DISCRETE PARTICLE SWARM OPTIMIZATION AND DIFFERENTIAL EVOLUTION ALGORITHMS FOR FLOW SHOP SCHEDULING PROBLEMS

by

Mehmet Seyyid ÖZTEKİN

August 2009

DISCRETE PARTICLE SWARM OPTIMIZATION AND DIFFERENTIAL EVOLUTION ALGORITHMS FOR FLOW SHOP SCHEDULING PROBLEMS

by

Mehmet Seyyid ÖZTEKİN

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

August 2009
Istanbul, Turkey

APPROVAL PAGE

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

_____
Assist. Prof. Mehmet Şevkli
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. Mehmet Şevkli
Supervisor

Examining Committee Members

Assist. Prof. Mehmet Şevkli _____

Assist. Prof. Özgür Uysal _____

Assist. Prof. İhsan Ömür Bucak _____

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.

_____
Assoc. Prof. Nurullah ARSLAN
Director

Date
August 2009

DISCRETE PARTICLE SWARM OPTIMIZATION AND DIFFERENTIAL EVOLUTION ALGORITHMS FOR FLOW SHOP SCHEDULING PROBLEMS

Mehmet Seyyid Öztekin

M. S. Thesis - Industrial Engineering
August 2009

Supervisor: Assist. Prof. Mehmet Şevkli

ABSTRACT

In this study, I tried to minimize the makespan of jobs in flow shops. The makespan criterion is a measure for total completion time of all jobs. Particle Swarm Optimization and Differential Evolution algorithms are heuristic methods used for solving combinatorial optimization problems like flow shop scheduling problems. These two different methods are proposed for the flow shop scheduling problem with minimizing makespan criterion. The algorithms are implemented using the processing time of Taillard benchmark data sets. The computational results are obtained to evaluate the fitness and cpu time for each algorithm. The performances of both algorithms to find optimal processing sequence for the jobs through m machines are compared. It is concluded from the experiments that SPPSO and DDE algorithms performed better results to compare PSO and DPSO algorithms.

Keywords: Scheduling, Flow shop, Discrete Particle Swarm Optimization, Discrete Differential Evolution Algorithm, Minimizing Makespan

AKIŞ TİPİ ÇİZELGELEME PROBLEMLERİ İÇİN KESİKLİ SÜRÜ PARÇACIK OPTİMİZASYONU VE DİFERANSİYEL EVRİMSEL ALGORİTMASI

Mehmet Seyyid Öztekin

Yüksek Lisans Tezi – Endüstri Mühendisliği
Ağustos 2009

Supervisor: Yard. Doç. Mehmet Şevkli

ÖZ

Bu çalışmada akış tipi atölyelerdeki işlerin toplam tamamlanma sürelerini minimize etmeye çalıştım. Toplam tamamlanma süresi işlerin bitiş zamanını gösteren performans kriteridir. Sürü parçacık optimizasyonu ve ayırtedici evrimsel algoritması akış tipi atölyelerdeki gibi kombinatoryel optimization problemlerini çözmek için kullanılan sezgisel yöntemlerdir. Bu iki farklı yöntem, akış tipi çizelgeleme problemleri için toplam tamamlanma süresini minimize etme kriteri altında önerildi. Algoritmalarda Taillard veri setinin işlem süreleri kullanıldı. Her bir algoritmanın işlemci süreleri ve performans değerleri hesapları elde edildi. Her iki algoritmanın m makinadaki en optimal iş sıralamasındaki performansları karşılaştırıldı. Yapılan deneylerde SPPSO ve DDE algoritmaları, PSO ve DPSO algoritmalarına karşın daha iyi sonuçlar vermiştir.

Anahtar Kelimeler: Çizelgeleme, Akış Tipi Atölyeler, Kesikli Sürü Parçacık Optimizasyonu, Ayırtedici Evrimsel Algoritması, Toplam Tamamlanma Süresini Minimize etme

To my family

ACKNOWLEDGEMENT

I would like to thank my thesis supervisor Assist. Prof. Mehmet ŞEVKLİ for his guidance throughout the research and invaluable help during the implementation phase of the thesis.

I express my thanks and appreciation to my family for their love, encouragement, understanding, and patience and I also thank my friends for their motivation.

I am greatful to my jury members, Assist. Prof. Özgür UYSAL and Assist. Prof. İhsan Ömür BUCAK for their valuable suggestions and comments.

I express my thanks and appreciation to TUBITAK (The Scientific and Technological Research Council of Turkey) R&D Human Resources Development for their support, during my master education with MSc fellowship.

And I would like to thank all my friends and colleagues at Fatih University, for their friendship and collaboration and companionship throughout this journey in graduate school life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYSMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION

p_j	: Processing time for Job j
C_{max}	: Completion time of all jobs in the flow shop
FSSP	: Flow Shop Sequencing Problem
$Fm \mid\mid C_{max}$	: Flowshop Sequencing Problem with Criterion of Makespan
PSO	: Particle Swarm Optimization Algorithm
DPSO	: Discrete Particle Swarm Optimization Algorithm
DDEA	: Discrete Differential Evolution Algorithm
SSPSO	: The Perturbed Particle Swarm Optimization Algorithm
SPV	: Smallest Position Value Heuristic Rule
CPU	: Central Processing Unit
NxM	: n Number of Jobs through m Number of Machines

CHAPTER 1

INTRODUCTION

OVERVIEW

Scheduling problems exist almost everywhere in real-world situations (Gen & Cheng, 1997), most of all in the industrial engineering world. Scheduling is a kind of decision making that plays importance role in both manufacturing and service industries. Moreover, effective scheduling increases productivity, utilization of resources and can yield cost saving.

Flow shop scheduling is one of the most popular problems in this area. The flow shop scheduling problem was first studied by Johnson (1954) for sequencing jobs on single or two machines with the objective of minimizing makespan. Various exact, heuristic and metaheuristic methods have been proposed after the Johnson's study and different objective criteria are used for evaluating the best sequence for all jobs on the set of machines.

More scheduling problems are very difficult and complex to solve by traditional methods. Many heuristic methods are developed to solve these difficult problems. We applied two metaheuristic algorithms to the $Fm \mid\mid C_{max}$ problem. Discrete particle swarm optimization algorithm and discrete differential evolution algorithm are our proposed algorithms to be used in the study. DPSO is used different problem types and applied to scheduling problems. DDE algorithm is a new form for scheduling problem. We will compare the fitness values and CPU times to see performance of each algorithm to solve the problems with the objectives of minimizing makespan and maximum lateness.

The organization of my thesis is as follows. Chapter 2 gives an overview of flow shop scheduling problems with classification, complexity and literature review about it.

An simple illustrated example is also given and some methods are introduced in this chapter. This chapter also includes some notations and assumptions.

In chapter 3, history of particle swarm optimization is mentioned. Discrete version of PSO and our proposed perturbed particle swarm optimization algorithm are explained. Like PSO, same procedure is followed for differential evolution algorithm in chapter 4. Literature review and the methodology of these methods are given. In chapter 5, these four methods' experimental results are presented and computational results of the study are summarized. Finally, chapter 6 discusses results and makes the conclusions.

CHAPTER 2

FLOW SHOP SCHEDULING

2.1 INTRODUCTION

There are many exact, heuristic and meta-heuristic solution techniques developed for machine scheduling problems. These problems are a decision making process with the goal of optimizing one or more objective.

The flow shop scheduling problem is a production planning problem. A flow shop can be described as n jobs which have to go in the same order on m different machines. There are many kinds of flow shops, such as no-wait flow shop, multi-objective flow shop, blocking flow shop, and so on. The sequence of the problem varies due to the objective of the problem. These objectives can be minimizing makespan, minimizing number of tardy jobs or tardiness, minimizing maximum lateness, minimizing flow time.

Given the processing times p_{jk} for job j on machine k, and a job permutation $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ where n jobs (j = 1, 2,…, n) will be sequenced through m machines (k = 1, 2,…,3) using the same permutation. Given the job permutation $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$, the calculation of completion time for n-job m-machine problem is given as follows:

$$C(\pi_1, 1) = P_{\pi_1,1} \qquad 2.1.1$$

$$C(\pi_j, 1) = C(\pi_{j-1}, 1) + P_{\pi_j,1} \qquad j = 2, 3, \dots, n \qquad 2.1.2$$

$$C(\pi_1, k) = C(\pi_1, k-1) + P_{\pi_1,k} \qquad k = 2, 3, \dots, m \qquad 2.1.3$$

$$C(\pi_j, k) = max\left\{C(\pi_{j-1}, k); C(\pi_j, k-1) + P_{\pi_j,k}\right\} \; j = 2, 3, \dots, n; \; k = 2, 3, .., m \qquad 2.1.4$$

2.2 COMPLEXITYS OF Fm || Cmax PROBLEMS

Many scheduling problems from manufacturing industries are quite complex in nature and very difficult to solve by conventional optimization techniques. They belong the class of NP-hard problems. There exists no polynomial time algorithm for the integer programming. $Fm \mid\mid C_{max}$ problems are strongly NP-hard (Pinedo, 1995).

2.3 PERFORMANCE MEASURE

The objective is to find a suitable allocation of jobs to machines, and a sequence for each machine which would optimize a performance measure. There are many performance measures for scheduling problems. (French, 1982)'s classification of performance measures:

- Completion time based performance measures:
  - Minimize the maximum flow time, F_{max}
  - Minimize the maximum completion time (makespan), C_{max}
  - Minimize mean flow time or total flow time, $\sum \frac{F_i}{n} \ \ or \ \ \sum F_i$
  - Minimize mean completion time or total completion time, $\sum \frac{C_i}{n} \ \ or \ \ \sum C_i$
  - Minimize weighted flow time, $\sum w_i \cdot F_i$
  - Minimize weighted completion time, $\sum w_i \cdot C_i$
- Due Date based performance measures:
  - Minimize maximum lateness, L_{max}
  - Minimize maximum tardiness, T_{max}
  - Minimize mean lateness or total lateness, $\sum \frac{L_i}{n} \ \ or \ \ \sum L_i$
  - Minimize mean tardiness or total tardiness, $\sum \frac{T_i}{n} \ \ or \ \ \sum T_i$
  - Minimize weighted lateness, $\sum w_i \cdot L_i$
  - Minimize weighted tardiness, $\sum w_i \cdot T_i$
- Utilization based performance measures:
  - Minimize idle time

- Maximize utilization of manpower
- Maximize machine utilization

In this study, we aim to determine a sequence that will minimize the maximum completion time (makespan) C_{max}.

$$C_{max}(\pi) = C(\pi_n, m) \quad 2.1.5$$

2.4 LITERATURE REVIEW

Flow shop scheduling problems are first introduced by Johnson (1954). A two and three machine problem is also discussed and solved for a restricted case.

Branch and bound (BB) technique have been used by Giglio and Wagner (1964) for three machine scheduling with objective of minimizing makespan, Dudek and Thuton (1964) used BB technique for scheduling n jobs through m with same performance criteria. Edward Ignall and Linus Schrage developed this method.

For the computational complexity of the PFSP with makespan and maximum lateness minimization, Rinnooy Kan (1976) proves that the makespan minimization is NP-complete, and Lenstra *et al*. (1977) prove that the two-machine flowshop with maximum lateness is NP-Complete, hence the PFSP is much more difficult to solve. Therefore, efforts have been devoted to finding high-quality solutions in a reasonable computational time by heuristic optimization techniques. Heuristics for the makespan minimization problem have been proposed by Taillard (1990), Framinan *et al*. (2002) and Framinan & Leisten (2003).

To achieve a better solution quality, modern meta-heuristics have been presented for the PFSP with makespan minimization such as Simulated Annealing by Osman & Potts (1989), Ogbu & Smith (1990), Tabu Search by Revees (1993), Nowicki & Smutnicki (1996), Grabowski & Wodecki (2004), and Watson *et al*. (2002), Genetic Algorithms by Revees (1995), Revees & Yamada (1998), and Ruiz *et al*. (2006), Ant Colony Optimization by Stützle (1998b), Rajendran & Ziegler (2004), Particle Swarm Optimization by Tasgetiren *et al*. (2004, 2007), Iterated Greedy Algorithm by Ruiz & Stutzle (2007), and Iterated Local Search by Stützle (1998a). An excellent review of flowshop heuristics and metaheuristics can be found in Ruiz & Maroto (2005).

Regarding the maximum lateness minimization, there exists a little research in the literature. However, the need to focus on the customer satisfaction made evident that due-date based objectives are very critical factors for management to be accomplished. On the other hand, the scheduling problem becomes more difficult when due-dates are concerned. This might be the reason why there exists fewer papers dealing with due-date based performance measure in flow shop scheduling.

Townsend (1977), Grabowski *et al*. (1983), and Kim (1995) used branch and bound methods to find optimal solutions. Kim (1995) dealt with the performance measure of total tardiness in two-machine flow shop whereas Grabowski *et al*. (1983) employed the performance measure of maximum lateness, and Townsend (1977) dealt with the minimization of maximum lateness in *m*-machine flow shop. In addition, some heuristic methods were presented by Rajendran & Chaudhuri (1991) and Kim (1993) for minimizing the maximum lateness and mean/total tardiness, respectively.

An example of a permutation flowshop problem schedule is shown in the below figure. The processing time for each job through four machine is generated. The table represent the processing time i_{th} job on j_{th} machine.

Table 2.1 Processing Times of The Example

	$J_{i=1}$	$J_{i=2}$	$J_{i=3}$	$J_{i=4}$
$M_{j=1}$	4	6	2	4
$M_{j=2}$	2	5	3	7
$M_{j=3}$	3	5	3	6
$M_{j=4}$	2	2	3	4

To calculate total completion time, we use the gant chart. Gant chart show start and finish time of each job. We can easily see the waiting time (idle time) of each machine.

From the graph: the makespan is 29 and total idle time is 55. Finish time of J_1 , J_2 , J_3 and J_4 is 25, 29, 13 and 23. As a result, our aim is minimize the total completion time which is 29.

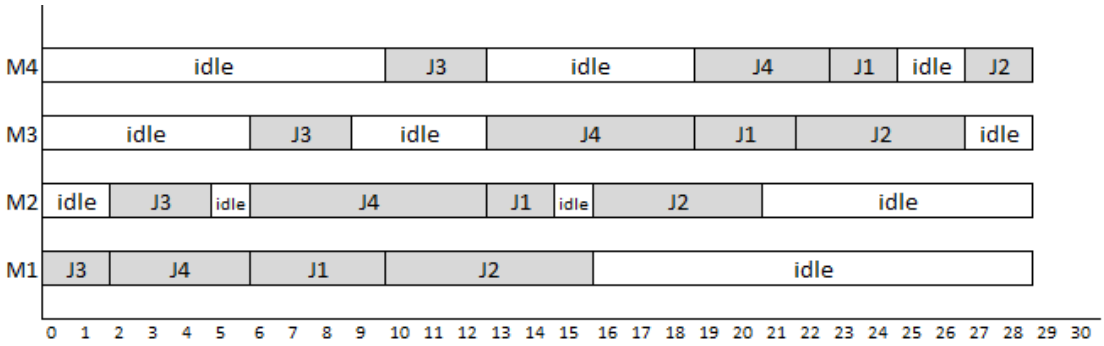

Figure 2.1 Gantt Chart of The Example

2.4.1 Johnson's Algorithm

A heuristic algorithm by S.M. Johnson can be used to solve the case of a 2 machine N job problem when all jobs are to be processed in the same order.The steps of algorithm are as follows:

Step 1: Set k=1 and m=1

Step 2: Set current list of unscheduled jobs $U = \{J_1, J_2, \dots, J_N\}$

Step 3: Find $\min\{a\}_i$ and $\min\{b\}_i$

Step 4: If the smallest time is for J_i on M_1 then

Schedule J_i in the k_{th} position of the processing sequence

Delete J_i from the current list of unscheduled jobs

Increment k to k+1,

Go to Step 6

Step 5: If the smallest time is for J_i on M_2 then;

Schedule J_i in the m_{th} position of the processing sequence

Delete J_i from the current list of unscheduled jobs

Reduce m to m-1,

Go to Step 6

Step 6: If $U = \{\}$, then stop and build the optimal schedule,

Otherwise go to Step 3

If the smallest takes place more than one job in Step 3, then pick arbitrarily.

2.4.2 Branch and Bound Algorithm

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.

Branch and bound algorithm is well known algorithm. This algorithm tries all possibilities of the jobs sequences through m machines. All fitness values are calculated and the best of them are taken.

2.4.3 Mixed Integer Programming

In order to formulate the problem as a Mixed Integer Programming a number of variables have to be defined: The decision variable $X_{j,k}$ equals 1 if job j is the k_{th} job in the sequence and 0 otherwise. The auxiliary variable $I_{i,k}$ denotes the idle time on machine *i* between the processing of the jobs in the k_{th} position and $(k+1)_{th}$ position and the auxiliary variable $W_{i,k}$ denotes the waiting time of the job in the k_{th} position in between machines *i* and *i+1*. There exists a strong relationship between the variables $W_{i,k}$ and the variables $I_{i,k}$. For example, if $I_{i,k} > 0$, then $W_{i-1,k+1}$ has to be zero. Formally, this relationship can be established by considering the difference between the time the job in the $(k+1)_{th}$ position starts on machine *i + 1* and the time the job in the k_{th} position completes its processing on machine *i*.

Define the following decision variables;

$$X_{j,k} = \begin{cases} 1, & if\ job\ j\ assigned\ to\ the\ k_{th}\ position\ in\ the\ permutation \\ 0, & otherwise \end{cases}$$

Note that minimizing the makespan is equivalent to minimizing the total idle time on the last machine, machine *m*. The following mathematical formulation of the permutation flow shop scheduling can be derived

$$\min \left(\sum_{i=1}^{m-1} \sum_{j=1}^{n} p_{i,j}\, X_{j,1} + \sum_{j=1}^{n-1} I_{mj} \right) \qquad 2.2.1$$

Subject to

$$\sum_{j=1}^{n} X_{j,k} = 1 \qquad \forall k \in 1,2,\dots,n \qquad 2.2.2$$

$$\sum_{k=1}^{n} X_{j,k} = 1 \qquad \forall j \in 1,2,\dots,n \qquad 2.2.3$$

$$I_{i,k} + \sum_{j=1}^{n} p_{i,j} X_{j,k+1} + W_{i,k+1} - W_{i,k} - \sum_{j=1}^{n} p_{i+1,j} X_{j,k} - I_{i+1,k} \quad \begin{matrix} \forall i \in 1,2,\dots,m-1 \\ \forall k \in 1,2,\dots,n-1 \end{matrix} \qquad 2.2.4$$

$$W_{i,1} = 0 \qquad \forall i \in 1,2,\dots,m-1 \qquad 2.2.5$$

$$I_{1,k} = 0 \qquad \forall k \in 1,2,\dots,n-1 \qquad 2.2.6$$

The first set of constraints specifies that exactly one job has to be assigned to position *k* for any *k*. The second set of constraints specifies that job *j* has to be assigned to exactly one position. The third set of constraints relate the decision variables $X_{j,k}$ to the physical constraints. These physical constraints enforce the necessary relationships between the idle time variables and the waiting time variables. Thus, the problem of minimizing the makespan in an *m* machine permutation flow shop is formulated as a MIP. The only integer variables are the binary (0−1) decision variables $X_{j,k}$. The idle time and waiting time variables are nonnegative continuous variables.

2.4.4 Heuristics and Metaheuristic Approach

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.

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.

(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.

2.5 ASSUMPTIONS

For the flow shop sequencing problem the following assumptions are made:

- Every job has to be processed at most once on machine 1 to m.
- Each machine processes only one job at a time
- Each job is processed at most on one machine at a time
- Preemption is not allowed.
- The set-up times of the operations are included in the processing time and do not depend on the sequence.
- The operating sequences of the jobs are the same on every machine and the common sequence has to be determined.
- All tasks release at work center at time zero.

CHAPTER 3

PARTICLE SWARM OPTIMIZATION

3.1 OVERVIEW

Exact methods are able to solve only small sized problems. As a matter of fact exact algorithms are useful when the number of sites and customers 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.

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.

Particle swarm optimization is a stochastic optimization technique was first introduced to optimize various continuous nonlinear functions by Dr. Eberhart and Dr. Kennedy in 1995. PSO is based on the social behavior and interaction of animals such as fish shoaling and bird flocking.

In PSO, the system is initialized with a population of random solutions. In that; each potential solution is also assigned a randomized velocity, and the potential solutions, called particles, are then “flown” through the problem space.

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called *pbest*. Another “best” value that is tracked by the global version of the particle swarm optimizer is the overall best value, and its location, obtained so far by any particle in the population. This location is called *gbest*.

The particle swarm optimization concept consists of, at each time step, changing the velocity (accelerating) each particle toward its *pbest* and *gbest* locations (global version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *gbest* locations.

There is also a local version of PSO in which, in addition to pbest, each particle keeps track of the best solution, called *lbest*, attained within a local topological neighborhood of particles.

Originally PSO is distinctly different from other evolutionary-type methods in a way that it does not use the filtering operation such as crossover or mutation, and the members of the entire population are maintained through the search procedure so that information is socially shared among individuals to direct the search towards the best position in the search space. Compared with the other methods, all the particles tend to converge to the best solution quickly even in the local version in most cases.

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.

3.2 LITERATURE REVIEW

The particle swarm concept originated as a simulation of a simplified social system. The original intent was to graphically simulate the graceful but unpredictable choreography of a bird flock. Initial simulations were modified to incorporate nearestneighbor velocity matching, eliminate ancillary variables, and incorporate multidimensional search and acceleration by distance (Kennedy and Eberhart, 1995, Eberhart and Kennedy, 1995). At some point in the evolution of the algorithm, it was realized that the conceptual model was, in fact, an optimizer. Through a process of trial and error, a number of parameters extraneous to optimization were eliminated from the algorithm, resulting in the very simple original implementation (Eberhart et al., 1996).

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,

$$V_{ij}^{t+1} = V_{ij}^{t} + c_1 r_1 \left(P_{ij}^{t} - X_{ij}^{t} \right) + c_2 r_2 \left(g_j^{t} - X_{ij}^{t} \right) \quad (3.1.1)$$

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \quad (3.1.2)$$

c_1: cognition learning rate

c_2: social learning rate

r_1 and r_2: random constant numbers

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

$$V_{ij}^{t+1} = w\, V_{ij}^{t} + c_1 r_1 \left(P_{ij}^{t} - X_{ij}^{t} \right) + c_2 r_2 \left(g_j^{t} - X_{ij}^{t} \right) \quad (3.1.3)$$

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \quad (3.1.4)$$

w: inertia weight

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

$$V_{ij}^{t+1} = \kappa\left[w\, V_{ij}^{t} + c_1\, r_1 \left(P_{ij}^{t} - X_{ij}^{t}\right) + c_2\, r_2 \left(g_j^{t} - X_{ij}^{t}\right)\right] \qquad (3.1.5.a)$$

$$\kappa = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}\ , where\ \varphi = c_1 + c_2, \varphi > 4 \qquad (3.1.5.b)$$

Eberhart and Shi (2001b), 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. When it reaches to 0.4 it is increased again to 0.9.

Six years after the introduction of PSO Eberhart and Shi reviewed the development, applications and the written books and articles (Eberhart and Shi, 2001).

In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods.

PSO has been successfully applied to a wide range of applications such as power and voltage control (Abido 2002), neural network training (Van den Bergh & Engelbecht 2000), mass-spring system (Brandstatter & Baumgartner 2002), task assignment (Salman et al. 2003), supplier selection and ordering problem (Yeh, 2003), permutation flowshop sequencing problems (Tasgetiren et al. 2004, 2007, Lio et al. 2005), lot sizing problem (Tasgetiren & Liang 2003), single machine total weighted tardiness problems (Tasgetiren et al. 2006a), Assembly scheduling (Allahverdi & Anzi 2006), and automated drilling (Onwubolu & Clerc 2004). PSO was compared with other computational intelligence methods in finding the Nash equilibrium in game theory (Pavlidis et al., 2004).

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. PSO was combined with genetic algorithm concepts and evaluated if it was competitive on function optimization (Løbjerg et al., 2001). 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.

A new hybrid particle swarm optimizer with passive congregation was presented (He et al., 2004). 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.

The major obstacle of successfully applying a PSO algorithm to combinatorial problems is due to their continuous nature. To remedy this drawback, the Smallest Position Value (SPV) rule borrowed from the random key representation of Bean (1994) is presented for the PSO algorithm to convert the continuous position values to a discrete job permutation in Tasgetiren et al. (2007). The SPV rule has been effectively applied to the single machine total weighted tardiness problem and the permutation flowshop sequencing problem.

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.

Yin (2004) proposed a discrete PSO algorithm for optimal polygonal approximation of digital curves. Liao et al.(2004). presented another discrete algorithm for flowshop scheduling problems and lastly Pan et al.(2007) proposed another for solving the no-wait flowshop scheduling problems.

3.2.1 Continuous Particle Swarm Optimization for FSSP

In the PSO algorithm, each solution is called a "particle". All the particles have the position, velocity, and fitness values. The particles fly through the n-dimensional space by learning from the historical information from the swarm population. For this reason, the particles are inclined to fly towards better search area over the course of evolution. Let NP denote the swarm population size represented as $X^t = [X_1^t,\ X_2^t, \dots, X_{NP}^t]$. Then each particle in the swarm population has the following attributes: A current position represented as $X_i^t = [x_{i1}^t,\ x_{i2}^t, \dots, x_{in}^t]$, a current velocity represented as $V_i^t = [V_{i1}^t,\ V_{i2}^t, \dots, V_{in}^t]$, a personal best position represented as $\mathrm{P_i^t} =$

$[p_{i1}^t, p_{i2}^t, \dots, p_{in}^t]$, and a global best position represented as $G^t = [g_1^t, g_2^t, \dots, g_n^t]$. Assuming that the function f is to be minimized, the current velocity of the j_{th} dimension of the i_{th} particle is updated as follows;

Step 1: Initialization: Set t=0, m=twice the number of dimensions. Generate m particles $X_i^0 = [x_{i1}^0 \dots x_{in}^0] \{i = 1 \dots m\}$ and and initial velocity of particles $V_i^0 = [v_{i1}^0 \dots v_{in}^0] \{i = 1 \dots m\}$ randomly. Apply the a heuristic rule like SPV and find the sequence of particle $S_i^0 = [s_{i1}^0 \dots s_{in}^0] \{i = 1 \dots m\}$. Evaluate each particle i in the swarm using objective function f_i^0. For each particle i in the swarm, set $PB_i^0 = X_i^0$, where $PB_i^0 = [pb_{i1}^0 = x_{i1}^0, \dots, pb_{in}^0 = x_{in}^0]$ along with its best fitness value. Find the best fitness value $f_1^0 = min\{f_i^0\}$ with its corresponding position x_1^0. Finally, set global best to $GB^0 = X_1^0$ where $GB^0 = [gb_1 = x_{11}, \dots, gb_n = x_{1n}]$ with its fitness value $f^{gb} = f_1^0$.

The initial continuous position and velocity vectors are generated randomly using the following formula:

$$X_{ij} = X_{min} + (X_{max} - X_{min}) \times r_1 \quad (3.2.1)$$

$$V_{ij} = V_{min} + (V_{max} - V_{min}) \times r_2 \quad (3.2.2)$$

where $V_{min} = -4.0$, $V_{max} = 4.0$, $X_{min} = -10.0$, and $X_{max} = 10.0$ which are consistent with the literature (Shi and Eberhart, 1998). $r_1$and r_2 are uniform random numbers between [0,1].

Step 2: Updating iteration counter,

$$t = t + 1 \quad (3.2.3)$$

Step 3: Updating inertia weight,

$$w^t = w^{t-1} \times \alpha \text{ where } \alpha \text{ is decrement factor.} \quad (3.2.4)$$

During each PSO iteration, particle i adjusts its velocity V_{ij} and position vector X_{ij} through each dimension j by referring to, with random multipliers, the personal best

vector (P_i) and the swarm's best vector (G_i, if the global version is adopted) using equations (3.1.b) and (3.2.a).

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

$$V_{ij}^{t+1} = w\, V_{ij}^{t} + c_1\, r_1 \left(P_{ij}^{t} - X_{ij}^{t} \right) + c_2\, r_2 \left(g_j^{t} - X_{ij}^{t} \right) \tag{3.2.5}$$

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.

Step 4: Updating velocity,

$$\mathrm{v}_{\mathrm{ij}}^{\mathrm{t}} = \mathrm{w}^{\mathrm{t}-1}\mathrm{v}_{\mathrm{ij}}^{\mathrm{t}-1} + \mathrm{c}_1\mathrm{r}_1\left(\mathrm{p}_{\mathrm{ij}}^{\mathrm{t}-1} - \mathrm{x}_{\mathrm{ij}}^{\mathrm{t}-1}\right) + \mathrm{c}_2\mathrm{r}_2\left(\mathrm{g}_{\mathrm{j}}^{\mathrm{t}-1} - \mathrm{x}_{\mathrm{ij}}^{\mathrm{t}-1}\right) \tag{3.2.6}$$

where w^{t} is the inertia weight which is a parameter to control the impact of previous velocities on the current velocity; c_1 and c_2 are acceleration coefficients and r_1 and r_2 are uniform random numbers between (0,1).

V_i^t is the velocity of particle i at iteration t. It can be defined as;

$$V_i^t = [V_{i1}^t,\ V_{i2}^t, \ldots, V_{in}^t] \tag{3.2.7}$$

where V_{ij} is the velocity of particle i at iteration t with respect to the j_{th} dimension.

The particle's velocity on each dimension is set restricted by a maximum velocity V_{max}, which controls the maximum travel speed during each iteration to avoid this particle flying past good solutions. 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.

Step 5: Updating position,

The current position of the j_{th} dimension of the t_{th} particle is updated using the previous position and current velocity of the particle i as follows:

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t} \tag{3.2.8}$$

X_i^t is the position of particle i at iteration t. It can be defined as;

$$X_i^t = [x_{i1}^t,\ x_{i2}^t, \dots, x_{in}^t] \tag{3.2.9}$$

where X_{ij}^t is the position of particle i at iteration t with respect to the j_{th} dimension.

The particle's position on each dimension is set restricted by a maximum position X_{max}, which controls the maximum travel distance during each iteration to avoid this particle flying past good solutions. If a position on a dimension of a particle exceeds X_{max}, then it is limited to X_{max}. X_{max} controls the exploration and exploitation ability of a particle. It helps to search the regions between the current position and the target position. It is usually set to 4 or 5 times of V_{max}.

Step 6: Find the permutation,

Apply a heuristic rule to find the sequence $S_i^k = [s_{i1}^t, \dots,\ s_{in}^t]$.

The position values of particles are in fact continuous. To discretize the positions, we use the SPV rule and we determine the sequence of jobs in the flow shop. SPV rule is a kind of heuristic rule. The smallest position value of each vectors assigned to first dimension of the permutation. In other words, dimension are sorted based on the SPV rule. The table below illustrate the solution represention of a particle for PSO algorithm.

Table 3.1 Solution Representation of a Particle

j	1	2	3	4	5
X_{ij}^t	1.60	3.03	-1.01	-2.15	0.83
V_{ij}^t	-0.13	3.14	2.73	1.42	-2.11
S_{ij}^t	4	3	5	1	2

According to the SPV rule, the smallest position value is X_{i4}^t= -2.15, so the dimension j=4 is aasigned to be the first job π_{i1}^t= 4 in the permutation π_i^t, the second smallest position value is X_{i3}^t= -1.01 so the dimension j=2 is assigned to be the second job π_{i2}^t= 3 in the permutation π_i^t, and so on.

Step 7: Update the personal best,

The personal best position of each particle is updated using

$$P_i^t = \begin{cases} P_i^{t-1} & if \quad f(X_i^t) \geq f(P_i^{t-1}) \\ X_i^t & if \quad f(X_i^t) < f(P_i^{t-1}) \end{cases} \tag{3.2.10}$$

P_i^t represents the best position of the particle i with the best fitness until iteration t, so the best position associated with the best fitness value of the particle i obtained so far is called the personal best.

For each particle, the personal best is defined as;

$$\mathrm{P_i^t} = [\mathrm{p_{i1}^t}, \ \mathrm{p_{i2}^t}, \dots, \mathrm{p_{in}^t}] \tag{3.2.11}$$

where $\mathrm{P_{ij}^t}$ is the position of particle $\mathrm{i_{th}}$ personal best with respectto the $\mathrm{j_{th}}$ dimension.

Step 8: Update the global best,

And the global best position found so far in the swarm population is obtained as

$$G^t = \begin{cases} \arg\min_{\mathrm{P_i^t}} f(P_i^t) & if \quad \min f(P_i^t) < f(G^{t-1}) \\ G^{t-1} & else \end{cases} \quad 1 \leq i \leq NP \tag{3.2.12}$$

G^t denotes the best position of the globally best particle achieved so far in the whole swarm. The global best is then defined as

$$G^t = [g_1^t, \ g_2^t, \dots, g_n^t] \qquad (3.2.13)$$

where g_j^t is the position of particle i_{th} global best with respectto the j_{th} dimension.

Step 9: Stopping Criterion

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

3.2.1.1 The Pseudocode of The CPSO SPV Algorithm

CPSO{
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 or mutation (optional)
} While (Termination) }

Figure 3.1 The Pseudocode of PSO_{SPV}

3.2.2 Discrete Particle Swarm Optimization Algorithm for FSSP

Pan, Tasgetiren and Liang (2008) applied DPSO algorithm to both 110 benchmark instances of Taillard [Benchmarks] for no wait flow shop scheduling problem. We coded this algorithm and we applied this algorithm to the flow shop scheduling problems.

In DPSO, Pan et al.(2008) proposed a new position update method for particles based on discrete job permutations. Since the behavior of a particle is a compromise among three possible choices: to follow its own position (X_i^t), to go towards its personal best position (P_i^t) and to go towards the best position of the particle in the whole swarm population (G^t), the position of the particle at iteration t can be updated as follows:

$$X_i^t = c_2 \oplus F_3(c_1 \oplus F_2(w \otimes F_1(X_i^{t-1}), P_i^{t-1}), G^{t-1}) \quad 3.3.1$$

Note that permutation π_i^t corresponds to the particle X_i^t. The update equation consists of three components: the first component is $\lambda_i^t = w \otimes F_1(X_i^{t-1})$, which represents the velocity of the particle. In the component $\lambda_i^t = w \otimes F_1(X_i^{t-1})$, F_1 represents the mutation operator with the probability of *w*. In other words, a uniform random number r is generated between 0 and 1. If r is less than w then the mutation operator is applied to generate a perturbed permutation of the particle by $\lambda_i^t = F_1(X_i^{t-1})$ otherwise current permutation is kept as $\lambda_i^t = X_i^{t-1}$.

The second component is $\delta_i^t = c_1 \oplus F_2(\lambda_i^t, P_i^{t-1})$ which is the "*cognition*" part of the particle representing the private thinking of the particle itself. In the component $\delta_i^t = c_1 \oplus F_2(\lambda_i^t, P_i^{t-1})$, F_2 represents the crossover operator with the probability of c_1. Note that λ_i^t and P_i^{t-1} will be the first and second parents for the crossover operator, respectively. It results either in $\delta_i^t = F_2(\lambda_i^t, P_i^{t-1})$ or in $\delta_i^t = \lambda_i^t$ depending on the choice of a uniform random number.

The third component is $X_i^t = c_2 \oplus F_3(\delta_i^t, G^{t-1})$, which is the "*social*" part of the particle representing the collaboration among particles. In the component $X_i^t = c_2 \oplus F_3(\delta_i^t, G^{t-1})$, F_3 represents the crossover operator with the probability of c_2. Note that δ_i^t and G^{t-1} will be the first and second parents for the crossover operator, respectively. It results either in $X_i^t = F_3(\delta_i^t, G^{t-1})$ or in $X_i^t = \delta_i^t$ depending on the choice of a uniform random number.

To be employed in the position update equation, a new crossover operator, called PTL crossover, is proposed. The PTL crossover is able to produce a pair of different permutations even from two identical parents. An illustration of the two-cut PTL

crossover and the traditional one-cut crossover operator is shown in Figure 3.8.4 and 3.8.5.

In the PTL crossover, a block of jobs from the first parent is determined by two-cut points randomly. This block is either moved to the right or left corner of the permutation. Then the offspring permutation is filled out with the remaining jobs from the second parent. This procedure will always produce two distinctive offspring even from the same two parents as shown in Figure 3.8.4 and 3.8.5. In this study, one of these two unique offspring is chosen randomly with an equal probability.

3.2.2.1 The Pseudocode of The DPSO Algorithm

DPSO{
Initialize parameters
Initialize population
Evaluate
Do{
Find the personal best (P_i)
Find the global best (G_i)
Update position (X_i)
Evaluate population
Apply local search (optional)
}While (Not Termination)}

Figure 3.2 The Pseudocode of DPSO

3.2.3 Stochastically Perturbed PSO Algorithm for FSSP

In this algorithm, a stochastically perturbed particle swarm optimization algorithm (SPPSO) is proposed for the flow shop scheduling problems.

In proposed stochastically perturbed Particle Swarm Optimization algorithm (SPPSO), the initial population is generated randomly. Initially, each individual with its position, and fitness value is assigned to its personal best (i.e., the best value of each individual found so far). The best individual in the whole swarm with its position and fitness value, on the other hand, is assigned to the global best (i.e., the best particle in

the whole swarm). Then, the position of each particle is updated based on the personal best and the global best. These operations in SPPSO are similar to classical PSO algorithm. However, the search strategy of SPPSO is different. That is, each particle in the swarm moves based on the following equations.

$$S_1 = w^t \oplus \eta(X_i^t) \quad 3.4.1$$

$$w^{t+1} = w.\beta \quad 3.4.2$$

$$S_2 = c_1 \oplus \eta(P_i^t) \quad 3.4.3$$

$$S_3 = c_2 \oplus \eta(G^t) \quad 3.4.4$$

$$X_i^{t+1} = best(S_1; S_3; S_3) \quad 3.4.5$$

In this algorithm, a stochastically perturbed particle swarm optimization algorithm (SPPSO) is proposed for the flow shop scheduling problems.

At each iteration, the position vector of each particle, its personal best and the global best are considered. First of all, a random number of $U(0,1)$ is generated to compare with the inertia weight to decide whether to apply Insert function (η) to the particle or not.

If the random number chosen is less than the inertia weight, the particle is manipulated with this Insert function, and the resulting solution, say S_1, is obtained. Meanwhile, the inertia weight is discounted by a constant factor at each iteration, in order to tighten the acceptability of the manipulated particle for the next generation, that is, to diminish the impact of the randomly operated solutions on the swarm evolution.

The next step is to generate another random number of U(0,1) to be compared with c_1, cognitive parameter, to make a decision whether to apply Insert function to personal best of the particle considered. If the random number is less than c_1, then the personal best of the particle undertaken is manipulated and the resulting solution is spared as S_2.

Likewise, a third random number of U(0,1) is generated for making a decision whether to manipulate the global best with the Insert function. If the random number is

less than c_2, social parameter, then Insert is applied to the global best to obtain a new solution of S_3.

Unlike the case of inertia weight, the values of c_1 and c_2 factors are not increased or decreased iteratively, but are fixed at 0.5. That means the probability of applying Insert function to the personal and global bests remains the same.

The new replacement solution is selected among S_1, S_2 and S_3, based on their fitness values. This solution may not always be better than the current solution. This is to keep the swarm diverse. The convergence is traced by checking the personal best of each new particle and the global best.

As it is seen, proposed equations have all major characteristics of the classical PSO equations. The following pseudo-code describes in detail the steps of the SPPSO algorithm.

3.2.3.1 The Pseudocode of The SPPSO Algorithm

Our proposed SPPSO algorithm are given below:

```
Begin
        Initialize particles (population) randomly
        For each particle
        Calculate fitness value
        Set to position vector and fitness value as personal best (Pit)
        Select the best particle and its position vector as global best(Gt)
        End
Do{
        Update inertia weight
        For each particle
        Apply insert with the probability of inertia weight (s1)
        Apply insert to (Pit) with the probability of c1 (s2)
        Apply insert to (Gt) with the probability of c2 (s3)
        Select the best one among the s1,s2 and s3
        Update personal best (Pit)
        End
Update global best (Gt)
}While (Maximum Iteration is not reached)
End
```

Figure 3.3 Pseudo code of the proposed SPPSO algorithm

3.3 NEIGHBOURHOOD STRUCTURES

The performance of the metaheuristic algorithm significantly depends on the efficiency of the neighbourhood structure. The solutions are determined to move with the neighbourhood structure. In this study, the following three neighbourhood structures are employed.

3.3.1 Interchange Operator

Interchange function changes the place of two different randomly chosen jobs in each other in the sequence. In order to apply interchange function, we also need to derive two random numbers.

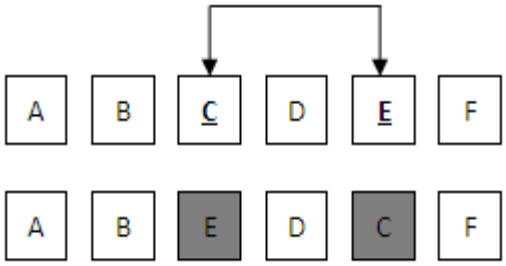

Figure 3.4 Interchange Operator

3.3.2 Insert Operator

Insert function (η) implies the insertion of a randomly chosen job in front (or back sometimes) of another randomly chosen job. In order to apply insert function, we also need to derive two random numbers; one is for determining the job to change place and the other is for the job in front of which the former job is to be inserted.

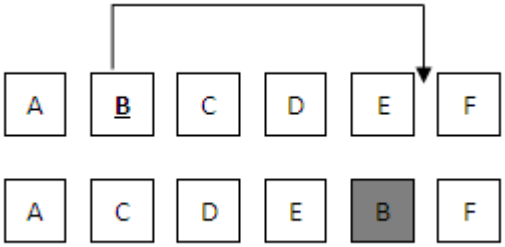

Figure 3.5 The Insert Operator

3.3.3 Swap Operator

Swap function changes the place of two neighbor randomly chosen jobs in each other in the sequence. In order to apply swap function, we also need to derive two random numbers. First random number determines the first job in the sequence, and then, second random number determines the left or right neighbor of job will be change.

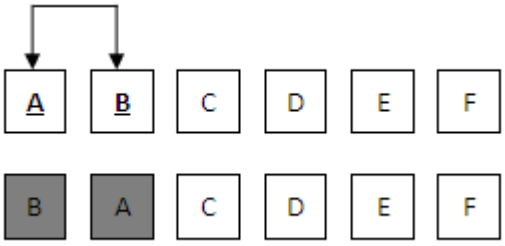

Figure 3.6 Swap Operator

CHAPTER 4

DIFFERENTIAL EVOLUTION ALGORITHM

4.1 LITERATURE REWIEV

Differential evolution (DE) is one of the latest evolutionary optimization methods proposed by Storn & Price (1995). They applied this heuristic approach for minimizing possibly nonlinear and non differentiable continuous space functions in 1995.

Differential Evolution(DE) is a parallel direct search method which utilizes NP D-dimensional parameter vectors as a population for each generation G. NP does not change during the minimization process. The initial vector population is chosen randomly and should cover the entire parameter space.

Like other evolutionary-type algorithms, DE is a population-based and stochastic global optimizer. In a DE algorithm, candidate solutions are represented by chromosomes based on floating-point numbers. In the mutation process of a DE algorithm, the weighted difference between two randomly selected population members is added to a third member to generate a mutated solution. Then, a crossover operator follows to combine the mutated solution with the target solution so as to generate a trial solution. Thereafter, a selection operator is applied to compare the fitness function value of both competing solutions, namely, target and trial solutions to determine who can survive for the next generation.

It has been succusfully applied in a variety of applications. DE's applications consist of aerodynamic design (Rogalsky et al. 2000), digital filter design (Storn, 1999), earthquake relocation (Ruzek & Kvasnicka 2001), microprocessor synthesis (Rae & Parameswaran 2001), neural network learning (Masters & Land 1997), permutation flowshop sequencing problems (Tasgetiren et. al. 2004), single machine tardiness

problem (Tasgetiren et. al. 2006), sequencing and scheduling problems (Andreas and Sotiris 2006).

To attain a better solution quality, modern metaheuristics have been recently presented for the PFSP with makespan / total flowtime minimization (Andreas & Omirou, 2006; Onwubolu & Davendra, 2006; Pan, Tasgetiren, & Liang, 2007; Tasgetiren, Liang, Sevkli, & Gencyilmaz, 2004b; Tasgetiren, Pan, Liang, & Suganthan, 2007b; Tasgetiren, Pan, Suganthan, & Liang, 2007c). Recent review of flowshop heuristics and metaheuristics can be found in Ruiz and Maroto (2005).

The applications of DE on combinatorial optimization problems are still considered limited, but the advantages of DE include a simple structure, immediately accessible for practical applications, ease of implementation, speed to acquire solutions, and robustness as demonstrated in the literature. However, the major obstacle of successfully applying a DE algorithm to solve combinatorial problems in the literature is due to its continuous nature. To remedy this drawback, this research proposes a discrete differential evolution (DDE) algorithm to solve the flowshop scheduling problem with the objective of minimizing total flowtime.

4.2 CONTINUOUS DIFFERENTIAL EVOLUTION ALGORITHM

DE algorithm, candidate solutions are represented by chromosomes based on floating-point numbers. In the mutation process of a DE algorithm, the weighted difference between two randomly selected population members is added to a third member to generate a mutated solution. Then, a crossover operator follows to combine the mutated solution with the target solution so as to generate a trial solution. Thereafter, a selection operator is applied to compare the fitness function value of both competing solutions, namely, target and trial solutions to determine who can survive for the next generation. This process is repeated until a convergence occurs.

As with all evolutionary optimization algorithms, DE works with a population of solutions, not with a single solution for the optimization problem. Population P of generation G contains NP solution vectors called individuals of the population and each vector represents potential solution for the optimization problem.

Step 1: Initialization: Firstly, The DE algorithm starts with initializing the initial target population with the size of NP. Target population are consist of individual vectors which are $X_i^t = [x_{i1}^t, \dots, x_{in}^t]$.

During each DE iteration, particle i adjusts its velocity V_{ij} and position vector

$$X^t = [X_1^t,\ X_2^t, \dots, X_n^t] \tag{4.1.1}$$

Step 2: Updating generation counter,

$$t = t + 1 \tag{4.1.2}$$

Step 3: Generate mutant population,

DE mutates vectors from the target population by adding the weighted difference between two randomly selected target population members to a third member at iteration t as follows:

$$V_{ij}^t = x_{aj}^{t-1} + F \times \left(x_{bj}^{t-1} - X_{cj}^{t-1} \right) \tag{4.1.2}$$

where *a*, *b*, and *c* are the three randomly chosen individuals from the target population such that $\left(a \neq b \neq b \in (1, \dots, NP)\right)$.

Step 4: Generate trial population,

Following the mutation phase, the crossover operator is applied to obtain the trial individual such that:

$$U_i^t = \left[u_{i1}^t,\ u_{i2}^t, \dots, u_{i,nm}^t\right] \tag{4.1.4}$$

$$u_{ik}^t = \begin{cases} v_{ik}^t\ , & if\ \ r_{ik}^t \leq CR\ or\ k = D_i \\ x_{ik}^{t-1}, & Otherwise \end{cases} \tag{4.1.5}$$

In other words, the trial individual is made up with some parameters of mutant individual, or at least one of the parameters randomly selected, and some other parameters of the target individual.

Note that; F and *CR* are differential evolution control parameters. Both values remain constant during the search process. Both values as well as the third control parameter, NP (population size), remain constant during the search process.

F is a mutation scale factor in range $[0.0,\ 1.0]$ that controls the amplification of differential variations. *F* affects the differential variation between two individuals.

CR is a real-valued crossover factor in the range $[0.0,\ 1.0]$ that controls the probability that a trial vector will be selected form the randomly chosen. r_{ik}^{t} is a uniform random number between 0 and 1. If generated random number is lower than *CR,* chose trial individual; otherwise select the target individual.

Generally, both *F* and *CR* affect the convergence rate and robustness of the search process. Their optimal values are dependent both on objective function characteristics and on the population size, *NP*. Usually, suitable values for *F*, *CR* and *NP* can be found by experimentation after a few tests using different values.

Step 5: Evaluate the trial population,

To decide whether or not the trial individual u_i^t should be a member of the target population for the next generation, it is compared to its counterpart target individual X_i^{t-1} at the previous generation.

Step 6: Do selection:

The selection is based on the survival of the fitness among the trial individual at the current iteration t and target individual at the previous iteration t-1 such that:

$$X_i^t = \begin{cases} U_i^t \ , & if\ f(U_i^t) \le f(X_i^{t-1}) \\ X_i^{t-1}, & Otherwise \end{cases} \qquad (4.1.6)$$

each individual of the temporary or trial population is compared with its counterpart in the current population. The one with the lower value of cost-function $f(X_i^t)$ to be minimized will propagate the population of the next generation.

Step 7: Stopping Criterion

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

Very recently, Tasgetiren et al. (2007b, 2007c) and Pan et al. (2007) proposed a simple and novel discrete DE (DDE) algorithm whose solutions are based on discrete job permutations.

We proposed a new DDE algorithm for flow shop scheduling problems.

4.2.1 The Standard Pseudocode of DE Algorithm

```
Begin
        Initialize parameters
        Initialize target population
        Evaluate target population

Do {

        Obtain mutant population
        Obtain trial population
        Evaluate trial population
        Make selection
Apply local search (optional)}
While (Not Termination)

End
```

Figure 4.1 The Standard Pseudocode of DE Algorithm

4.3 DISCRETE DIFFERENTIAL EVOLUTION ALGORITHM

The proposed DDE algorithm, the basic idea is to take advantage of the best solution obtained from the previous generation in the target population during the search procedure. Unlike its continuous counterpart, the differential variation is stochastically achieved through perturbation of the best solution from the previous generation in the target population. Then a crossover operator is used to obtain the trial individual to be compared to its counterpart individual in the target population.

In the DDE algorithm, the target population is constructed based on discrete flow shop permutation by $X^t = [X_1^t,\ X_2^t, \dots, X_n^t]$. For the mutant population the following equation are used:

$$V_{ij}^{t} = x_{aj}^{t-1} \oplus F \otimes \left(x_{bj}^{t-1} \ominus X_{cj}^{t-1} \right) \quad (4.2.1)$$

where *a*, *b*, and *c* are the three randomly chosen individuals from the target population such that $\left(a \neq b \neq b \in (1, \ldots, NP)\right)$. The mutant population consist of three components: the first component is $\tau_i^t = x_{bj}^{t-1} \ominus X_{cj}^{t-1}$, which represents the one cut operator is applied to generate a pair of different permutations even from two different parents.

The second component is $F \otimes \tau_i^t$, F represent the exchanged operator with probability of c_1. In other words, a uniform random number c_1 is generated between 0 and 1. If c_1 is less than F then the exchanged operator is applied to generate a perturbed permutation of the particle by $\lambda_i^t = F \otimes \tau_i^t$, otherwise current permutation is kept as $\lambda_i^t = \tau_i^t$.

The third component is $V_{ij}^t = x_{aj}^{t-1} \oplus \lambda_i^t$, which represents the two cut operator is applied to generate a pair og different permutations even from two different parents.

To be employed in the position update equation, a new crossover operator, called is proposed. An illustration of the two-cut crossover and the traditional one-cut crossover operator is shown in Figure 3.8.4 and 3.8.5.

The mutation phase, the crossover operators are applied to determine the trial individual such that;

$$u_{ik}^t = c_2 \odot CR\ (v_{ik}^t) \quad (4.2.2)$$

CR is a user defined crossover constant in the range [0.0, 1.0], and c_2 is a uniform random number between [0.0, 1.0]. If c_2 is less than *CR* then the insert operator ($\odot$) is applied to generate a perturbed permutation of the particle by V_i^t, otherwise current permutation is kept as V_i^t . Finally, the trial population is generated as follow:

$$U_i^t = \left[u_{i1}^t,\ u_{i2}^t, \ldots, u_{i,nm}^t\right] \quad (4.2.3)$$

The other step same with the classical DE. The selection is based on the survival of the fitness among the trial population and target population such that:

$$X_i^t = \begin{cases} U_i^t \ , & if\ f(U_i^t) \le f(X_i^{t-1}) \\ X_i^{t-1}, & Otherwise \end{cases} \qquad (4.2.4)$$

4.3.1 The Pseudocode Of Proposed DDE Algorithm

Our proposed discrete differential evolution algorithm are given below:

Begin
    Initialize parameters
    Initialize target population
    Evaluate target population
    End
Do{
    Obtain mutant population
    For each individual
      Apply one cut to b and c individual target (τ_i^t)
      Apply exchanged to (τ_i^t) with the probability of c_1 (λ_i^t)
      Apply two cut to (λ_i^t)
    Obtain trial population
    For each individual
      Apply insert to (V_i^t) with the probability of c_2
    Evaluate trial population
    Make selection
Apply local search(optimal)
}While (Not Termination)

Figure 4.2 The Pseudocode Of Proposed DDE Algorithm

4.4 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.

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.

4.4.1 One Cut

First parent chromosome is cut from one point randomly. This block is either moved to the right or left corner of the permutation. Then the offspring permutation is filled out with the remaining jobs from the second parent. This procedure will always produce two distinctive offspring even from the different two parents as shown below.

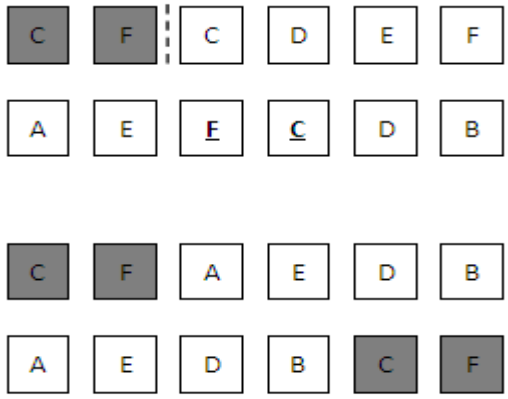

Figure 4.3 One Cut Operator

4.4.2 Two cut

First parent chromosome is cut from two different point randomly. This block is either moved to the right or left corner of the permutation. Then the offspring permutation is filled out with the remaining jobs from the second parent. This procedure will always produce two distinctive offspring even from the different two parents as shown below.

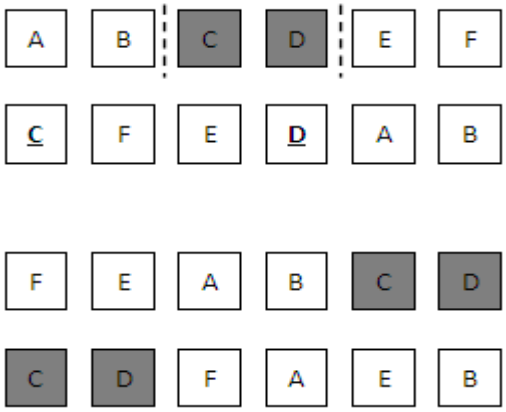

Figure 4.4 Two Cut Operator

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 INTRODUCTION

The objective of my study is to determine a sequence of jobs which processed through m machines in a permutation flow shop. The sequence should be arranged in order to minimize makespan. The problem is denoted as $Fm \mid\mid C_{max}$ 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 mixed integer 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 particle swarm optimization and differential evolution algorithm.

The algorithms were coded in Borland C++ programming language and runs were done on a Intel Core 2 Due Centrino 2.20 GHz computer with 2,00 GB memory. In this section, a comparison study is carried out on the effectiveness of the proposed SPPSO and DDE algorithm. SPPSO and DDE were exclusively tested in comparison with two other recently introduced PSO algorithms which are PSOspv algorithm of Tasgetiren et al. and DPSO algorithm of Pan et al. We coded these two algorithms.

For SPPSO, DPSO and PSOspv, the social and cognitive parameters were taken as $c_1 = c_2 = 0.5$, initial inertia weight is set to 0.9 and never decreased below 0.40, and the decrement factor β is fixed at 0.99999. The algorithms were run for 1000 iterations for each problem and 10 replications were carried out to collect the statistics for the performance measures. Mutant constant and crossover rate were taken as $F = 0.5$ $CR = 0.4$ at DDE algorithm. In PSOspv, 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.

Taillard (1993) have provided an extensive set of randomly generated test problems for minimizing makespan in flow shop. They have only provided 11 problem instances for makespan criterion for the flow shop problems. Each instance include 10 different problem set. A variety of problem sizes ranging from 20 to 200 jobs with 5 and 20 machines. Thus, were run for 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, and 200x20, problem sets, e.g., in 20x15 problem, 20 corresponds to the number of jobs and 10 corresponds to the number of machines.

In section 5.2, the performance of the algorihms were compared with respect to the deviation of the best, worst and average makespan achieved from the best known makespan and t-test result can be found in section 5.3.

5.2 RESULTS

The performance of the PSOspv, DPSO, SPPSO and DDE algorithms is evaluated by using the benchmark suite of Taillard (1993). The solution quality is measured with the mean percent relative increase in makespan (Δ) with respect to the upper bounds provided by Taillard (1993). To be more specific, Δ_{avg} is computed as follows:

$$\Delta_{avg} = \sum_{i=1}^{R} \left(\frac{(H_i - U_i) \times 100}{U_i} \right) / R$$

Where H_i denotes the value of the makespan that the PSOspv, DPSO, SPPSO and DDE algorithms generated, whereas U_i is the value of upper bounds for makespan provided by Taillard (1993), and R is the total number of replications, i.e., R is equal to the number of replications for each instance. For convenience, Δ_{min}, Δ_{max}, and Δ_{avg} denote the deviation of the best, worst and average makespan achieved from the best known makespan, Δ_{std} represents the average standard deviation of makespan. For the computational effort consideration, t_{avg} denotes the average CPU times over R runs in seconds.

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, Table 5.2 Table 5.3 and Table 5.4.

5.2.1 PSO_{SPV} Results

We did not use any local search operator in the PSO algorithm The SPV rule is used to convert a position vector to a job permutation. The algorithm found four optimal solutions in the problem sets (Table A.1). The PSO produces slightly better results if we compare with DPSO. The average CPU time for PSO_{SPV} was better than DPSO and DDE algorithm.

Table 5.1 Results of PSO_{SPV}

PSO_{SPV} FITNESS					
Problem set	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	t_{avg}
tai 20 X 5	0.93	2.73	4.99	16.40	7.25
tai 20 X 10	2.20	4.05	6.19	19.35	8.45
tai 20 X 20	1.55	3.33	5.52	26.96	8.49
tai 50 X 5	0.25	0.84	1.87	14.49	9.06
tai 50 X 10	3.67	4.93	6.29	25.16	9.40
tai 50 X 20	5.36	6.76	8.48	35.69	9.72
tai 100 X 5	0.21	0.45	0.92	12.38	14.13
tai 100 X 10	1.72	2.65	3.77	36.30	14.75
tai 100 X 20	6.31	7.43	9.15	54.95	15.87
tai 200 X 10	1.44	2.09	2.92	47.23	52.54
tai 200 X 20	5.87	7.05	8.90	102.27	56.92
AVRG	**2.68**	**3.85**	**5.36**	**35.56**	**18.78**

5.2.2 DPSO Results

One cut, two cut mutation operators and insert fuction were applied to the algorithm. The algorithm found three optimal solutions in the problem sets (Table A.2). The DPSO produces slightly worst results if we compare with the other entire algorithm. Use of local search increased the CPU time for the PSO_{SPV} algorithm from 18.78 to 76.87.

Table 5.2 Results of DPSO

DPSO FITNESS					
Problem set	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	t_{avg}
tai 20 X 5	0.59	1.72	3.49	10.94	72.43
tai 20 X 10	1.69	3.37	5.72	19.16	74.05
tai 20 X 20	1.67	3.03	5.06	24.47	73.85
tai 50 X 5	0.50	1.28	2.26	16.52	73.73
tai 50 X 10	4.78	6.05	7.95	29.78	74.03
tai 50 X 20	6.26	7.89	9.70	40.12	74.46
tai 100 X 5	0.29	0.70	1.18	15.18	77.54
tai 100 X 10	2.54	3.55	4.84	39.97	76.60
tai 100 X 20	7.16	8.58	10.07	55.84	81.84
tai 200 X 10	1.94	2.58	3.24	43.71	83.43
tai 200 X 20	7.29	8.25	9.22	65.69	83.57
AVRG	**3.16**	**4.27**	**5.70**	**32.85**	**76.87**

5.2.3 SPPSO Results

Insert operator was applied to the algorithm. The algorithm found twelve optimal solutions in the problem sets (Table A.3). The SPPSO produces slightly better results if we compare with the other entire algorithm. Use of local search did not increase the CPU time. In addition the average CPU is the best time in contrast to the other algorithms.

Table 5.3 Results of SPPSO

SPPSO FITNESS					
Problem set	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	t_{avg}
tai 20 X 5	0.08	0.80	1.45	6.60	7.70
tai 20 X 10	0.60	1.95	3.41	13.77	9.65
tai 20 X 20	0.75	1.87	3.24	17.40	10.29
tai 50 X 5	0.14	0.38	0.90	7.17	9.38
tai 50 X 10	1.89	3.19	4.44	24.49	9.59
tai 50 X 20	3.44	4.52	5.71	29.08	9.77
tai 100 X 5	0.14	0.28	0.49	6.74	10.79
tai 100 X 10	0.82	1.37	2.03	22.03	10.15
tai 100 X 20	3.34	4.33	5.42	40.40	12.34
tai 200 X 10	0.70	0.97	1.51	28.77	14.82
tai 200 X 20	3.02	3.86	4.64	58.98	21.21
AVRG	**1.36**	**2.14**	**3.02**	**23.22**	**11.43**

5.2.4 DDE Results

Two cut and one cut mutation operators were applied to the algorithm. The algorithm found fifteen optimal solutions in the problem sets (Table A.4). The DDE beat the SPPSO algoritm in the deviation of the worst makespan achieved from the best known makespan (Δ_{max}). The algorithm produces better solution if we compare with the PSO_{SPV} and DPSO algorithm. Using of mutation operators highly increased the CPU time. In addition the average CPU is the worst time in contrast to the other algorithms.

Table 5.4 Results of DDE

DDE FITNESS					
Problem set	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	t_{avg}
tai 20 X 5	0.34	1.02	1.58	5.59	9.39
tai 20 X 10	0.66	1.84	3.28	12.78	12.02
tai 20 X 20	0.50	1.65	3.27	18.75	14.38
tai 50 X 5	0.20	0.45	0.79	5.92	42.71
tai 50 X 10	2.06	3.01	4.02	18.80	56.16
tai 50 X 20	3.37	4.32	5.60	26.68	91.06
tai 100 X 5	0.05	0.29	0.52	8.06	311.66
tai 100 X 10	0.93	1.49	2.12	21.36	436.40
tai 100 X 20	3.48	4.23	5.05	31.22	631.40
tai 200 X 10	0.76	1.02	1.42	24.25	3136.35
tai 200 X 20	3.01	3.69	4.64	60.14	4213.96
AVRG	**1.40**	**2.09**	**2.94**	**21.23**	**814.14**

5.2.5 Graphical Representation of The Results

The proposed algorithms which are SPPSO and DDE slightly better than PSO_{SPV} and DPSO which are the Tasgetiren's and Pan's algorithms and have been coded by me. First three graph shows the deviation of the best, average and worst makespan achieved from the best known makespan and the last graph shows the average standard deviation of makespan. It can summarized from the graphs, SPPSO is the best algorithm. SPPSO work faster and more accurate than the other algorithms. On the other hand, in some problem instances, DDE reveal good performce than SPPSO. But, DDE have a disadvantage compared with SPPSO in CPU time. DDE becomes clumsy if problem size will increase. The graphical representation of results were given below.

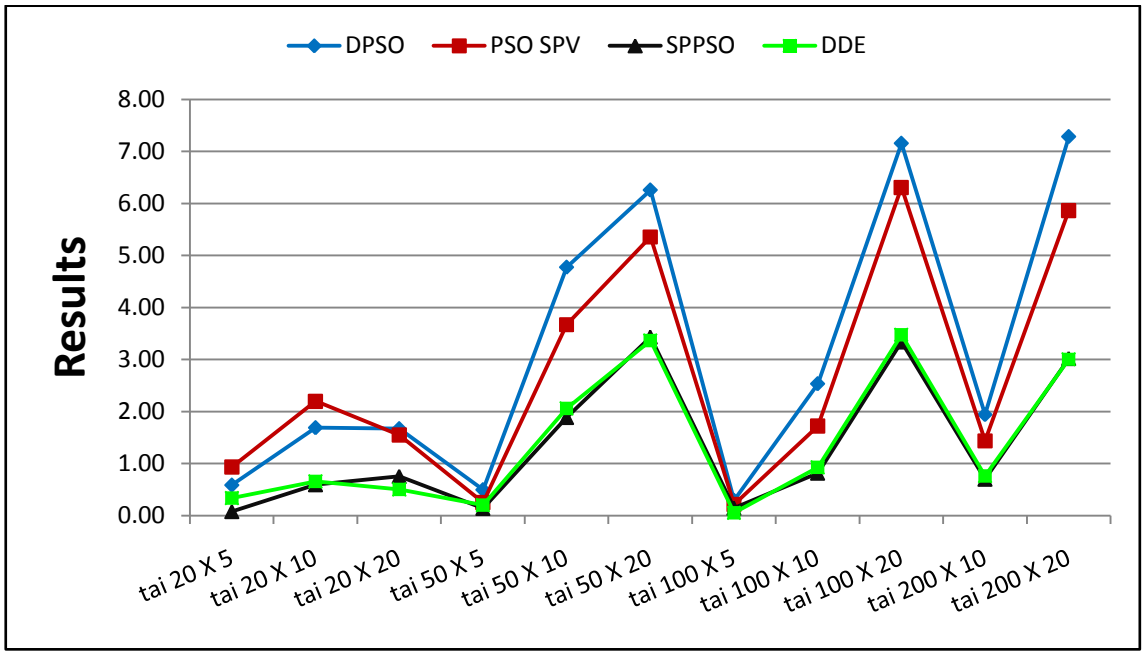

Figure 5.1 The deviation of the best makespan ($\Delta_{\min}$)

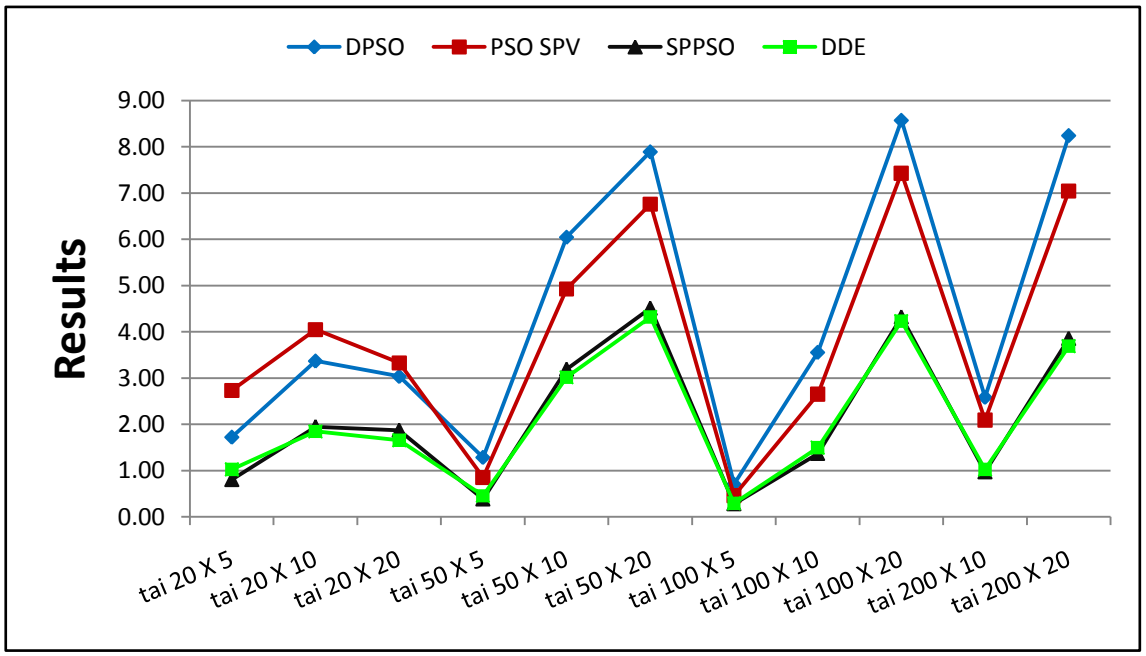

Figure 5.2 The deviation of the average makespan (Δ_{avg})

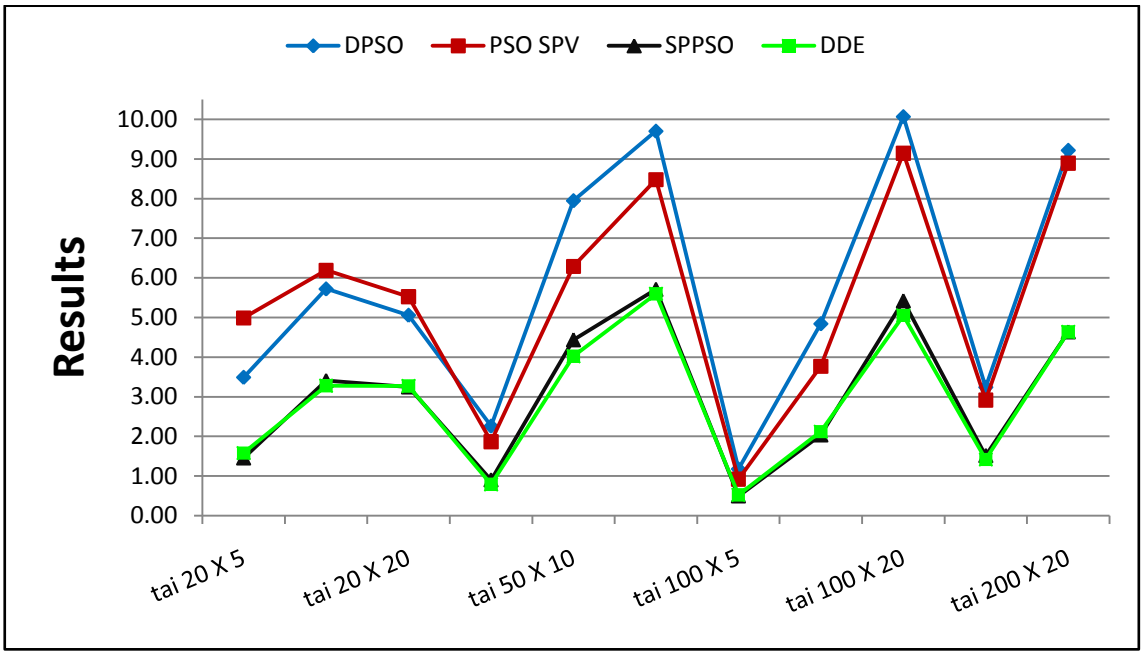

Figure 5.3 The deviation of the worst makespan ($\Delta_{\max}$)

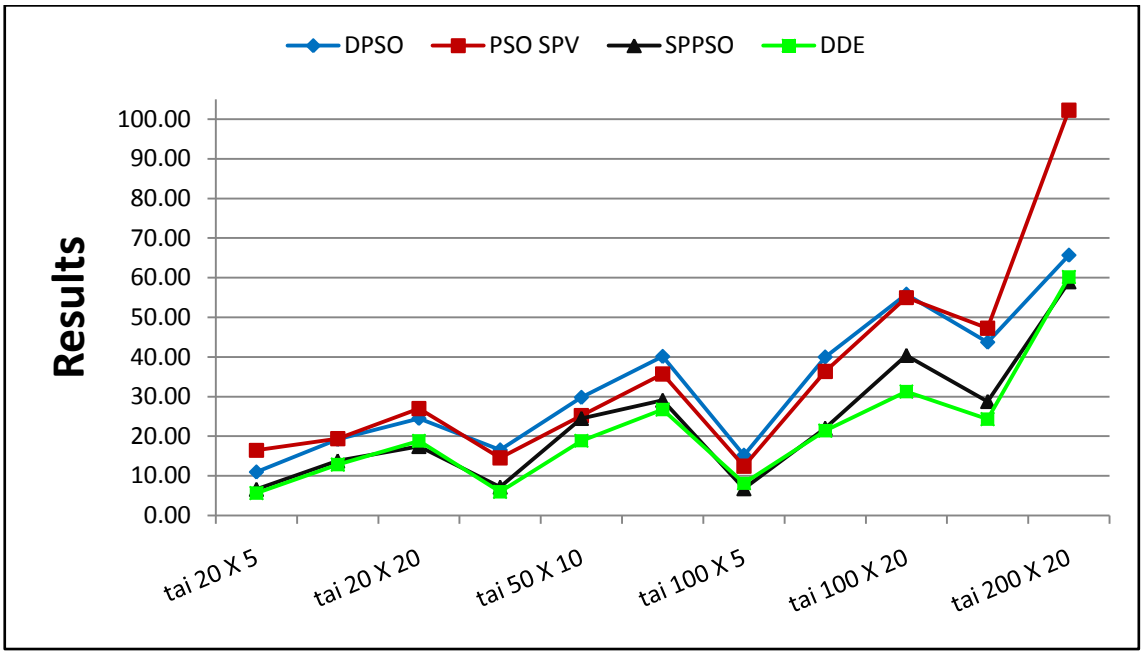

Figure 5.4 The average standard deviation of makespan (Δ_{std})

Table 5.5 Results of The Algorithms

	DPSO					PSO SPV					SPPSO					DDE				
problem set	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	t_{avg}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	t_{avg}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	t_{avg}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{std}	t_{avg}
tai 20 X 5	0,59	1,72	3,49	10,94	72,43	0,93	2,73	4,99	16,40	7,25	0,08	0,80	1,45	6,60	7,70	0,34	1,02	1,58	5,59	9,39
tai 20 X 10	1,69	3,37	5,72	19,16	74,05	2,20	4,05	6,19	19,35	8,45	0,60	1,95	3,41	13,77	9,65	0,66	1,84	3,28	12,78	12,02
tai 20 X 20	1,67	3,03	5,06	24,47	73,85	1,55	3,33	5,52	26,96	8,49	0,75	1,87	3,24	17,40	10,29	0,50	1,65	3,27	18,75	14,38
tai 50 X 5	0,50	1,28	2,26	16,52	73,73	0,25	0,84	1,87	14,49	9,06	0,14	0,38	0,90	7,17	9,38	0,20	0,45	0,79	5,92	42,71
tai 50 X 10	4,78	6,05	7,95	29,78	74,03	3,67	4,93	6,29	25,16	9,40	1,89	3,19	4,44	24,49	9,59	2,06	3,01	4,02	18,80	56,16
tai 50 X 20	6,26	7,89	9,70	40,12	74,46	5,36	6,76	8,48	35,69	9,72	3,44	4,52	5,71	29,08	9,77	3,37	4,32	5,60	26,68	91,06
tai 100 X 5	0,29	0,70	1,18	15,18	77,54	0,21	0,45	0,92	12,38	14,13	0,14	0,28	0,49	6,74	10,79	0,05	0,29	0,52	8,06	311,66
tai 100 X 10	2,54	3,55	4,84	39,97	76,60	1,72	2,65	3,77	36,30	14,75	0,82	1,37	2,03	22,03	10,15	0,93	1,49	2,12	21,36	436,40
tai 100 X 20	7,16	8,58	10,07	55,84	81,84	6,31	7,43	9,15	54,95	15,87	3,34	4,33	5,42	40,40	12,34	3,48	4,23	5,05	31,22	631,40
tai 200 X 10	1,94	2,58	3,24	43,71	83,43	1,44	2,09	2,92	47,23	52,54	0,70	0,97	1,51	28,77	14,82	0,76	1,02	1,42	24,25	3136,35
tai 200 X 20	7,29	8,25	9,22	65,69	83,57	5,87	7,05	8,90	102,27	56,92	3,02	3,86	4,64	58,98	21,21	3,01	3,69	4,64	60,14	4213,96
AVRG	**3,16**	**4,27**	**5,70**	**32,85**	**76,87**	**2,68**	**3,85**	**5,36**	**35,56**	**18,78**	**1,36**	**2,14**	**3,02**	**23,22**	**11,43**	**1,40**	**2,09**	**2,94**	**21,23**	**814,14**

5.3 STATISTICALLY TESTING

On the other hand, It is difficult to evaluate the performances only looking to these results. I performed statistical T-tests, to find out if one of the algorithms performed significantly better for a certain instance. Confidence intervals of 90%, 95% and 99.5% were used. I did the t-test when we want to compare two different independent observations. All of the algorithms were compared with each other by one by.

We established our hypothesis as;

$$H_0 : \mu_\alpha - \mu_\beta = 0$$
$$H_1 : \mu_\alpha - \mu_\beta > 0$$

α and β represent two different algorithms. Each algorithm was compared with the other algorithm by one by. The t-critical and t-observed values are evaluated and compared. If t-observed is greater than t-critical, we reject the null hypothesis. It means that there is a significant difference between the algorithms.

To test hypotheses about D_1: $\mu_1 - \mu_2$ when data is paired, the differences $D_1, D_2 \dots, D_n$ are formed and a one-sample t-test on the differences is carried out.

$$t_o = \frac{\bar{d}}{S_D/\sqrt{n}}$$

where $\bar{d}$ and S_D are the sample mean and the sample standard deviation, respectively. $t - critical = t_{\alpha,n-1}$ where α and $n - 1$ are the confidence level and the degrees of freedom respectively (n is 10 in our cases).

Table 5.6 Overall Fitness Results

PSO-DPSO		PSO-SPPSO		PSO-DDE		DPSO-SPPSO		DPSO-DDE		SPPSO-DDE	
PSO Better	82	**PSO Better**	0	**PSO Better**	0	**DPSO Better**	0	**DDE Better**	0	**SPPSO Better**	53
DPSO Better	28	**SPPSO Better**	110	**DDE Better**	110	**SPPSO Better**	110	**SPPSO Better**	110	**DDE Better**	56
PSO=DPSO	0	**PSO=SPPSO**	0	**PSO=DDE**	0	**DPSO=SPPSO**	0	**DPSO=DDE**	0	**SPPSO=DDE**	1
Best PSO	-2,41	**Best PSO**	0,01	**Best PSO**	0,00	**Best DPSO**	0,21	**Best DPSO**	0,13	**Best SPPSO**	-0,70
Best DPSO	2,91	**Best SPPSO**	4,06	**Best DDE**	3,82	**Best SPPSO**	4,79	**Best DDE**	4,88	**Best DDE**	0,86
Average	-0,39	**Average**	1,66	**Average**	1,71	**Average**	2,07	**Average**	2,12	**Average**	0,04

For the algorithms, as seen in this table, SPPSO and DDE outperform PSO_{SPV} and DPSO for each problem instances. The worst algorithm is DPSO. On the other hand there is no big difference between SPPSO and DDE.

Table 5.7 Average Percent Deviation Fitness Results

Problem Sets	PSO-DPSO	PSO-SPPSO	PSO-DDE	DPSO-SPPSO	DPSO-DDE	SPPSO-DDE
20X05	0,994	1,913	1,690	0,909	0,688	-0,219
20X10	0,659	2,058	2,163	1,391	1,495	0,104
20X20	0,287	1,427	1,649	1,138	1,359	0,219
50X05	-0,429	0,460	0,389	0,894	0,822	-0,071
50X10	-1,052	1,681	1,857	2,765	2,942	0,174
50X20	-1,044	2,152	2,345	3,234	3,429	0,190
100X05	-0,247	0,173	0,162	0,422	0,410	-0,011
100X10	-0,872	1,263	1,138	2,155	2,028	-0,123
100X20	-1,053	2,979	3,075	4,075	4,173	0,094
200X10	-0,478	1,108	1,054	1,595	1,541	-0,054
200X20	-1,107	3,072	3,240	4,226	2,198	0,163

For the PSO and DPSO comparison, it should be noted here that, a minus(-) sign in a table means that PSO performs better otherwise DPSO performs better for that occasion and for the other comparison it follows same procedures.

Table 5.8 T-test Values for The Algorithms

Problem Sets	PSO-DPSO	PSO-SPPSO	PSO-DDE	DPSO-SPPSO	DPSO-DDE	SPPSO-DDE
20X05	3,779	6,001	5,382	6,443	5,849	-4,786
20X10	3,913	11,060	12,848	9,484	18,080	0,896
20X20	2,489	11,256	14,449	8,315	12,541	2,297
50X05	-4,848	9,480	5,735	7,800	7,978	-1,527
50X10	-6,210	15,470	18,510	13,364	21,348	1,316
50X20	-4,818	17,581	16,298	14,734	15,241	2,048
100X05	-5,025	5,058	4,937	8,067	6,980	-0,385
100X10	-8,422	10,412	11,233	11,719	12,959	-3,145
100X20	-17,786	22,840	27,621	39,528	41,593	0,932
200X10	-4,057	11,420	8,880	7,684	7,037	-1,090
200X20	-12,811	34,158	27,081	69,463	48,248	2,564

You can see the t-values calculated for different problem sets and generation numbers. As known, t values are calculated as:

$$t_o = \frac{\bar{d}}{S_D/\sqrt{n}}$$

where $\bar{d}$ and S_D are the sample mean and the sample standard deviation, respectively.

5.3.1 Hypothesis Testing

The paired t-test determines whether they differ from each other in a significant way under the assumptions that the paired differences are independent and identically normally distributed. From the problem context, I identified the parameters of interest. The null hypothesis and alternative hypothesis were stated. Three different significance level α were chosen. The hypothesis is as follows;

$$H_0 : \text{Two algorithms are not significantly different}$$
$$H_1 : \text{One algorithm performs significantly better}$$

Table 5.9 T-test Critical Values for $N = 10$

0.1 (90%)	**0.05 (95%)**	**0.005 (99.5%)**
1,3830	1,8330	3,2500

In this table t-test critical values were given. Each observed t values were compared with these three different rejection criteria.

5.3.2 T-test Results for The Algorithms

Results are investigated with 90%, 95%, 99.5% levels of confidence. The SPPSO and DDE produced significantly better fitness results than PSO_{SPV} and DPSO for all problems for 90%, 95%, 99.5% level of confidence. If we compare the SPPSO with DDE, the algorithms are not significantly different.

Table 5.10 T-test Results for The Algorithms

Problem Sets	PSO-DPSO			PSO-SPPSO			PSO-DDE			DPSO-SPPSO			DPSO-DDE			SPPSO-DDE		
	90%	**95%**	**99,5%**	**90%**	**95%**	**99,5%**	**90%**	**95%**	**99,5%**	**90%**	**95%**	**99,5%**	**90%**	**95%**	**99,5%**	**90%**	**95%**	**99,5%**
20X05	DPSO	DPSO	DPSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO
20X10	DPSO	DPSO	DPSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	NS	NS	NS
20X20	DPSO	DPSO	NS	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	DDE	DDE	NS
50X05	PSO	PSO	PSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	NS	NS
50X10	PSO	PSO	PSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	NS	NS	NS
50X20	PSO	PSO	PSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	DDE	DDE	NS
100X05	PSO	PSO	PSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	NS	NS	NS
100X10	PSO	PSO	PSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	NS
100X20	PSO	PSO	PSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	NS	NS	NS
200X10	PSO	PSO	PSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	NS	NS	NS
200X20	PSO	PSO	PSO	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	SPPSO	SPPSO	SPPSO	DDE	DDE	DDE	DDE	DDE	NS

In this table, you can see the results of the hypothesis testing, that is if one algorithm performs significantly better or not. First row shows algorithms which are compared with each other. For example, PSO-DPSO column reveal that, t-observed values were computed and compared with t-critical values in three different confidence levels for these two algorithms and best algorithm for each problem instances were chosen.

NS: The algorithms are not significantly different

CHAPTER 6

CONCLUSION

In this study, a stochastically perturbed particle swarm optimization algorithm (SPPSO) and a discrete differential evolution algotihm (DDE) are proposed for flow shop scheduling problems (FSSP) and PSO_{SPV} and DPSO algorithms which developed by Tasgetiren and Pan et al. are coded. These four heuristic algorithms compared with each other. Taillard benchmark was used and the algorithms were applied flow shop scheduling problems to minimize the makespan of the jobs.

The SPPSO has all major characteristics of the classical PSO. However, the search strategy of SPPSO is different. The algorithm is applied to (FSSP) problem and compared with two recent PSO algorithms. It also should be noted that, since PSOspv considers each particle based on three key vectors; position (X_i), velocity (V_i), and permutation (π_i), it consumes more memory than SPPSO. In addition, since DPSO uses one and two cut crossover operators in every iteration, implementation of DPSO to combinatorial optimization problems is rather cumbersome. In DDE algorithm, the differential variation is stochastically achieved through perturbation of the best solution from the previous generation in the target population. The two cut and one cur crossover operators are used to obtain trial individual. The proposed algorithm can be applied to other combinatorial optimization problems such as flow shop scheduling, job shop scheduling etc. as future work.

It is concluded that SPPSO produced better results than DPSO and PSOspv in terms of number of optimum solutions obtained. In terms of the deviation of the best, worst and average makespan achieved from the best known makespan, SPPSO is slightly better than DPSO and PSOspv. However, PSOspv is better than DPSO.

The proposed DDE algorithm was produced better results than DPSO and PSO_{SPV} in term of the deviation of the best, worst and average makespan achieved from the best known makespan and in terms of number of optimum solutions obtained, on the other hand, the algorithm very cumbersome.

DDE algorithm presents better solution than SPPSO in terms of number of optimum solutions obtained and the deviation of the worst makespan achieved from the best known makespan. But SPPSO produces better results than DDE in term of the deviation of the best makespan achieved from the best known makespan. SPPSO is the fastest algorithm.

In statistically, Results are investigated with 90%, 95%, 99.5% levels of confidence. I prove the significant difference between the proposed algorithms (SPPSO, DDE) and the coded algorithms (PSO_{SPV}, DPSO). The SPPSO and DDE produced significantly better fitness results than PSO_{SPV} and DPSO for all problems for 90%, 95%, 99.5% level of confidence.

If we compare the SPPSO with DDE, the algorithms are not significantly different. Only, SPPSO algorithm beat DDE algorithm on 20X05 problem set for 90%, 95%, 99.5% level of confidence. In addition, DDE algorithms decelerates, if problem size increases, on the other hand, SPPSO progresses almost same. The cpu results of SPPSO is superior to DDE for data sets of 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, and 200x20.

REFERENCES

Al-Anzi F. S., Allahverdi A., A Self-Adaptive Differential Evolution Heuristic For Two-Stage Assembly Scheduling Problem To Minimize Maximum Lateness With Setup Times, European Journal of Operational Research, Vol. 182, pp: 80-94, 2007

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

Bolat, A., Al-Harkan, _., 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.

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

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

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 61

Kashan A.H., Karimi B., A Discrete Particle Swarm Optimization Algorithm For Scheduling Parallel Machines, Computers & Industrial Engineering, Vol. 56, pp: 216-22, 2009

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

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 62

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

Liao C.J., Tseng C.T., Luarn P., A Discrete Version Of Particle Swarm Optimization For Flowshop Scheduling Problems, Computers & Operations Research, Vol. 34, pp:3099-3111, 2007

Lobjerg, 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

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

Nearchou, A. C., A Differential Evolution Approach For The Common Due Date Early/Tardy Job Scheduling Problem, Computers & Operations Research, Vol. 35, pp: 1329-1343, 2008

Onwubolu, G., Davendra, D., Scheduling Flow Shops Using Differential Evolution Algorithm, European Journal of Operational Research Vol. 171, pp: 674- 692, 2006

Pan Q-K, Tasgetiren M. F, Liang Y.C., A Discrete Particle Swarm Optimization Algorithm for the No-Wait Flowshop Scheduling Problem with Makespan and Total Flowtime Criteria, Bio-inspired metaheuristics for combinatorial optimization problems. Special issue of Computers & Operations Research, 2005

Pan Q-K, Tasgetiren M. F, Liang Y-C., A Discrete Differential Evolution Algorithm for the permutation flowshop scheduling problem, Computer & Industrial Engineering, Vol. 55, pp: 795-816, 2008

Pan Q.K., Tasgetiren M. F., Liang Y.C., A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, European Journal of Operational Research, Vol. 177, pp: 1930-1947, 2007

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

Pinedo, M., Chao, X., Operations Scheduling: with Applications in Manufacturing and Services, Boston, Mass.: Irwin/McGraw-Hill, 1999 63

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

Price, K., An introduction to differential evolution. In: Corne, D., Dorigo, M., Glover Eds., New Ideas in Optimization. McGraw Hill, International UK, pp. 79–10, 1999

Qian B., Wang L., Hu R., Huang D.X., Wang X., A DE-based approach to no-wait flow-shop scheduling, Computers & Industrial Engineering, 2009

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

Storn, R. and Price, K., Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces, Technical Report TR-95-012, ICSI, 1995

Storn, R. and Price, K., Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Space, Journal of Global Optimization, vol. 11, pp. 341-359 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

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

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 and Differential Evolution Algorithms for the Single Machine Total Weighted Tardiness Problem, 2005

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

Tasgetiren M. F, Pan Q.K, Suganthan P.N, Liang Y.C., A Discrete Differential Evolution Algorithm for the No-Wait Flowshop Scheduling Problem with Total Flowtime Criterion. IEEE Symposium on Computational Intelligence in Scheduling, 2007

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

Wang L., Pan Q.K., Suganthan P.N., Wang W.H., Wang Y.M., A Novel Hybrid Discrete Differential Evolution Algorithm For Blocking Flow Shop Scheduling Problems, Computers & Operations Research, 2008

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

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

Zhang C., Ning J., Lu S., Ouyang D., Ding T., A Novel Hybrid Differential Evolution And Particle Swarm Optimization Algorithm For Unconstrained Optimization, Operations Research Letters, Vol. 37, pp: 117-12, 2009

Ziaee M., Sadjadi S.J., Mixed binary integer programming formulations for the flow shop scheduling problems. A case study: ISD projects scheduling, Applied Mathematics and Computation, Vol. 185, pp: 218-228, 2007

Zobolas G.I., Tarantilis C.D., Ioannou G., Minimizing Makespan in Permutation Flow Shop Scheduling Problems Using a Hybrid Metaheuristic Algorithm, Computers & Operations Research, Vol. 36, pp: 1249-1267, 2009

APPENDIX A

RESULTS OF THE ALGORITHMS

Table A.1 Detailed Results of PSO_{SPV}

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 20 X 5	1297	1297	1297	1324	1315	1339	1297	1278	1297	1339
	1366	1368	1373	1368	1366	1366	1383	1377	1373	1373
	1132	1132	1143	1132	1102	1120	1162	1098	1140	1163
	1309	1331	1315	1359	1315	1331	1315	1357	1309	1316
	1244	1250	1250	1244	1283	1250	1250	1265	1244	1291
	1210	1272	1217	1219	1272	1258	1233	1258	1224	1224
	1257	1276	1277	1276	1259	1251	1276	1251	1282	1256
	1254	1251	1268	1217	1231	1249	1283	1268	1253	1251
	1257	1255	1255	1247	1255	1287	1261	1263	1255	1253
	1127	1127	1153	1151	1127	1131	1112	1131	1138	1161
tai 20 X 10	1653	1627	1671	1646	1628	1644	1633	1637	1625	1619
	1701	1733	1709	1740	1704	1745	1737	1750	1761	1745
	1542	1536	1557	1569	1536	1577	1545	1528	1547	1556
	1492	1414	1491	1481	1423	1470	1418	1435	1445	1415
	1477	1474	1513	1498	1526	1489	1534	1504	1483	1429
	1429	1431	1443	1475	1436	1433	1450	1434	1443	1435
	1502	1549	1566	1558	1540	1531	1554	1529	1536	1520
	1596	1590	1606	1627	1580	1582	1607	1604	1613	1581
	1648	1676	1653	1654	1639	1638	1660	1654	1644	1652
	1650	1642	1657	1672	1687	1631	1689	1650	1646	1637
tai 20X 20	2403	2422	2384	2405	2403	2475	2389	2370	2335	2329
	2154	2129	2160	2152	2165	2180	2169	2175	2165	2210
	2359	2388	2353	2378	2390	2372	2446	2413	2394	2411
	2275	2251	2312	2307	2252	2272	2262	2297	2264	2306
	2376	2364	2382	2372	2367	2371	2401	2367	2348	2413
	2278	2280	2346	2330	2327	2270	2271	2308	2262	2280
	2329	2354	2390	2337	2361	2372	2330	2378	2364	2352
	2249	2257	2264	2291	2291	2279	2290	2301	2268	2241
	2292	2310	2379	2326	2291	2293	2304	2319	2269	2352
	2259	2245	2314	2186	2266	2218	2254	2267	2263	2247

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 50 X 5	2729	2729	2752	2735	2735	2752	2724	2745	2752	2748
	2898	2882	2882	2838	2882	2882	2863	2882	2853	2882
	2637	2648	2624	2653	2634	2626	2631	2696	2622	2665
	2777	2843	2777	2782	2777	2777	2782	2770	2784	2770
	2864	2864	2891	2864	2864	2891	2891	2900	2891	2897
	2856	2852	2835	2835	2835	2849	2832	2832	2857	2832
	2776	2746	2743	2745	2743	2736	2735	2736	2746	2766
	2707	2707	2704	2713	2710	2704	2707	2705	2700	2705
	2606	2571	2584	2583	2564	2583	2565	2578	2579	2588
	2796	2792	2791	2792	2818	2832	2784	2782	2783	2783
tai 50 X 10	3189	3143	3135	3181	3163	3150	3131	3165	3161	3155
	3020	2983	3003	2994	3049	3012	3024	2998	3047	3013
	3038	3006	3014	2971	2986	3003	2987	2994	3014	2986
	3208	3143	3170	3169	3218	3177	3170	3149	3131	3179
	3155	3156	3143	3143	3102	3141	3172	3143	3105	3083
	3152	3115	3180	3200	3160	3130	3135	3155	3138	3169
	3227	3251	3276	3192	3217	3254	3271	3231	3268	3217
	3165	3200	3164	3179	3166	3115	3170	3127	3107	3131
	3077	3021	3093	3079	3042	3024	3032	3032	3025	3057
	3245	3210	3209	3234	3217	3255	3191	3273	3202	3254
tai 50 X 20	4093	4066	4027	4101	4108	4056	4103	4089	4152	4080
	3934	3992	3945	3954	3974	3893	3940	3933	3921	3973
	3899	3971	3910	3901	3888	3883	3875	3919	3962	3904
	3974	3956	3889	3957	4014	3958	3997	3984	3945	3956
	3886	3836	3897	3861	3940	3876	3933	3859	3827	3897
	3889	3886	3893	3987	3905	3957	3973	3895	3953	3930
	3971	3954	3948	3892	4039	3959	3914	3936	3935	3918
	4025	3923	3962	4003	3947	3913	4082	3971	3971	3938
	3979	4018	3948	4016	3978	4060	4009	3980	4050	3969
	4012	3941	3947	3998	3967	4001	4000	3953	3959	3982
tai 100 X 5	5495	5495	5495	5503	5495	5527	5505	5505	5495	5495
	5284	5290	5290	5290	5283	5289	5284	5290	5290	5296
	5213	5216	5219	5193	5219	5213	5219	5221	5213	5213
	5035	5044	5044	5035	5027	5023	5044	5023	5035	5032
	5275	5268	5255	5255	5255	5253	5265	5311	5255	5255
	5187	5150	5135	5146	5152	5135	5139	5146	5146	5139
	5305	5276	5262	5260	5261	5305	5264	5305	5305	5264
	5184	5137	5127	5108	5106	5122	5106	5105	5124	5105
	5504	5473	5484	5489	5472	5484	5467	5473	5504	5484
	5346	5346	5346	5346	5342	5342	5346	5346	5346	5342

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 100 X 10	5999	5905	5893	5931	5918	5866	5958	5923	5950	5920
	5572	5440	5458	5465	5476	5506	5459	5505	5468	5552
	5825	5787	5783	5781	5771	5719	5764	5840	5778	5757
	6058	6008	5982	5955	5985	6020	6003	5982	5984	5942
	5714	5715	5645	5630	5746	5645	5678	5644	5690	5675
	5427	5437	5432	5414	5415	5417	5394	5345	5438	5455
	5788	5691	5696	5734	5687	5737	5720	5734	5733	5740
	5838	5755	5778	5735	5761	5796	5836	5825	5793	5762
	6024	5979	5977	6008	6000	5979	6024	5979	6029	6023
	6059	5915	5965	6069	5903	5935	5931	5960	5983	5903
tai 100 X 20	6727	6615	6691	6783	6659	6672	6677	6747	6602	6629
	6802	6597	6691	6654	6696	6613	6652	6750	6718	6634
	6703	6728	6727	6719	6794	6744	6698	6800	6743	6746
	6700	6657	6728	6688	6700	6704	6682	6676	6623	6628
	6858	6768	6775	6713	6695	6752	6765	6776	6707	6742
	6949	6716	6742	6850	6806	6901	6789	6782	6824	6811
	6897	6739	6724	6853	6747	6827	6798	6717	6718	6757
	7086	6881	6907	6826	6904	6847	6940	6932	6889	6870
	6825	6704	6756	6681	6775	6732	6707	6720	6809	6722
	6998	6844	6782	6829	6883	6798	6852	6915	6866	6861
tai 200 X 10	11139	10992	11079	11027	11027	10999	11079	11041	11069	11054
	10834	10867	10731	10771	10771	10835	10801	10751	10752	10692
	11243	11132	11159	11112	11118	11123	11160	11137	11156	11131
	11010	11057	10977	11005	11015	10999	10954	11057	10984	11005
	10809	10735	10754	10771	10714	10720	10757	10679	10764	10787
	10671	10528	10561	10578	10501	10594	10602	10564	10470	10512
	11090	11047	11110	11189	11025	11091	11084	11093	11060	11114
	11118	10927	10943	10927	11039	10985	10886	10948	10972	11018
	10737	10650	10584	10637	10708	10623	10676	10651	10689	10641
	10975	10980	10958	10899	10906	10884	10839	10850	10938	10850
tai 200 X 20	12123	11885	11886	11973	11942	11815	12031	11756	11766	11869
	12231	12232	12007	12027	11910	12113	12112	12016	12040	11998
	12333	12213	12082	11975	11983	12079	12081	12102	12081	12014
	12210	12205	12105	12073	12018	12038	12112	11963	12075	12067
	12121	11983	11924	11963	11940	11895	11899	11889	12172	12052
	12122	12020	11945	12036	11871	12020	12079	11820	12004	11967
	12327	12160	12196	12152	12125	12100	12075	12144	12108	12167
	12358	12061	12149	12116	12036	12063	12146	11978	12067	12071
	12343	11948	11992	12093	12074	12147	12099	11897	12071	11992
	12437	12052	11983	12074	12194	12200	12044	12021	12092	12125

Table A.2 Detailed Results of DPSO

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 20 X 5	1297	1297	1297	1297	1297	1297	1297	1297	1297	1297
	1366	1360	1367	1373	1366	1365	1366	1365	1370	1366
	1100	1098	1099	1098	1088	1098	1127	1098	1098	1100
	1309	1309	1301	1315	1325	1301	1318	1345	1315	1340
	1250	1248	1244	1250	1244	1250	1315	1243	1271	1252
	1219	1224	1210	1210	1226	1219	1233	1210	1210	1210
	1251	1251	1266	1251	1257	1257	1251	1274	1242	1257
	1239	1213	1211	1221	1227	1249	1224	1250	1206	1264
	1253	1258	1255	1236	1258	1255	1247	1261	1258	1258
	1124	1144	1127	1151	1145	1108	1127	1142	1136	1108
tai 20 X 10	1627	1611	1619	1607	1637	1607	1669	1647	1650	1607
	1737	1722	1710	1740	1759	1699	1689	1703	1715	1698
	1533	1540	1535	1542	1530	1525	1534	1554	1574	1532
	1424	1398	1430	1427	1447	1398	1427	1433	1420	1429
	1478	1506	1504	1479	1439	1433	1480	1486	1482	1475
	1434	1450	1433	1451	1435	1457	1432	1440	1463	1444
	1522	1507	1538	1494	1531	1554	1603	1512	1539	1538
	1586	1601	1612	1608	1559	1568	1585	1609	1583	1653
	1636	1664	1672	1639	1631	1636	1638	1669	1641	1636
	1649	1654	1656	1633	1628	1645	1625	1630	1637	1639
tai 20X 20	2369	2347	2351	2347	2433	2410	2403	2381	2374	2344
	2171	2168	2132	2130	2152	2147	2165	2128	2197	2180
	2351	2444	2377	2425	2351	2413	2408	2400	2381	2410
	2282	2277	2247	2297	2291	2303	2301	2253	2263	2288
	2336	2355	2381	2381	2445	2354	2365	2339	2388	2389
	2280	2285	2280	2259	2354	2269	2262	2265	2265	2306
	2347	2329	2320	2347	2361	2367	2344	2335	2339	2318
	2246	2242	2267	2275	2242	2249	2266	2257	2259	2263
	2282	2348	2341	2317	2348	2304	2289	2311	2315	2298
	2229	2242	2258	2316	2253	2280	2217	2234	2238	2243

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 50 X 5	2774	2729	2735	2735	2774	2745	2740	2735	2774	2745
	2882	2863	2863	2882	2882	2863	2882	2853	2848	2871
	2655	2647	2667	2687	2624	2655	2667	2646	2648	2671
	2822	2782	2770	2788	2843	2850	2782	2801	2776	2782
	2887	2922	2904	2894	2887	2891	2891	2887	2891	2908
	2874	2873	2832	2853	2843	2839	2855	2874	2839	2876
	2767	2791	2761	2756	2758	2781	2812	2755	2758	2756
	2707	2707	2723	2720	2707	2707	2733	2722	2725	2707
	2565	2599	2606	2604	2605	2602	2580	2611	2604	2614
	2818	2786	2783	2815	2802	2789	2830	2802	2792	2802
tai 50 X 10	3187	3163	3208	3196	3172	3192	3197	3190	3207	3207
	3054	3071	3117	3017	3035	3065	3045	3033	3070	3054
	3037	3041	3083	3048	3135	3048	3033	3168	3101	3047
	3192	3178	3216	3263	3203	3220	3238	3178	3199	3201
	3125	3157	3133	3169	3177	3165	3141	3186	3169	3121
	3167	3177	3199	3181	3152	3177	3145	3150	3233	3170
	3234	3289	3354	3271	3241	3253	3289	3289	3247	3239
	3163	3177	3144	3187	3216	3165	3206	3209	3216	3154
	3031	3051	3145	3025	3090	3065	3052	3051	3032	3060
	3290	3260	3291	3244	3245	3273	3307	3221	3194	3249
tai 50 X 20	4197	4147	4143	4044	4135	4095	4155	4119	4172	4106
	3987	3977	3992	4063	3991	3976	3970	3997	4016	3943
	3932	3923	3926	3894	3980	3907	3923	3905	3889	3907
	3952	3973	4058	3949	3963	4045	4054	4008	4052	4007
	3924	3914	4004	4032	3928	3909	3961	3896	3968	4021
	3998	3952	3977	4035	3931	3977	3969	3876	3972	3983
	3989	4004	4090	4085	4015	3998	4033	4037	3987	3966
	4018	3977	3963	3948	4031	4011	4099	3924	4011	4032
	3958	3986	3955	3986	4045	4010	3974	4039	3975	4000
	4014	4005	4083	4036	4103	4007	4071	4087	4080	3983
tai 100 X 5	5529	5529	5527	5527	5504	5529	5538	5493	5495	5500
	5284	5290	5302	5296	5316	5284	5316	5290	5327	5290
	5219	5220	5219	5221	5221	5221	5219	5221	5219	5229
	5029	5044	5044	5044	5030	5044	5035	5032	5044	5057
	5267	5305	5294	5257	5312	5258	5270	5255	5267	5272
	5139	5146	5150	5161	5161	5150	5156	5146	5150	5146
	5304	5305	5305	5262	5264	5305	5304	5316	5333	5263
	5133	5134	5137	5148	5134	5108	5141	5141	5140	5137
	5487	5492	5538	5498	5504	5510	5504	5492	5467	5504
	5386	5346	5383	5383	5386	5411	5342	5374	5345	5422

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 100 X 10	6011	6001	5989	6002	6025	5972	5943	5992	5955	6036
	5546	5498	5585	5526	5531	5506	5615	5626	5494	5554
	5871	5811	5866	5923	5858	5855	5855	5868	5815	5859
	6036	5987	6056	6053	6035	6005	6121	6048	5984	6004
	5745	5773	5746	5758	5725	5793	5773	5670	5674	5766
	5372	5462	5574	5460	5421	5519	5456	5433	5474	5513
	5734	5814	5781	5759	5772	5886	5810	5754	5741	5811
	5868	5834	5812	5819	5846	5822	5894	5828	5831	5776
	6055	6014	6038	5995	6068	6012	5979	5979	6022	6021
	5983	5951	6069	5981	6068	5938	5997	5966	5981	5946
tai 100 X 20	6736	6663	6712	6673	6749	6838	6826	6758	6704	6698
	6635	6754	6682	6761	6771	6830	6742	6775	6739	6676
	6798	6804	6815	6864	6771	6782	6683	6764	6879	6939
	6724	6705	6776	6792	6819	6776	6848	6750	6795	6756
	6807	6825	6896	6837	6848	6846	6754	6806	6831	6856
	6866	6871	6987	6751	6924	6923	6852	6860	6913	6969
	6812	6835	6848	6846	6811	6886	6833	6947	6868	6835
	6889	6955	6928	7035	6985	6988	7040	6912	7009	6984
	6787	6853	6935	6746	6862	6762	6757	6816	6896	6900
	6899	6835	6931	6900	6887	6912	7043	6932	7040	6912
tai 200 X 10	11026	11075	11040	11086	11092	11027	11066	11077	11039	11037
	10940	10839	10929	10832	10841	10816	10920	10901	10851	10869
	11146	11168	11234	11182	11187	11171	11175	11126	11175	11172
	10966	11010	11010	11010	11014	11035	11074	10999	11062	11005
	10863	10836	10809	10818	10768	10833	10946	10897	10916	10946
	10655	10624	10658	10610	10678	10591	10585	10691	10613	10632
	11077	11195	11112	11081	11043	11129	11164	11153	11057	11129
	10906	11041	10998	11099	11056	11029	11036	11055	10992	11044
	10708	10725	10721	10670	10783	10883	10771	10798	10720	10814
	10984	10993	10874	10941	10939	10877	10896	10931	10967	10860
tai 200 X 20	11930	12088	12076	12127	12025	12033	12056	12109	12212	12096
	12185	12270	12353	12239	12236	12201	12324	12144	12188	12181
	12177	12246	12253	12238	12309	12273	12144	12180	12230	12269
	12105	12174	12271	12211	12196	12213	12247	12182	12185	12097
	12184	12165	12031	12064	12098	12105	12274	12176	12135	12193
	12127	12178	12106	12130	12087	12085	12222	12180	12163	12174
	12284	12161	12464	12350	12313	12321	12313	12328	12253	12282
	12225	12306	12240	12095	12059	12279	12166	12169	12328	12252
	12133	12112	12114	12242	12117	12048	12206	12135	12023	12190
	12183	12171	12317	12248	12305	12176	12295	12345	12282	12254

Table A.3 Detailed Results of SPPSO

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 20 X 5	1297	1297	1278	1297	1297	1278	1297	1297	1278	1297
	1362	1360	1366	1365	1366	1360	1365	1360	1360	1366
	1098	1081	1087	1098	1088	1088	1081	1088	1085	1088
	1315	1301	1308	1301	1300	1297	1309	1297	1299	1309
	1250	1244	1235	1235	1235	1250	1235	1250	1243	1250
	1210	1210	1210	1195	1210	1210	1210	1210	1210	1195
	1251	1251	1251	1251	1251	1251	1251	1251	1251	1239
	1206	1211	1211	1214	1208	1217	1214	1220	1206	1217
	1236	1230	1230	1230	1255	1255	1255	1261	1261	1255
	1108	1120	1113	1127	1108	1111	1108	1108	1108	1112
tai 20 X 10	1612	1617	1614	1619	1619	1586	1618	1605	1597	1607
	1704	1695	1702	1694	1702	1709	1718	1691	1698	1705
	1508	1506	1538	1532	1517	1510	1551	1547	1542	1532
	1443	1396	1381	1397	1397	1405	1429	1389	1412	1414
	1440	1423	1449	1440	1430	1468	1450	1434	1465	1465
	1408	1414	1400	1424	1407	1425	1411	1439	1426	1436
	1515	1502	1499	1494	1504	1487	1494	1494	1509	1488
	1592	1569	1609	1568	1557	1600	1575	1566	1555	1579
	1630	1594	1624	1612	1636	1630	1636	1612	1648	1615
	1624	1619	1640	1612	1606	1615	1636	1616	1637	1614
tai 20X 20	2367	2369	2336	2305	2358	2376	2351	2391	2343	2323
	2120	2141	2149	2131	2135	2113	2129	2120	2155	2144
	2359	2343	2370	2368	2372	2390	2401	2345	2374	2374
	2276	2248	2248	2229	2252	2260	2305	2240	2252	2235
	2352	2298	2318	2308	2323	2345	2319	2352	2351	2345
	2263	2274	2264	2273	2261	2280	2293	2279	2257	2260
	2309	2312	2298	2313	2316	2345	2306	2319	2305	2295
	2245	2245	2228	2221	2250	2256	2214	2273	2246	2252
	2260	2274	2263	2280	2286	2309	2262	2283	2264	2259
	2252	2242	2248	2224	2210	2234	2217	2226	2205	2230

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 50 X 5	2724	2729	2729	2724	2752	2729	2724	2724	2724	2729
	2863	2847	2848	2863	2863	2853	2882	2882	2863	2848
	2624	2624	2641	2624	2644	2624	2624	2624	2630	2622
	2762	2770	2782	2770	2782	2762	2762	2762	2762	2762
	2864	2864	2864	2887	2864	2864	2864	2864	2864	2864
	2829	2832	2833	2839	2836	2835	2835	2835	2831	2831
	2732	2732	2732	2725	2725	2732	2725	2725	2732	2732
	2704	2704	2686	2704	2705	2694	2707	2707	2704	2707
	2564	2561	2583	2564	2576	2564	2564	2561	2564	2568
	2782	2782	2782	2782	2802	2782	2782	2782	2782	2782
tai 50 X 10	3131	3108	3126	3062	3167	3144	3108	3122	3162	3083
	2975	3012	2947	2975	2964	2963	2961	2948	2952	2914
	2975	2984	2961	2932	2961	2984	2918	2884	2968	2953
	3111	3129	3158	3130	3110	3136	3127	3112	3103	3103
	3071	3099	3060	3067	3119	3135	3093	3051	3117	3050
	3114	3114	3115	3127	3083	3092	3078	3091	3114	3083
	3148	3203	3196	3182	3165	3201	3184	3165	3191	3176
	3062	3094	3061	3072	3058	3080	3084	3131	3088	3062
	3032	3025	3025	3017	2965	3025	3017	2964	3005	3014
	3204	3182	3172	3136	3143	3203	3148	3162	3155	3195
tai 50 X 20	3965	4002	3980	4000	4053	4003	3993	3974	4009	3988
	3903	3910	3901	3891	3814	3842	3929	3909	3931	3817
	3761	3878	3779	3859	3819	3869	3830	3817	3768	3756
	3902	3839	3890	3916	3893	3877	3834	3842	3882	3874
	3819	3750	3771	3828	3806	3805	3794	3781	3764	3777
	3901	3817	3825	3837	3837	3850	3875	3830	3832	3825
	3865	3866	3872	3828	3881	3868	3862	3844	3828	3825
	3897	3846	3872	3899	3848	3834	3893	3869	3934	3943
	3928	3903	3892	3904	3895	3919	3934	3920	3938	3909
	3920	3900	3912	3899	3952	3945	3953	3892	3924	3952
tai 100 X 5	5493	5493	5493	5495	5493	5493	5493	5495	5493	5493
	5289	5284	5290	5290	5290	5284	5290	5290	5290	5284
	5193	5179	5221	5193	5193	5193	5213	5193	5193	5193
	5023	5023	5035	5023	5023	5023	5029	5023	5044	5044
	5255	5255	5255	5255	5255	5255	5255	5255	5255	5255
	5139	5139	5139	5146	5139	5139	5139	5139	5139	5139
	5284	5261	5263	5259	5259	5276	5259	5263	5263	5263
	5106	5108	5137	5106	5106	5106	5106	5127	5106	5106
	5454	5454	5484	5484	5484	5484	5454	5484	5454	5454
	5346	5342	5346	5346	5346	5346	5342	5346	5342	5328

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 100 X 10	5857	5836	5836	5909	5827	5795	5817	5836	5836	5825
	5400	5429	5453	5411	5403	5402	5421	5391	5424	5424
	5745	5750	5691	5692	5691	5691	5759	5752	5752	5736
	5926	5931	5935	5896	5875	5864	5965	5916	5905	5913
	5595	5613	5560	5553	5531	5521	5557	5544	5605	5562
	5362	5371	5356	5346	5344	5335	5328	5328	5370	5366
	5677	5655	5673	5726	5692	5649	5678	5653	5704	5704
	5693	5695	5689	5709	5695	5741	5687	5695	5695	5695
	5940	5979	5979	5960	5957	5957	5963	5928	5967	5940
	5903	5903	5881	5903	5903	5903	5903	5903	5903	5903
tai 100 X 20	6457	6496	6496	6444	6435	6469	6570	6542	6508	6474
	6446	6413	6444	6411	6360	6412	6452	6404	6501	6442
	6578	6612	6572	6496	6495	6515	6559	6473	6581	6515
	6473	6567	6528	6573	6478	6485	6558	6493	6550	6493
	6578	6586	6602	6613	6543	6555	6540	6629	6572	6568
	6627	6654	6584	6621	6603	6598	6570	6655	6708	6668
	6644	6519	6550	6576	6526	6480	6584	6610	6567	6494
	6727	6687	6713	6687	6678	6606	6757	6742	6699	6689
	6561	6487	6651	6566	6576	6569	6519	6578	6575	6542
	6667	6706	6732	6689	6644	6644	6672	6708	6710	6705
tai 200 X 10	10980	11044	10952	10950	10952	10957	10948	10957	10993	10993
	10625	10625	10639	10626	10613	10645	10605	10635	10583	10633
	10995	11120	11017	11017	11045	11015	11045	11082	11017	11045
	10950	10939	10974	10948	10939	10939	10939	10948	10948	10939
	10760	10575	10575	10575	10574	10627	10625	10575	10575	10582
	10394	10412	10398	10375	10415	10425	10410	10428	10417	10419
	10976	11069	10998	10962	10962	11069	10976	10966	10998	10998
	10856	10828	10828	10828	10849	10856	10832	10856	10821	10865
	10503	10529	10497	10497	10497	10536	10553	10515	10498	10566
	10786	10794	10758	10758	10758	10758	10835	10758	10783	10850
tai 200 X 20	11591	11570	11639	11604	11524	11693	11517	11649	11575	11519
	11656	11801	11791	11729	11713	11665	11701	11835	11697	11805
	11779	11759	11758	11702	11725	11737	11706	11737	11766	11682
	11750	11682	11688	11655	11647	11733	11754	11670	11710	11738
	11747	11666	11701	11629	11524	11729	11594	11661	11669	11637
	11671	11643	11651	11544	11596	11569	11629	11657	11648	11517
	11819	11805	11782	11678	11683	11771	11937	11778	11817	11848
	11893	11808	11690	11643	11669	11851	11748	11748	11850	11759
	11671	11637	11575	11543	11495	11672	11716	11646	11697	11717
	11735	11774	11832	11771	11709	11726	11680	11779	11761	11756

Table A.4 Detailed Results of DDE

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 20 X 5	1297	1297	1278	1297	1297	1297	1278	1297	1297	1297
	1366	1366	1365	1360	1359	1366	1366	1366	1366	1366
	1099	1081	1087	1099	1098	1098	1089	1081	1088	1087
	1309	1309	1309	1309	1309	1305	1293	1309	1301	1306
	1244	1244	1250	1250	1258	1244	1239	1250	1244	1250
	1210	1210	1210	1210	1210	1210	1210	1210	1213	1210
	1251	1251	1251	1251	1251	1251	1256	1256	1251	1251
	1214	1214	1211	1214	1214	1214	1211	1224	1214	1214
	1240	1230	1230	1255	1253	1230	1261	1253	1261	1253
	1120	1108	1127	1108	1120	1127	1127	1120	1108	1120
tai 20 X 10	1597	1596	1620	1619	1624	1613	1619	1586	1598	1599
	1715	1667	1693	1680	1692	1688	1686	1688	1701	1684
	1511	1511	1513	1518	1536	1533	1516	1517	1519	1509
	1388	1398	1424	1400	1392	1391	1397	1439	1399	1406
	1438	1474	1430	1457	1487	1427	1436	1497	1440	1436
	1424	1407	1433	1422	1401	1431	1424	1430	1424	1424
	1493	1514	1526	1486	1508	1499	1492	1493	1500	1520
	1566	1572	1578	1590	1570	1573	1577	1570	1548	1566
	1636	1612	1632	1636	1627	1632	1636	1617	1612	1624
	1613	1629	1612	1626	1624	1618	1613	1626	1627	1632
tai 20X 20	2364	2367	2344	2343	2307	2326	2364	2351	2362	2363
	2121	2134	2140	2116	2138	2130	2122	2115	2112	2119
	2354	2420	2368	2344	2354	2362	2337	2384	2356	2350
	2244	2238	2248	2286	2233	2248	2270	2265	2252	2276
	2325	2316	2328	2322	2299	2329	2320	2372	2313	2326
	2261	2272	2248	2264	2247	2244	2246	2267	2299	2243
	2318	2329	2305	2287	2355	2303	2299	2303	2302	2319
	2232	2248	2234	2238	2268	2239	2229	2211	2220	2225
	2264	2287	2324	2288	2263	2285	2281	2248	2272	2310
	2196	2201	2213	2219	2214	2242	2253	2189	2185	2232

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 50 X 5	2735	2735	2735	2729	2735	2735	2735	2735	2729	2735
	2838	2848	2882	2838	2882	2848	2848	2848	2882	2839
	2638	2624	2621	2624	2639	2644	2624	2624	2634	2621
	2770	2768	2770	2770	2777	2782	2777	2782	2762	2770
	2864	2864	2864	2864	2864	2864	2864	2864	2864	2864
	2838	2832	2835	2835	2832	2836	2832	2835	2829	2832
	2736	2736	2732	2732	2732	2725	2732	2736	2725	2732
	2705	2707	2707	2707	2707	2705	2707	2705	2707	2704
	2577	2577	2587	2564	2577	2568	2588	2568	2568	2565
	2802	2782	2783	2783	2784	2783	2784	2783	2791	2784
tai 50 X 10	3101	3076	3126	3126	3102	3077	3126	3126	3126	3126
	2979	2971	2974	2924	2956	2969	2972	2940	2970	2983
	2956	2915	2977	2960	2970	2960	2926	2988	2928	2960
	3120	3136	3167	3074	3098	3116	3102	3120	3141	3140
	3065	3068	3064	3052	3102	3062	3049	3082	3058	3115
	3115	3075	3104	3083	3093	3100	3083	3098	3100	3103
	3156	3156	3165	3165	3172	3165	3165	3189	3176	3165
	3114	3132	3082	3085	3094	3105	3119	3081	3111	3087
	2980	2970	2987	2970	2992	3005	2970	2975	3025	2957
	3190	3141	3166	3152	3190	3154	3161	3156	3152	3140
tai 50 X 20	3964	4009	3949	3973	3965	3997	3974	4007	4056	3950
	3833	3897	3833	3928	3879	3930	3874	3894	3856	3881
	3822	3792	3805	3785	3803	3798	3783	3802	3796	3831
	3848	3872	3873	3884	3879	3850	3876	3839	3913	3848
	3798	3766	3800	3791	3785	3776	3794	3740	3800	3729
	3843	3854	3857	3912	3855	3874	3901	3829	3834	3851
	3866	3854	3863	3860	3827	3836	3897	3850	3855	3866
	3858	3851	3839	3877	3837	3938	3817	3895	3845	3870
	3875	3956	3924	3890	3874	3892	3882	3939	3949	3876
	3910	3899	3936	3877	3902	3872	3949	3915	3899	3899
tai 100 X 5	5495	5495	5493	5493	5495	5495	5493	5495	5495	5493
	5290	5285	5275	5284	5286	5284	5284	5289	5284	5290
	5193	5179	5205	5221	5193	5221	5221	5193	5179	5205
	5023	5023	5029	5021	5023	5021	5023	5023	5023	5023
	5255	5253	5266	5255	5265	5255	5255	5255	5267	5255
	5146	5146	5139	5135	5139	5150	5136	5146	5146	5146
	5262	5262	5264	5246	5255	5262	5263	5255	5255	5304
	5108	5123	5134	5137	5108	5108	5134	5133	5133	5094
	5475	5467	5467	5479	5473	5473	5466	5465	5448	5467
	5342	5328	5330	5346	5336	5346	5334	5346	5342	5342

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 100 X 10	5868	5818	5837	5863	5863	5856	5821	5897	5860	5821
	5396	5411	5408	5412	5458	5412	5400	5456	5400	5415
	5752	5752	5731	5736	5712	5778	5745	5734	5702	5762
	5937	5905	5939	5912	5933	5935	5890	5905	6001	5897
	5594	5562	5580	5535	5583	5612	5604	5569	5570	5579
	5344	5370	5342	5328	5386	5372	5332	5356	5352	5334
	5706	5684	5711	5706	5659	5692	5670	5691	5675	5643
	5721	5704	5713	5721	5695	5722	5742	5733	5705	5674
	5951	5979	5940	5979	5940	5979	5930	5969	5940	5960
	5903	5903	5903	5881	5903	5903	5904	5881	5903	5903
tai 100 X 20	6487	6487	6425	6498	6470	6484	6451	6471	6455	6452
	6396	6434	6433	6390	6490	6462	6492	6447	6408	6438
	6488	6619	6482	6659	6552	6543	6526	6595	6571	6531
	6495	6448	6513	6478	6479	6533	6498	6453	6469	6531
	6598	6577	6566	6569	6568	6605	6618	6600	6619	6662
	6596	6613	6602	6605	6583	6673	6609	6618	6580	6638
	6574	6516	6580	6571	6605	6498	6552	6594	6509	6548
	6704	6638	6721	6761	6713	6723	6738	6704	6716	6718
	6556	6569	6552	6558	6529	6520	6511	6509	6539	6582
	6644	6646	6673	6674	6679	6651	6642	6619	6646	6652
tai 200 X 10	10950	10992	10957	10938	10992	10957	10957	10957	10993	11027
	10571	10609	10634	10598	10595	10586	10630	10627	10648	10570
	11045	11049	11079	11079	11070	11133	11049	11070	11049	11133
	10939	10993	10984	11010	10995	10948	10939	10939	10961	10973
	10637	10625	10604	10582	10610	10647	10604	10652	10640	10650
	10440	10427	10411	10429	10427	10426	10409	10416	10458	10409
	10958	10998	10985	10979	11011	10990	11022	10992	10998	10962
	10866	10821	10819	10858	10849	10856	10832	10856	10821	10861
	10497	10497	10497	10497	10497	10536	10553	10526	10526	10566
	10767	10786	10768	10758	10758	10760	10835	10758	10758	10758
tai 200 X 20	11565	11633	11553	11665	11586	11577	11527	11599	11523	11608
	11741	11668	11782	11762	11705	11701	11691	11652	11668	11660
	11756	11736	11735	11679	11702	11714	11683	11714	11743	11659
	11825	11772	11696	11685	11627	11627	11627	11627	11648	11648
	11857	11766	11710	11680	11669	11629	11724	11652	11629	11629
	11566	11566	11670	11628	11630	11600	11566	11538	11520	11520
	11742	11745	11937	11778	11683	11771	11937	11688	11713	11686
	11871	11782	11668	11631	11647	11729	11756	11726	11728	11737
	11691	11622	11563	11443	11585	11659	11616	11686	11711	11711
	11785	11728	11736	11736	11709	11656	11710	11685	11661	11736

APPENDIX B

CPU TIMES OF THE ALGORITHMS

Table B.1 CPU Times of PSO_{SPV}

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 20 X 5	6.178	6.521	6.146	6.302	6.084	6.178	6.318	1.311	6.505	6.396
	6.271	6.287	6.224	6.256	6.302	6.35	6.38	6.365	6.396	6.521
	6.568	6.598	6.63	6.677	6.708	6.724	7.051	6.895	6.833	6.895
	6.88	6.849	6.91	6.88	7.005	6.942	7.02	7.035	7.051	7.083
	7.036	7.082	7.02	7.098	7.036	7.098	7.254	7.27	7.254	7.207
	7.145	7.831	8.096	8.066	7.924	7.566	7.239	7.332	7.332	7.41
	7.441	7.676	7.519	7.737	7.598	7.566	7.566	7.519	7.519	7.629
	7.597	7.691	7.769	7.878	7.94	7.878	7.91	7.971	7.909	7.972
	8.159	7.987	8.034	8.065	8.113	8.049	8.128	8.19	8.174	8.206
	8.221	8.331	8.33	8.424	8.393	8.315	8.346	8.518	8.595	8.409
tai 20 X 10	8.393	8.377	8.361	8.378	9.048	8.689	8.611	8.331	8.346	8.455
	8.393	8.642	8.471	8.362	8.486	8.378	8.455	8.486	8.456	8.33
	8.362	8.533	8.564	8.565	8.471	8.439	8.362	8.346	8.533	8.565
	8.564	8.455	8.721	8.377	8.409	8.361	8.471	8.393	8.377	8.284
	8.299	8.284	8.268	8.346	8.486	8.565	8.283	8.315	8.393	8.362
	8.517	8.425	8.704	8.284	8.315	8.284	8.595	8.518	8.346	8.346
	8.705	8.471	8.408	8.487	8.47	8.393	8.549	8.393	8.299	8.44
	8.268	8.471	8.533	8.533	8.892	9.111	8.471	8.486	8.533	8.518
	8.362	8.315	8.33	8.486	8.674	8.377	8.331	8.44	8.642	8.408
	8.393	8.362	8.424	8.627	8.517	8.487	8.299	8.268	8.362	8.377
tai 20X 20	8.362	8.471	8.595	8.378	8.377	8.424	8.611	8.955	8.58	8.392
	8.44	8.377	8.331	8.346	8.471	8.393	8.548	8.503	8.299	8.471
	8.33	8.362	8.471	8.408	8.362	8.705	8.611	8.611	8.549	8.315
	8.439	8.534	8.517	8.471	8.799	8.673	8.284	8.44	8.517	8.658
	8.362	8.502	8.58	8.533	8.549	8.424	9.033	8.377	8.408	8.487
	8.315	8.361	8.456	8.564	8.549	8.471	8.486	8.393	8.362	8.861
	8.626	8.534	8.455	8.44	8.439	8.409	8.361	8.378	8.424	8.502
	8.502	8.517	8.815	8.377	8.408	8.455	8.471	8.549	8.721	8.689
	8.627	8.58	8.502	8.533	8.502	8.455	8.346	8.581	8.72	8.518
	8.517	8.253	8.408	8.299	8.378	8.439	8.58	8.502	8.378	8.283

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 50 X 5	9.08	9.313	9.282	9.36	9.282	9.173	1.108	9.11	9.251	9.688
	9.235	9.438	9.235	9.126	9.111	9.36	9.547	9.391	9.251	9.126
	9.298	9.297	9.454	9.579	9.063	9.048	9.064	9.11	9.033	9.048
	9.173	9.126	9.329	9.204	9.329	9.297	9.064	9.22	9.172	9.111
	9.298	9.313	9.173	9.063	9.095	9.064	9.095	9.173	9.235	9.079
	9.189	9.001	9.079	9.079	9.204	9.251	9.236	9.36	9.36	8.97
	9.017	9.547	9.313	9.282	9.095	9.173	9.095	9.297	9.173	9.22
	9.079	9.064	9.063	9.064	9.048	9.079	9.345	9.266	9.329	9.173
	9.189	9.765	9.157	9.111	9.188	9.423	9.344	9.267	9.235	9.204
	8.97	9.173	9.204	9.454	9.266	9.126	9.251	1.622	9.033	9.235
tai 50 X 10	9.485	9.345	9.422	9.438	9.703	9.282	9.173	9.407	9.563	9.594
	9.672	9.469	9.657	10.06	9.703	9.407	9.391	9.579	9.344	9.501
	9.516	9.547	9.266	9.376	9.282	9.282	9.298	9.282	9.376	9.438
	9.562	9.47	9.219	9.36	9.657	9.407	9.344	9.329	9.407	9.282
	9.329	9.251	9.422	9.579	9.266	9.469	9.813	9.36	9.157	9.267
	9.406	9.938	9.438	9.453	9.345	9.266	9.126	9.439	9.36	9.36
	9.375	9.173	9.329	9.485	9.578	9.36	9.423	9.344	9.126	9.204
	9.407	9.345	9.375	9.314	9.251	9.219	9.36	9.61	9.485	9.329
	9.235	9.313	9.22	9.219	9.47	9.297	9.236	9.25	9.283	9.64
	9.376	9.454	9.453	9.345	9.235	9.469	9.423	9.391	9.516	9.376
tai 50 X 20	9.734	9.548	9.547	9.687	9.735	9.641	9.516	9.532	9.609	9.719
	10.22	10.19	9.859	9.61	9.937	9.922	9.75	9.625	9.703	9.61
	9.781	9.938	9.765	9.766	9.641	9.578	9.61	9.688	9.703	9.609
	9.626	9.609	9.672	9.626	9.547	9.719	10.03	9.531	9.516	9.61
	9.657	9.609	9.563	9.594	9.594	9.657	9.578	9.844	9.937	9.75
	9.75	9.953	9.734	9.547	9.735	9.641	9.609	9.516	9.75	9.829
	9.937	10.12	9.594	9.735	9.625	9.61	9.75	9.781	9.719	9.703
	10.05	10.51	9.906	9.641	9.781	9.657	9.531	9.501	9.485	9.672
	9.797	9.875	9.812	9.578	9.782	9.687	9.657	9.766	9.796	9.844
	9.828	9.657	9.609	9.516	9.735	9.719	9.718	9.844	9.969	9.796
tai 100 X 5	14.13	14.23	14.15	14.57	14.12	14.74	13.98	13.96	14.12	14.13
	13.71	13.99	13.81	13.56	13.73	14.31	13.88	14.34	13.74	14.4
	14.17	14.07	13.99	14.38	15.26	14.55	14.06	14.76	14.43	14.31
	13.95	14.35	14.45	14.54	14.26	14.27	14.48	14.12	14.6	14.12
	14.34	14.87	13.9	14.12	14.17	14.06	14.06	14.48	14.29	14.2
	14.07	14.93	10.51	14.35	14.15	5.35	14.07	14.24	14.24	14.35
	14.37	14.31	14.21	14.23	14.38	14.46	14.79	14.21	14.52	14.35
	14.15	14.4	14.23	14.95	14.23	14.23	14.34	14.1	14.21	14.46
	14.09	14.13	14.15	14.18	14.18	14.26	14.03	14.07	13.98	14.02
	14.13	14.23	14.31	14.15	14.46	14.2	14.27	14.68	14.15	14.42

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 100 X 10	14.8	14.57	14.88	14.87	14.7	14.54	15.02	14.68	14.82	14.74
	15.15	14.51	14.87	14.7	14.6	14.74	14.62	14.8	14.43	14.98
	14.77	14.79	15.04	14.62	14.87	14.68	14.56	14.7	15.02	15.13
	14.63	15.16	14.56	14.43	14.4	14.62	14.62	14.51	14.85	14.68
	14.65	14.54	15.21	14.9	14.6	14.79	14.77	14.87	14.91	14.84
	14.48	14.62	14.68	14.46	14.71	14.62	14.73	14.76	14.76	14.77
	14.73	14.96	14.55	14.48	14.81	14.71	14.49	14.6	15.18	14.82
	14.56	14.63	14.9	14.73	14.65	14.76	14.63	15.3	14.48	14.68
	14.71	14.74	14.7	14.7	14.54	14.9	14.7	14.74	14.73	15.32
	14.9	14.51	14.77	14.65	15.27	15.02	14.62	14.9	14.76	14.65
tai 100 X 20	15.82	15.74	16.55	15.85	15.6	15.59	16.33	15.76	15.9	15.62
	16.3	16.13	15.91	15.87	15.69	15.74	15.87	15.76	15.66	16.19
	15.91	15.85	15.91	15.82	15.66	15.52	16.08	16.3	16.04	15.93
	15.91	16.02	15.9	15.83	15.79	15.98	15.69	15.76	15.85	16.22
	15.91	16.1	15.94	15.66	15.8	15.66	15.82	15.79	16.01	15.57
	15.8	15.88	15.91	15.88	15.91	15.59	15.83	15.94	15.86	15.44
	15.76	15.74	15.9	15.85	15.73	16.01	15.87	15.99	15.96	15.71
	15.8	16.01	15.55	15.76	15.85	15.99	15.98	15.55	15.71	15.9
	15.87	15.87	15.71	16.08	15.94	15.98	16.1	16.04	15.83	15.9
	15.8	16.18	15.96	15.87	15.71	15.91	16.16	15.96	15.86	15.79
tai 200 X 10	53.14	52.74	53.01	52.34	54.01	52.18	51.56	52.95	53.71	53.93
	52.51	52.45	53.8	52.4	51.29	51.7	53.2	53.1	52.14	51.96
	53.24	53.46	52.74	53.71	52.2	52.98	53.29	51.96	53.82	52.93
	51.81	52.73	52.35	52.67	51.67	53.34	52.96	52.17	52.99	51.51
	52.79	54.12	53.15	52.79	51.03	53.17	52.82	51.71	52.23	52.48
	52.68	52.09	51.82	52.45	53.2	52.04	51.78	52.26	52.21	52.06
	52.84	53.04	52.14	53.43	52.29	52.48	52.64	53.96	53.7	52.45
	53.01	52.89	52.7	53.2	51.59	51.03	51.79	51.61	51.95	52.9
	52.62	51.17	52.04	52.09	52.82	53.38	52.09	51.19	52.81	52.81
	52.46	52.45	52.09	53.27	51.44	51.67	51.5	52.85	52.06	52.28
tai 200 X 20	56.96	57.03	56.33	57.5	57.95	58.06	56.24	57.13	57.6	56.77
	56.64	57.05	55.43	57.2	55.04	56.52	55.71	56.69	55.38	56.94
	56.99	57.77	56.54	58.19	58.3	56.88	56.96	56.71	58.1	56.61
	56.6	57.16	55.6	56.71	56.61	56.04	55.57	55.5	55.97	55.44
	56.58	55.99	56.44	56.13	54.88	56.93	56.33	55.3	57.86	56.5
	56.74	58.14	69.92	58.13	58.15	58.98	58.89	59.4	63.88	56.65
	56.83	55.33	58.14	57.15	57.25	57.85	57.42	58.17	55.91	55.24
	57.06	55.3	55.4	56.47	57.41	55.49	57.11	58.06	56.16	57.11
	56.47	56.04	55.15	57.24	55.16	55.91	56.5	56.27	56.93	57.24
	56.71	56.65	55.43	55.49	57.42	57.11	57.1	56.16	56.25	55.38

Table B.2 CPU Times of DPSO

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 20 X 5	72.49	73.5	78.27	75.42	77.69	75.34	73.19	72.38	73.14	74.22
	72.03	72.22	74.05	74.08	73.05	76.09	74.27	72.24	75.05	71.22
	76.19	75.08	71.06	74.98	73.25	77.11	70.11	72.06	74.11	73.28
	72	72.77	72.83	73.75	75.88	70.84	75.77	71.69	69.92	70.64
	71.83	73.73	75.78	75.86	72.73	74.91	72.8	72.69	75.89	69.7
	72.83	72.73	77	77.19	74.91	69.88	71.97	69.72	70.69	73.73
	72.72	73.78	72.66	72.66	69.61	73.78	73.69	71.73	70.63	70.72
	76.77	70.75	74.77	70.7	73.98	76.67	71.73	73.55	45.55	70.72
	72.94	71.94	71.78	72.73	74.83	71.64	73.7	72.75	72.8	70.77
	70.72	72.77	75.77	74.88	74.63	64.38	75.83	72.86	71.74	41.59
tai 20 X 10	69.59	71.78	70.88	74.78	77.89	71.64	72.7	77.97	71.77	71.74
	75.7	74.73	69.74	72.83	69.64	74.94	73.7	75.64	72.92	73.77
	74.69	71.03	77.19	73.06	74.1	72.23	73.02	75.08	74.17	74.2
	79.16	74.16	72.09	72.16	77.06	98.08	73.13	74.73	75.75	71.83
	80.5	72.77	73.61	74.75	71.61	71.78	75.73	73.61	72.74	74.69
	75.77	74.05	77.52	71.77	70.66	74.74	78.86	71.83	72.8	70.83
	80.81	72.94	71.84	72.75	73.94	71.92	70.75	69.95	73.63	73.7
	72.75	71.72	75.95	71.81	77.7	73.77	74.8	74.81	72.83	73.89
	74	73.88	74.8	74.89	72.3	72.8	78.95	75.83	74.7	75.22
	71.83	72.86	74.77	72.89	72.86	71.56	72.61	74.81	73.99	74.67
tai 20X 20	72.81	81.17	69.98	71.83	73.88	72.86	70.92	75.94	70.92	71.03
	78.22	72.36	77.33	77.11	72.23	73.17	73.19	70.08	76.16	73.08
	72.3	73.28	74.2	75.06	75.02	73	73.88	70.89	74.81	73.83
	73.69	71.72	73.81	71.81	75.09	72.63	73.59	73.73	73.89	74.91
	70.8	74.7	71.85	73.97	76.84	72.28	71.05	76.19	75.27	72.13
	76.81	74.14	71.31	71.13	72.05	72.2	74.31	72.23	75.13	75.06
	72.86	73.76	75.66	71.89	72.8	72.92	70.88	71.84	75.83	74.81
	74.81	73.83	75.09	76.13	72.47	72.2	72.96	74.17	73.99	75.16
	75.16	77.22	75.22	76.14	71.14	78.03	72.03	73.28	72.06	77.11
	75.11	75.19	74.2	73.08	72.19	72.16	76.19	72.23	77.39	72.94

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 50 X 5	70.78	74.83	79.06	71.88	73.02	73.98	73.98	71.86	74.89	74.95
	75.03	74.8	72.84	72.97	73.97	72.95	76.89	71.81	74.96	69.89
	73.88	70.95	72.22	74.2	75.28	73.08	74.11	76.4	73.23	70.31
	73.27	73.28	76.45	73.2	76.17	73.19	74.31	77.16	72	76.92
	71.98	73	76.84	73.06	70.89	72.99	72.02	75.19	75.06	77.66
	73.27	72.25	75.28	75.31	75.34	71.17	72.36	73.22	76.25	75.28
	74.17	74.14	71.27	77.08	74.11	72.22	77	75.08	76.2	74.16
	76.41	74.16	77.22	73.25	70.97	70.81	73.83	75.08	75.22	73.99
	72.25	71.28	73.14	73.17	74.13	73.16	75.47	74.2	75.22	72.28
	72.31	71.21	71.42	74.17	70.95	72.06	78.19	74.84	71.86	74.08
tai 50 X 10	71.02	74.78	73.13	72.91	73	69.98	75.88	71.84	72.91	72
	71	70.91	71.77	71.02	71.92	74.88	76.92	80	73.92	74.89
	72.8	74	76.03	73	76	73.81	76.17	76.11	79.03	72.16
	74.02	70.89	73.86	71.97	73.02	76.97	73.05	75.31	76.3	74.44
	74.22	72.17	72.44	77.3	74.33	77.8	74.17	73.19	72.48	75.17
	72.33	74.33	74.14	74.09	72.91	71.84	72.77	73.05	71.97	74.95
	73.94	70.93	80.02	72.91	72.89	72.02	72.95	73.97	75.1	73.92
	73.61	74.61	73.38	78.52	72.28	74.42	72.44	75.27	74.3	74.53
	71.33	76.31	71.43	74.36	74.5	72.36	77.31	74.28	72.38	72.28
	74.34	73.34	76.3	76.8	73.48	72.34	74.36	74.06	70.97	73.95
tai 50 X 20	72.3	72.17	73.33	76.2	74.11	73.41	77.13	72.02	79.05	73.25
	71.97	72.09	73.09	74.08	71.13	74.33	73.3	75.34	75.45	73.38
	71.64	72.52	71.36	74.3	76.45	72.36	77.23	73.33	75.3	76.52
	72.48	72.5	73.34	80.38	76.33	71.11	73.19	75.25	72.31	75.55
	77.47	72.66	81.03	74.31	73.63	75.69	82.66	74.55	72.44	73.56
	75.59	73.42	73.41	73.41	72.38	73.39	75.44	74.39	74.38	73.53
	75.52	73.59	72.47	75.39	70.58	75.56	71.52	76.66	72.66	74.52
	74.42	70.56	72.45	76.67	72.53	77.31	72.56	74.36	73.42	74.38
	77.58	76.63	71.5	74.44	73.13	76.06	72.17	98.78	72.47	73.16
	75.81	77.41	72.13	76.3	80.28	77.1	74.42	74.27	71.02	72.23
tai 100 X 5	79.47	73.67	73.52	75.78	72.81	74.53	76.84	63.7	74.86	72.09
	75.59	73.31	75.92	85.13	85.2	80.38	74.49	77.08	74.52	75.58
	80.99	74.86	74.47	77.88	72.55	72.88	78.17	74.89	80.7	74.53
	76.91	77.28	74.34	77.95	71.39	78.66	77.5	72.48	76.5	78.33
	78.45	77	81.59	79.78	78.73	76.45	75.08	74.36	78.63	80.75
	78.11	75.72	79.41	78.13	80.91	78.08	77.97	75.58	77.53	77.99
	77.72	78.88	74.8	73.74	74.47	77.55	78.74	74.69	73.58	74.64
	75.81	74.95	72.61	80.14	75.11	77.58	81.78	77.09	81.03	76.99
	105.3	77.91	74.7	76.78	75.8	74.47	76.59	75.25	75.11	86.67
	87.47	77.84	76.52	103.7	76.88	77.66	78.69	75.63	73.67	74.78

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 100 X 10	79.34	74.13	75.86	74.86	76.99	76.92	75.06	74.09	75.13	77.53
	80.16	75.08	79.17	78	79.08	78.56	73.42	76.89	74.22	78.47
	74.56	76.34	78.13	81.75	74.13	73.8	76.36	79.8	79.47	73.11
	80.98	75.66	79.75	75.7	75.78	105	77.81	76	74.92	75.75
	72.33	73.02	75.97	74.78	76.25	81.45	74.89	76.86	77.02	77.78
	77.81	98.33	84.92	72.64	74.64	77.75	80.3	74.59	74.8	74.72
	80.16	71.72	73.88	73.67	74.67	73.88	78.75	73.8	75.89	70.94
	82.52	76.86	82.02	74.81	77.08	83.25	80.89	75.13	75.8	71.75
	73.8	79.02	75.73	72.13	73.81	74.75	74.42	72.45	76.41	77.63
	75.66	72.34	73.41	74.47	73.42	72.5	72.5	70.47	75.39	74.45
tai 100 X 20	77.28	74.92	77.05	76.83	74.89	75.85	76.89	76.94	73.08	75.22
	75.14	73.92	73.94	71	75	73.91	78.92	76.42	74.36	77.02
	74.13	78.88	73.99	76.94	77.03	74.92	76.05	73.91	73.88	73.06
	72.31	78.14	73.89	74.92	76.06	71.84	75.06	74.91	77.02	72.98
	73.14	72.92	72.94	77.2	73.95	75.02	75.94	77.16	76.49	82.28
	75.78	77.02	75.91	79.8	83.89	99.91	117.1	83.59	119.6	101.4
	101.8	115.8	82.03	73.92	75.98	85.75	98.9	80.19	80.23	75.06
	78.77	88.88	78.09	78.62	79.88	73.38	73.08	75.47	77.33	79.3
	75.25	71.77	79.25	77.28	91.96	98.75	87.7	90.7	77.53	80.58
	88.41	90.72	77.8	73.69	74.75	80.97	95	97.77	88.84	99.37
tai 200 X 10	107.1	107.2	88.64	80.53	78.66	101	91.64	88.44	81.81	88.13
	105.3	97.06	91.03	98.5	98.25	89.55	78.7	97.44	89.36	96.64
	82.55	77.42	75.61	81.83	81.61	78.5	78.67	82.8	87.22	102.3
	94.44	84.56	88.03	86.17	78.65	83.45	86.28	107.4	90.31	93.16
	79.83	90.09	77.92	76.67	79.41	77.53	84.28	80.53	74.42	80.39
	77.75	78.77	83.52	74.44	79.51	80.45	84.25	82.36	75.38	77.44
	82.78	74.44	77.45	78.64	82.42	76.11	77.05	79.13	76.97	79.33
	81.06	80.55	81.84	109.1	85.47	91.95	81.22	84.52	77.64	85.62
	78	78.55	75.42	77.49	82.36	78.39	78.36	77.42	80.34	77.56
	77.23	84.25	83.91	75.67	84.8	83.81	89.33	77.03	77.66	78.64
tai 200 X 20	86.11	89.22	85.95	83.74	85.7	82.63	81.8	85.59	81.97	82.56
	83.72	82.94	79.86	84.86	81.59	84	83.99	79.81	85.63	80.92
	81.52	84.22	89.05	82.94	97.09	81.25	87.11	78.86	81	80.84
	81.64	79.83	83.69	85.78	82.81	81.53	108.8	87.67	85.72	78.63
	83.75	83.02	81.83	84.97	80.91	82.63	82.38	84.34	81.39	80.28
	80.14	84.48	81.41	83.55	82.39	80.59	82.34	81.33	84.16	97.5
	81.09	84.2	85.53	80.64	86.2	83.44	82.36	82.33	86.17	85.44
	83.22	87.7	82.72	86.02	82.78	79.55	85.39	89.16	82.19	82.58
	85.3	87.31	84.2	81.98	78.73	85.86	83.56	83.95	82.83	83.91
	81.52	83.83	84.77	80.99	80.03	81.7	81.88	82.39	81.73	86.34

Table B.3 CPU Times of SPPSO

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 20 X 5	8.829	7.816	3.214	7.971	10.59	4.383	9.033	7.02	7.113	8.456
	10.92	7.363	8.003	7.488	7.628	7.894	6.926	7.176	7.66	8.471
	7.488	0.749	7.566	9.703	8.58	7.223	4.399	7.551	7.878	7.488
	9.235	8.533	8.097	9.453	7.457	8.066	8.517	9.126	8.627	9.173
	8.767	8.097	4.789	7.597	7.519	7.676	3.775	9.126	7.55	8.112
	8.253	9.282	10.23	0.733	10.56	8.315	7.956	9.734	9.267	3.229
	8.284	8.814	9.391	9.235	8.097	8.143	7.753	8.877	7.737	9.376
	3.276	7.941	8.08	9.017	9.017	9.204	9.781	9.283	1.388	8.299
	9.641	0.998	2.949	4.773	9.189	9.532	10.34	9.844	8.923	9.672
	1.264	9.859	9.875	9.033	7.737	9.407	4.103	8.876	3.526	10.16
tai 20 X 10	9.204	9.953	9.765	9.423	12.06	9.719	8.876	14.77	9.142	10.86
	9.734	8.643	8.892	9.578	10.23	10.44	9.672	10.61	12.01	9.08
	11.14	9.922	10.94	9.345	9.454	9.172	10.53	9.517	10.05	10.27
	9.313	12.36	10.11	9.75	8.892	8.799	8.705	8.939	8.704	11.2
	9.251	9.984	9.875	9.126	9.095	8.908	9.204	9.001	8.767	9.079
	9.283	8.798	8.97	9.251	9.204	9.376	9.391	9.656	9.095	9.033
	9.251	9.36	9.204	9.641	9.204	9.578	9.235	9.111	8.767	9.594
	11.84	11.01	10.59	10.61	11.7	10.17	8.627	8.549	8.939	11.17
	10.62	9.313	9.282	9.126	8.986	9.048	9.235	9.095	9.376	10.44
	9.797	8.783	8.721	8.829	8.908	9.329	8.907	8.861	8.923	8.799
tai 20X 20	8.798	8.924	9.126	8.736	8.767	8.845	8.814	9.017	9.251	8.861
	8.908	8.751	8.705	9.36	11.22	8.517	8.596	17.53	8.721	8.767
	8.268	8.253	8.221	8.237	11.67	16.21	13.74	9.906	17.44	16.54
	11.81	14.98	9.688	10.66	9.563	10.44	10.44	9.5	10.14	9.532
	9.937	9.89	9.392	9.188	9.719	14.56	9.469	8.877	10.06	9.5
	9.937	8.955	8.72	8.783	8.471	8.767	9.064	9.017	8.954	15.34
	10.48	10.02	9.875	9.016	8.705	9.048	8.612	8.829	8.908	8.876
	9.485	9.189	8.939	9.344	8.923	8.752	8.486	9.173	9.735	9.625
	9.376	13.82	10.61	9.158	19.09	16.82	15.27	10.51	10.03	10.81
	15.34	8.876	8.814	9.033	9.032	8.955	8.751	16.86	15.43	8.393

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 50 X 5	2.59	9.079	9.984	11.31	12.9	10.19	3.432	1.202	7.191	15.77
	14.18	14.32	18.63	16.94	15.8	16.27	17.4	17.78	15.71	17.94
	16.91	17.69	16.05	19.27	15.58	8.97	8.721	8.595	8.892	9.064
	8.861	9.859	9.797	9.672	9.423	9.672	8.985	9.189	8.814	9.001
	9.033	9.484	9.002	8.611	8.502	8.986	8.892	9.251	8.892	9.141
	5.523	9.11	8.58	9.095	8.643	8.954	9.61	9.953	10.53	10.37
	9.048	9.297	8.861	5.32	1.591	9.033	5.522	0.203	8.642	8.939
	8.736	8.627	9.407	9.656	9.158	9.625	8.986	8.673	8.877	10.67
	9.657	9.75	10.12	8.877	8.751	9.079	9.017	9.173	9.547	10.16
	1.81	1.31	2.699	5.507	9.079	2.808	8.736	0.156	0.702	0.328
tai 50 X 10	9.734	9.22	8.939	9.095	10.03	9.516	10.36	14.74	10.08	14.79
	9.828	9.204	9.235	9.142	9.094	9.205	9.048	9.672	12.03	9.314
	9.453	9.563	9.282	9.251	9.079	9.251	9.329	9.095	9.095	9.188
	8.845	9.142	9.345	9.578	9.251	8.83	8.736	8.892	9.937	8.83
	9.157	8.892	8.83	8.673	8.767	8.908	9.079	10.45	9.626	9.5
	9.625	9.205	9.625	9.188	9.142	9.017	8.97	10.45	8.986	9.016
	9.813	9.11	9.08	8.845	18.35	8.97	8.892	8.767	8.783	8.783
	8.923	8.97	10.97	8.72	8.674	8.705	9.001	8.705	8.658	8.658
	8.736	8.721	8.658	8.845	8.673	10.81	10.72	11.51	11.55	11.54
	11.09	14.42	11.89	8.752	8.751	8.892	8.799	8.954	8.658	8.736
tai 50 X 20	9.563	9.485	9.423	9.095	9.032	9.095	9.407	9.422	9.36	9.594
	9.501	9.687	10.25	9.625	9.797	9.173	9.016	9.501	9.313	9.688
	9.734	9.75	9.423	9.89	10.05	10.55	10.53	10.11	9.61	9.563
	9.454	10.44	10.53	10.94	9.453	10.98	10.17	10.08	9.968	10.31
	10.27	10.17	10.19	10.23	9.999	10.23	14.41	11.59	9.688	9.859
	9.719	9.657	9.406	9.407	9.298	9.204	9.485	9.282	9.204	9.391
	9.782	9.594	9.126	9.328	9.236	9.391	9.61	9.375	9.501	9.5
	9.423	9.765	9.626	9.344	9.579	9.859	9.656	9.563	9.516	9.516
	9.672	9.906	10.13	9.422	9.454	10.06	9.766	9.391	9.5	11.3
	9.719	10.19	10.12	9.454	9.656	9.672	9.423	9.531	9.563	9.906
tai 100 X 5	3.37	6.396	4.945	9.781	0.11	0.608	2.403	9.984	3.541	2.075
	10.02	9.766	11.58	13.84	13.82	11.81	12.28	12.9	11.29	15.41
	9.766	9.843	9.828	9.22	9.251	10.17	9.969	9.999	9.782	9.079
	9.547	10.06	12.09	10.17	10.28	9.859	9.376	15.77	15.88	11.9
	9.641	9.844	9.329	9.173	9.157	12.18	15.52	9.11	9.095	9.22
	9.36	9.266	9.173	9.188	9.236	9.141	9.158	9.235	9.11	9.111
	12.17	17.53	9.407	9.158	9.11	9.188	9.282	9.329	9.267	9.251
	9.235	12.65	16.8	14.49	11.05	9.251	9.407	9.22	12.54	19.33
	19.39	18.42	18.46	18.86	17.61	17.19	17.86	12.37	10.92	20.67
	14.26	10.91	10.78	10.7	10.89	10.81	11.23	9.391	9.298	9.313

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 100 X 10	10.16	9.937	10.23	10.31	10.27	10.02	10.52	9.968	9.953	10.22
	10.25	9.796	10.08	9.828	9.953	9.859	9.875	9.641	9.766	9.828
	10.14	9.968	10.06	10.17	10.08	10.05	10.59	10	9.922	9.843
	10.13	10.08	9.828	9.984	9.875	9.984	10.14	10.19	10.2	9.906
	10.2	10.13	9.906	10.67	10.44	10.05	9.953	9.937	10.36	9.812
	10.28	9.843	9.938	9.937	9.844	9.968	9.719	9.781	10.17	9.891
	10.06	9.922	9.968	9.844	9.859	10.17	10.2	10.06	10.23	10.06
	10.22	10.25	10.16	10.36	10.47	10.59	10.3	10.27	10.31	10.33
	10.53	10.42	10.14	10.2	10.19	10.14	10.09	10.25	10.67	10.41
	10.8	10.27	10.36	11.47	11.19	10.44	10.08	10.69	10.27	10.27
tai 100 X 20	11.84	11.37	11.3	11.53	11.69	12.67	12.42	11.97	12.42	12.57
	12.45	11.7	13	12.73	11.98	11.78	11.81	12.25	11.97	12.87
	12.85	12.39	11.64	11.62	12.67	14.01	14.15	14.1	14.24	13.48
	13.49	11.51	11.31	11.42	14.46	13.98	11.4	11.33	11.43	13.42
	12.28	11.76	11.73	12.67	13.14	13.45	12.03	11.86	12.01	12.11
	12.03	11.67	11.56	11.93	12.15	11.51	11.84	11.59	11.44	11.42
	11.92	11.51	12.26	11.51	11.4	11.36	11.47	11.34	11.42	11.33
	11.86	11.65	12.35	11.61	12.31	11.78	11.67	15.71	16.6	15.51
	14.02	14.7	15.46	11.9	11.62	11.64	11.89	12.71	16.86	11.98
	12.18	12.12	12.53	12	11.61	11.7	11.67	11.2	11.25	11.48
tai 200 X 10	15.54	15.12	14.76	14.77	14.87	14.73	15.13	14.51	14.42	14.57
	15.07	14.07	13.96	14.04	13.98	14.1	13.99	14.06	14.01	14.18
	15.66	14.7	14.55	14.62	14.84	15.37	15.4	15.41	16.65	15.62
	18.27	21.79	18.36	19.42	15.01	14.24	14.37	14.4	14.32	14.27
	14.85	14.29	14.26	14.01	13.99	14.13	14.1	13.99	14.01	14.32
	14.84	14.24	14.12	14.1	14.21	14.12	14.9	16.22	19.73	17.6
	14.74	14.03	14.1	14.01	13.99	14.04	13.9	13.88	14.03	14.01
	14.99	14.15	14.46	14.54	14.27	14.18	14.31	14.26	15.1	19.05
	17.66	13.84	13.84	14.2	13.9	13.99	13.92	13.84	13.9	13.85
	15.29	14.27	14.26	14.27	16.18	14.18	14.35	14.23	14.37	14.57
tai 200 X 20	21.89	20.39	23.32	25.77	20.17	20.06	20.27	20.2	20.39	21.06
	21.75	19.86	19.91	19.59	19.52	19.77	19.81	20.27	20.11	20.17
	26.16	21.5	25.68	25.33	29.58	23.87	24.59	20.36	19.72	19.77
	21.03	19.48	19.66	19.55	23.77	28.39	28.22	28.36	26.47	20.16
	21.92	20.26	20.33	20.36	20.03	20.36	20.31	20.62	20.47	20.47
	21.73	20.11	20.22	20.14	19.98	20.8	20.47	20.45	20.67	20.22
	22.57	20.56	20.25	21.14	20.16	20.97	19.98	20.05	20.31	20.55
	22.34	20.34	20.97	20.12	20.65	20.37	20.9	20.36	20.39	20.41
	21.56	20.41	20.69	20.2	20.06	20.64	20.67	20.72	20.73	20.17
	21.81	20.03	20.34	20.3	20.39	21.09	20.59	20.4	20.84	20.45

Table B.4 CPU Times of DDE

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 20 X 5	10	11.19	0.998	9.938	10.11	9.376	4.618	9.656	10.06	9.626
	9.796	9.953	9.782	9.531	2.761	9.158	9.516	9.563	9.5	9.438
	9.641	6.505	10.09	9.673	9.828	9.594	9.547	7.254	9.438	9.641
	9.625	9.516	9.563	9.454	9.531	9.641	3.463	9.47	10.17	9.532
	9.656	9.594	9.719	9.641	9.656	9.828	9.922	9.828	9.75	9.891
	10.17	9.703	9.703	9.797	9.844	9.859	9.828	9.938	9.89	9.906
	9.859	10.02	10.55	9.922	9.844	9.921	9.906	9.891	9.937	12.26
	9.968	10.06	10.06	10.02	10.61	10.31	10.16	10.17	10.19	10.2
	10.2	8.455	8.627	10.27	10.37	2.418	10.27	10.84	10.3	10.39
	10.48	2.683	10.58	3.916	10.64	10.58	10.7	10.67	10.22	10.83
tai 20 X 10	11.43	11.55	11.43	12.43	11.54	11.65	11.69	11.62	11.62	11.75
	11.59	11.53	11.59	11.67	11.65	11.67	11.67	11.72	11.73	13.28
	11.84	11.95	11.9	11.95	11.97	11.98	11.89	12.15	12.01	11.98
	12.01	12.37	12.15	12.37	12.17	11.92	11.9	12.51	12.56	12.5
	12.54	12.4	12.32	11.93	12	11.83	11.9	12.2	11.81	11.79
	11.89	11.93	11.84	11.87	12.08	11.98	11.73	12.03	11.97	11.81
	12	12.4	12.5	12.06	11.89	12.06	12.62	12.31	11.94	12.03
	12.14	12.23	11.68	11.78	11.97	12.09	12.54	12.42	12.54	12.15
	12.42	12.37	12.36	12.32	11.97	11.73	11.67	12.09	12.31	11.87
	12.08	12.14	11.93	11.92	12.08	12.12	12.15	11.92	12	12.31
tai 20X 20	14.71	14.43	14.8	14.71	14.34	14.74	14.68	14.49	14.2	14.23
	14.32	14.23	14.29	14.21	14.09	13.82	14.01	14.63	14.07	13.9
	14.51	14.1	14.07	14.6	14.34	14.2	14.52	14.6	14.63	14.85
	14.32	14.29	14.67	14.45	14.12	14.18	14.04	13.81	13.81	14.27
	13.95	14.07	14.6	14.34	14.31	14.32	14.43	14.98	14.68	14.79
	14.38	14.32	14.43	14.37	14.18	14.7	14.38	14.27	14.29	14.35
	14.17	14.15	14.18	14.09	14.23	14.26	14.1	14.63	14.74	14.77
	14.63	14.81	14.49	15.46	14.37	14.42	14.46	14.42	14.34	14.45
	14.49	14.48	14.27	14.21	14.09	14.1	14.46	14.09	14.51	14.48
	14.57	14.35	14.26	14.15	14.32	14.18	14.21	14.73	14.51	14.49

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 50 X 5	46.8	46.57	46.5	46.96	46.88	46.54	46.47	46.16	45.96	46.16
	44.54	44.46	44.84	44.93	44.57	45.15	44.68	44.51	44.96	44.38
	44.23	43.99	19.81	44.02	43.96	44.57	45.05	45.01	45.02	27.94
	45.38	45.6	45.55	43.46	42.5	42.74	42.59	42.76	42.89	42.51
	44.04	44.2	43.98	44.18	44.13	44.07	44.24	44.04	43.91	44.2
	42.7	42.39	42.11	42.09	42.42	42.36	42.64	42.31	25.8	42.01
	43.49	44.09	43.68	43.57	44.12	34.88	43.95	44.1	22.57	43.84
	43.2	42.89	43.18	43.07	43.46	43.2	42.76	43.1	43.1	43.26
	43.37	43.24	43.54	43.4	43.63	43.38	43.26	43.37	43.4	43.51
	42	17.55	42.04	42.34	42.62	42.1	42.14	42.25	42.51	42.06
tai 50 X 10	56.36	54.49	54.94	54.73	55.47	55.55	55.27	55.62	54.93	56.38
	56.85	56.58	55.9	57.02	55.99	56.11	57.27	57.33	56.71	56.68
	54.3	54.87	55.26	54.8	54.19	54.04	54.9	55.23	54.9	54.65
	55.82	54.85	55.18	55.24	55.35	55.8	55.71	55.58	55.76	56.19
	55.19	56.19	54.01	55.19	54.94	55.85	54.77	55.16	55.47	54.34
	55.04	57.47	56.21	56.55	56.77	56.02	56.83	57.58	56.68	55.57
	57.58	56.94	57.63	58.3	57.28	58.22	57.81	57.69	57.27	57.33
	56.61	55.99	56.89	56.69	55.52	56.8	56.79	56.3	57.61	56.11
	56.41	55.63	55.79	56.24	56.65	56.88	55.97	55.97	57.1	56.39
	56.71	57.28	57.11	56.91	56.53	57.72	58.06	57.58	57.94	57.66
tai 50 X 20	97.94	93.4	92.49	93.6	92.68	92.48	91.25	92.28	89.15	89.11
	89.54	89.67	92.04	89.95	88.61	88.25	87.91	87.58	89.47	86.16
	86.92	88.34	89.45	89.53	90.39	86.25	85.49	86.8	86.83	85.5
	86.25	86.41	84.07	83.96	85.02	87.1	85.91	85.15	87.03	86.58
	86.38	85.6	85.71	88.76	86.66	85.41	87.38	88.39	89.72	92.12
	87.28	87.99	91.01	90.56	87.89	88.67	87.86	85.83	90.65	86.67
	89.76	89.9	89.03	90.86	93.18	92.17	92.45	93.16	92.43	92.79
	90.54	95.24	93.45	93.38	94.91	97.44	96.8	92.45	92.98	92.59
	92.56	93.9	97.13	100.3	95.65	96.32	94.35	94.96	93.13	94.63
	93.37	95.22	94.93	99.42	103.1	103.4	103.9	96.02	98.48	102.1
tai 100 X 5	339.4	325.5	174.2	158.1	320.3	322.7	140.4	324.1	328.5	183
	306.9	307.8	288.2	288.1	286.1	287.4	318.6	320	314.5	291.8
	324.3	334.9	329.3	322.5	323.5	315	314	312.7	313.9	322.4
	317.6	315.2	314.5	328	331.4	321.4	324.7	322.6	314.5	317.3
	311.4	314.2	312.6	318	313.1	325.5	324.8	318.2	316.9	317.9
	318.4	316.2	317.7	193.8	320.5	325.8	317.6	317.5	315.3	318.8
	321.5	323.7	330.7	313.4	329.4	303.2	303.3	303.7	302	303.5
	341.1	333	340.9	339.3	330	326.8	323.9	342	314.8	221.6
	326.3	318.3	315	318.6	326.3	328.9	333	335.4	310.5	327
	325.7	331.8	332.4	330.5	339.1	331.2	318.9	325.6	346	318.7

Problem	REPLICATION NUMBER									
	R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
tai 100 X 10	491.6	475.3	489.3	492.8	432.6	429.8	440.9	501	473.9	437.4
	434	434.1	429	439.8	436.9	431.5	425.1	423.9	435.4	446.6
	430.6	437.1	423.9	427.4	430.4	430.6	423.1	423.3	432.7	439.3
	409.3	407	488.6	448.6	408.5	421.3	435.9	408.8	405.6	443.5
	466.2	473.3	420.5	438.1	415.9	417.7	425.3	427.5	416.2	412.7
	404.5	425.1	419.7	415.7	434.2	426.3	427.9	420.1	414.1	413.1
	439.2	432.8	426.9	417.8	411.4	416.8	418.9	423.3	414.6	445.9
	417.9	414	415	410.3	417.5	408.7	405.7	438.5	441.6	491.1
	475.3	491.3	481.5	477	465.2	439.4	437.2	437.6	436.1	448.2
	435.5	447.6	440.2	460.4	444.5	436.8	448.5	450.6	448.5	440.2
tai 100 X 20	653.8	665.1	663.1	684	652.4	655.7	656.3	666.2	653.3	665.2
	635.7	654.3	624.3	628.3	647.1	640.5	649.3	656.3	661.3	660.6
	680.9	671.1	661.4	652.7	660.2	659.7	676.5	668.1	671.3	657.1
	633.6	634.5	627.8	644.1	637.4	640.9	643.9	635	640.1	640.2
	675.8	687.5	672.1	687.5	678.6	655.1	673.4	684.2	671.4	659.8
	653.6	660.8	639.3	649.8	578.8	593.3	598.8	590.8	595.4	587.7
	582.6	585.9	582	584	590.2	579.9	581.3	582.4	597.7	583.3
	612.6	607	611	600.7	604.8	596.7	593.8	595.7	585	600.9
	595.7	592.3	585.9	601.7	593.8	597.6	600.9	597.1	600.2	593.3
	628.6	636.6	634.6	618.9	625.4	630.5	642.7	630.5	642.7	630.7
tai 200 X 10	3266	3264	3146	3154	3145	3151	3164	3151	3164	3158
	3071	3071	3080	3079	3063	3075	3081	3075	3081	3084
	3162	3158	3164	3170	3160	3184	3161	3168	3166	3167
	3701	3669	3669	3164	3158	3625	3630	642.7	3151	3071
	3071	3071	3158	3081	3084	3145	3151	3164	3075	3158
	3162	3158	4669	3166	3167	3063	3075	3081	3184	4669
	4701	4669	3079	3087	3071	3145	3151	3071	3125	3158
	3215	3101	3170	3160	3184	3063	3075	3162	3166	4669
	3164	3158	625.4	630.5	642.7	3160	3184	3001	3151	3071
	3158	3166	3167	3063	3075	3081	3184	3269	3075	3167
tai 200 X 20	4233	4197	4199	4239	4248	4223	4234	4238	4235	4256
	4135	4223	4183	4208	4178	4290	4236	4251	4217	4338
	4333	4197	4199	4119	4148	4111	4114	4338	4335	4356
	4135	4334	4181	4118	4208	4178	4290	4236	4251	4217
	4348	4333	4114	4118	4223	4181	4208	4135	4112	4183
	4208	4178	4290	4216	4251	4217	4118	4335	4356	4256
	4133	4197	4199	4119	4148	4111	4114	4138	4135	4156
	4135	4112	4181	4108	4178	4190	4116	4151	4117	4338
	4333	4197	4199	4339	4348	4333	4334	4338	4335	4356
	4135	4334	4183	4338	4108	4178	4190	4136	4151	4117

APPENDIX C

UPPER BOUND OF THE PROBLEMS

TAILLARD'S BENCHMARK									
Benchmark	**Prob. Set**	**N**	**M**	**Upper Bound**	**Benchmark**	**Prob. Set**	**N**	**M**	**Upper Bound**
tai 20 X 5	tai001	20	5	1278	**tai 50 X 5**	tai031	50	5	2724
	tai002	20	5	1290		tai032	50	5	2834
	tai003	20	5	1081		tai033	50	5	2621
	tai004	20	5	1293		tai034	50	5	2751
	tai005	20	5	1236		tai035	50	5	2863
	tai006	20	5	1195		tai036	50	5	2829
	tai007	20	5	1239		tai037	50	5	2725
	tai008	20	5	1206		tai038	50	5	2683
	tai009	20	5	1230		tai039	50	5	2552
	tai010	20	5	1108		tai040	50	5	2782
tai 20 X 10	tai011	20	10	1582	**tai 50 X 10**	tai041	50	10	2991
	tai012	20	10	1659		tai042	50	10	2867
	tai013	20	10	1496		tai043	50	10	2839
	tai014	20	10	1378		tai044	50	10	3063
	tai015	20	10	1419		tai045	50	10	2976
	tai016	20	10	1397		tai046	50	10	3006
	tai017	20	10	1484		tai047	50	10	3093
	tai018	20	10	1538		tai048	50	10	3037
	tai019	20	10	1593		tai049	50	10	2897
	tai020	20	10	1591		tai050	50	10	3065
tai 20X 20	tai021	20	20	2297	**tai 50 X 20**	tai051	50	20	3850
	tai022	20	20	2099		tai052	50	20	3704
	tai023	20	20	2326		tai053	50	20	3641
	tai024	20	20	2223		tai054	50	20	3723
	tai025	20	20	2291		tai055	50	20	3611
	tai026	20	20	2226		tai056	50	20	3681
	tai027	20	20	2273		tai057	50	20	3705
	tai028	20	20	2200		tai058	50	20	3691
	tai029	20	20	2237		tai059	50	20	3743
	tai030	20	20	2178		tai060	50	20	3756

TAILLARD'S BENCHMARK									
Benchmark	**Prob. Set**	**N**	**M**	**Upper Bound**	**Benchmark**	**Prob. Set**	**N**	**M**	**Upper Bound**
tai 100 X 5	tai061	100	5	5493	**tai 100 X 20 (cont.)**	tai086	100	20	6364
	tai062	100	5	5268		tai087	100	20	6268
	tai063	100	5	5175		tai088	100	20	6401
	tai064	100	5	5014		tai089	100	20	6275
	tai065	100	5	5250		tai090	100	20	6434
	tai066	100	5	5135	**tai 200 X 10**	tai091	200	10	10862
	tai067	100	5	5246		tai092	200	10	10480
	tai068	100	5	5094		tai093	200	10	10922
	tai069	100	5	5448		tai094	200	10	10889
	tai070	100	5	5322		tai095	200	10	10524
tai 100 X 10	tai071	100	10	5770		tai096	200	10	10329
	tai072	100	10	5349		tai097	200	10	10854
	tai073	100	10	5676		tai098	200	10	10730
	tai074	100	10	5781		tai099	200	10	10438
	tai075	100	10	5467		tai100	200	10	10675
	tai076	100	10	5303	**tai 200 X 20**	tai101	200	20	11195
	tai077	100	10	5595		tai102	200	20	11203
	tai078	100	10	5617		tai103	200	20	11331
	tai079	100	10	5871		tai104	200	20	11294
	tai080	100	10	5845		tai105	200	20	11259
tai 100 X 20	tai081	100	20	6202		tai106	200	20	11176
	tai082	100	20	6186		tai107	200	20	11368
	tai083	100	20	6252		tai108	200	20	11334
	tai084	100	20	6269		tai109	200	20	11192
	tai085	100	20	6314		tai110	200	20	11284

APPENDIX D

EXAMPLE OF A RUN'S EXPORTED DATA