Chapter 4

Particle Swarm Optimization for Single machine Total Tardiness Problem with Sequence-Dependent Setup Time

This chapter introduces an application of Particle Swarm Optimization algorithm (PSO) to the single machine job scheduling problem with the objective of minimizing total tardiness where the setup time is sequence-dependent. In order to apply the PSO concept to the scheduling problem, a particle is defined as multi-dimensional point in space. The ranking of position values of each particle enable the generation of a feasible job sequence, which represent the potential solution to the problem. The performance of PSO algorithm is compared with Ant Colony Optimization algorithm by Gagné *et al*. (2001), the best known solution approaches in the literature. Finally, PSO is extended to cover the scheduling problem proposed in Chapter 3.

4.1 Problem Description

The following notation will be used in the following sections.

X	set of N independent jobs, $X = \{x_1, x_2, ..., x_N\}$
N	total number of jobs
ST	sequence-dependent setup time matrix, **ST** = $[ST_{ji}]$
ST_{ji}	setup time of switching from job j to i
d_i	due date of job i
p_i	processing time of job i
C_i	completion time of job i
T_i	tardiness of job i, $T_i = max\ (0,\ C_i - d_i)$.

The single machine scheduling problem to minimize total tardiness with sequence-dependent setup times can be describes as follows.

There are N jobs to be processed on a single processor. All jobs are available at the beginning of planning period. Each job requires a processing time, p_i. The setup time (ST_{ji}) depends on the job sequence, or sequence-dependent setup. ST_{ji} is the setup time incurred when switching from processing job j to job i. The distinct due date d_i is known. The completion time of job i, C_i is equal to the completion time of job j, C_j plus sequence-dependent setup time from job j to job i, $C_i = C_j + ST_{ji}$. If the job completion time C_i is later than its due date, the tardiness of that job can be calculated as $T_i = \max(0, C_i - d_i)$.

4.2 Objective Function

The objective of the problem is to minimize total tardiness of all jobs. For sequence Q, the objective function can be written as:

$$f(Q) \quad = \quad \text{Min} \sum_{i=1}^{N} T_i \tag{4.1}$$

subject to

$$T_i \geq 0 \tag{4.2}$$

$$T_i \geq C_i - d_i \tag{4.3}$$

4.3 Application of PSO Algorithm to SMTST Problem

PSO algorithm is applied to solve the single machine total tardiness problem with sequence-dependent setup time. In order to apply PSO to the scheduling problem, each particle represents a multi-dimensional point in space. The main variables in PSO can be summarized as follows: $X_t[n][d]$ denotes the position value of dimension n^{th} of particle d^{th} in the swarm at iteration t. At each iteration t the position value of each particle is represented by $X_t[d] = \{X_t[1][d], X_t[2][d],....., X_t[N][d]\}$, where N is the total number of jobs to be scheduled. The whole swarm is the set of all particles, $X_t = \{X_t[1], X_t[2], …., X_t[D]\}$, whereas D is the number of agents or population size. Therefore, the position values of all particles in the swarm are represented by an $N \times D$ matrix. The particle will move to the search space by using its associated velocity value, $V_t[n][d]$. The $V_t[n][d]$ refers to velocity of dimension n^{th} of particle d^{th} in the swarm at iteration t. The best position with the best fitness value until iteration t will be identified as previous best position. The previous best position until iteration t is represented by $pbest_t[n][d]$ while the previous best fitness of particle d will be recognized as $pbest_t[d]$. Similarly, $gbest_t$ represents the dimension index of the global best particle of the whole swarm achieved so far. The pseudo code and step-by-step PSO algorithm for SMTST problem is presented below.

……

Pseudo code 4.1: PSO algorithm for SMTST

Start
Initialize Random $X^0[n][d]$**,** *popsize*, *maxiter*, χ, w^0, c_1, c_2, v_{max}, V^0_k, $pbest^0_k$ $gbest^0$ for all k
Do{
For { *d = 1 to popsize*
Solution representation: *Job sequence generation* (see *Step 2*)
Fitness value evaluation: calculate $f(Seq(X_t[d]))$ using Eq. 4.1
Apply Local Improvement Procedure: to some selected previous best (*pbest*) particles and global best (*gbest*) particle
a. Regroup and Resize
b. Local Improvement Procedure, or
c. Variable Neighborhood Search, or
and apply *Repairing algorithm* (where necessary)
Update: objective value after LIP, *pbest* and *gbest*
if $f(Seq(X_t[d])) < f(Seq(pbest_{t-1}[d]))$
$pbest_t[d] = f(Seq(X_t[d]))$
$pbest_t[n][d] = X_t[n][d]$
if $f(Seq(pbest_t[d])) < f(Seq(pbest_{t-1}[gbest_{t-1}]))$
$gbest_t = d$
Calculate velocity: using Eq. 4.6

Update position value: using Eq. 4.7

}End for

} while (termination)

End

……

Step of PSO Algorithm

Step 1. Initialization

The population of particles is initialized randomly. The continuous position values of each dimension of each particle are generated by using the following formula.

$$X_0[n][d] = X_{\min} + (X_{\max} - X_{\min}) \times U(0,1) \quad (4.4)$$

where, X_{min} and X_{max} are the pre-defined range of position value (in this case, X_{min} = 0.0, X_{max} = N). $U(0,1)$ is the uniform random number from 0 to 1. Similarly, their associated random velocities can be generated according to the following formula.

$$V_0[n][d] = V_{\min} + (V_{\max} - V_{\min}) \times U(0,1) \quad (4.5)$$

where, V_{min} and V_{max} are the pre-defined range of velocity value (in this case, V_{min} = 0.0, V_{max} = N).

Suppose, there are 4 jobs to be scheduled (n = 1,2,.., N and N = 4). The random position values of the first particle (d = 1) is shown in Table 4.1.

Table 4.1 Initial position values and velocities

X_0 [n][d]	X_0 [1][1]	X_0 [2][1]	X_0 [3][1]	X_0 [4][1]
	2.1	0.5	3.8	1.6
V_0 [n][d]	V_0 [1][1]	V_0 [2][1]	V_0 [3][1]	V_0 [4][1]
	1.3	-0.4	-1.8	3.2

Step 2. Job sequence generation

The technique similar to Tasgetiren *et al.* (2004) is used to translate the continuous position value to the job sequence. Each dimension of particle represents a corresponding job, i.e. dimension 1 represents job1. In order to generate the feasible solution to the problem, position values of each particle will be sorted in ascending order. The job with the lowest position value is the first job in the production sequence. This procedure is illustrated in Table 4.2.

Let $Seq(X_t[d])$ be the corresponding sequence at iteration t of particle $X_t[d]$. From Table 4.2, there are 4 jobs to be scheduled (N = 4). The position values of all dimensions are represented in the second row. The position value will be ranked according to its value. In this case, position value of dimension 2 or (X_t[2][1]) which has the lowest

position value, is ordered first. Therefore, job 2 will be the first job in the sequence, and so on.

Table 4.2 Solution Representation

$X_t[n][d]$	$X_t[1][1]$	$X_t[2][1]$	$X_t[3][1]$	$X_t[4][1]$
	2.1	0.5	3.8	1.6
Ranking	0.5	1.6	2.1	3.8
$SeqX_t[d]$	2	4	1	3

Step 3. Fitness value evaluation

Once the job sequences of all particles are constructed, fitness value of each particle will evaluated by calculating its objective function value $f(Seq(X_t[d]))$ using Eq. 4.1.

Step 4. Update previous best position value and previous best fitness value

Initially, the previous best position value and previous best fitness value of particle d^{th}, i.e. $pbest_0[n][d]$ and $pbest_0[d]$, are set equal to the initial position value and initial objective value of each particle, respectively. At each iteration t, the current fitness value $f(Seq(X_t[d]))$ will be compared with the previous best fitness value $f(Seq(pbest_{t-1}[d]))$. If $f(Seq(X_t[d]))$ is less than $f(Seq(pbest_{t-1}[d]))$, set $pbest_t[d]$ equal to $f(Seq(X_t[d]))$ and $pbest_t[n][d]$ equal to $X_t[n][d]$, which is the current position values of particle d^{th}.

Step5. Update global best's dimension index

The initial dimension index of global best particle, i.e. $gbest_0$, is the array index of particle which has the minimum value among $pbest_0[d]$. Consequently, the initial global best position value and the initial global best fitness value will be equal to $pbest_0[n][gbest_0]$ and $pbest_0[gbest_0]$, respectively. At each iteration t, the $gbest_t$ will be updated. By comparing a new previous best fitness value with the global best fitness value, if $f(Seq(pbest_t[d]))$is less than $f(Seq(pbest_{t-1}[gbest_{t-1}]))$, set $gbest_t = d$

Step 6. Velocity calculation

The velocity of each particle can be calculated by using the following formula.

$$V_{t+1}[n][d] = \chi\,[\, w_t \times V_t[n][d] + c_1 \times r_1 \times (pbest_t[n][d] - X_t[n][d]) + c_2 \times r_2 \times (pbest_t[n][gbest_t] - X_t[n][d])] \quad (4.6)$$

The velocity is limited by a predefined maximum velocity, V_{max}. If $V_{t+1}[n][d] > V_{max}$, set $V_{t+1}[n][d] = V_{max}$.

Step 7. Update position value

$$X_{t+1}[n][d] = X_t[n][d] + V_{t+1}[n][d] \quad (4.7)$$

Step 8. Update inertia weight

The inertia weight will be updated at each iteration according to the following formula.

$$w_t = w_t \times \alpha \tag{4.8}$$

where, α is the decrement factor.

Step 9. Termination

Repeat Steps 2 to 8 until stopping criteria is met. In this case the maximum number of iterations is used as the stopping criteria.

4.4 Modification and Improvement Techniques for PSO Algorithm

It has been reported in the literature that the original PSO has difficulty in controlling the balance between exploration (global investigation of a search space) and exploitation (fine-tune search around a local area). PSO is quickly converging to a local optimum within a few iterations. Three improvement techniques are proposed and tested in order to find the best technique to integrate into original PSO, which are Regroup and Resize techniques, Local Improvement Procedure, and Variable Neighborhood Search.

4.4.1 Regroup and Resize Techniques

PSO concept follows a collaborative population-based search, which models over the social behavior of particles. The particles normally remain in cluster, i.e., bird flocking and fish schooling. In order to overcome the fast convergence problem of PSO, the degree of local and global exploration need to be balanced. Regroup and resize technique are employed to facilitate the information sharing between groups and help PSO to escape from local optima. For these techniques, instead of only one group, the particles are divided into many groups. The number of groups is specified by the parameter, *numgroup*. The number of member in each group (*groupsize*) can be found by dividing the total number of populations (*popsize*) with the specified number of groups.

$$\text{groupsize} = \text{popsize} / \text{numgroup} \tag{4.9}$$

Each group of particle will explore its search space independently in order to facilitate global exploration. Each particle in each group will move to a new position according to its own previous experience and its own group experience. Therefore, many groups of particles will search for the solution in parallel. At each iteration, the overall best solution (*tbest*) of all groups can be obtained.

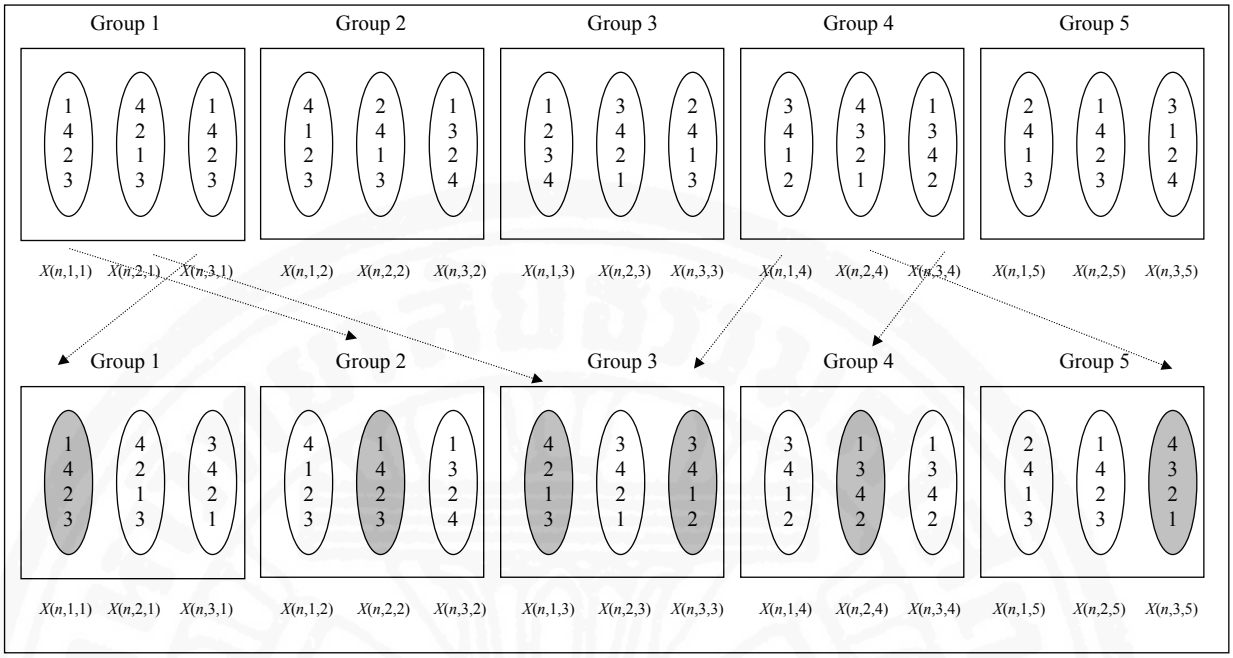

Figure 4.1 Regroup of particles

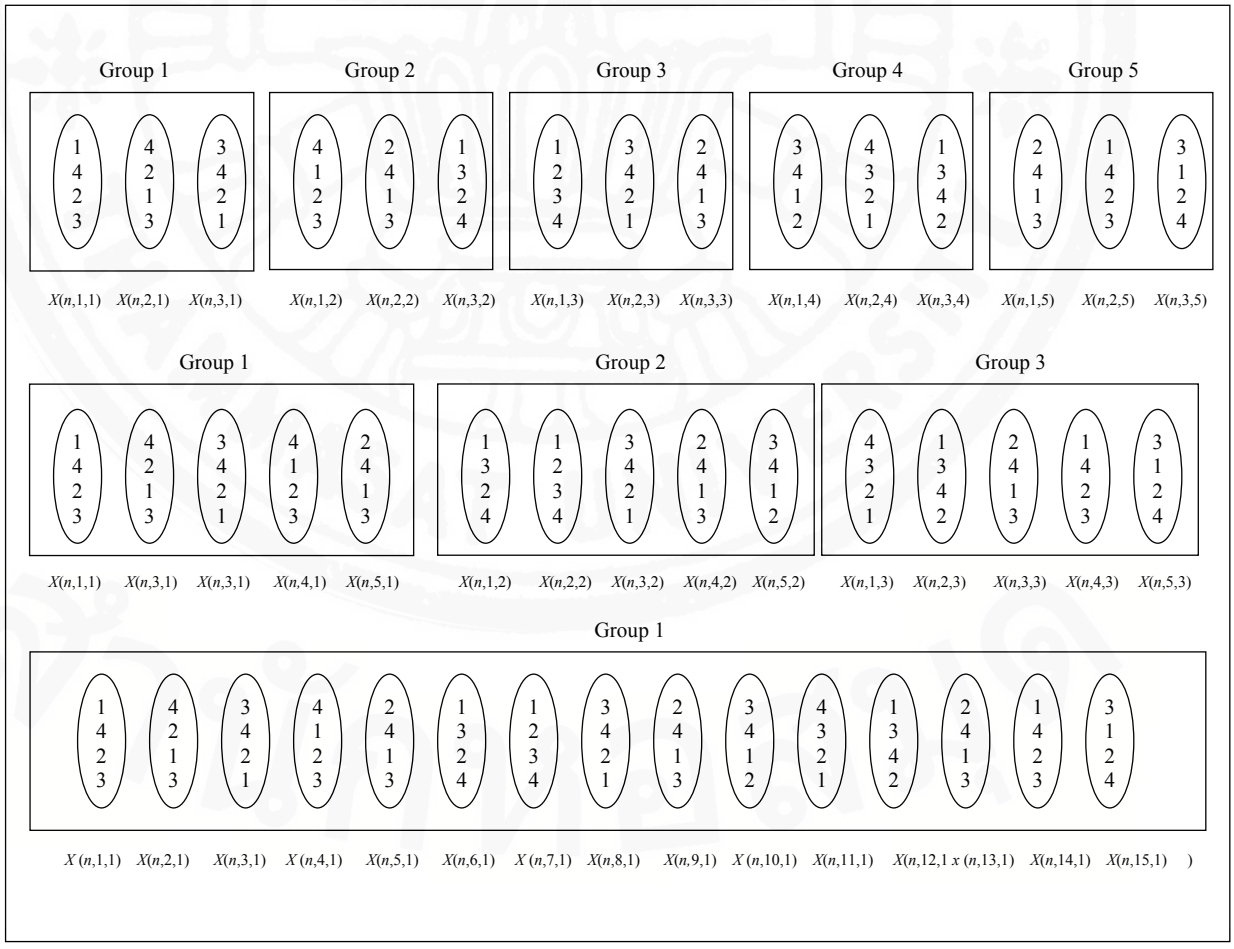

Figure 4.2 Resize of particles

When the solutions of all groups are not improved for a certain number of iterations, the particles will change their groups. This technique will make particles escape from local optima and move their positions into new search region. After searching for some period of time, the number of groups of particles will be reduced where the number of members in each group will be increased. Thus, the numbers of search regions are reduced. This technique will facilitate the local exploitation or allow the particles to search better solutions more precisely in the limited number of search space. The group size and number of member in each group will be dynamically updated. The schematic view of regroup and resize techniques can be found in Figure 4.1 and 4.2. In this figure, each particle is represented by $X(n,d,g)$, where n is the jobs to be scheduled, d is the d^{th} particle in group g

4.4.2 Local Improvement Procedure

The local improvement procedure (LIP) is a simple and effective search algorithm, which can efficiently apply to standard PSO. LIP is embedded in standard PSO in order to allow the particle to explore more to other search region and escape from the local minimum, hence, improve the solution quality.

LIP is applied to the sequence *gbest* solution and randomly selected sequence of *pbest* solution. The number of selected *pbest* solution can be found as follows:

$$\text{number of selected } pbest \text{ solution} = probselect_{lip} * popsize \tag{4.10}$$

where, $probselect_{lip}$ is the percentage of *pbest* solution from total population.

LIP consists of all-pairwise swap and all-pairs insertion schemes, which can be done in the following manner. LIP is started from the initial permutation of selected sequence. Firstly, the swap procedure is applied by selecting job A as the job in position i, $i = 1, 2,\ldots, N$. Then, do the forward move by selecting job B as the job in position $j = j + 1$ while j_0 is initially set equal to $j_0 = i + 1$. Compute the objective value after interchanging job A and job B. If the improving move is founded, that sequence is adopted as the new starting sequence. Repeat this procedure until $j = N$ and start the backward move. The procedure of backward move is the same as forward move except that job B is selected as the job in position $j = j - 1$ while j_0 is initially set equal to $j_0 = i + 1$. Repeat the backward swap until $j = 1$. After forward and backward move, the position of job A is moved by one unit, $i = i + 1$. The process is repeated until $i = N$. Secondly, the insertion procedure is applied to sequence after swap procedure. Job A and Job B are selected in the same manner as a swap procedure. Then remove job A and insert it in the position of job B, calculate and compare the objective value. Keep the new sequence if the objective value is improved. Finally, repeat all procedures until $i = N$. The overall processes of LIP are illustrated in Figure 4.3.

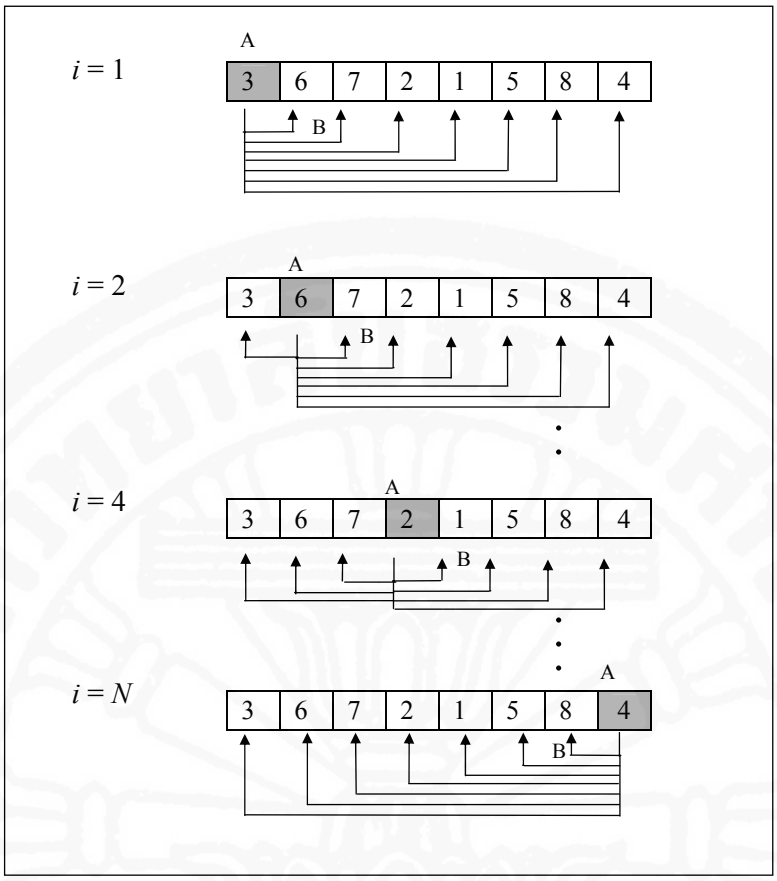

Figure 4.3 The selection scheme of job A and B of LIP

Pseudo code 4.2: Local Improvement Procedure for SMTST

```
Do {  Swap Procedure A= job in position i;
      Do {forward swap
          π = searchseq; sequence of gbest or selected pbest solution ;
          B= job in position j, j0 = i + 1;
          π1 =  SWAP(π, A, B);
          if (f(π1) <= f(searchseq))  then  searchseq = π1 ; repair(searchseq);
          j= j+1;
        } while (j <= N);
     Do { backward swap
          π = searchseq;
          B= job in position j, j0 = i - 1;
          π1 =  SWAP(π, A, B);
          if (f(π1) <= f(searchseq))  then  searchseq = π1 ; repair(searchseq);
          j = j - 1;
        } while (j >= 1);
     i = i + 1;
  } while (i <= N);
Do {  Insertion Procedure A= job in position i;
      Do {forward swap
          π = searchseq; sequence of gbest or selected pbest solution ;
```

```
        B= job in position j, j0 = i + 1;
        π1 =  INSERT(π, A, B);
        if (f(π1) <= f(searchseq))  then  searchseq = π1 ; repair(searchseq);
        j= j+1;
      } while (j <= N);
    Do { backward swap
         π = searchseq;
        B= job in position j, j0 = i - 1;
        π1 =  INSERT(π, A, B);
        if (f(π1) <= f(searchseq))  then  searchseq = π1 ; repair(searchseq);
        j = j - 1;
      } while (j >= 1);
   i = i + 1;
 } while (i <= N);
```

4.4.3 Variable Neighborhood Search

Variable neighborhood search (VNS) is a randomized local search algorithm which can easily applied to a combinatorial optimization problem. VNS is based on a simple principle, i.e., systematic change of neighborhood within the search. PSO combined with VNS is successfully applied to a single machine total tardiness problem and flowshop sequencing problem by Tasgetiren *et al*. (2004).

VNS is applied to the sequence *gbest* solution and randomly selected sequence of *pbest* solution. The number of selected *pbest* solution can be found as follows:

$$\text{No. of selected } pbest \text{ solution} = probselect_{vns} * popsize \qquad (4.11)$$

where, is the percentage of *pbest* solution from total population.

VNS is based on the *INSERT* and *SWAP* methods. The number *firstpoint* and *secondpoint* are integer random number between 1 and *N*. According to Pseudo code 4.3, the basic neighborhood technique of VNS, *INSERT* and *SWAP*, can be explained as follows.

INSERT (π, firstpoint, secondpoint) refers to removing the job at the $firstpoint^{th}$ dimension and inserting it in the $secondpoint^{th}$ dimension, $firstpoint^{th} \neq secondpoint^{th}$, in the permutation sequence π.

SWAP (π, firstpoint, secondpoint) means interchanging two jobs between $firstpoint^{th}$ and $secondpoint^{th}$ dimension, $firstpoint^{th} \neq secondpoint^{th}$, in the permutation sequence π.

Pseudo code 4.3: Variable Neighborhood Search for SMTST

```
outloop = 0;
Do { π0 = searchseq, sequence of global best or selected previous best solution ;
     firstpoint = rand(1,n);
     secondpoint = rand(1,n);
     π =  INSERT (π0, firstpoint, secondpoint);
     inloop = 0;
     Do {count = 0 ; maxmethod = 2;
```

```
    Do { firstpoint = rand(1,n);
         secondpoint = rand(1,n);
         if (count = 0)  then π1 =  INSERT (π, firstpoint, secondpoint);
        if (count = 1)  then π1 =  SWAP (π, firstpoint, secondpoint);
      if ( f(π1)  < f(π) ) then count  = 0 and   π = π1 ;
      else count++;
    } while (inloop < N * (N-1))
  outloop++;
  if (f(π) <= f(searchseq))  then  searchseq = π ; repair(searchseq);
} while (outloop < pvns * popsize);
```

4.4.4 Repairing Algorithm

The application of variable neighborhood search or local improvement procedure may result in the changing of job sequence. If this is the case, the associated particle position value must be reordered according to new job sequence. The repairing approach is illustrated in Figure 4.4. Suppose job2 and job 1 are interchanged, the position value of job2 and job1 are also changed.

$X[n][d]$	$X[1][1]$	$X[2][1]$	$X[3][1]$	$X[4][1]$
	2.1	0.5	3.8	1.6
Ranking	0.5	1.6	2.1	3.8
Job sequence	2	4	1	3
New sequence	1	4	2	3
New $X[n][d]$	0.5	2.1	3.8	1.6

Figure 4.4 Repairing approach after LIP and VNS

4.5 Hybrid-PSO with Heuristic Techniques

This section presents the procedures of combining PSO with the above mentioned improvement techniques as in Pseudo code 4.4 – 4.6. There are five types of Hybrid-PSO, which are:

- Hybrid-PSO with LIP
- Hybrid-PSO with VNS
- Hybrid-PSO with Regroup & Resize
- Hybrid-PSO with Regroup & Resize and LIP
- Hybrid-PSO with Regroup & Resize and VNS

……

Pseudo code 4.4: Hybrid-PSO with LIP and Hybrid-PSO with VNS

Start
Initialize: PSO parameters
Do{
For { *d = 1 to popsize*
Solution representation
Fitness value evaluation
Find: *pbest* and *gbest*

Random *(probselect$_{lip,vns}$) % of pbest and gbest particles to do LIP or VNS*
Apply *LIP* or *VNS* *to the selected pbest and gbest particles*
and apply *Repairing algorithm* (where necessary)

Update: objective value after *LIP* or *VNS*, *pbest* and *gbest*
Calculate velocity
Update position value
}End for
} while (termination)
End

……

……

Pseudo code 4.5: Hybrid-PSO with Regroup & Resize

Start
Initialize: PSO parameters
Initialize: population by dividing particles into many groups
Do{
For { *a = 1 to numgroup*
For { *b = 1 to groupsize*
Solution representation
Fitness value evaluation
Find: *pbest* & *gbest* of each group, and *tbest* (or *gbest* of all groups)

% Compare tbest with previous tbest
if (tbest < previous tbest) set regroupcounter = 0;
regroupcounter++;
if (regroupcounter = specified number of iteration to *regroup)*
do regroup;
set regoupcounter = 0;
resizecounter++;
if (resizecounter = specified number of iteration to *resize)*
do resize;
set resizecounter = 0;
and apply *Repairing algorithm* (where necessary)

Update: objective value after *Regroup & Resize*, *pbest* , *gbest*, and *tbest*
Calculate velocity
Update position value
}End for
}End for
} while (termination)
End

……

……

Pseudo code 4.6: Hybrid-PSO with Regroup & Resize and LIP or VNS

Start
Initialize: PSO parameters
Initialize: population by dividing particles into many groups
Do{
For { *a = 1 to numgroup*
For { *b = 1 to groupsize*
Solution representation
Fitness value evaluation
Find: *pbest* & *gbest* of each group, and *tbest* (or *gbest* of all groups)

Apply *LIP or VNS* *to the gbest particles of each group*
and apply *Repairing algorithm* (where necessary)
Update: *tbest*

% Compare *tbest* with *previous tbest*
if (tbest < previous tbest) set regroupcounter = 0;
regroupcounter++;
if (regroupcounter = specified number of iteration to *regroup)*
do regroup;
set regoupcounter = 0;
resizecounter++;
if (resizecounter = specified number of iteration to *resize)*
do resize;
set resizecounter = 0;
and apply *Repairing algorithm* (where necessary)

Update: objective value after *Regroup & Resize*, *pbest* , *gbest*, and *tbest*
Calculate velocity
Update position value
}End for
}End for
} while (termination)
End

……

4.6 Analysis of Computational Performance

To test the performance of the Hybrid-PSO algorithm, the benchmark problem of Rubin and Ragatz (1995) are used. The test problem is available on the website (http://mgt.bus.msu.edu/datafiles.htm). The algorithm has been coded in MATLAB program and all testes have been performed on a Pentium IV computer, 1.8 GHz CPU speed, 512MB RAM.

The characteristic of problem is varied according to the processing time variance (PTV), tardiness factor (TF), relative range of due date (DDR). The PTV, TF, and DDR have two levels, which are low and high, low and moderate, narrow and wide, respectively. The problem setting and the branch and bound solutions are shown in Table 4.3.

Table 4.3 Characteristic of test problems and B&B solutions

Problem	PTV	TF	DDR	B&B
Prob701	L	L	N	116
Prob702	L	L	W	0*
Prob703	L	M	N	27097
Prob704	L	M	W	15941
Prob705	H	L	N	234
Prob706	H	L	W	0*
Prob707	H	M	N	25070
Prob708	H	M	W	24123

* Optimal solution

4.6.1 Experiment 1: Evaluation of various heuristic techniques with PSO

This experiment is to test the computational performance of Hybrid-PSO in order to find the best heuristic to combine with original PSO. Two benchmark problems of Rubin and Ragatz (1995) (T701 and T703) with 45 jobs are selected for testing. The results are compared with the optimal results from Branch and Bound algorithm of Tan *et al*. (2000). The experiments were run 20 times for each case. For all test cases, the acceleration constants (c_1 *and* c_2) are set to 2.0 according to Eberhart and Shi (2001) and the constriction factor (χ) is set to 0.729 as suggested by Clerc (1999). The linearly varying inertia weight (w), which varies from 0.9 to 0.4, is used. The range of initial position value of particles is the random number from $U[0,N]$, whereas the initial velocity is the random number from $U[0,N]$. The maximum velocity (v_{max}) is set equal to N.

The following types of Hybrid-PSO are tested:

i. ***PSO*$_{lip}$ and *PSO*$_{backlip}$**
A Hybrid-PSO with *LIP* using $probselect_{lip}$ = 5%, *popsize* = 2**N*, *maxiter* = 50.

ii. ***PSO*$_{vns}$**
A Hybrid-PSO with *VNS* using $probselect_{vns}$ = 5%, p_{vns} = 10%, *popsize* = 2**N*, *maxiter* = 50.

iii. ***RPSO***

A Hybrid-PSO with regroup&resize using *popsize* = 240, *no. of iter to regroup*= 20, *no. of iter to resize* = 20, *numgroup* is set as 20, 16, 12, 8, 4, 2, and 1. The termination criterion is 20 iterations after the last *resize*.

iv. ***$RPSO_{lip}$***

A Hybrid-PSO with regroup&resize and LIP using *popsize* = 240, $probselect_{lip}$ = 5%, *no. of iter to regroup*= 10, *no. of iter to resize* = 10, *numgroup* is set as 20, 16, 12, 8, 4, 2, and 1. The termination criterion is 20 iterations after the last *resize*.

v. ***$RPSO_{vns}$***

A Hybrid-PSO with regroup&resize and VNS using *popsize* = 240, $probselect_{vns}$ = 5%, *no. of iter to regroup*= 10, *no. of iter to resize* = 10, *numgroup* is set as 20, 16, 12, 8, 4, 2, and 1. The termination criterion is 20 iterations after the last *resize*.

There are two versions of LIP in the experiment, with backward search and without backward search. It should be noted that, there is no backward search for in PSO_{lip} and $RPSO_{lip}$.

The computational results of all Hybrid-PSO are shown in Table 4.4. The results are compared with the Branch and Bound results from Tan *et al*. (2000). For the Branch and Bound results, the computation was stopped after a maximum of five million nodes were generated.

It can be seen from Table 4.4 that, Hybrid-PSO with LIP and backward search ($PSO_{backlip}$) has the least deviation from B&B solutions for both test problems. The Hybrid-PSO with VNS (PSO_{vns}) has the highest deviation from B&B solutions. Even though the inclusion of *Regroup & Resize* technique help to improve the solution quality of PSO_{vns} but the solution quality still much worse than $PSO_{backlip}$.

In addition, the average deviation from B&B solution of PSO_{lip} and $RPSO_{lip}$ are almost the same, but the computational time of $RPSO_{lip}$ is much larger. So, the Regroup & Resize technique is not worth to include in Hybrid-PSO.

Table 4.4 Computational results of Hybrid-PSO algorithm

Type of Hybrid-PSO	Test problem	Solution from B&B	Avg. Obj. value	Std.dev of obj.value	Deviation from B&B solution	Avg. time (sec.)
PSO_{lip}	T701	116	152.8	12.87	31.72	17.65
	T703	27097	27,256.85	65.45	0.59	11.65
$PSO_{backlip}$	T701	116	131.36	5.51	14.56	22.29
	T703	27097	26,927.15	89.78	-0.62	22.47
PSO_{vns}	T701	116	489.6	102.89	322.06	7.28
	T703	27097	31,561.85	588.07	16.47	6.59
RPSO	T701	116	520.6	108.06	348.79	78.73
	T703	27097	30,421.33	353.67	12.27	76.48
$RPSO_{lip}$	T701	116	152.33	11.77	31.32	528.45
	T703	27097	27,148.4	112.715	0.19	524.53
$RPSO_{vns}$	T701	116	285.17	15.406	145.83	104.20
	T703	27097	29,853.35	587.738	10.17	128.41

Moreover, comparing $PSO_{backlip}$ and PSO_{lip}, the average computational time of $PSO_{backlip}$ is slightly longer than PSO_{lip}, but the solution quality is much better. For the second test problem, the solution obtained from $PSO_{backlip}$ is even better than the B&B solution. It can be concluded that the best heuristic technique to combine with PSO is *LIP* with backward search.

4.6.2. Experiment 2: Study of the effect of different parameter settings

This experiment is to study the effect of effect of different parameter settings on the performance of *Hybrid-*$PSO_{backlip}$. Three out of eight benchmark problems of Rubin and Ragatz (1995) are selected for testing, which are T701, T704, and T708.

The experimental Design and Parameters Settings are as follow:

The *Hybrid-*$PSO_{backlip}$ has six main parameters, which are acceleration constants (c_1 & c_2), inertia weight (ω), constriction factor (χ), maximum velocity (V_{max}), population size (*popsize*), and probability of selection for local improvement ($probselect_{lip}$). The 2^k factorial design is used. This experimental design is commonly used when the experiments involve several factors and it is necessary to study the effect of the interaction between these factors.

Table 4.5 Parameters Settings

Factor	Parameters	Level (1)	Level (-1)
A	c_1 & c_2	$c_1 = 2$ & $c_2 = 2$	$c_1 = 1$ & $c_2 = 3$
B	ω	Decrease ω from 0.9 to 0.4 by ω* 0.975 at each iteration	$\omega = 0.9$ and non-decreasing
C	χ	0.729	1
D	V_{max}	N	12% * N
E	*popsize*	2 * N	4 * N
F	*probselect*	5%	15 %

Since 6 main factors are involved, the 2^6 full factorial design with 64 treatment combinations are tested. Each factor was set at two levels. All the settings follow the recommendation from the previous literature. For 2^6 factorial design, there are 6 main effects, 15 two-factor interactions, 20 three-factor interactions, 15 four-factor interaction, 6 five-factor interactions, and 1 six-factor interaction. In this experiment, the three-factor and higher order interactions are ignored. The experiment concerns only the main effects and two-factor interactions. The settings of all parameters are shown in Table 4.5, while the design of the experiment is shown in Table 4.6. Each treatment combination is run for five replications.

Table 4.6 Design of Experiment

RunOrder	A	B	C	D	E	F	RunOrder	A	B	C	D	E	F
1	1	-1	-1	1	1	1	33	-1	-1	1	-1	1	-1
2	1	1	-1	1	-1	1	34	1	-1	1	1	-1	-1
3	-1	1	1	-1	-1	-1	35	1	-1	1	1	1	1
4	1	-1	1	-1	1	1	36	-1	-1	1	1	1	1
5	-1	-1	1	1	1	-1	37	1	-1	-1	-1	1	1
6	-1	1	-1	-1	-1	-1	38	1	1	-1	1	1	1
7	1	1	-1	-1	1	1	39	-1	-1	-1	-1	-1	1
8	1	1	1	1	-1	-1	40	-1	-1	1	1	-1	1
9	1	1	1	1	-1	1	41	-1	1	1	1	1	-1
10	1	1	1	-1	1	-1	42	1	1	1	1	1	1
11	1	-1	-1	-1	1	-1	43	-1	1	1	1	1	1
12	1	1	-1	-1	-1	-1	44	-1	1	-1	-1	1	1
13	1	1	1	1	1	-1	45	1	1	1	-1	-1	1
14	-1	1	1	1	-1	-1	46	1	1	-1	-1	-1	1
15	-1	1	1	-1	1	1	47	-1	1	-1	1	1	1
16	-1	-1	-1	-1	-1	-1	48	-1	-1	1	-1	-1	1
17	-1	-1	-1	1	1	-1	49	-1	-1	-1	1	1	1
18	1	1	-1	1	-1	-1	50	-1	1	-1	-1	1	-1
19	1	-1	1	1	-1	1	51	1	-1	-1	-1	-1	-1
20	1	1	-1	-1	1	-1	52	-1	-1	-1	-1	1	1
21	-1	1	1	-1	1	-1	53	-1	1	-1	1	-1	-1
22	-1	1	-1	1	1	-1	54	-1	1	1	-1	-1	1
23	1	-1	-1	1	-1	1	55	-1	-1	-1	1	-1	-1
24	-1	1	-1	-1	-1	1	56	1	-1	1	-1	1	-1
25	1	1	1	-1	1	1	57	-1	-1	1	1	-1	-1
26	1	-1	-1	1	-1	-1	58	1	-1	-1	1	1	-1
27	1	1	1	-1	-1	-1	59	1	-1	1	1	1	-1
28	1	-1	-1	-1	-1	1	60	-1	1	1	1	-1	1
29	-1	-1	1	-1	1	1	61	1	-1	1	-1	-1	-1
30	1	1	-1	1	1	-1	62	-1	1	-1	1	-1	1
31	-1	-1	1	-1	-1	-1	63	-1	-1	-1	1	-1	1
32	-1	-1	-1	-1	1	-1	64	1	-1	1	-1	-1	1

Table 4.7 ANOVA result for problem T701

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	6	243.0	243.0	40.50	0.74	0.614
2-Way Interactions	15	685.0	685.0	45.67	0.84	0.633
Residual Error	298	16212.5	16212.5	54.40		
Lack of Fit	42	1811.7	1811.7	43.14	0.77	0.849
Pure Error	256	14400.8	14400.8	56.25		
Total	319	17140.5				

Table 4.8 ANOVA result for problem T704

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	6	356420460	356420460	59403410	1.00	0.426
2-Way Interactions	15	895688460	895688460	59712564	1.00	0.450
Residual Error	298	17712623083	1.7713E+10	59438332		
Lack of Fit	42	2498813654	2498813654	59495563	1.00	0.476
Pure Error	256	15213809429	1.5214E+10	59428943		
Total	319	18964732003				

Table 4.9 ANOVA result for problem T708

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Main Effects	6	91049	91049	15175	1.83	0.093
2-Way Interactions	15	130408	130408	8694	1.05	0.406
Residual Error	298	2472959	2472959	8299		
Lack of Fit	42	406917	406917	9688	1.20	0.198
Pure Error	256	2066042	2066042	8070		
Total	319	2694416				

The Minitab program was used to analyze the result. The ANOVA results of all test problems are shown in Table 4.7 to Table 4.9. The 95% confidence level is used in the experiment. The analysis of variance results indicated that there is no significant different among different parameters setting of both main effects and 2-way interaction for all test problems. Since the *p*-value of all test problems is greater than 0.05, it is fail to reject null hypothesis. This can be concluded that changing the parameters do not contribute to the improvement of the solution quality.

According to the ANOVA results, the estimated effects of factor A*B for T701 is significant (*p*-value ≅ 0.051). The results suggest the use of c_1 & c_2 =2 and non-decreasing inertia weight. Similarly, the estimated effects of factor F, A*B, and E*F for T708 are significant (*p*-value ≅ 0.03, 0.056, 0.045). The results suggest the use of c_1 & c_2 =2, decreasing inertia weight, *popsize* = 2**N*, and *probselect* = 15%.

Since the analysis of variance was not indicate the significant different for all three test problems, all parameters will be set at level 1. These parameters values are recommended by other authors.

4.6.3 Experiment 3: Comparison of Hybrid-$PSO_{backlip}$ with previous solution techniques

By using the above mentioned parameter settings, the experiment was proceeded to compare the result of *Hybrid-$PSO_{backlip}$* with *Ant Colony Optimization* (ACO) by Gagné *et al*. (2001), which is the best-known solution techniques in the literature. In order to compare the performance, the experiment was performed on all eight test cases (T701 to T708). Each experiment was run for 20 replications. The computational results of Hybrid-PSO_{lip} and ant colony optimization are shown in Table 4.10 and Table 4.11, respectively.

Table 4.10 Computational results of *Hybrid-PSO$_{backlip}$* with *B&B*

Problem	% to B&B				Average time (sec)
	Best	Median	Worse	Mean	
Prob701	-1.75	12.93	21.55	13.68	11.356
Prob702	0.0*	0.0*	0.0*	0.0*	11.186
Prob703	-1.43	-0.89	-0.40	-0.78	11.669
Prob704	-4.61	-3.72	-2.82	-3.77	11.197
Prob705	2.56	12.82	16.23	11.53	11.246
Prob706	0.0*	0.0*	0.0*	0.0*	11.136
Prob707	-4.45	-4.05	-3.57	-3.98	11.206
Prob708	-5.45	-4.99	-4.24	-4.91	11.297

Table 4.11 Computational results of *ACO* with *B&B*

Problem	% to B&B			Average time (sec)
	Best	Median	Worse	
Prob701	-11.2	-5.5	0.0	1197.95
Prob702	0.0*	0.0*	0.0*	11.05
Prob703	-1.6	-1.0	-0.5	2638.25
Prob704	-2.8	-1.1	-0.3	1886.65
Prob705	-5.1	5.6	15.4	1168.85
Prob706	0.0*	0.0*	0.0*	9.60
Prob707	-4.2	-3.3	-2.9	2524.65
Prob708	-3.2	-1.7	-0.6	2336.10

The results are represented by the percent deviation from Branch and Bound solutions. The known optimal solutions are indicated by an asterisk. It should be noted that the computation of branch and bound was stopped after a maximum of five million nodes were generated. The *Hybrid-PSO$_{backlip}$* demonstrates better results for problem T704, T707, and T708, whereas the *ACO* provides better results for problem T701, T703, and T705. For problem T702 and T706, the optimal solutions are achieved. So, the performances of both techniques are the same. The computational time cannot be compared since the computer with different CPU speed are used for running the experiment.

4.6.4 Experiment 4: Comparison of *Hybrid-PSO$_{backlip}$* with B&B2 for Single machine Earliness/ Tardiness Problem with Sequence-Dependent Setup Cost

Hybrid-PSO$_{backlip}$ is extended to cover single machine earliness/tardiness problem with sequence-dependent setup cost in order to evaluate the performance of PSO algorithm for the problem proposed in chapter 3. The minor modification is performed by changing only the objective function part to earliness/tardiness problem and sequence-dependent setup cost while keeping the same assignment technique and local improvement procedure. The experiment is performed on the same set of three test cases (where, number of jobs = 20, 30, 40). Two problems are generated for each case according to the parameter settings in Table 2.6. *B&B*2 is run for one replication since

the optimal solution is obtained while PSO is executed for 10 replications. The results are shown in Table 4.12.

Table 4.12. Computational results of $PSO_{backlip}$ and *B&B*2

No. of jobs	Test problem	*B&B*2	*Hybrid-* $MPSO_{backlip}$ (Avg. solution)	*Hybrid-* $BPSO_{backlip}$ (Best solution)	Deviation of $BPSO_{backlip}$ from *B&B*2 solution	Time of *B&B*2	Average time of $PSO_{backlip}$
20	T1	534	541.4	534	0	5.24	3.23
	T2	1345	1367.2	1345	0	4.45	3.84
30	T1	2647	2690.3	2659	12	83.26	14.23
	T2	2803	2847.9	2822	19	95.98	14.31
40	T1	3391	3467.6	3445	54	2829.12	22.12
	T2	4289	4353.8	4312	23	2825.35	22.45

It can be seen from the table that, the results of *Hybrid-*$PSO_{backlip}$ are satisfactory for all test problems. For small-sized problem, i.e. 20 jobs, within 10 replications *Hybrid-*$BPSO_{backlip}$ can achieved the optimal results. The deviation from *B&B*2 solution tends to be increased as the problem size increased. However, the deviation from *B&B*2 solution is acceptable for all problems. The average computational time of *Hybrid-*$PSO_{backlip}$ is much less than *B&B*2 and is not increased exponentially with the problem size. It can be seen that for large-sized problem (i.e., number of jobs > 40) PSO is more appropriate for this problem than branch and bound algorithm since the acceptable solution quality can be obtained in a short time. On the other hand, for small-sized to medium-sized problem, branch and bound is superior to PSO since it can guarantee optimal solutions in a reasonable time.

4.7 Concluding Remark

In this chapter, a meta-heuristic called Particle Swarm Optimization (PSO), is applied to the single machine scheduling problem with the objective of minimizing total tardiness where setup time is sequence-dependent. The concept of PSO is very simple, easy to implement, and computational efficient. Since this problem is NP-hard, the result of basic PSO is not satisfied. The PSO algorithm always converges too fast, which make the solution trapped in local optima.

The Hybrid-PSO by combining basic PSO with heuristic techniques are proposed, which are *Regroup & Resize* techniques, *Local Improvement Procedure* and *Variable Neighborhood Search*. The computational results reported that the combination of standard PSO and heuristic techniques can effectively enhance the searching efficiency and greatly improve the solution quality. Hybrid-PSO with LIP and backward search (*Hybrid-*$PSO_{backlip}$) is superior to other techniques in term of solution quality and computational time. The Hybrid-PSO with LIP is much better than Hybrid-PSO with VNS. On contrary, the Hybrid-PSO with VNS gives a satisfactory result for the weighted tardiness scheduling problem of Tasgetiren *et al*. (2004). This can be concluded that the performance of Hybrid-PSO is specific to the problem.

Then, the computational experiment was further conducted in order to find the best parameters for this problem, There are six main parameters in *Hybrid-*$PSO_{backlip}$ algorithm, which are acceleration constants (c_1 & c_2), inertia weight (ω), constriction

factor (χ), maximum velocity (V_{max}), population size (*popsize*), and probability of selection for local improvement (*probselect*). The 2^6 factorial design experiment was performed to evaluate six main effects and two-way interactions of these effects. The analysis of variance results (ANOVA) of the test problems have shown that there is no significant different among all treatment combinations in all test problems, The statistical analysis indicated that varying any parameters can not contribute to the improvement of the *Hybrid-PSO*$_{backlip}$ performance for the scheduling problem. Since the quality of solution is not changed by adjusting the parameters, all parameters will be set by following the recommendation by other authors and minimizes the computational time.

By using these parameters setting, the experiment was further carry out to compare the performance of *Hybrid-PSO*$_{backlip}$ and *ACO* algorithm (the best known solution techniques in the literature). The overall solution of have performed competitively with the best results obtain by Gagné *et al*. (2001). It can be concluded that *Hybrid-PSO*$_{backlip}$ and *ACO* perform differently with different problem characteristics. However, the experiment still limit to 45 jobs problem. For larger or smaller sized problem, the comparative performance still can not conclude.

Lastly, *Hybrid-PSO*$_{backlip}$ is modified and tested on single machine earliness/tardiness problem with sequence-dependent setup cost as proposed in Chapter 3. The solution quality of *Hybrid-PSO*$_{backlip}$ is satisfactory when compared with branch and bound solution while the computational time is much less than *B&B* especially for large-sized problems. It can be concluded that PSO is suitable for large-sized problems but B&B is more appropriate for small- to medium-sized problems.