents are broken randomly at a point.

P1	2 1 3 4 5 6 7	O1	2 1 4 3 6 7 5
	*		
P2	4 3 6 2 7 1 5	O2	4 3 2 1 5 6 7

As seen in the first offspring, the alleles 2 and 1 are erased from the second parent and the rest are directly replaced beyond the breaking point of the first parent without changing the sequence the second parent. The second offspring is reproduced in the same way.

2-point, 3- point and multi-point crossover have been developed from 1-point crossover.

3.5.2 Cycle Crossover (CX)

A single crossover point is selected. From starting at this point, elements from one parent is inherited to the offspring, as soon as the cycle is completed, the values are inherited from the other parent. Let's explain it on the example,

Parent 1	7 3 4 2 1 5 6
Crossover points	*
Parent 2	3 1 5 2 4 7 6
Offspring	7 3 4 2 1 5 6

Third point is selected as the crossover point. 4 is inherited from parent 1. We move on the second parent until we see 4, and inherit the across value to the offspring. This continues until a cycle is completed. The cycle is completed at location 3. After than the remaining loci is filled with the elements from the second parent.

3.5.3 Order Crossover (OX)

This is a 2-point crossover operator. The points are randomly selected. The offspring inherits the alleles between the selected points from one of the parents. The remaining locations are filled with the alleles from the alternate parent. The alleles are inherited from the beginning allele to the end if they don't appear in the offspring so far. Filling of offspring loci begins beyond the second crossover point.

Parent 1	7 3 4 2 1 5 6
Crossover points	* *
Parent 2	3 1 5 2 4 7 6
Offspring	1 6 5 2 4 7 3

3.5.4 Partially Mapped Crossover (PMX)

Two points are selected randomly. The elements among the points are inherited from one of the parents. Other unfilled loci are inherited from the alternate parent.

Parent 1	7 3 4 2 1 5 6
Crossover points	* *
Parent 2	3 1 5 2 4 7 6
Offspring	3 4 4 2 1 7 6

At this moment, if we met in the alternate parent the previously inherited elements, they are replaced with the other elements from the previous parent across them.

In the example, 4 duplicates; so the offspring is mutated and the second locus of the child chromosome is changed with 5; since 5 hasn't appeared in the sequence. So the new sequence is

Mutated offspring	3 5 4 2 1 7 6

3.5.5 Position-Based Crossover (PBX)

Some points are selected randomly and the alleles at these points are inherited from one of the parents to the offspring. The remaining gene loci are inherited from the latter parent. To avoid from the duplication of alleles, the gene values aligned with the crossover points should be replaced with the alleles across the points.

Parent 1	7 3 4 2 1 5 6
Crossover points	* * *
Parent 2	3 1 5 2 4 7 6
Offspring	7 1 4 2 5 5 6

As seen in the representation above, the allele 5 duplicates and 3, is not available in the offspring sequence. Let's do a mutation on the reproduced chromosome.

Mutated offspring	7 1 4 2 3 5 6

3.6 MUTATION

Mutation changes the values of genes at some locations in the chromosome. It helps randomizing the search with a very low probability and finds solutions that cannot be encountered by crossover. It enables movement in the search space and restores lost information to the population.

Mutation has less impact near the beginning of a run, and more near the end while the crossover is more effective at the beginning and less at the end.

3.7 INVERSION

It is rarely used. The order of a part of a chromosome is reversed, e.g., {1,2,**3,4,5,6,7,**8,9} is changed into {1,2,**7,6,5,4,3**,8,9}.

3.8 REPLACEMENT

During replacement, generally the aim is to eliminate the worst or the most similar individuals. Because first we want to get children with good properties and second when the population includes similar individuals, the likelihood of finding new solutions decreases. It is important to consider carefully the replacement strategy. There are some approaches for deletion.

3.9 TERMINATION CRITERIA

The algorithm is terminated whenever a certain number of iterations are reached or the population converges. Each iteration is called a generation. Typically a GA can be iterated from 50 to 500 or more generations.

CHAPTER 4

PARTICLE SWARM OPTIMIZATION ALGORITHM

4.1 AN OVERVIEW

Particle Swarm Optimization (PSO) is a population based metaheuristic which was developed by Eberhart and Kennedy, in 1995 and introduced as an alternative to Genetic Algorithms (GA). It was inspired by the social behavior of flocking organisms such as bird swarms and fish shoals, which benefit from their previous experience or from the experience of the previous generation while they are searching for food and mate.

PSO is a rather successful method for the continuous optimization problems; however it is very difficult to adapt it for the discrete case. Researches were already done for the adaptation of the algorithm for the discrete case. These approaches can solve the combinatorial problems to some extent.

The PSO paradigm resembles to GA at some points. The initialization of the algorithm is done with a population of random solutions. It searches the optimal value by updating generations. Solutions are not generated by the crossover and mutation operators as in GAs. Instead, in PSO new generations are formed by means of velocity updates. The potential solutions, called particles, fly through the multi-dimensional search space, and follow the current optimum particles. Execution of the algorithm is terminated as soon as the maximum number of iteration is or maximum CPU time exceeded.

There is no replacement in PSO, all particles are kept in the population during the whole run. PSO does not incorporate the survival of the fittest, whereas all other evolutionary algorithms do.

Each particle has a velocity. Particles are carried to new positions with this velocity. The fitness values of particles are evaluated according to their positions at each iteration. The velocity, position and fitness of a particle are stored in a short term memory. The best position and fitness values of the particle are stored in the long term memory; which is named by Kennedy & Eberhart as autobiographical memory. The best experience stored in this memory is named as personal best; *pbest*. The particle with the best fitness in the neighborhood is named as the local best; *lbest* and the best particle in the whole swarm is called as the global best; *gbest*.

PSO has two versions; the local version and the global version. According to the local neighborhood, each particle moves towards its best previous position, *pbest* and towards the best particle in its restricted neighborhood, *lbest*; rather than moving towards the best of the entire group, *gbest*. In the global neighborhood, each particle moves towards its best previous position, *pbest* and towards the best particle in the whole swarm, *gbest*.

There is a communication between particles, each particle shares its information with others. A particle exchanges its information with the particles in the neighborhood or a predetermined set of particles in the search space. Therefore; after some number of iterations the swarm loses its diversity and solutions progress to uniformity. If the convergence occurs too early, the probability of being stuck in local minima increases.

In recent years, PSO has been successfully applied in many areas. It solves a variety of optimization problems in a faster and cheaper way than the evolutionary algorithms in the early iterations, but its computational efficiency may reduce as the number of generations increases. In addition to this, PSO has few parameters to adjust. It works well for different kind of problems when the algorithm is slightly modified.

4.2 THE PSEUDOCODE OF THE ALGORITHM

```
Initialize parameters
Initialize population
Find permutation
Evaluate
Do
{
        Find the personal best
        Find the global best
        Update velocity
        Update position
        Find permutation
        Evaluate
        Apply local search (optional)
}
While (Termination)
```

4.3 NOTATION

X_i^t : ith particle in the swarm at iteration t; $X_i^t = \left[x_{i1}^t, x_{i2}^t, .., x_{in}^t\right]$

x_{ij}^t : Position value of the ith particle with respect to the jth dimension ($j = 1,2,...,n$).

pop^t : Set of ρ particles in the swarm at iteration t, i.e., $pop^t = \left[X_1^t, X_2^t, ..., X_\rho^t\right]$

π_i^t : Permutation of jobs implied by the particle X_i^t ; $\pi_i^t = \left[\pi_{i1}^t, \pi_{i2}^t, .., \pi_{in}^t\right]$

π_{ij}^t : Assignment of job *j* of the particle *i* in the permutation at iteration *t*.

V_i^t : Velocity of particle i at iteration t; $V_i^t = \left[v_{i1}^t, v_{i2}^t, ..., v_{in}^t\right]$

v_{ij}^t : Velocity of particle i at iteration t with respect to the jth dimension.

w^t : Inertia weight; a parameter to control the impact of the previous velocities on the current velocity.

P_i^t : The best position of the particle i with the best fitness until iteration t, personal best; $P_i^t = \left[p_{i1}^t, p_{i2}^t, ..., p_{in}^t\right]$

p_{ij}^t : Position value of the ith personal best with respect to the jth dimension ($j = 1,2,...,n$).

G^t : The best position of the globally best particle achieved so far, global best; $G^t = \left[g_1^t, g_2^t, ..., g_n^t\right]$

g_j^t : Position value of the global best with respect to the jth dimension ($j = 1,2,...,n$)

4.4 ORIGINAL PSO ALGORITHM

Each particle updates its velocity and position according to its previous velocity and the distances of its current position from its own best experience and the group's best experience according to the equation 4.1.a given below:

$$v_{ij}^{k} = v_{ij}^{k-1} + c_1 r_1 (pb_{ij}^{k-1} - x_{ij}^{k-1}) + c_2 r_2 (gb_{j}^{k-1} - x_{ij}^{k-1}) \qquad (4.1.a)$$

$$x_{ij}^{k} = x_{ij}^{k-1} + v_{ij}^{k} \qquad (4.1.b)$$

Here, c_1 and c_2 are cognitive and social components respectively. These terms pull each particle to *pbest* and *gbest* locations. They are both set to 2 for almost all applications; which is obtained from early experience. High or low values of these terms may hinder particles to reach the target.

r_1, r_2 are random numbers uniformly distributed in the interval [0,1]. Particles fly to new position with this velocity and their new position is calculated by the equation 4.1.b.

4.5 SOLUTION REPRESENTATION

Solution representation is a very important issue in PSO algorithm. The representation changes depending on the type of the problem. For the PFSP, we present *n* number of dimensions for *n* number of jobs ($j = 1,..,n$). Each dimension in the sequence corresponds to a certain job. In addition, the particle $X_i^k = [x_{i1}^k ,.., x_{in}^k]$ corresponds to the position values for *n* number of jobs in the PFSP problem. The position values of particles are in fact continuous. To discretize the positions, we use the SPV rule and by this way, determine processing sequence of jobs in the flow shop.

Table 4.1 illustrates the solution representation of particle X_i^t for the PSO algorithm for the PFSP together with its corresponding velocity and permutation. According to the proposed SPV rule, the smallest position value is $x_{i5}^t = -1.20$, so the dimension j=5 is assigned to be the first job $\pi_{i1}^t = 5$ in the permutation π_i^t; the second smallest position value is $x_{i2}^{t} = -0.99$, so the dimension j=2 is assigned to be the second job $\pi_{i2}^{t} = 2$ in the permutation π_{i}^{t}, and so on. In other words, dimensions are sorted according to the SPV rule.

This representation is unique in terms of finding new solutions since positions of each particle are updated at each iteration *k* in the PSO algorithm, thus resulting in different sequences at each iteration *k*.

Table 4.1 Solution Representation of a Particle

j	1	**2**	3	4	**5**	6
x_{ij}^{k}	1.80	**-0.99**	3.01	-0.72	**-1.20**	2.15
v_{ij}^{k}	3.89	2.94	3.08	-0.87	-0.20	3.16
s_{ij}^{k}	**5**	**2**	4	1	6	3

4.6 INITIAL POPULATION

In PSO, the population is initialized randomly and the initial continuous position values are generated randomly using the following formula: $x_{ij}^{0} = x_{\min} + (x_{\max} - x_{\min}) * U(0,1)$ where $x_{\min} = -1.0, x_{\max} = 1.0$. Initial continuous velocities are generated by similar formula as follows: $v_{ij}^{0} = v_{\min} + (v_{\max} - v_{\min}) * U(0,1)$ where $v_{\min} = -1.0, v_{\max} = 1.0$. $U(0,1)$ is a uniform random number between 0 and 1. Continuous velocity values are restricted to some range, namely $v_{ij}^{k} = [v_{\min}, v_{\max}] = [-4.0, 4.0]$ where $v_{\min} = -v_{\max}$.

As the objective of our problem is to minimize the number of tardy jobs in permutation flow shops, the fitness function includes the total number of tardy jobs for the particle *i*. That is,

$$f_{i}^{t}(\pi_{i}^{t}) = \sum_{j=1}^{n} U(\pi_{n}^{t}, m) \tag{4.2}$$

where π_{i}^{t} is the corresponding permutation of particle X_{i}^{t} and U is the number of tardy jobs from job 1 to n. The complete computational procedure of the PSO algorithm for the PFSP can be summarized as follows:

Step 1: Initialization

- Set k=0, m=twice the number of dimensions.
- Generate m particles randomly as explained before, $\{X_i^0, i = 1,..,m\}$ where $X_i^0 = [x_{i1}^0,..,x_{in}^0]$.
- Generate initial velocities of particles randomly $\{V_i^0, i = 1,..,m\}$ where $V_i^0 = [v_{i1}^0,..,v_{in}^0]$
- Apply the SPV rule to find the sequence $S_i^0 = [s_{i1}^0,...,s_{in}^0]$ of particle X_i^0 for $i = 1,..,m$.
- Evaluate each particle i in the swarm using the objective function f_i^0 for $i = 1,..,m$.
- For each particle i in the swarm, set $PB_i^0 = X_i^0$, where $PB_i^0 = [pb_{i1}^0 = x_{i1}^0,.., pb_{in}^0 = x_{in}^0]$ along with its best fitness value, $f_i^{pb} = f_i^0$ for $i = 1,..,m$.
- Find the best fitness value $f_l^0 = \min\{f_i^0\}$ for $i = 1,..,m$ with its corresponding position X_l^0 .
- Set global best to $GB^0 = X_l^0$ where $GB^0 = [gb_1 = x_{l,1},.., gb_n = x_{l,n}]$ with its fitness value $f^{gb} = f_l^0$

Step 2: Update iteration counter

- $k = k + 1$

Step3: Update inertia weight

- $w^k = w^{k-1} * \alpha$ where α is decrement factor.

Step 4: Update velocity

$$v_{ij}^k = w^{k-1} v_{ij}^{k-1} + c_1 r_1 \left(pb_{ij}^{k-1} - x_{ij}^{k-1}\right) + c_2 r_2 \left(gb_j^{k-1} - x_{ij}^{k-1}\right)$$

Step 5: Update position

- $x_{ij}^k = x_{ij}^{k-1} + v_{ij}^k$

Step 6: Find Sequence

- Apply the SPV rule to find the sequence $S_i^k = [s_{i1}^k,...,s_{in}^k]$ for $i = 1,..,m$.

Step 7: Update personal best

- Each particle is evaluated by using its sequence to see if personal best will improve. That is, if $f_i^k < f_i^{pb}$ for $i = 1,..,m$, then personal best is updated as $PB_i^k = X_i^k$ and $f_i^{pb} = f_i^k$ for $i = 1,..,m$.

Step 8: Update global best

- Find the minimum value of personal best.

$$f_l^k = min\{f_i^{pb}\} for \quad i = 1,..,m, l \in \{i; i = 1,..,m\}$$

- If $f_l^k < f^{gb}$, then the global best is updated as $GB^k = X_l^k$ and $f^{gb} = f_l^k$

***Step 9:* Stopping criterion**

- If the number of iteration exceeds the maximum number of iteration, or maximum CPU time, then stop, otherwise go to step 2.

4.7 MAXIMUM VELOCITY

The velocities of particles are constrained to a maximum velocity, V_{max}. If a velocity on a dimension of a particle exceeds V_{max}, then it is limited to V_{max}. V_{max} controls the exploration and exploitation ability of a particle. It helps to search the regions between the current position and the target position.

Fine-tuning V_{max} is so important that a large value of V_{max} facilitates global exploration, while a smaller V_{max} encourages local exploitation. If V_{max} is set too high or too small, the particles can't explore the search space sufficiently and they could stuck at local optima.

4.8 INERTIA WEIGHT

Eberhart & Shi introduced a new concept to PSO in 1998; the inertia weight, *w* which highly increased the performance of PSO in a number of applications. Before, PSO was not searching neighbors sufficiently. Dynamically adjusting the velocity by means of *w* provided the local search.

The inertia weight controls the effect of previous velocity of the particle to its current velocity as seen in the formula is as;

$$v_{ij}^{k} = wv_{ij}^{k-1} + c_1 r_1 (pb_{ij}^{k-1} - x_{ij}^{k-1}) + c_2 r_2 (gb_{j}^{k-1} - x_{ij}^{k-1}) \qquad (4.3.a)$$

$$x_{ij}^{k} = x_{ij}^{k-1} + v_{ij}^{k} \qquad (4.3.b)$$

Setting high values to *w* at the beginning and small values at the end of the search is found to be better. It is generally reduced linearly from 1.2 to 0.4 during a run, but these values may change from application to application.

When suitably set, the inertia weight helps to balance the local and global exploration, thus the optimal value can be obtained in a few iterations. High values encourage global exploration, while low values facilitate local exploitation.

4.9 CONSTRICTION FACTOR

Maurice Clerc has introduced in 1999 the *constriction factor, K*, which highly increases the performance of the algorithm by constraining and controlling the velocity of the particles. Shi and Eberhart found that when the constriction factor is used with *Vmax* constraint, the performance PSO improves.

The velocity formula using *K* is stated in equation 4.3.a. Clerc used *K* to be 0.729 in calculations.

$$v_{id}^{k} = \kappa\left[v_{id}^{k-1} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id})\right] \qquad (4.4.a)$$

$$\kappa = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}, \text{ where } \varphi = c_1 + c_2, \quad \varphi > 4 \qquad (4.4.b)$$

Both the constriction factor, *K*, and the inertia weight, *w*, are used to control the velocities of particles. Therefore, they both prevent the particles from explosion. Since they have some computational differences, they correspond to different PSO variants.

4.10 PSO MODELS

There are four PSO models defined by Kennedy. The complete velocity update formula is named as the *Full Model.* If the cognition component, c_1 is omitted, it is defined as the *Social-Only Model* and if social component, c_2 is omitted, then the model is called the *Cognition-Only Model.* And the fourth model is the *Selfless Model* which is a kind of *Social-Only Model.* In this model, it selects its global best only from its neighbors.

CHAPTER 5

EXPERIMENTAL DESIGN

5.1 INTRODUCTION

The objective of our study is to determine a sequence of jobs to are processed through m machines in a permutation flow shop. The sequence should be arranged in order to minimize the number of tardy jobs. The problem is denoted as $Fm|\ |\sum U_i$ in scheduling.

Since the problem is known to be NP-hard, it is computationally expensive to solve the problem with exact algorithms such as branch & bound or dynamic programming algorithms. It is better to develop metaheuristic algorithms.

The selection of the search algorithm has an important impact on quality of results. We have chosen the well-known genetic algorithm and the particle swarm optimization algorithm.

Genetic algorithms are mostly heard with scheduling problems, but we cannot say the same thing for particle swarm optimization. It has been used for optimizing continuous functions in the past, but recently it is also used for the combinatorial scheduling problems.

In this study, we have compared the computational performances of these two algorithms. Two metrics are measured as the performance criteria: the execution time (cpu) and the number of tardy jobs. If the number of tardy jobs and the cpu time of one algorithm is less than the other, this shows the superiority of that algorithm against the latter one.

The algorithms were coded in Borland C++ and runs were done on a Centrino 1.5 GHz computer with 256 MB memory. 10 replications of 20 instances were run for 20x15, 20x20, 30x15, 30x20, 40x15, 40x20, 50x15 and 50x20 problem sets, e.g., in 20x15 problem, 20 corresponds to the number of jobs and 15 corresponds to the number of machines. We obtained the data for the due dates from the benchmark suite of (Demirkol et al., 1998).

(Demirkol et al., 1998) has provided a set of randomly generated test problems which includes four different due date configurations. The problem sizes range from 20 to 50 jobs with 15 to 20 machines. Results are presented for 160 problem instances.

Experimentally, the parameters are set to some special values. In both algorithms the population size is taken to be equal to the number of jobs. Based on the experiments, both algorithms converged after 2500 generations so, we terminated the run after 2500 generations.

The developed genetic algorithm is the traditional one; it includes the evolutionary stochastic operators: the selection, crossover and the mutation. The crossover probability is tuned to 70 % and the mutation probability to 5 %. Selection is done randomly. Initially, a population of size λ is constructed probabilistically. At each generation, two parents with a random selection are determined to produce an offspring through order-based crossover.

This process is maintained as a loop until μ offspring (pop size*crossover probability) are produced. Some of the offspring are mutated with a certain probability. The size of the population increased to $(\lambda + \mu)$ at the end of each generation. In order to maintain the population size of the next generation the same, μ individuals are replaced among $(\lambda + \mu)$ individuals. The tournament selection of size 2 is used to establish the next population. This procedure is repeated until the stopping criterion is achieved.

In the proposed particle swarm optimization algorithm, we generated ρ particles randomly. Initial velocities are generated according to the following formula:

$v_{ij}^0 = v_{\min} + (v_{\max} - v_{\min}) * r_2$, where $v_{\min} = -1.0, v_{\max} = 1.0,$ and r2 is a uniform random number between 0 and 1. The velocities and the position values are updated at each iteration. The social and cognitive parameters are taken as c1=c2= 2 consistent with the literature. Initial inertia weight is set to w0= 1.2 and it decreased linearly until the weight becomes 0.4.

The following formula is used to construct the initial continuous position values of the particle uniformly: $x_{ij}^0 = x_{\min} + (x_{\max} - x_{\min}) * r_1$, where $x_{\min} = -1.0, x_{\max} = 1.0,$ and r1 is a uniform random number between 0 and 1. The SPV rule is used to convert a position vector to a job permutation. The obtained personal best and the global best values are recorded in the memory.

The particles are mutated with a probability of 5 %. When the algorithm converges to an optimal value the algorithm stops.

5.2 RESULTS

We compared the cpu time and the fitness values that we collected from the output files of the algorithms. The statistics of these two performance metrics such as the average, standard deviation, minimum and the maximum values are calculated in MS Excel file. The fitness statistics are given in Table 5.1.a and Table 5.1.b. As seen from the tables, the average fitness values of PSO algorithm are less than those of GA.

Table 5.1.a GA Fitness Statistics

GA FITNESS				
J x M	**AVERAGE**	**STD. DEV**	**MAX**	**MIN**
20X15	17,21	2,53	20	10
20X20	18,37	2,13	20	13
30X15	23,26	4,04	29	14
30X20	25,44	4,20	30	16
40X15	27,27	6,38	36	15
40X20	31,17	5,40	39	20
50X15	31,90	7,76	44	17
50X20	35,99	7,78	47	21

Table 5.1.b PSO Fitness Statistics

PSO FITNESS				
J x M	**AVERAGE**	**STD. DEV**	**MAX**	**MIN**
20X15	16,88	2,71	20	10
20X20	18,24	2,34	20	13
30X15	22,80	4,16	29	14
30X20	25,03	4,29	30	16
40X15	26,87	6,17	36	15
40X20	30,85	5,27	39	20
50X15	31,56	7,32	43	18
50X20	35,58	7,38	46	21

The cpu statistics are given in Table 5.2.a and Table 5.2.b. PSO seems to be better for the data sets 20x15, 20x20 and 30x15 problems. GA is faster to solve the rest of the data set which are more complex.

Table 5.2.a GA CPU Statistics

GA CPU				
J x M	**AVERAGE**	**STD. DEV**	**MAX**	**MIN**
20X15	1,46	0,60	5,54	1,02
20X20	1,71	0,59	5,07	1,11
30X15	2,51	0,95	9,15	1,66
30X20	1,93	0,07	2,39	1,8
40X15	2,62	0,09	2,8	2,45
40X20	2,96	0,05	3,11	2,8
50X15	3,69	0,08	4,26	3,45
50X20	4,23	0,08	4,5	3,9

Table 5.2.b PSO CPU Statistics

PSO CPU				
J x M	**AVERAGE**	**STD. DEV**	**MAX**	**MIN**
20X15	0,97	0,05	1,08	0,90
20X20	1,07	0,05	1,17	1,00
30X15	1,86	0,08	2,00	1,69
30X20	2,10	0,23	4,37	1,94
40X15	3,22	0,03	3,33	3,11
40X20	3,64	0,03	3,72	3,58
50X15	5,14	0,04	5,27	5,01
50X20	5,81	0,05	6,06	5,68

We also computed the relative percent deviation for fitness values to see which algorithm is better to solve the problem. The relative percent deviation (Δ) is calculated using the formula:

$$\Delta = \frac{\mu_{GA} - \mu_{PSO}}{\mu_{PSO}} \times 100\% \qquad (5.1)$$

Here, $\mu_{a\lg orithm}$ shows the number of tardy jobs obtained by the related algorithm. Positive values of Δ show that PSO performs better than GA. The Δ values are given in Table 5.3.a.

Table 5.3.a Relative Percent Deviation for Fitness Values

JOBxM/C	**RELATIVE PERCENT DEVIATION**									
	REPLICATION NUMBER									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
20X15	0,00	0,00	0,00	0,00	0,00	5,88	0,00	0,00	0,00	0,00
	5,88	6,25	6,25	6,25	12,50	5,88	12,50	6,25	0,00	0,00
	12,50	0,00	6,25	6,25	6,25	6,25	6,25	6,25	0,00	0,00
	0,00	5,88	5,88	0,00	5,88	5,88	5,88	5,88	5,88	0,00
	0,00	0,00	0,00	5,88	0,00	0,00	0,00	0,00	0,00	5,88
	-8,33	9,09	8,33	0,00	9,09	8,33	18,18	18,18	0,00	-7,69
	0,00	0,00	-9,09	9,09	8,33	0,00	9,09	9,09	9,09	10,00
	7,14	7,14	0,00	-6,67	7,14	7,69	7,14	7,14	0,00	7,14
	8,33	16,67	0,00	-7,69	8,33	8,33	0,00	8,33	8,33	16,67
	0,00	0,00	0,00	6,25	0,00	6,25	0,00	6,25	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	5,56	0,00	0,00	0,00	0,00	0,00	5,56
	0,00	0,00	0,00	5,88	0,00	0,00	5,88	0,00	0,00	0,00
	0,00	0,00	5,56	0,00	5,56	0,00	0,00	5,56	5,56	5,56
	0,00	5,56	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Table 5.3.a Relative Percent Deviation for Fitness Values (continued)

JOBxM/C	**RELATIVE PERCENT DEVIATION**									
	REPLICATION NUMBER									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
20X20	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	7,14	7,14	0,00	0,00	0,00	6,67	7,14	14,29	7,14
	7,14	14,29	7,14	0,00	0,00	7,14	7,14	0,00	6,67	0,00
	-6,25	6,67	6,67	0,00	0,00	0,00	0,00	0,00	0,00	6,67
	-6,67	14,29	7,14	0,00	-6,67	7,14	-6,67	7,14	0,00	0,00
	7,14	0,00	-6,67	15,38	7,14	7,69	7,69	7,14	0,00	7,69
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
30X15	0,00	10,00	5,00	4,76	4,76	-4,55	0,00	0,00	4,76	10,00
	9,09	0,00	0,00	0,00	0,00	4,55	4,55	-4,35	14,29	0,00
	9,52	14,29	4,55	0,00	9,52	0,00	4,76	9,52	9,09	9,52
	0,00	4,35	0,00	9,09	9,09	0,00	0,00	4,35	4,55	4,55
	4,55	0,00	4,55	0,00	9,09	4,55	0,00	4,55	0,00	0,00
	0,00	-6,67	0,00	0,00	0,00	0,00	-6,25	-6,67	-6,67	7,14
	5,88	6,25	0,00	0,00	12,50	6,25	6,25	0,00	13,33	12,50
	-5,00	5,00	5,00	15,79	5,00	4,76	0,00	5,00	0,00	0,00
	0,00	5,56	0,00	5,56	0,00	0,00	0,00	0,00	5,56	5,56
	12,50	6,25	0,00	12,50	0,00	-5,56	0,00	5,88	-5,56	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	3,57	0,00	0,00	0,00	0,00	0,00	0,00	3,57	0,00
	0,00	3,57	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	4,35	8,70	4,35	4,35	0,00	8,70	0,00	0,00	4,35	0,00
	-4,35	4,35	4,35	0,00	0,00	0,00	4,55	4,35	9,09	4,35
	0,00	0,00	4,17	0,00	-4,17	0,00	4,35	4,35	0,00	4,35
	0,00	-4,00	-4,00	4,17	0,00	0,00	-4,00	0,00	0,00	0,00
	0,00	4,35	4,35	0,00	8,70	0,00	8,70	4,35	0,00	13,04

Table 5.3.a Relative Percent Deviation for Fitness Values (continued)

JOBxM/C	RELATIVE PERCENT DEVIATION									
	REPLICATION NUMBER									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
30X20	0,00	0,00	0,00	0,00	4,17	4,00	0,00	4,17	-3,85	0,00
	4,00	0,00	8,00	0,00	4,00	4,00	4,00	4,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	3,85	0,00	0,00	8,00
	3,85	0,00	0,00	0,00	0,00	7,69	0,00	0,00	3,85	0,00
	4,17	8,33	4,17	8,33	0,00	4,17	4,17	4,17	0,00	0,00
	5,00	5,26	-4,76	0,00	5,00	-4,76	0,00	5,26	5,26	0,00
	0,00	5,88	5,88	5,88	0,00	5,88	-5,56	5,88	0,00	5,56
	5,26	0,00	16,67	0,00	0,00	5,26	11,11	-5,00	5,26	-5,00
	0,00	11,11	5,56	5,26	11,11	10,53	5,56	-5,00	0,00	5,56
	-5,88	0,00	-5,56	0,00	12,50	-5,88	-5,88	5,88	12,50	-5,56
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	7,41	7,41	7,41	3,70	3,70	0,00	7,41	7,41	3,70	7,41
	4,00	0,00	4,00	3,85	0,00	4,00	0,00	0,00	0,00	0,00
	8,00	0,00	3,85	4,00	8,00	0,00	4,00	4,00	3,85	4,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	3,70	0,00	0,00	0,00	3,70	3,70	3,70	0,00	3,70
40X15	4,17	0,00	8,33	0,00	0,00	-4,00	12,50	8,33	12,50	4,17
	0,00	0,00	11,54	-3,85	3,85	-3,70	7,69	8,00	7,69	8,00
	8,33	-3,85	8,00	3,85	8,00	8,00	3,85	4,00	8,00	3,85
	4,00	3,85	8,33	8,33	4,00	4,00	8,70	3,85	8,00	8,00
	8,00	0,00	12,00	0,00	0,00	4,00	3,85	0,00	4,00	4,00
	11,11	0,00	5,00	0,00	-5,26	-4,76	-4,76	-4,76	0,00	-5,00
	0,00	-10,00	-15,00	-10,00	5,26	0,00	-5,26	-5,00	-5,26	-5,00
	-6,25	-5,88	-5,88	6,25	6,25	12,50	20,00	-5,56	6,67	0,00
	-5,26	-5,26	-14,29	0,00	-15,00	-10,00	0,00	-5,56	-5,00	5,88
	-5,56	0,00	-10,53	11,76	0,00	0,00	0,00	-5,26	-10,00	12,50
	-2,78	2,86	0,00	2,86	0,00	-2,78	0,00	2,86	-2,78	0,00
	2,86	2,86	2,86	2,86	0,00	2,86	2,86	2,86	0,00	0,00
	2,86	2,86	0,00	-2,78	0,00	-2,78	0,00	0,00	0,00	0,00
	2,86	2,86	2,86	0,00	2,86	2,86	2,86	-2,78	2,86	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	2,86	0,00	0,00
	6,90	3,45	3,45	3,57	3,45	3,57	3,57	3,45	0,00	-3,45
	0,00	3,45	0,00	6,90	3,57	6,90	0,00	7,14	3,45	0,00
	7,41	3,70	0,00	11,11	7,41	0,00	3,57	7,41	7,14	3,70
	0,00	3,57	0,00	0,00	3,70	0,00	-3,45	0,00	0,00	-3,57
	-3,57	-3,45	7,14	0,00	-3,45	3,57	-6,67	7,41	0,00	-6,67

Table 5.3.a Relative Percent Deviation for Fitness Values (continued)

JOBxM/C	RELATIVE PERCENT DEVIATION									
	REPLICATION NUMBER									
	1	2	3	4	5	6	7	8	9	10
40X20	0,00	3,33	0,00	6,90	0,00	0,00	6,90	3,45	3,33	-3,23
	0,00	0,00	0,00	3,33	0,00	6,67	3,23	6,67	0,00	-3,23
	3,23	3,23	3,23	10,00	0,00	0,00	6,67	3,33	6,67	3,23
	3,23	0,00	3,33	6,45	-3,23	0,00	3,33	3,33	-3,23	0,00
	0,00	0,00	-6,45	3,33	3,45	6,67	0,00	-3,33	0,00	-3,33
	-4,17	0,00	-4,35	-4,17	-12,00	-4,17	0,00	0,00	-4,00	4,35
	0,00	0,00	4,17	0,00	-4,00	4,17	4,17	0,00	8,70	4,17
	0,00	-3,85	8,33	-3,85	8,00	0,00	-3,85	-3,70	0,00	0,00
	-4,35	0,00	-4,35	-4,35	4,55	0,00	0,00	4,76	4,55	9,09
	0,00	0,00	-9,09	-4,55	0,00	-4,76	-4,35	0,00	0,00	9,52
	0,00	0,00	0,00	2,63	0,00	0,00	0,00	0,00	2,63	0,00
	2,70	0,00	2,70	0,00	0,00	2,70	0,00	2,70	0,00	2,70
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	0,00	2,63	2,63	0,00	0,00	2,63	0,00	0,00	0,00	0,00
	6,67	-6,06	-6,06	0,00	0,00	0,00	0,00	0,00	3,33	0,00
	-3,23	0,00	3,23	0,00	0,00	0,00	6,67	3,23	3,23	3,23
	0,00	6,45	-3,13	0,00	-3,03	6,45	0,00	3,13	0,00	0,00
	0,00	6,45	3,13	6,67	0,00	3,23	6,67	6,45	6,45	3,23
	6,25	0,00	3,13	6,25	3,03	0,00	0,00	3,03	0,00	6,25
50X15	-6,67	3,33	7,14	-3,23	0,00	3,33	3,45	6,90	-6,67	0,00
	3,33	3,33	3,33	6,90	0,00	3,45	3,33	3,33	3,33	7,14
	-3,23	0,00	-3,33	-6,45	3,57	-3,23	3,45	0,00	-3,33	7,14
	-6,67	3,33	-3,45	11,11	3,33	7,14	11,11	0,00	3,33	0,00
	7,14	-6,67	3,45	7,14	-3,33	0,00	-6,67	0,00	-6,67	3,57
	-12,00	0,00	4,55	0,00	-11,54	8,70	0,00	-11,54	-4,17	-7,69
	0,00	5,00	-4,76	-4,55	0,00	5,00	-4,55	4,76	0,00	-8,70
	-10,53	-5,26	11,11	-15,00	-10,00	5,26	5,26	11,11	-19,05	-10,00
	-16,00	0,00	-4,35	4,55	-20,00	0,00	-8,70	-8,33	-8,70	-4,35
	4,55	4,55	4,76	4,55	9,52	-8,70	0,00	4,76	4,55	4,76
	0,00	2,38	2,44	0,00	0,00	-2,38	7,32	0,00	2,38	2,44
	0,00	2,38	2,38	-2,33	0,00	0,00	2,38	0,00	0,00	-2,33
	0,00	7,32	2,38	0,00	0,00	4,76	4,88	4,88	4,76	4,88
	2,38	-2,33	4,88	-2,33	2,44	2,38	4,88	4,88	2,38	2,38
	-2,33	0,00	0,00	0,00	0,00	4,88	-2,38	2,38	4,76	4,76
	0,00	3,03	0,00	2,94	12,90	6,25	2,94	0,00	3,03	-3,03
	-2,94	3,03	6,25	-2,94	9,38	0,00	6,06	6,25	3,03	-2,86
	0,00	6,45	0,00	0,00	0,00	6,06	6,25	6,45	-5,88	6,25
	-3,03	3,03	3,23	6,45	0,00	3,03	3,03	3,03	3,03	0,00
	0,00	6,25	3,03	0,00	9,68	0,00	3,13	0,00	6,06	6,06

Table 5.3.a Relative Percent Deviation for Fitness Values (continued)

JOBxM/C	**RELATIVE PERCENT DEVIATION**									
	REPLICATION NUMBER									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
50X20	5,88	2,94	8,82	5,88	2,78	0,00	9,09	2,94	9,09	0,00
	-5,71	2,86	2,94	-2,86	5,71	5,88	0,00	-2,78	0,00	3,03
	9,09	3,03	0,00	-2,86	0,00	3,03	3,13	2,94	3,03	2,94
	5,88	8,82	2,86	2,94	0,00	-2,86	9,09	2,78	0,00	2,86
	0,00	0,00	0,00	6,06	5,88	-2,86	0,00	6,06	9,09	0,00
	0,00	0,00	3,45	11,11	-6,67	-3,23	0,00	0,00	6,67	-6,45
	-16,00	0,00	4,76	-4,17	-8,33	-4,35	14,29	8,70	-4,17	-15,38
	4,17	4,35	4,17	-7,69	-3,85	-3,85	-7,69	0,00	-3,85	0,00
	-4,17	4,00	4,00	-10,71	-7,41	4,00	0,00	-11,11	-4,00	4,17
	4,35	0,00	0,00	-4,00	0,00	-11,54	-4,35	-15,38	9,52	-8,33
	2,22	0,00	0,00	4,55	2,22	0,00	2,22	2,22	0,00	0,00
	0,00	0,00	0,00	0,00	2,17	2,17	0,00	-2,17	2,17	0,00
	2,17	2,17	0,00	4,44	2,17	0,00	0,00	2,17	2,17	0,00
	2,22	0,00	0,00	2,17	0,00	0,00	2,17	0,00	4,44	4,44
	4,44	2,17	0,00	2,17	2,17	4,44	2,17	0,00	2,17	2,22
	5,41	2,70	0,00	5,41	5,41	5,41	2,70	5,56	0,00	0,00
	0,00	-2,70	2,86	5,41	-2,70	2,70	5,56	5,56	-5,26	8,57
	0,00	-2,63	5,41	2,63	2,63	2,70	2,70	2,70	2,70	0,00
	0,00	-2,78	0,00	-5,56	2,86	-2,63	2,94	0,00	2,86	-5,41
	0,00	2,63	0,00	0,00	0,00	2,63	0,00	5,56	2,63	0,00

The computed relative percent deviation for cpu values of the algorithms are as seen in Table 5.3.b. In this table, $\mu_{a\lg orithm}$ is taken as the cpu time of the related algorithm. Positive deviation values show the superiority of PSO against GA.

Table 5.3.b Relative Percent Deviation for CPU Values

JOBxM/C	**RELATIVE PERCENT DEVIATION**									
	REPLICATION NUMBER									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
20X15	20,4	3,8	29,3	14,0	19,4	20,4	48,9	47,4	43,9	44,6
	58,1	34,3	22,6	41,9	51,0	63,4	47,8	10,7	66,7	24,0
	20,9	25,8	33,3	105,4	80,6	62,6	60,9	43,1	20,7	13,7
	21,5	8,7	35,2	14,6	15,2	5,8	44,6	63,0	285,4	293,5
	195,7	443,1	182,6	30,1	76,1	12,6	17,2	46,2	25,2	17,4
	16,7	26,1	82,5	84,6	47,8	36,9	27,5	13,6	18,5	29,1
	107,5	67,4	55,4	39,4	54,4	38,0	52,1	49,5	26,1	26,0
	82,4	46,7	57,7	35,9	99,0	101,1	271,7	332,4	312,0	185,9
	150,5	20,4	57,3	35,9	73,1	47,3	26,6	22,5	28,3	18,3
	5,8	31,5	32,0	68,5	21,4	37,4	35,5	5,9	30,1	43,1
	22,6	12,5	19,6	13,6	29,3	32,7	54,8	22,3	23,9	19,2
	18,3	98,9	70,9	58,7	11,7	19,4	18,3	41,3	62,5	32,6
	85,4	29,0	7,9	12,5	47,9	141,6	31,5	34,8	43,8	72,8
	17,5	16,5	43,0	29,3	121,6	37,6	49,5	20,4	79,6	20,4
	59,1	27,9	24,7	79,6	116,1	46,2	9,8	16,0	16,5	63,0
	25,2	27,2	6,8	35,5	4,9	16,8	5,9	31,2	18,9	19,8
	30,1	62,6	147,8	66,7	48,9	96,1	138,2	150,5	29,6	33,3
	44,6	7,7	33,7	7,8	40,9	192,2	23,7	28,2	20,4	24,3
	39,4	6,8	28,0	32,0	43,0	91,2	48,4	38,0	12,1	1,0
	27,2	11,7	13,0	10,9	13,6	10,9	0,0	12,1	10,8	1,0
20X20	18,3	8,0	7,8	-1,8	18,4	-1,7	0,9	21,4	0,0	10,7
	8,7	10,5	9,7	10,5	9,6	18,4	41,7	0,9	17,5	3,5
	129,4	397,1	342,0	341,6	154,4	35,7	37,3	69,0	48,0	61,8
	26,3	36,9	58,4	43,7	36,3	31,1	93,2	109,6	157,3	24,1
	106,8	13,3	49,0	22,3	0,9	21,4	23,3	112,7	25,2	21,4
	57,3	56,6	64,7	47,8	63,7	60,2	70,9	55,8	68,9	44,2
	50,0	26,5	114,3	50,0	35,3	50,0	48,0	25,4	43,1	73,7
	27,5	15,0	18,4	55,8	39,2	46,6	29,8	23,0	37,6	12,6
	7,1	12,7	10,7	18,6	8,8	22,5	33,3	19,6	3,5	24,8
	35,1	159,8	84,2	208,8	83,3	49,0	262,1	185,0	287,4	32,7
	68,0	1,8	20,6	16,3	29,8	113,6	62,3	101,9	133,9	136,7
	42,5	72,1	33,3	79,8	71,8	104,9	79,8	66,3	110,7	33,3
	149,5	88,2	14,2	41,2	15,0	61,2	27,7	44,0	41,5	63,7
	57,3	28,9	29,4	12,2	16,5	37,2	59,2	82,5	94,2	43,4
	40,4	60,2	48,2	112,6	46,1	43,9	128,4	22,1	46,1	49,5
	22,3	57,3	39,3	62,7	64,8	52,3	24,6	22,5	55,9	46,0
	53,4	56,1	84,5	82,4	41,2	82,4	36,8	77,7	65,5	92,1
	118,4	64,9	99,0	100,0	139,8	42,1	117,5	72,8	69,0	81,7
	41,6	35,7	43,0	50,5	84,8	92,1	76,5	37,5	17,9	68,3
	27,2	52,9	46,0	47,6	55,8	69,9	128,3	128,7	59,0	65,8

Table 5.3.b Relative Percent Deviation for CPU Values (continued)

JOBxM/C	RELATIVE PERCENT DEVIATION									
	REPLICATION NUMBER									
	1	2	3	4	5	6	7	8	9	10
30X15	33,3	0,5	33,0	109,8	14,1	5,7	-0,5	-10,4	-6,7	28,1
	20,0	2,6	0,5	-2,7	-13,5	-7,3	45,0	13,0	8,9	22,9
	20,1	7,8	20,2	22,4	43,6	8,8	-3,1	-6,7	-4,2	-10,4
	-7,3	-4,6	-5,7	-5,2	0,0	97,9	33,0	-2,6	-11,5	-3,6
	-4,7	-4,1	1,0	22,3	25,4	17,6	11,2	36,9	38,1	1,0
	8,2	18,5	-8,2	1,5	29,7	-8,2	-12,7	-5,6	-6,6	-3,6
	-6,2	26,5	51,6	53,9	4,6	24,2	-7,3	-8,2	-10,8	-9,8
	-3,1	-3,6	-5,1	9,3	34,2	26,8	8,4	16,0	17,5	15,5
	20,7	17,9	14,0	23,0	19,2	35,9	10,9	13,4	8,0	2,6
	259,0	357,7	369,2	275,3	-0,5	7,6	76,3	41,8	73,0	39,0
	32,6	25,6	114,6	67,9	104,9	34,8	34,1	25,8	61,2	-4,4
	9,3	6,0	24,0	20,5	18,5	119,6	67,4	118,6	45,9	0,0
	31,1	34,6	49,1	33,7	37,2	21,7	41,7	39,4	40,2	14,3
	61,9	51,9	35,7	24,0	43,7	52,5	59,0	37,9	45,3	30,6
	30,4	36,6	45,9	39,3	59,9	27,6	70,9	34,6	44,0	41,9
	50,8	41,8	40,7	39,8	21,3	39,2	49,5	50,3	40,0	54,3
	92,4	74,2	75,8	70,2	76,9	52,3	43,4	40,1	63,7	70,3
	62,0	69,2	64,1	72,5	63,2	51,6	91,2	72,4	89,1	63,2
	49,7	64,3	31,1	37,2	70,4	55,8	39,4	33,7	50,6	30,9
	17,1	39,2	4,4	-1,1	3,5	-3,3	-7,7	-1,7	-3,8	2,9
30X20	-10,6	-4,4	-4,9	-4,4	-4,8	-7,5	-6,6	-5,8	-3,4	-6,8
	-6,3	-7,7	-7,7	-7,6	-5,8	-6,3	-7,2	-6,3	-6,8	-7,2
	-12,6	-10,7	-9,5	-12,6	-7,2	-12,1	-8,6	-6,7	-5,9	-2,4
	-7,2	-2,9	-11,7	-8,2	-6,3	-6,8	-7,3	-12,6	-6,3	-7,7
	-13,4	-10,7	-1,5	-5,4	-5,3	-11,2	-5,3	-4,9	-4,9	-4,4
	0,5	-15,1	-20,2	-18,3	-29,1	-26,2	-20,3	-8,9	-2,0	-2,0
	-7,0	-3,0	-2,5	-0,5	-5,6	-6,7	0,0	-9,1	-5,8	-1,0
	-15,1	-51,3	-55,8	-5,7	-12,3	-6,2	-4,4	-5,8	-4,9	-0,5
	-10,1	-3,8	-9,2	-1,0	-6,2	0,0	-4,8	-5,8	-7,2	-9,6
	-3,5	-5,3	-7,2	-15,1	1,4	-8,7	-6,2	-0,5	-2,8	-10,0
	-1,5	-7,6	-5,8	-6,7	-7,6	-7,1	-13,2	-10,0	-7,2	-11,5
	-6,3	-7,2	-1,0	-13,0	-7,2	-6,8	-8,2	-4,1	-7,7	-5,9
	-5,9	-6,3	-7,6	-14,1	-6,8	-1,0	-6,8	-7,7	-10,2	-7,2
	-11,5	-1,4	-7,1	-8,5	-5,8	-7,2	-7,6	-2,0	-8,5	-9,3
	-7,2	-6,7	-6,7	-11,0	4,1	-12,0	-5,7	-12,4	-3,3	-9,5
	-6,3	-9,8	-7,6	-6,3	-7,7	-6,3	-5,9	-5,8	-10,7	-0,5
	-4,8	0,5	-10,1	-5,3	-5,3	-8,7	-5,8	-4,3	-4,8	-9,2
	-8,1	-6,8	-5,9	-12,1	-5,4	-2,9	-8,3	-5,3	-7,7	-5,4
	-5,8	-5,3	-6,3	-10,7	-5,8	-0,5	-9,3	-2,4	-9,9	-8,6
	-7,2	-5,3	-6,3	-5,8	-5,3	1,0	14,9	-6,1	11,5	-11,1

Table 5.3.b Relative Percent Deviation for CPU Values (continued)

JOBxM/C	RELATIVE PERCENT DEVIATION									
	REPLICATION NUMBER									
	1	2	3	4	5	6	7	8	9	10
40X15	-18,3	-20,7	-16,4	-22,0	-14,0	-18,3	-22,0	-14,3	-18,9	-21,5
	-14,5	-18,8	-23,4	-18,8	-17,9	-15,0	-19,4	-17,8	-20,2	-18,9
	-17,5	-18,0	-17,4	-16,1	-20,7	-18,7	-16,6	-16,2	-18,3	-18,6
	-20,4	-15,5	-21,7	-18,4	-15,5	-16,9	-18,1	-21,4	-18,6	-11,9
	-23,8	-15,8	-22,4	-13,8	-19,1	-22,5	-15,0	-21,7	-19,8	-18,7
	-17,7	-13,7	-17,4	-18,3	-17,3	-19,4	-21,5	-15,1	-20,9	-18,1
	-14,7	-21,1	-14,1	-18,6	-17,7	-22,2	-15,6	-22,3	-15,9	-19,2
	-21,0	-16,4	-22,7	-14,7	-17,0	-19,7	-11,9	-18,0	-23,9	-15,2
	-20,7	-17,8	-14,5	-17,5	-20,5	-17,9	-18,0	-15,2	-17,4	-21,1
	-13,5	-16,5	-21,4	-14,3	-18,0	-21,7	-16,6	-17,4	-13,7	-16,8
	-21,3	-15,8	-18,9	-17,9	-19,0	-18,7	-20,5	-17,4	-22,4	-14,6
	-22,6	-13,4	-22,7	-18,5	-19,6	-23,8	-16,5	-20,5	-23,8	-16,5
	-22,2	-12,9	-22,7	-15,9	-17,7	-18,8	-20,8	-18,8	-15,0	-17,8
	-22,0	-16,4	-22,0	-15,5	-19,4	-21,7	-16,3	-21,4	-18,3	-17,3
	-22,7	-19,5	-19,3	-15,6	-20,9	-17,2	-19,3	-18,8	-19,8	-23,0
	-17,7	-19,6	-18,6	-14,9	-18,0	-22,8	-15,5	-21,3	-15,3	-16,9
	-21,8	-15,0	-20,6	-18,3	-19,3	-17,4	-15,0	-20,1	-17,1	-18,6
	-19,1	-17,9	-21,3	-17,4	-18,4	-21,9	-15,9	-18,4	-22,9	-14,7
	-21,7	-14,9	-21,5	-16,6	-19,2	-23,8	-19,7	-20,4	-16,4	-18,9
	-17,9	-18,7	-23,0	-15,7	-21,7	-16,5	-18,9	-22,4	-16,1	-22,4
40X20	-14,7	-18,2	-18,0	-17,5	-20,3	-16,6	-21,8	-14,8	-18,4	-18,7
	-17,3	-19,2	-16,9	-20,5	-16,3	-21,7	-15,5	-18,8	-18,0	-18,4
	-17,8	-18,4	-18,3	-18,1	-19,2	-18,6	-18,4	-21,0	-16,2	-18,1
	-18,2	-17,5	-18,1	-19,1	-18,3	-18,6	-20,0	-17,3	-20,4	-17,0
	-18,1	-18,7	-18,2	-13,4	-18,8	-18,2	-17,0	-18,1	-18,8	-18,1
	-17,6	-18,2	-18,5	-18,3	-18,7	-18,6	-18,6	-17,9	-19,9	-19,0
	-18,8	-17,2	-19,2	-17,7	-18,2	-18,3	-18,9	-16,9	-18,9	-17,8
	-18,0	-19,3	-16,4	-20,8	-16,1	-18,1	-22,3	-15,0	-16,5	-18,8
	-16,7	-18,4	-19,2	-18,6	-17,9	-18,1	-18,0	-18,3	-18,0	-18,4
	-17,8	-18,9	-18,9	-18,0	-17,4	-17,4	-19,5	-19,1	-17,3	-18,8
	-18,9	-18,9	-18,7	-18,8	-19,4	-19,3	-18,2	-19,4	-19,3	-19,1
	-19,0	-19,0	-19,3	-17,5	-18,3	-19,8	-15,5	-21,0	-16,2	-18,0
	-18,1	-19,6	-19,4	-19,3	-19,4	-17,0	-19,8	-18,5	-19,3	-19,2
	-19,7	-21,1	-16,7	-23,5	-17,0	-20,0	-20,3	-20,3	-20,2	-20,2
	-20,4	-19,7	-21,9	-21,7	-17,7	-19,4	-20,0	-19,6	-20,2	-20,2
	-19,2	-19,9	-20,4	-17,7	-21,6	-15,5	-19,1	-18,6	-17,5	-18,7
	-20,2	-16,3	-17,8	-18,0	-18,2	-18,8	-19,6	-20,5	-20,5	-15,5
	-22,5	-17,9	-19,2	-15,9	-18,4	-18,1	-17,8	-19,4	-20,3	-16,4
	-18,2	-17,5	-19,4	-15,8	-18,0	-17,5	-18,2	-19,5	-20,7	-19,6
	-20,4	-16,2	-19,3	-18,0	-19,2	-17,5	-19,7	-18,6	-20,0	-18,8

Table 5.3.b Relative Percent Deviation for CPU Values (continued)

JOBxM/C	RELATIVE PERCENT DEVIATION									
	REPLICATION NUMBER									
	1	2	3	4	5	6	7	8	9	10
50X15	-26,9	-26,9	-27,6	-28,1	-28,2	-28,7	-27,8	-28,8	-16,8	-29,1
	-32,0	-27,0	-28,0	-27,6	-27,0	-28,2	-25,7	-29,3	-28,3	-27,8
	-29,1	-28,1	-27,9	-28,4	-29,2	-28,7	-27,6	-26,9	-28,9	-27,3
	-27,6	-27,7	-27,6	-28,1	-27,5	-28,8	-28,2	-26,2	-28,1	-28,8
	-27,0	-28,2	-28,4	-27,4	-29,0	-28,1	-27,6	-28,9	-28,1	-28,9
	-28,1	-28,7	-27,7	-28,6	-27,6	-27,8	-28,3	-29,0	-28,2	-28,4
	-28,3	-27,3	-27,6	-28,4	-27,9	-28,7	-28,9	-27,1	-28,5	-29,0
	-29,1	-27,1	-28,6	-28,8	-27,0	-23,4	-33,0	-28,9	-32,1	-31,0
	-28,1	-27,3	-27,7	-28,6	-29,2	-27,9	-28,6	-27,8	-29,5	-28,7
	-24,7	-28,1	-27,9	-27,7	-27,1	-28,4	-27,5	-28,1	-27,1	-25,7
	-31,3	-31,0	-28,6	-26,7	-27,4	-30,3	-27,4	-27,9	-28,5	-28,7
	-28,2	-27,9	-27,8	-28,3	-28,5	-27,6	-27,2	-28,8	-28,8	-28,7
	-29,7	-27,9	-28,3	-27,8	-28,0	-28,4	-27,9	-27,8	-28,0	-28,1
	-28,9	-28,7	-28,4	-27,8	-29,1	-28,3	-28,7	-27,8	-28,9	-27,8
	-26,9	-29,0	-29,6	-28,7	-28,6	-29,0	-28,1	-28,0	-27,9	-29,8
	-28,8	-29,4	-29,6	-30,0	-29,2	-26,9	-30,7	-26,0	-28,1	-27,9
	-24,8	-32,6	-30,5	-28,4	-28,3	-29,3	-29,1	-31,3	-27,7	-28,1
	-29,0	-29,5	-27,7	-31,7	-27,0	-29,2	-30,2	-29,3	-28,5	-28,3
	-28,6	-30,1	-25,0	-27,5	-28,2	-27,6	-28,9	-28,3	-28,6	-29,3
	-30,2	-26,5	-28,1	-28,8	-29,0	-28,5	-27,5	-28,3	-31,6	-33,0
50X20	-29,4	-28,9	-26,0	-22,4	-23,0	-26,9	-25,5	-26,6	-27,9	-25,6
	-29,3	-30,0	-25,7	-27,4	-29,5	-27,8	-25,6	-26,4	-27,0	-26,7
	-27,6	-26,6	-27,6	-26,6	-28,2	-27,3	-26,2	-25,9	-26,4	-27,1
	-26,4	-27,1	-27,2	-26,0	-25,5	-26,2	-26,6	-27,7	-28,3	-27,3
	-26,2	-28,0	-28,0	-27,3	-27,8	-27,2	-26,3	-26,5	-27,6	-28,3
	-29,2	-25,9	-25,6	-26,5	-26,3	-25,8	-25,0	-26,3	-26,8	-27,2
	-25,5	-26,3	-26,6	-27,2	-26,8	-27,5	-27,6	-26,9	-26,9	-27,2
	-26,1	-27,8	-26,3	-25,3	-25,5	-28,9	-28,8	-27,9	-27,4	-29,3
	-25,0	-26,9	-26,2	-26,8	-25,3	-26,7	-28,6	-24,8	-27,1	-28,5
	-26,1	-27,1	-27,0	-26,7	-26,8	-26,9	-26,3	-26,9	-26,2	-26,1
	-27,6	-26,7	-25,7	-28,0	-28,2	-28,3	-27,6	-28,0	-28,0	-27,1
	-27,5	-27,9	-28,4	-28,4	-28,8	-28,2	-26,9	-28,1	-27,2	-28,1
	-28,7	-26,9	-27,6	-25,7	-27,8	-27,7	-27,8	-28,1	-27,8	-27,8
	-26,7	-27,5	-27,8	-26,9	-26,9	-27,5	-27,0	-27,4	-26,4	-27,4
	-27,4	-27,6	-27,5	-27,5	-28,0	-27,5	-27,5	-27,6	-26,8	-26,9
	-26,3	-27,4	-27,4	-27,5	-27,9	-28,3	-26,7	-27,4	-27,2	-27,2
	-27,2	-27,3	-26,5	-27,2	-26,6	-26,6	-26,8	-26,8	-27,5	-27,4
	-26,7	-26,9	-27,6	-26,6	-25,5	-33,0	-32,9	-32,5	-31,8	-31,9
	-29,5	-24,9	-26,3	-26,4	-27,0	-27,8	-26,4	-27,4	-26,6	-27,2
	-26,5	-26,4	-26,4	-27,3	-27,6	-25,9	-26,8	-26,4	-27,6	-27,8

But it is difficult to evaluate the performances only looking to these results. Therefore, we did a paired t-test in order to draw a significant conclusion, assuming that the obtained results come from a normal distribution.

We do the paired t-test when we want to compare two different independent observations. It establishes a hypothesis over the difference of the pairs. In the test, we have two hypotheses, a null hypothesis and an alternative hypothesis. We test the acceptability of the null hypothesis. If the null hypothesis is rejected, the alternative hypothesis seems to be accepted. We established our hypotheses as;

$$\begin{aligned} H_0 &: \mu_{GA} - \mu_{PSO} = 0 \\ H_1 &: \mu_{GA} - \mu_{PSO} > 0 \end{aligned} \tag{5.2}$$

The t-critical and t-observed values are computed and compared. If t-observed is greater than t-critical, we reject the null hypothesis.

$t-critical = t_{\alpha,n-1}$, where α and $n-1$ are the confidence level and the degrees of freedom respectively.

$t\text{-}observed = \dfrac{\overline{d}}{s_D / \sqrt{n}}$, where $\overline{d}$ and s_D are the sample mean and the sample standard deviation, respectively.

We did separate tests for each data set, including 200 (20 instance*10 replication) instances. The statistical Statgraphics and Minitab programs are used to apply the tests and draw the related graphics.

Table 5.4.a Paired t-test for GA vs PSO fitness

20x15	**20x20**
Average = 0,33	Average = 0,125
Variance = 0,342814	Variance = 0,200377
Standard deviation = 0,585503	Standard deviation = 0,447635
Minimum = -1,0	Minimum = -1,0
Maximum = 2,0	Maximum = 2,0
Range = 3,0	Range = 3,0
Stnd. skewness = 4,85824	Stnd. skewness = 10,9963
Stnd. kurtosis = 1,78297	Stnd. kurtosis = 17,4157
t-test	**t-test**
------	**------**
Null hypothesis: mean = 0,0	Null hypothesis: mean = 0,0
Alternative: greater than	Alternative: greater than
Computed t statistic = 7,97076	Computed t statistic = 3,94913
P-Value = 1,06289E-7	P-Value = 0,0000544921
Reject the null hypothesis for alpha = 0,05.	Reject the null hypothesis for alpha = 0,05.

Table 5.4.b Paired t-test for GA vs PSO fitness

30x15	**30x20**
Average = 0,46	Average = 0,41
Variance = 0,72201	Variance = 0,584824
Standard deviation = 0,849712	Standard deviation = 0,764738
Minimum = -1,0	Minimum = -1,0
Maximum = 3,0	Maximum = 3,0
Range = 4,0	Range = 4,0
Stnd. skewness = 4,59787	Stnd. skewness = 4,13262
Stnd. kurtosis = 1,46099	Stnd. kurtosis = 0,978816
t-test	**t-test**
------	**------**
Null hypothesis: mean = 0,0	Null hypothesis: mean = 0,0
Alternative: greater than	Alternative: greater than
Computed t statistic = 7,65599	Computed t statistic = 7,58204
P-Value = 9,51225E-8	P-Value = 9,32017E-8
Reject the null hypothesis for alpha = 0,05.	Reject the null hypothesis for alpha = 0,05.

Table 5.4.c Paired t-test for GA vs PSO fitness

40x15	**40x20**
Average = 0,4	Average = 0,315
Variance = 1,47739	Variance = 0,930427
Standard deviation = 1,21548	Standard deviation = 0,964587
Minimum = -3,0	Minimum = -3,0
Maximum = 3,0	Maximum = 3,0
Range = 6,0	Range = 6,0
Stnd. skewness = -1,13584	Stnd. skewness = 0,253956
Stnd. kurtosis = -0,203015	Stnd. kurtosis = 1,13774
t-test	**t-test**
------	**------**
Null hypothesis: mean = 0,0	Null hypothesis: mean = 0,0
Alternative: greater than	Alternative: greater than
Computed t statistic = 4,65402	Computed t statistic = 4,61832
P-Value = 0,00000304868	P-Value = 0,00000354851
Reject the null hypothesis for alpha = 0,05.	Reject the null hypothesis for alpha = 0,05.

Table 5.4.d Paired t-test for GA vs PSO fitness

50x15	**50x20**
Average = 0,34	Average = 0,41
Variance = 2,19538	Variance = 1,85116
Standard deviation = 1,48168	Standard deviation = 1,36057
Minimum = -5,0	Minimum = -4,0
Maximum = 4,0	Maximum = 3,0
Range = 9,0	Range = 7,0
Stnd. skewness = -3,63503	Stnd. skewness = -3,50104
Stnd. kurtosis = 1,6892	Stnd. kurtosis = 2,68294
t-test	**t-test**
------	**------**
Null hypothesis: mean = 0,0	Null hypothesis: mean = 0,0
Alternative: greater than	Alternative: greater than
Computed t statistic = 3,24518	Computed t statistic = 4,26165
P-Value = 0,000688569	P-Value = 0,0000157335
Reject the null hypothesis for alpha = 0,05.	Reject the null hypothesis for alpha = 0,05.

In all data sets, it is seen that the difference of the fitness values of the genetic algorithm and the particle swarm optimization algorithm is greater than zero, which means that PSO gives more promising results with respect to GA.

A sample box and whiskers plot is depicted for the 20x15 problem instances in Figure 5-1. As seen in the figure, the difference is located at the right of the zero. This approved the acceptability of the alternative hypothesis which states that PSO gives better yield than that of GA.

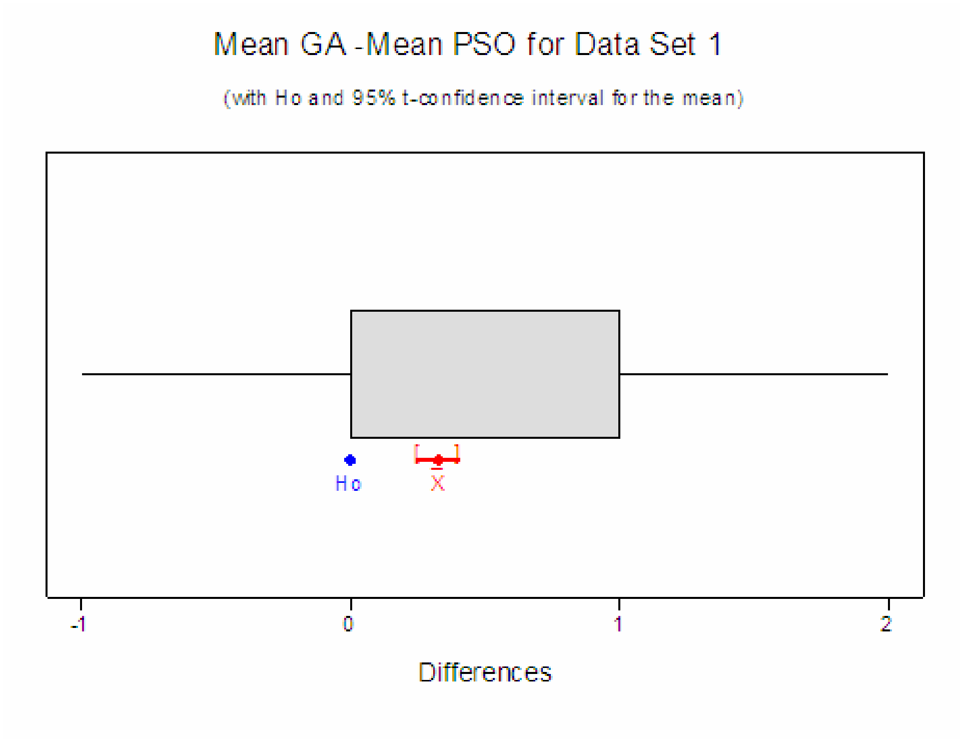

Figure 5-1 A Boxplot for the Difference of Fitness Values

The Figure 5-2 shows a normal probability plot for a data set (40 jobx15 machine) among 8 data sets. The difference values draw a straight pattern which supports the normality assumption.

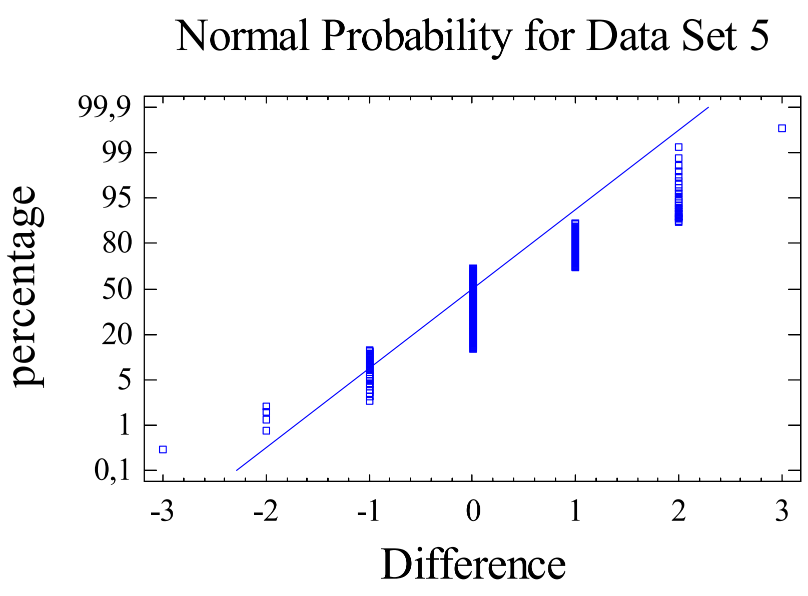

Figure 5-2 A Normal Probability Plot for the Difference of Fitness Values

In Table 5.5.a and Table 5.5.b, the best sequences for n number of jobs and m number of machines are given. The J x M column depicts the number of jobs and machines. j is for the number of jobs and m is for the number of machines. The sequences are formed randomly from different seeds for both GA and the PSO. In the seed column, the seeds that find the best sequences are stated.

Table 5.5.a Best GA Job Sequences

J x M	SEED	FITNESS	GA JOB SEQUENCE
20X15	856767	10	1 8 18 9 10 14 13 0 3 2 7 17 19 5 6 11 4 16 15 12
20X20	856773	13	17 4 12 2 18 0 10 3 13 9 6 14 5 11 7 1 15 16 8 19
30X15	856773	14	5 11 16 20 13 8 23 24 25 18 2 28 10 12 14 27 19 15 1 29 21 9 22 0 26 4 7 3 17 6
30X20	856771	16	11 23 4 2 14 7 1 9 13 21 28 0 16 3 24 12 5 25 20 8 6 15 27 10 22 29 17 19 18 26
40X15	856765	15	32 2 14 33 13 37 15 23 36 19 16 9 8 27 7 21 26 22 6 20 12 38 11 5 30 29 25 17 18 0 1 34 10 4 3 31 24 28 35 39
40X20	856770	20	8 7 13 4 12 27 11 19 14 6 32 38 3 26 15 24 2 33 23 9 0 17 25 18 37 5 16 21 30 1 28 35 10 29 22 20 36 34 39 31
50X15	856773	17	31 48 49 34 18 27 35 4 20 32 23 40 21 38 24 44 2 6 1 25 10 33 37 28 7 5 9 17 30 26 15 39 3 43 14 16 22 45 36 12 11 13 19 46 41 47 8 42 29 0
50X20	856765	21	12 9 15 3 11 40 16 25 33 42 6 1 27 23 14 21 2 19 4 32 22 43 13 44 38 20 24 5 8 7 29 31 28 0 17 26 34 35 47 10 18 36 45 30 39 37 46 41 48 49

Table 5.5.b Best PSO Job Sequences

J x M	SEED	FITNESS	PSO JOB SEQUENCE
20X15	856774	10	1 8 16 18 0 10 14 7 2 3 19 6 12 13 15 11 5 17 9 4
20X20	856774	13	17 4 2 12 10 0 18 11 19 3 1 9 6 15 14 5 16 7 13 8
30X15	856774	14	25 15 16 11 13 8 27 20 23 1 2 12 22 24 28 14 18 10 26 29 19 6 21 0 4 7 9 5 3 17
30X20	856773	16	11 7 15 6 17 0 2 19 24 21 28 9 13 23 27 3 22 4 12 10 18 20 1 29 16 5 26 25 14 8
40X15	856773	15	28 33 32 27 13 15 0 36 14 23 6 16 37 26 18 21 22 8 38 12 1 7 5 11 29 4 34 20 2 30 19 31 24 10 3 17 35 39 25 9
40X20	856765	20	12 10 6 22 27 9 39 19 25 4 38 11 26 32 3 24 14 15 30 2 34 23 33 37 0 5 1 21 20 17 31 29 7 16 35 28 18 36 8 13
50X15	856772	18	9 0 1 32 18 30 20 44 23 4 47 31 21 49 48 40 39 24 25 8 38 33 5 2 34 7 10 43 45 37 15 14 17 46 26 36 3 42 35 22 13 11 29 27 28 41 6 12 16 19
50X20	856771	21	0 12 40 44 1 11 37 20 16 25 33 3 23 2 6 47 27 14 43 39 21 7 13 19 32 30 22 38 24 8 34 5 35 26 36 41 17 10 45 46 28 15 4 9 18 31 42 48 29 49

CHAPTER 6

CONCLUSIONS

Minimization of number of tardy jobs is an important objective in both manufacturing and service industries. When the customers are supplied on the scheduled time, they will have good feelings about the company. In our study we aimed to minimize the number of tardy jobs in permutation flow shops.

To the best of our knowledge, our study stands to be the first to apply the genetic and the particle swarm optimization algorithms to this NP-hard sequencing problem. We used the traditional genetic algorithm and the discrete particle swarm optimization algorithm. However, PSO is used for optimizing continuous functions; we enabled it to be applied to the discrete case by using a simple SPV (Smallest Position Value) heuristic rule.

After implementing the GA and PSO algorithms to the data sets, we compared the fitness values and the cpu times to see the performance of each algorithm to solve the problem. This comparison was very fair.

From the statistics, graphs and hypothesis tests that we obtain from MS Excel, Minitab and Statgraphics programs, we saw that the PSO algorithm gave promising results for finding better sequences. The cpu results of PSO is superior to GA for the data sets of 20x15, 20x20 and 30x15. For the rest of data sets GA leads PSO.

We used in the study the due dates given in the data set of Demirkol et al. It will be more meaningful if we generate the due dates which are more realistic. It is also among our further studies to develop more meaningful upper and lower bounds for due dates. This study is still one of a few studies in this area and it will be more valuable if we brush it with further studies.

REFERENCES

Angeline, P., Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference, the 7th Annual Conference on Evolutionary Programming, San Diego, USA, 1998.

Baptiste, P., Peridy, L., Pinson, E., A branch and bound to minimize the number of late jobs on a single machine with release time constraints, European Journal of Operational Research, Vol. 144, pp: 1–11, 2003.

Bean, J. C., Genetic Algorithm and Random Keys for Sequencing and Optimization, ORSA Journal on Computing, Vol. 6 (2), pp: 154-160, 1994

Beasley, D., Bull D. R., Martin, R. R., An Overview of Genetic Algorithms: Part 1, Fundamentals. University Computing, Vol. 15 (2), pp: 58 -69, 1993.

Beasley, D., Bull D. R., Martin, R. R., An Overview of Genetic Algorithms: Part 2, Research Topics. University Computing, Vol. 15 (4), pp: 170-181, 1993

Bertel, S., Billaut, J., C., A genetic algorithm for an industrial multiprocessor flowshop scheduling problem with recirculation, European Journal of Operational Research, Vol. 159, pp: 651–662, 2004.

Bolat, A., Al-Harkan, İ., Al-Harbi, B., Flow-shop scheduling for three serial stations with the last two duplicate, Computers & Operations Research, Vol. 32, pp: 647–667, 2005.

Bulfin, R.L., Hallah, R., Minimizing the weighted number of tardy jobs on a two-machine flow shop, Computers & Operations Research, Vol. 30, pp: 1887–1900, 2003.

Carlisle, A., and Dozier, G., Adapting Particle Swarm Optimization to dynamic environments ,WAC 2002 Proceedings,Orlando,Florida, June 9-13, 2002

Carlisle, A., and Dozier, G., An Off-the-Shelf PSO, Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis. In: Purdue School of Engineering & Technology, 2001.

Cavory, G., Dupas, R., Goncalves, G., A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints, European Journal of Operational Research, Vol. 161, pp:73–85, 2005

Chan, F., T., S., Chung, S.H., Multicriterion genetic optimization for due date assigned distribution network problems, Decision Support Systems, 2004.

Chang, P., Su, L., Scheduling n Jobs on One Machine to Minimize the Maximum Lateness with Minimum Number of Tardy Jobs, Computers & Industrial Engineering, Vol. 40, pp: 349-360, 2001

Clerc, M., The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceedings, 1999 ICEC, Washington, DC, pp 1951-1957, 1999.

Croce, F. D., Gupta, J., N., D., Roberto Tadei, Minimizing tardy jobs in a flowshop with common due date, European Journal of Operational Research, Vol.120, pp: 375-381, 2000.

Dauzére-Pérés, S., Minimizing late jobs in the general one machine scheduling problem, European Journal of Operational Research, Vol. 81 pp: 134–142, 1995.

Dauzére-Pérés, S., Sevaux, M., A Branch and Bound Method to Minimize the Number of Late Jobs on a Single Machine In National contribution for the 15th triennal conference, IFORS'99, Beijin, P.R. of China, 16-20 August 1999

Dauzére-Pérés, S., Sevaux, M., Using Lagrangean Relaxation to Minimize the (Weighted) Number of Late Jobs on a Single Machine, 1999.

Demirkol E., Mehta S. and Uzsoy R., Benchmarks for shop scheduling problems, European Journal of Operational Research, Vol. 109, pp: 137-141, 1998

Devore, J. L., Probability and Statistics for Engineering and the Sciences, Duxbury Thomson Learning, Pacific Grove, CA, 2001

Eberhart, R. C.& Hu, X., Human Tremor Analysis Using Particle Swarm Optimization., Proc. Congress on Evolutionary computation, Wachington, DC, pp 1927-1930, Piscataway, NJ: IEEE Service Center, 1999

Eberhart, R. C., and Kennedy, J., A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, Piscataway, NJ: IEEE Service Center, pp: 39-43, 1995.

Eberhart, R. C., and Shi, Y., Comparison between Genetic Algorithms and Particle Swarm Optimization. In V. W. Porto, N. Saravanan, D. Waagen, A. E. Eiben, Eds. Evolutionary Programming VII: Proc. Seventh Ann. Conf. on Evolutionary Programming Conf., San Diego, CA. Berlin: Springer-Verlag, 2001.

Eberhart, R. C., and Shi, Y., Particle Swarm Optimization: Developments, Applications and Resources, Proc. congress on evolutionary computation 2001 IEEE service center, Piscataway, NJ., Seoul, Korea., 2001

Eberhart, R. C., and Shi, Y., Tracking and Optimizing Dynamic Systems with Particle Swarms, Proc. Congress on Evolutionary computation, Seoul, Korea. Piscataway, NJ: IEEE Service Center, 2001

Eberhart, R. C., Simpson, P. K., and Dobbins, R. W., Computational Intelligent PC Tools, Boston, MA: Academic Press Professional, 1996

Eberhart, R.C., and Shi, Y., Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization, Congress on Evolutionary Computing, Vol. 1, pp. 84-88, 2000

Gao, Y., Rong, H., Huang, J., Z., Adaptive grid job scheduling with genetic algorithms, Future Generation Computer Systems, 2004

Ghoshal, S. P., Optimizations of PID gains by particle swarm optimizations in fuzzy based automatic generation control, Electric Power Systems Research Vol. 72, pp: 203–212, 2004

Gonçalves, J., Mendes, J., Resende, M., G., C., A hybrid genetic algorithm for the job shop scheduling problem, European Journal of Operational Research, 2004

Gordon, V., Kubiak, W., Single Machine Scheduling with Release and Due Date Assignment to Minimize the Weighted Number of Tardy Jobs, Information Processing Letters, Vol. 68, pp: 153-159, 1998.

Güner, E., Erol, S., Tani, K., One machine scheduling to minimize the maximum earliness with minimum number of tardy jobs, Int. J. Production Economics, Vol. 55, pp: 213-219, 1998.

Gupta, J. N. D., Hariri, A.M.A., Two-machine flowshop scheduling to minimize the number of tardy jobs, Journal of the Operational Research Society, Vol. 48, pp: 212–220, 1997

Hallah R., M., Bulfin, R., L., Minimizing the weighted number of tardy jobs on a single machine, European Journal of Operational Research, Vol. 145, pp: 45-56, 2003

Hariri, A.M.A., Potts, C.N., A Branch and Bound Algorithm to Minimize the Number of Late Jobs in a Permutation Flowshop, European Journal of Operational Research, Vol. 38 pp: 228-237, 1989

Hariri, A.M.A., Potts, C.N., Single machine scheduling with deadlines to minimize the weighted number of tardy jobs, Management Science, Vol. 40, pp: 1712-1719, 1994

He, S., Wu, Q. H., Wen, J. Y., Saunders, J. R., Paton, R. C., A Particle Swarm Optimizer With Passive Congregation, BioSystems, 2004

Held, M., Karp, R. M., A Dynamic Programming Approach to Sequencing Problems, Journal of the Society for Industrial and Applied Mathematics, Vol. 10(1), pp: 196-210, 1962

Hino, C. M., Ronconi, D. P., Mendes A. B., Minimizing earliness and tardiness penalties in a single-machine problem with a common due date, European Journal of Operational Research, Vol. 160, pp: 190–201, 2005

Holland, J., Adaptation in natural and artificial system. Ann Arbor, MI: The University of Michigan Press, 1975

Iyer, S., K., Saxena, B., Improved Genetic Algorithm for the Permutation Flowshop Scheduling Problem, Computers & Operations Research, Vol. 31, pp: 593–606, 2004

Karp, R., M. Reducibility Among Combinatorial Problems.. In: R. E. Miller and J. W. Thatcher, Eds., Complexity of Computer Computations, pp. 85.103, Plenum Press, New York, NY, USA, 1972.

Kennedy, J. and Spears, W., Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and some Genetic Algorithms on the Multimodal Problem Generator, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998

Kennedy, J. Eberhart, R. C., Shi, Y., Swarm Intelligence, San Francisco: Morgan Kaufmann Publishers, 2001

Kennedy, J., and Eberhart, R. C., Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks, IV, 1942-1948. Piscataway, NJ: IEEE Service Center, 1995

Kennedy, J., Eberhart, R. C., and Shi, Y., Swarm Intelligence, San Francisco: Morgan Kaufmann Publishers, 2001

Kennedy, J., Eberhart, R., Particle Swarm Optimization, IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, NJ, Volume: 4, pp. 1942-1948, 1995

Kethley, R., B., Alidaee, B., Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms, Computers & Industrial Engineering, Vol. 43, pp: 509–528, 2002

Kim, K. W., Gen, M., Yamazaki, G. Hybrid genetic algorithm with fuzzy logic for resource-constrained project scheduling, Applied Soft Computing, 2003

Kise, H., Ibaraki, T., Mine, H, A solvable case of the one machine scheduling problem with ready and due times, Operations Research, Vol. 26 (1), pp: 121–126, 1978

Köksalan, M., Keha, A. B., Using genetic algorithms for single-machine bicriteria scheduling problems, European Journal of Operational Research, Vol. 145, pp: 543–556, 2003

Lawler, E. L., Moore, C. U., A Functional Equation and its Application to Resource Allocation and Sequencing Problems, Management Science, Vol. 16, pp. 77.84, 1969

Lawler, E., L. Sequencing to minimize the weighted number of tardy jobs, RAIRO Rech. Optr, Vol. 10, pp: 27-33, 1976.

Lawler, E., L., A Dynamic Programming Algorithm for Preemptive Scheduling of a Single Machine to Minimize the Number of Late Jobs, Annals of Operations Research, Vol. 26, pp: 125-133, 1990

Lenstra, J.K., Rinnooy Kan, A.H.G., Bruker, P., Complexity of machine scheduling problems, Annals of Discrete Mathematics, Vol. 1, pp: 343-362, 1977

Leu, S., Hwang, S., GA-based resource-constrained flow-shop scheduling model for mixed precast production, Automation in Construction, Vol. 11, pp: 439– 452, 2002.

Lin, B.M.T., Cheng, T.C.E., Minimizing the weighted number of tardy jobs and maximum tardiness in relocation problem with due date constraints, European Journal of Operational Research, Vol. 116, pp: 183-193, 1999

Lin, B.M.T., Scheduling in the two-machine flowshop with due date constraints Int. J. Production Economics, Vol. 70, pp: 117-123, 2001

Løbjerg, M., Rasmussen, T.K., Krink, K., Hybrid particle swarm optimizer with breeding and subpopulations, In: Proceedings of the Third Genetic and Evolutionary Computation Conference (GECCO-2001), Vol. 1. pp. 469–476, 2001

Lodree, E., Jang, W., Klein, C., M., A New Rule for Minimizing the Nuımber of Tardy Jobs in Dynamic Flowshops, European Journal of Operational Research, Vol. 159, pp: 258-263, 2004

Moore, J. M., A n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs, Management Science, Vol. 15, No. 1, pp. 102.109, 1968.

Mosheiov, G., Sidney, J. B., Note on Scheduling with General Learning Curves to Minimize the Number of Tardy Jobs, Journal of Operational Research Society, pp: 1-3, 2004

Nearchou, A. C., A novel metaheuristic approach for the flowshop scheduling problem, Engineering Applications of Artificial Intelligence, Vol. 17, pp: 289-300, 2004

Pavlidis, N.G., Parsopoulos, K.E, Vrahatis, M. N., Computing Nash Equilibria Through Computational Intelligence Methods, 2004

Penev, K., Littlefair, G., Free Search—a comparative analysis, Information Sciences, 2004

Peridy, L., Pinson, É., Rivreau, D., Using Short-Term Memory to Minimize the Weighted Number of Late Jobs on a Single Machine, European Journal of Operational Research, Vol. 148, pp: 591-603, 2001

Pinedo, M., Chao, X., Operations Scheduling: with Applications in Manufacturing and Services, Boston, Mass.: Irwin/McGraw-Hill, 1999

Pinedo, M., Scheduling: Theory, Algorithms, and Systems, Englewood Cliffs, N.J.: Prentice Hall, 1995

Potts, C.N., Wassenhove, L. N., Algorithms for scheduling a single machine to minimize the weighted number of late jobs. Management Science Vol. 34, pp: 843–858, 1988.

Reeves, C., R., Yamada, T, Genetic Algorithms, Path Relinking and the Flowshop Sequencing Problem, Evolutionary Computation journal (MIT press), Vol.6 No.1, pp. 230-234, 1998

Rote, G., Woeginger, G. J., Minimizing the Number of Tardy Jobs on a Single Machine with Batch Setup Times, START Project Y43-MAT Combinatorial Approximation Algorithms, 1998

Ruiz, R., Maroto, C., Alcaraz, J., Solving the flowshop scheduling problem with sequence-dependent setup times using advanced metaheuristics, European Journal of Operational Research, 2004

Sahni, S. K., Algorithms for Scheduling Independent Jobs, J. Assoc. Comput. Mach, Vol. 23, pp: 116-127, 1976

Sevaux, M., Dauzere-Peres, S., Genetic algorithms to Minimize the Weighted Number of Late Jobs on a Single Machine, European Journal of Operational Research, Vol. 151, pp: 296-306, 2003

Shi, Y., Eberhart, R. C., Parameter selection in particle swarm optimization, Evolutionary Programming VII, Lecture Notes in Computer Science, vol. 1447. Springer, pp. 591–600, 1998

Shi, Y., Eberhart, R.C., A modified particle swarm optimizer, Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 303–308, 1998

Sipper, D., Bulfin, R. L., Production: Planning, Control and Integration, McGraw Hill, 1998

Steiner, G., Minimizing the number of tardy jobs with precedence constraints and agreeable due dates, Discrete Applied Mathematics, Vol. 72, pp: 167- 177, 1997

Taillard, E., “Benchmarks For Basic Scheduling Problems”, European Journal of Operational Research, Vol. 64, pp. 278-285, 1993.

Taillard, E., Some Efficient Heuristic Methods for the FlowShop Sequencing Problem, European Journal of Operational Research, Vol. 47, pp: 65–74, 1990

Tandon, V., Closing the gap between CAD/CAM and optimized CNC end milling, Master’s thesis, Purdue School of Engineering and Technology, Indiana University, Purdue University Indianapolis, 2000

Tasgetiren M. F., Sevkli M., Yun-Chia Liang, Gencyilmaz G, 2004, Particle Swarm Optimization Algorithm for the Single Machine Total Weighted Tardiness Problem, World Congress on Evolutionary Computation, CEC2004,p.1412-1419.

Tasgetiren, M. F, Liang Y. C., A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem, Journal of Economic and Social Research, Vol.5 No.2, 2003.

Tasgetiren, M. F., Liang Y. C., Sevkli M., Yenisey, M. M., Particle Swarm Optimization and Differential Evolution Algorithms for Job Shop Scheduling, 2005 (submitted)

Tasgetiren. M. F., Liang Y. C., Sevkli, M., Gencyilmaz, G., Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem, 4th International Workshop on Ant Colony Optimization and Swarm Intelligence, ANTS2004, LNCS 3172 by Springer-Verlag, pp.382-390, September 5-8, 2004

Tasgetiren. M. F., Liang Y. C., Sevkli, M., Gencyilmaz, G., Particle Swarm Optimization Algorithm for Makespan and Total Flowtime Minimization in Permutation Flowshop Sequencing Problem, European Journal of Operational Research, 2004

Tasgetiren. M. F., Liang Y. C., Sevkli, M., Gencyilmaz, G., Particle Swarm Optimization Algorithm for Makespan and Maximum Lateness Minimization in Permutation Flowshop Sequencing Problem, 4th International Symposium on Intelligent Manufacturing Systems, IMS2004, pp.431-441, Sakarya, Turkey, 6-8 Sep 2004

Tasgetiren. M. F., Liang Y. C., Sevkli, M., Gencyilmaz, G., Particle Swarm Optimization and Differential Evolution Algorithms for the Single Machine Total Weighted Tardiness Problem, 2005 (Submitted)

Tasgetiren. M. F., Sevkli, M., Liang Y. C., Gencyilmaz, G., Particle Swarm Optimization Algorithm for Single Machine Total Weighted Tardiness Problem, 4th International Symposium on Intelligent Manufacturing Systems, IMS2004, pp.431-441, Sakarya,Turkey 6-8 Sep 2004

Trelea, I. C., The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection, Information Processing Letters, Vol. 85, pp: 317-325, 2003

Tsujimura, Y., Mafune, Y., Gen, M., Effects of Symbiotic Evolution in Genetic Algorithms for Job-Shop Scheduling, Proceedings of the 34th Hawaii International Conference on System Sciences, 2001

Villarreal, F., J., Bulfin, R., L., Scheduling a single machine to minimize the weighted number of tardy jobs, IIE Transactions, Vol 15, pp: 337–343, 1983

Wang, H., Wu, K., Hybrid genetic algorithm for optimization problems with permutation property, Computers & Operations Research, Vol. 31, pp: 2453–2471, 2004

Wang, Y., A GA-based methodology to determine an optimal curriculum for schools, Expert Systems with Applications, Vol. 28, pp: 163-174, 2005

Werner J. C., Aydin, M. E., Fogarty, T., C., Evolving genetic algorithm for Job Shop Scheduling problems, Proceedings of ACDM, 2000

Yinga, K. Liaoa, C., An Ant Colony System for Permutation Flow-Shop Sequencing, Computers & Operations Research, Vol. 31, pp: 791–801, 2004

Yoo, W. S., Martin-Vega, L. A., Scheduling Single Machine Problems for On-time Delivery, Computers & Industrial Engineering, Vol. 39, pp: 371-392, 2001

Zacharia, P. T., Aspragathos, N. A., Optimal robot task scheduling based on genetic algorithms, Robotics and Computer-Integrated Manufacturing, Vol. 21, pp: 67–79, 2005

Zhang, H., Li, H., Huang, F., Particle Swarm Optimization-Based Schemes for Resource-Constrained Project Scheduling, Automation in Construction, 2004