J Intell Manuf (2010) 21:525–537
DOI 10.1007/s10845-008-0236-8

Optimization of parts scheduling in multiple cells considering intercell move using scatter search approach

Jiafu Tang · Xiaoqing Wang · Iko Kaku · Kai-leung Yung

Received: 25 March 2008 / Accepted: 19 December 2008 / Published online: 15 January 2009

Abstract This paper addresses the problem of parts scheduling in a cellular manufacturing system (CMS) by considering exceptional parts processed on machines located in multiple cells. To optimize the scheduling of parts as well as to minimize material handling between cells, the practice has to develop processing sequences for the parts in cells. A commonly chosen objective is to find part sequences within cells which results in a minimum tardiness. This paper proposes a nonlinear mathematical programming model of the problem by minimizing the total weighted tardiness in a CMS. To solve the mathematical model, a scatter search approach is developed, in which the common components of scatter search are redefined and redesigned so as to better fit the problem. This scatter search approach considers two different methods to generate diverse initial solutions and two improvement methods, and adopts the roulette wheel selection in the combination method to further expand the conceptual framework and implementation of the scatter search. The proposed approach is compared with the commercial solver CPLEX on a set of test problems, some of which are large dimensions. Computational results have demonstrated the effectiveness of this scatter search approach.

Keywords Cellular manufacturing · Parts scheduling · Exceptional parts · Scatter search

J. Tang · X. Wang (✉)
Department of Systems Engineering, Key Lab of Integrated Automation of Process Industry of MOE, Northeastern University (NEU), Shenyang, Liaoning 110004, People's Republic of China
e-mail: xqwang001@gmail.com

J. Tang
e-mail: jftang@mail.neu.edu.cn

I. Kaku
Department of Management Science and Engineering, Akita Prefectural University, Honjo, Japan

K.-l. Yung
Department of Industrial and Systems Engineering, The Polytechnic University of Hong Kong, Hung Hom, Kowloon, Hong Kong

Introduction

In a cellular manufacturing system, machines are grouped into several cells, where families of parts are produced in manufacturing cells or a group of machines that are physically close together but could be dissimilar in function. CMS incorporates the flexibility of job shops and the high production rate of flow shop lines. It overcomes the inefficiencies of traditional batch and job shop manufacturing through reductions in transportation distance/time, response time to orders, throughput time, setup time, work-in-process and finished goods inventory (Wemmerlov and Johnson 1997).

Implementing an effective CMS involves three phase, i.e. cell formation (CF), layout of the cell (CL), parts scheduling in cells (CPS). The first two phases are design problems, while the last phase is production planning problem. CF is the process of designing a cellular manufacturing system, and dealing with the identification of part family or families and the associated machine groups that constitute each cell (Askin et al. 1997). CF is the first problem of implementing a CMS. With the development of CMS in the earlier 30s, a large number of approaches have been proposed for cell formation (e.g., Billo et al. 1996; Selim et al. 1998; Liang and Zolfaghari 1999; Willow 2002; Vin et al. 2005; Safaei et al. 2007; Wei and Mejabi 2008). CL concerns the problem of laying out machines within each cell (intracell layout) and cells with respect to one another (intercell layout). A cell configuration can be two types, namely a flow-line layout

or a job-shop layout. In a flow-line layout cell, all parts are required to visit all machines in the same sequence. But parts may visit machines with diverse routings in a job-shop layout cell, and the job-shop layout is general form in a manufacturing cell. So this paper is to study CPS problem in job-shop layout cells. There have been some researches devoted to the CL decision (e.g., Hsu and Su 1998; Bazargan-lari et al. 2000; Wang 2001; Wu et al. 2007).

CPS emerges as a problem following the formation of manufacturing cells, and concerns the scheduling of parts in cells for production. CPS problem deals with the allocation of manufacturing resources over time to perform a collection of parts in cells, and it is a decision-making process that plays an important role in a CMS. The objective of CPS is to identify a sequence of parts in part families that minimizes some measures of effectiveness. These measures of effectiveness include total completion time, total weighted flow time, and total weighted tardiness, amongst others. The issues of CPS have been mentioned by some researchers in the literature. Most of them addressed CPS in a flow-line manufacturing cell. Baker (1990) addressed the problem of scheduling parts in a two-machine flow-line cell, and time lags or sequence-independent setup times were considered. Wemmerlov and Vakharia (1991) developed two heuristic procedures for part family scheduling in a flow-line manufacturing cell with three or more machines, and both procedures were extensions of flow shop scheduling to include a family sequence. Logendran et al. (1995) and Schaller (2000) investigated and compared the performance of several combined heuristics for scheduling parts within a part family in a flow-line cell. Reddy and Narendran (2003) proposed a heuristic for scheduling parts considering sequence-dependent setup times to improve the use of machines within a flow-line cell. Some meta-heuristics have been proposed for the problem. Skorin-Kapov and Vakharia (1993) developed six tabu search heuristic procedures, and Sridhar and Rajendran (1994) developed a genetic algorithm, and Venkataramanaiah (2008) developed a simulated annealing approach for scheduling of parts in a flow-line manufacturing cell for the objective of minimizing makespan. Gupta and Schaller (2006) proposed a branch-and-bound algorithm capable of solving the moderate sized problems with independent setup times in a flow-line manufacturing cell. A few of researchers addressed CPS in a job-shop manufacturing cell. Mahmood et al. (1990) developed dynamic scheduling heuristics to stress good due date performance while reducing overall set-up time in a job-shop cell. Ruben et al. (1993) described a broad-based simulation study of the performance of parts scheduling heuristics in a job-shop cell. Tsai and Li (2000) presented a due-date oriented scheduling heuristic algorithm for a job-shop CMS based on capacity constraint resource.

However, the contributions in their researches have been made to CPS problem in an isolated manufacturing cell, and the above researches assumed that all parts in a part family are processed in one cell and there is no intercell move. Ideally, all machines required for producing a part can be allocated to a cell.

However, this is hard to achieve in practice because parts processed in different cells may all require a particular machine (Arikan and Gungor 2005). Although the difficulty can be resolved by purchasing additional machines, it may not be economical to achieve such a cell independence. Thus, it is not unusual to transport parts between two or more cells. The parts required processing on machines in two or more cells are termed exceptional parts (EP). Such a movement of EP is usually called lntercell move. The interaction between cells caused by the existence of EP disrupts the CM philosophy of creating independent cells, but it is essential for enterprises to reduce production cost. So there is a need to study scheduling in the context of multiple cells (Krishnamoorthy and Kamath 2000). There has been relatively little research on CPS problem considering multiple cells and intercell move. Only two papers addressed the issue according to our limited knowledge. Yang and Liao (1996) assumed that a CMS consists of two cells and each part cannot have more than one operation in each cell, despite the fact that a CMS contains multiple cells and each part might have more than one operation in a cell. Solimanpur et al. (2004) assumed that each part must be processed through machines in the same order, despite the fact that each part may have a different required sequencing of operations in a CMS.

Hence, in this paper, CPS problem connotes that a large number of parts with diverse routings compete for time on

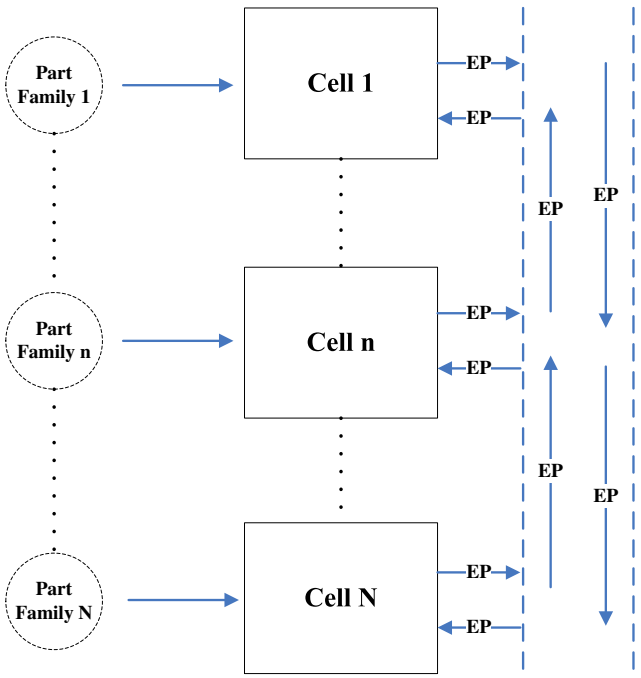


Fig. 1 ACMS including N cells and N part families

common machine facilities in a CMS including multiple cells. Figure 1 shows that a CMS contains N cells and N part families, and some parts require processing on machines in multiple cells.

Based on the above discussion, this paper addresses the problem of parts scheduling in a CMS by considering exceptional parts processed on machines in multiple cells. A mathematical model of the problem is proposed with the objective of minimizing the total weighted tardiness in a cellular manufacturing system.

Another significant contribution of this paper is to develop an efficient scatter search approach for the problem discussed above, and this paper is the first one to adopt scatter search procedure to solve CPS problem. Scatter search was first introduced in Glover (1977) as a heuristic for integer programming, and it is quite an established meta-heuristics that has received renewed attention over the last decades. From the meta-heuristics classification, scatter search can be viewed as an evolutionary algorithm that constructs solutions by combining others, and the goal of this methodology is to enable the implementation of solution procedures that can derive new solutions from combined elements (Marti et al. 2006). The scatter search methodology is very flexible, and each of its elements can be implemented in a variety of ways. Moreover, the algorithm incorporates procedures based on different strategies, such as diversification, local search, tabu search or path relinking (Glover 1998).

In this paper, a scatter search approach is developed, which redesigns the common components of scatter search and incorporates diversification generation method, local search method and other improvement mechanisms to provide a wide exploration of the search space through intensification and diversification. Two different methods of generating diverse initial solutions and two improvement methods are considered in the proposed approach, and the roulette wheel selection method based on the probability of total weighted tardiness values is adopted in the combination method. To validate the effectiveness of the proposed approach, a computational study is performed on a set of test problems with various dimensions. The computational results indicate that the proposed scatter search approach is able to successfully solve all the test problems with significantly improved solutions and less computational effort, even when the standard software behaved poorly.

The structure of this paper is given as follows: in section "Problem description and formulation", CPS problem is described and a mathematical model is proposed; in section "A Scatter search approach", the logic of the scatter search procedure is developed in detail; in section "Computational experiment and results", a series of experimental tests and computational results are presented; in section "Conclusions", conclusions of the study are presented.

Problem description and formulation

Problem description

A cellular manufacturing system consists of a number of manufacturing cells. Part family or families and the associated machine groups that constitute each cell have been identified. Machines have been laid out within each cell. Each type machine can process different parts and only one part at a time. Some parts require processing on machines in two or more cells. The intercell move time between different cells may be different for each part. Compared with the intercell move time, the intracell move time of parts is negligible. Each part must be processed on each machine at most once. Each part has a due date representing a desired completion time and a sequence of operations processed in the given order. The processing time for operations of each part type on a machine is known and fixed. Set-up times for the operations on the machines are independent of operation sequence and are included in the processing times. Preemption of each operation is not permitted. Assumed that all parts are released at time zero and there is no shortage of materials, machine breakdown and absence of operators. In order to avoid tardiness penalties, including the possibility of losing customers, the addressed problem is to develop processing sequences for the parts within cells which results in the minimum total weighted tardiness.

For the convenience to illuminate the problem, a simple instance is given. The part–machine load matrix in a CMS is shown in Table 1, and data include the processing time for each operation and the routing of each part through each machine. The machine groups and part families assigned to each cell are displayed in Table 2. Since the intracell move time of parts is minor compared with the intercell move time, the intracell move time is negligible. The intercell move time is set to 2.5 seconds for each part. Then the scheduling of production in the CMS is obtained, and one feasible solution of the problem represented by a Gantt chart can be given in Fig. 2.

List of symbols

The following notations are adopted to describe the above problem:

Indices:

j = index for part types ($j = 1, \ldots, J$)
i = index for operations required by part j ($i = 1, \ldots, I_j$)
k = index for machine types ($k = 1, \ldots, K$)
l = index for cells ($l = 1, \ldots, L$)

Parameters:

O_{ij} = ith operation of part j
P_{ij} = standard time to process operation O_{ij}

Table 1 Part–machine load matrix

Machines	Parts							
	P1	P2	P3	P4	P5	P6	P7	P8
M1	0.8			4.4			0.8	
M2			2.3			0.6	1.2	
M3					1.6	1.5		0.7
M4	2.1	1.7			3.2			
M5	1.8	2.5		1.1				
M6					0.9	0.4		2.2
M7		0.6	1.2				2.3	
Routing	4-1-5	5-7-4	7-2	1-5	4-3-6	3-6-2	7-2-1	6-3

Table 2 Machine groups and part families assigned to each cell

	Machines in cells			Parts in cells		
	Cell 1	Cell 2	Cell 3	Cell 1	Cell 2	Cell 3
Cell formation	M3, M6	M1, M4, M5	M2, M7	P5, P6, P8	P1, P2, P4	P3, P7

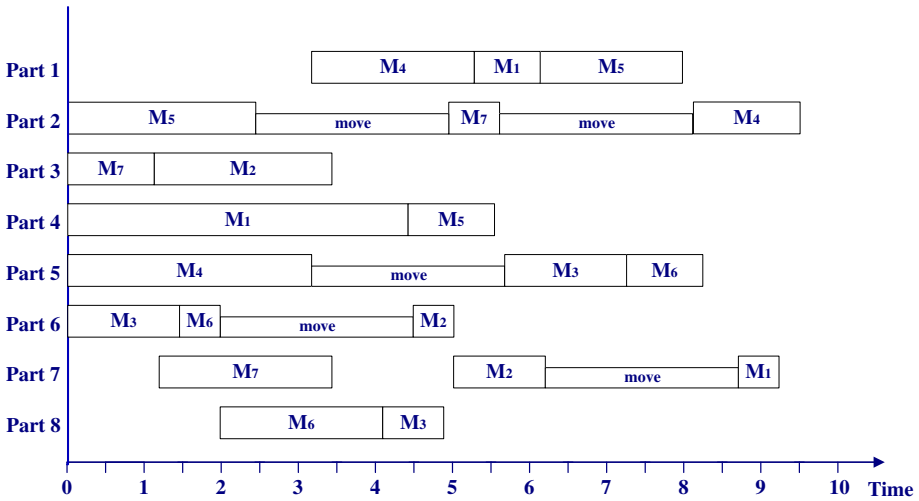


Fig. 2 Gantt chart of a feasible solution to the scheduling problem

M_j = intercell transfer time of part j in unit distance
$H_{kk'}$ = transportation distance between the cell including machine k and the other cell includingmachine k'
R_j = set of pairs of operation (i, i') for part j, where operation O_{ij} precedes operation $O_{i'j}$
D_j = due date of part j
W_j = weight of tardiness penalty for part j

$$\alpha_{ijk} = \begin{cases} 1, & \text{if operation } O_{ij} \text{ is required processing on machine } k \\ 0, & \text{otherwise} \end{cases}$$

$$\beta_{kk'} = \begin{cases} 1, & \text{if machine } k \text{ and machine } k' \text{ are arranged in the same cell} \\ 0, & \text{otherwise} \end{cases}$$

Decision variables:
X_{ij} = starting time of processing for operation O_{ij}
C_j = completion time of part j
T_j = tardiness of part j

Objective function and constraints

Following the problem description and notation given in sections "Problem description" and "List of symbols", the mathematical model of the problem of parts scheduling in a CMS considering intercell move is presented below.

Meeting due dates has always been one of the most important objectives in scheduling management. In order to avoid tardiness penalties, including the possibility of losing customers, the objective of the addressed scheduling problem is to minimize the total weighted tardiness in a CMS, and can be expressed mathematically as follows:

$$\text{Minimize} \quad \sum_{j=1}^{J} W_j T_j \tag{1}$$

The actual production has many peculiar characteristics and must be subject to some constraints. We describe mathematically the characteristics and constraints below.

Constraint (2) states that an operation cannot be started before its preceding operation is completed and the part is transported to the corresponding machine in a cell. When machines k and k' are assigned in the same cell, the intercell move time of each part is zero.

$$\begin{aligned} &\alpha_{i'jk'} X_{i'j} - \alpha_{ijk} \left(X_{ij} + P_{ij}\right) \geq (1 - \beta_{kk'}) M_j H_{kk'} \\ &\forall i, i', j, k, k', i \neq i', k \neq k', \\ &\alpha_{ijk}\alpha_{i'jk'} = 1 \text{ and } \left(i, i'\right) \in R_j \end{aligned} \tag{2}$$

Constraint (3) ensures completion time of part j is larger than or equal to the completion time of all operations of part j. This constraint set can be included in the mathematical model when minimizing machine idle is desired.

$$C_j - \sum_{k=1}^{K} \alpha_{ijk} X_{ij} \geq \sum_{k=1}^{K} \alpha_{ijk} P_{ij}, \quad \forall i, j \tag{3}$$

Constraint (4) is the constraint of the operational sequence of the operations that are processed on the same machine k, and $\vee$ denotes OR. This constraint set imposes the requirement that no two operations are processed on a machine at any time.

$$\begin{aligned} &\left(\alpha_{ijk} X_{ij} \geq \alpha_{i'jk} X_{i'j'} + \alpha_{i'jk} p_{i'j'}\right) \vee \\ &\left(\alpha_{i'jk} X_{i'j'} \geq \alpha_{ijk} X_{ij} + \alpha_{ijk} p_{ij}\right) \\ &\forall i, i', j', j, k, \alpha_{ijk}\alpha_{i'j'k} = 1, i \neq i', j \neq j' \end{aligned} \tag{4}$$

Constraint (5) gives the definition of tardiness for each part. This constraint can only be included in the model minimizing total weighted tardiness.

$$T_j = \max\left\{C_j - D_j, 0\right\}, \quad \forall j \tag{5}$$

Constraint (6) is the integrality requirement. The nonnegativity constraints for X_{ij}, C_j and T_j are altogether specified.

$$X_{ij}, C_j, T_j \geq 0, \quad \forall i, j \tag{6}$$

Summarizing the objective and all the constraints, a nonlinear mathematical model (CPSM) of the proposed problem can be attained. CPSM is a fractional nonlinear mathematical programming model that is neither convex nor concave, and it is difficult to be solved by traditional methods. Moreover, as the number of decision variables and constraints increase substantially with increased number of operations, parts, machines and cells, the computation will also become more complex and difficult. Hence, an efficient scatter search approach is developed for the above model with greatly reduced computational effort in next section.

A scatter search approach

In this paper, a scatter search approach is developed to determine a near-optimum strategy for the above problem, and it includes the general 5 steps (Laguna and Marti 2003) and other heuristic techniques. The framework of the proposed scatter search approach is presented in Fig. 3, and the detailed, step-by-step description of its related implementations will be presented in the following paragraphs.

Generating diverse initial solutions

Representation

In order to solve CPS problem by scatter search, the first task is to represent a solution of the problem. An operation-based representation is adopted that uses an unpartitioned permutation with multiple repetitions of part numbers (Cheng et al. 1996). This formulation names all operations for a part with the same symbol and then interprets it according to the order of occurrence in the representation. Each part appears in the permutation many times and each repeating does not indicate a concrete operation of a part but refers to a unique operation which is context-dependent. The length of the representation is the total number of operations required by all parts in a CMS. Figure 4 shows the encoding of a solution. As it is shown, n is the length of the encoding, and p_j denotes part j, and p'_k denotes the part in kth position in the permutation, and a_{kj} is the sum of times of part j appearing before the $(k+1)$ th position in the permutation. a_{kj} is determined as follows:

$$a_{kj} = \begin{cases} a_{k-1,j} + 1 & \forall j,\ k \geq 1 \text{ and } p'_k = p_j \\ a_{k-1,j} & \forall j,\ k \geq 1 \text{ and } p'_k \neq j \\ 0 & \forall j,\ k = 0 \end{cases} \tag{7}$$

Accordingly, part p'_k in the encoding indicates a unique operation $O_{a_{kj}p_j}$, which presents a_{kj}th operation required by p_j. Meanwhile, any permutation of the representation always yields a feasible schedule in a CMS. Based on the representation, an initial solution can be generated randomly by an encoding method, and the encoding method of the problem is presented in Algorithm 1.

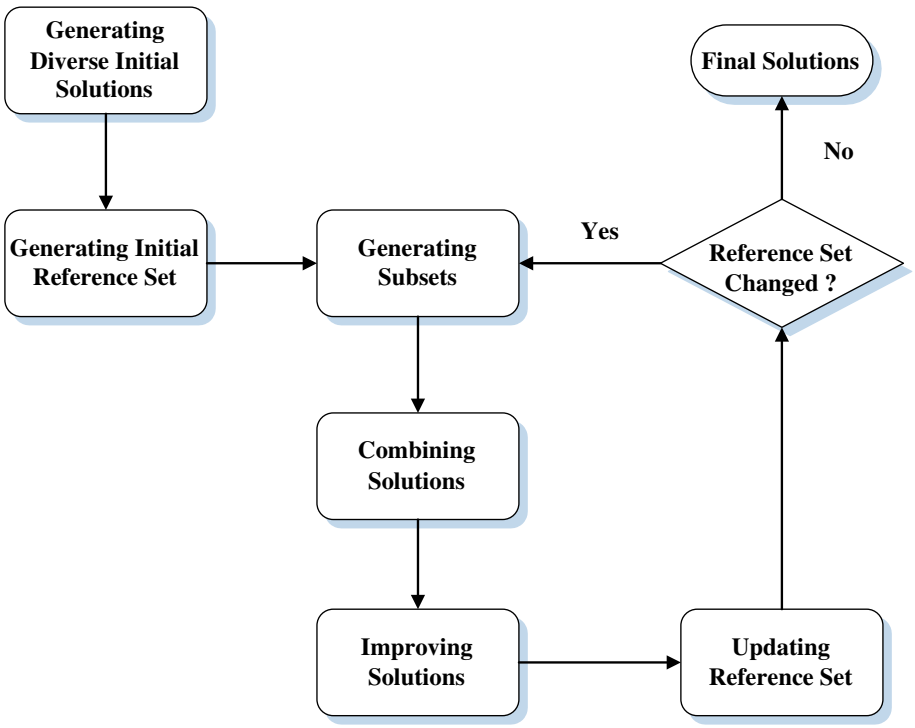


Fig. 3 Framework of the scatter search approach

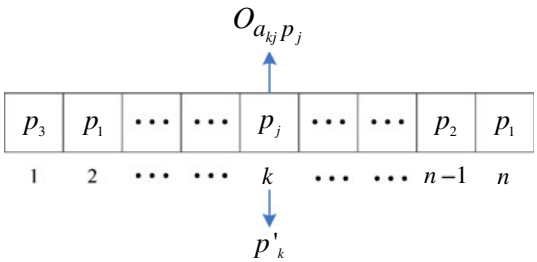


Fig. 4 The encoding of a solution

Algorithm 1: The encoding method of the CPS problem

Step 1:	Let **C** represent an initial solution of the CPS problem, where each element of the solution is empty; Let set **P** contain all parts;
Step 2:	Calculate *len*, which is the total number of operations required by all parts in a CMS;
Step 3:	Generate a sequence $\mathbf{A} = (1, 2, \ldots, len)$, and **B** is a random permutation of **A**;
Step 4:	Arbitrarily select a part $p_j (p_j \in \mathbf{P})$, and delete p_j from **P**;
Step 5:	Calculate I_j, which is the total number of operations required by p_j; Let $i \leftarrow 1$;
Step 6:	Let $\mathbf{C}(\mathbf{B}(i)) \leftarrow p_j$; Let $i = i + 1$;
Step 7:	If $i \leq I_j$, go to Step 6; If $i > I_j$ and $\mathbf{P} \neq \emptyset$, go to Step 4; If $i > I_j$ and $\mathbf{P} = \emptyset$, stop.

The decision variable X_{ij} is determined by a decoding method under consideration. Suppose $X_{a_{kj}p_j}$ presents starting time of processing for operation $O_{a_{kj}p_j}$. The decoding method of the problem is presented in Algorithm 2.

Diversification generation method

Diversification generation method is used to generate a collection of diverse initial solutions (or seed solutions) while keeping these solutions uniformly distributed in the feasible region. In this paper, two different methods of generating diverse initial solutions are considered. The first one is called the random generation method (RGM), which is to generate initial solutions randomly and is adopted widely in the original scatter search approach. The second one is called the permutation generation method (PGM), which applies and improves diversification generator proposed by Glover (1998).

Algorithm 2: The decoding method of the CPS problem

Step 1:	Let $k \leftarrow 1$;
Step 2:	Determine the part in kth position in the encoding, p_j, and calculate a_{kj} based on the Eq. 7;
Step3:	Find out machine m processing operation $O_{a_{kj}p_j}$ based on technological constraints;
Step 4:	If the operations sequence on machine m is not empty, go to Step 6;
Step 5:	If $a_{kj} = 1$, $X_{a_{kj}p_j}$ is zero; otherwise, $X_{a_{kj}p_j}$ is equal to $CT\left(O_{a_{kj}p_j}\right)$ that is obtained adding completed time of processing for $(a_{kj} - 1)$ th operation of part p_j and transfer time of part p_j transported to machine m. Go to Step 8;
Step 6:	Based on the operations sequence on machine m, determine pm that is the position of operation $O_{a_{kj}p_j}$ inserted into the operations sequence;

Algorithm 2: continued

Step 7:	If $pm = 1$ and $a_{kj} = 1$, $X_{a_{kj}p_j}$ is zero; If $pm = 1$ and $a_{kj} > 1$, $X_{a_{kj}p_j}$ is $CT\left(O_{a_{kj}p_j}\right)$; If $pm > 1$ and $a_{kj} = 1$, $X_{a_{kj}p_j}$ is equal to $CE\,(pm-1)$ that is completed time of processing for the operation in the $(pm-1)$ th position of the operations sequence on machine m; If $pm > 1$ and $a_{kj} > 1$, $X_{a_{kj}p_j}$ is the maximum value of $CT\left(O_{a_{kj}p_j}\right)$ and $CE\,(pm-1)$;
Step 8:	If $k \le n$, $k = k+1$ and go to Step 2; otherwise, stop.

Definition 1 Given a sequence P = (1, 2, . . . , n), the subsequence of P is defined as: P(h:s) = (s, s + h, s + 2h,. . .,s + rh), where h is a positive integer; s is a positive integer between 1 and h; r is the largest nonnegative integer such that s + rh ≤ n.

Definition 2 Given a sequence P = (1, 2,. . .,n), and h ≤ n, the permutation P(h) = (P(h:h), P(h:h − 1), . . ., P(h:1)).

Definition 3 Given two permutations of equal length, C = (c_1, . . ., c_n) and P(h), L(C | P(h)) is a permutation of C, where each element is filled by element in the permutation C according to the order of the permutation P(h), and the length of permutation L(C | P(h)) is equal to the length of C.

For example, given a set P = (1, 2, 3, 4, 5,6, 7, 8), choose h = 3, the permutation P(3) can be represented as, P(3) = (3, 6, 2, 5, 8, 1, 4, 7) (Glover 1998). And, for the given sequence C = (a, b, c, d, e, f, g, h), the permutation L(C | P(3)) can be represented as, L(C | P(3)) = (c, f, b, e, h, a, d, g).

Based on the above definition, PGM is presented in Algorithm 3.

Algorithm 3: The permutation generation method

```
Begin
  Let i ← 1;
  Let N1 be equal to the number of initial solutions
  While i ≤ N1 do
      Arbitrarily generate a parts permutation to represent
      a feasible solution s = (s1, . . . , sn);
      Set N2 = Length(s)/M, and M is a random
      number range from 3 to Length(s)/2;
    For j = 1 to N2 do
      Generate the permutation P(j) based on Defined
      2;
      Generate the permutation L(s| P(j)) based on
      Defined 3;
      i ← i + 1,   j ← j + 1;
      If (i ≤ N1) then
        Take L(s| P(j)) as one solution;
      End If
    End For
  End While
End
```

Generating initial reference set

The reference set is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by way of applying the combination method (Marti et al. 2006). The reference set is constructed in the initial step, and it consists of b_1 high quality solutions and b_2 diverse solutions from the diverse initial solutions. The size of the reference set is denoted by $b = b_1 + b_2$. The b_1 high quality solutions are selected based on the objective function values, i.e. the solutions with less total weighted tardiness values are selected as high quality solutions. In order to find b_2 diverse solutions, a diversity measure needs to be defined. The distance calculated via the l_1-norm is used as a diversity measure between two solutions (Burcu and Halit 2006). That is

$$d(\mathbf{s_1}, \mathbf{s_2}) = \sum_{i=1}^{n} \left| s_1^i - s_2^i \right| \tag{8}$$

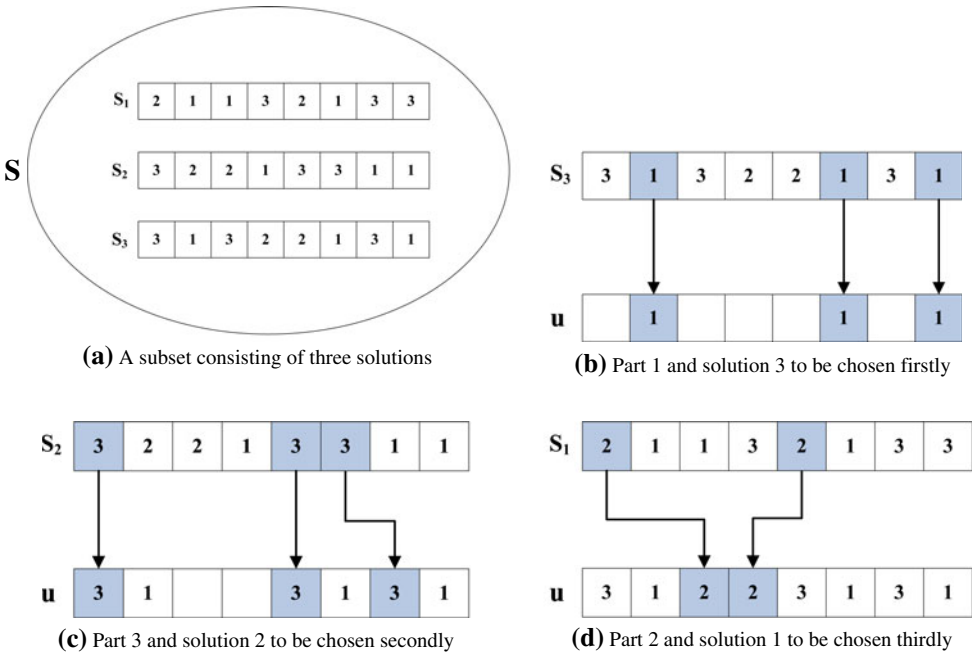
where $d(\mathbf{s_1}, \mathbf{s_2})$ connotes the minimum distance between two solutions $\mathbf{s_1} = (s_1^1, \ldots, s_1^n)$ and $\mathbf{s_2} = (s_2^1, \ldots, s_2^n)$.

After high quality solutions in the reference set are included, the minimum of the distances to the solutions in reference set is computed for each solution in the initial population. Then, the solution with maximum of those minimum distances is selected. This solution is added to reference set and deleted from the initial population. Repeat the manner until b_2 diverse solutions are found. The resulting reference set contains b_1 high quality solutions and b_2 diverse solutions.

Generating subsets

The subset generation method operates on the reference set to produce a subset of its solutions as a basis for creating combined solutions. There are several systematic ways to generate these subsets. The number of subsets has an impact on the overall duration of the algorithm. To decrease the overall duration of the algorithm and simplify the subset generation method, a procedure for generating subsets of reference solutions is adopted (Burcu and Halit 2006). The strategy expands pairs into subsets of larger size while controlling the total number of subsets to be generated, and avoids the extreme type of process that creates all the subsets of size 2, then all the subsets of size 3, and so on until reaching the subsets of size $b - 1$ and finally the entire reference set. The four types of subset are presented as follows:

- Subset type 1: all 2-element subsets.
- Subset type 2: 3-element subsets derived from the 2-element subsets by augmenting each 2-element subset to include the best solution not in this subset.
- Subset type 3: 4-element subsets derived from the 3-element subsets by augmenting each 3-element subset to include the best solutions not in this subset.
- Subset type 4: the subsets consisting of the best i elements, for $i = 5$ to b.



(a) A subset consisting of three solutions

(b) Part 1 and solution 3 to be chosen firstly

(c) Part 3 and solution 2 to be chosen secondly

(d) Part 2 and solution 1 to be chosen thirdly

Fig. 5 A example of the solution combination method

Combining solutions

The solution combination method is used to transform a given subset of solutions produced by the subset generation method into a combined solution. The roulette wheel selection method based on the probability of total weighted tardiness values is adopted in this combination method. Let **S** be a subset under consideration for generating a new solution **u**, and assume that **S** contains the solutions $\{\mathbf{s_1}, \ldots, \mathbf{s_n}\}$. In order to find the value of each element u_i in **u**, a solution combination method is proposed in Algorithm 4.

As shown in Fig. 5a, considering a subset **S** consists of three solutions, $\mathbf{s_1}$, $\mathbf{s}_2$ and $\mathbf{s}_3$. Percentages of roulette wheel based on the weighted tardiness values of solutions are calculated, and selection probability of each solution is corresponding to its percentage. Assuming part 1 and solution $\mathbf{s}_3$ are chosen firstly, a temporary solution **u** is obtained in Fig. 5b. Then part 3 and solution $\mathbf{s}_2$ are chosen secondly, a temporary solution **u** is obtained in Fig. 5c. Finally, part 2 and solution $\mathbf{s_1}$ are chosen, a new solution **u** can be obtained in Fig. 5d.

Improving solutions

The improvement method is used to transform the solution generated into an enhanced solution via a local search procedure. If the improvement method yields a better value than the one from the original solution, the new solution will replace the original solution. If no improvement has been found after the local search, no replacement will be made. A scatter search procedure can be implemented without improvement method, but it is essential for high quality solutions. Assuming that one solution $\mathbf{u} = (u_1, \ldots, u_n)$, and u_i represents the ith element of the solution. Two different methods for improving solutions are considered below.

Algorithm 4: The solution combination method

Step 1:	Let set **P** contain all parts;
Step 2:	Calculate total weighted tardiness values of the solutions in subset **S**;
Step3:	Arbitrarily select a part $p(p \in \mathbf{P})$, and delete p from **P**;
Step4:	Select a solution $\mathbf{s}_k$ with roulette wheel selection method based on the probability of total weighted tardiness values of the solutions in the subset;
Step 5:	Find out elements whose value is equal to p in the solution $\mathbf{s}_k$, and mark their positions;
Step 6:	If elements in the corresponding positions in **u** are empty, place p to the elements; otherwise, place p to empty elements that are nearest to the elements;
Step 7:	If $\mathbf{P} \neq \emptyset$, go to Step 3; otherwise, stop.

The first improvement method (IM1) mainly keeps the same sequence information of the original solution by only relocating one randomly selected element to the neighborhood of the other randomly selected element. First, one element u_j is randomly selected from **u**, followed by randomly choosing another element u_i. If u_i is placed in sequence before u_j, the new solution will be generated by placing u_i immediately after u_j after the improvement method. If, on the contrary, u_i is placed in sequence after u_j, the new solution will be generated by placing u_i immediately before u_j after the improvement method. For example, shown in Fig. 6a, for the above new solution **u**, the fourth element is randomly selected from **u**, and then the seventh element is randomly selected. The new solution $\mathbf{u}'$ is obtained after the improvement method.

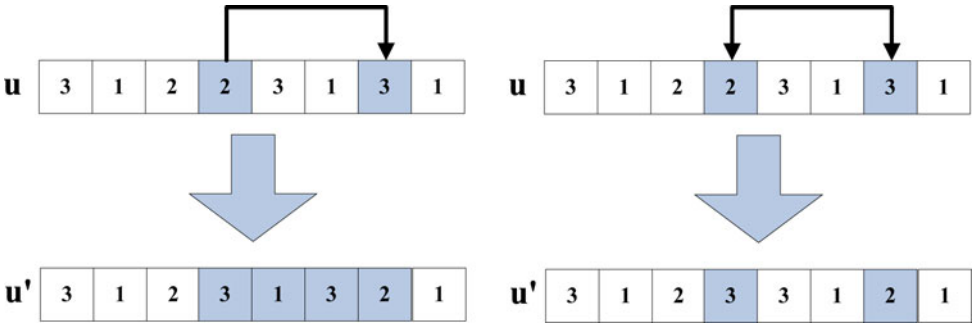


Fig. 6 A example of the two improvement method

(a) The first improvement method **(b)** The second improvement method

The second improvement method (IM2) keeps same sequence information of the original solution by only swapping two randomly selected elements. First two element u_i and u_j is randomly selected from **u**, then the new solution will be generated by swapping the positions of elements u_i and u_j after the improvement method. As shown in Fig. 6b, for the above new solution **u**, the fourth and seventh elements are selected and swapped, and then the new solution $\mathbf{u}'$ is obtained after the improvement method.

Updating reference set

The reference set update method accompanies each application of the improvement method, and the update operation consists of maintaining the record of b_1 high quality solutions and b_2 diverse solutions, where the value of $b(b = b_1 + b_2)$ is stated as a chosen constant, but may readily be allowed to vary. Once the new solutions have been generated, the better ones in the reference set are included and the worst ones are excluded from this reference set. There are two update strategies of the reference set, static update strategy and dynamic update strategy. Dynamic update strategy can replace solutions more quickly in reference set for the future combinations, but it neglects some potentially promising combinations (Marti et al. 2006). For obtaining high quality solutions, the static update strategy of the reference set was chosen. And the stopping criterion is defined by the convergence of the RefSet, i.e. when in any iteration no new solutions are included in the RefSet.

Computational experiments and results

In order to evaluate the performance of the proposed algorithm we have run a series of computational experiments. In this section we report on the obtained results. All the algorithms have been coded in C and run on a PC with a Pentium IV processor, a 3.0 GHz internal clock and 1G RAM memory.

Since there are no benchmarks test sets available for the above problem, 13 test problems are generated randomly. The test problems include small-medium size problems and large size problems. The machine groups assigned to each cell are shown in Table 3. The number of parts varies from 5 to 50, the number of machines is from 4 to 24, and the number of cells is from 2 to 6. The distance between different cells is randomly generated in [2, 20]. The due date of each part type is randomly generated in [1, 25]. The weight of tardiness penalty for each part type is randomly generated in [1, 10]. The processing time for operations of each part on a machine is randomly generated in [0.1, 0.9]. The above random numbers are generated with normal distribution.

Based on the above diversification generation methods and improvement methods, the four different scatter search strategies were compared and shown in Table 4. The above test problems are solved respectively by using SSRI, SSRM, SSGI, and SSGM, and the obtained results for the problems are displayed in Table 5.

As it can be seen in Table 5, all test problems are successfully solved by SSRI, SSRM, SSGI, and SSGM, and the total weighted tardiness and execution time (in seconds) for each instance are obtained. The least mean execution time is observed in SSRI, and the least mean total weighted tardiness is observed in SSGM. Comparing with SSRI, SSRM and SSGI, the mean total weighted tardiness obtained by SSGM is improved respectively about 10.93, 8.22 and 2.01%. It can be seen that SSGM outperforms other scatter search procedures achieving better solutions for the test problems. Meanwhile, comparing with SSRM, SSGI and SSGM, the mean computation time taken by SSRI is shorter respectively about 16.20, 13.50 and 28.88%. Hence, PGM and IM2 are essential to this scatter search approach for obtaining high quality solutions, but take more computation time. Figure 7 shows a Gantt chart of one solution obtained by SSGM for test problem 4, which is taken as an example of the test problems.

In order to further evaluate the performance of the proposed algorithm, the MIP optimizer of the CPLEX 10.1 is used for the optimal solutions. The above four scatter search

Table 3 Machine groups assigned to each cell

Problem	Parts/machines/cells	Cell 1	Cell 2	Cell 3	Cell4	Cell 5	Cell 6
1	5/4/2	M2, M3	M1, M4				
2	8/5/2	M1, M2	M3, M4, M5				
3	10/6/3	M2, M5	M1, M4	M3, M6			
4	12/7/3	M3, M4, M5	M1, M7	M2, M6			
5	16/8/3	M3, M4, M5	M6, M8	M1, M2, M7			
6	20/10/4	M3, M5, M8	M2, M7	M1, M4, M10	M6, M9		
7	22/12/4	M2, M7, M12	M1, M10	M4, M8, M9	M3, M5, M6, M11		
8	26/14/5	M5, M12, M13	M6, M7	M1, M4, M11	M3, M9, M10, M14	M2, M8	
9	30/16/5	M1, M2	M10, M13, M14, M15,	M6, M9, M11, M16	M7, M8	M3, M4, M5, M12	
10	35/18/5	M1, M9, M10	M4, M8, M14, M15	M6, M7, M16, M17	M2, M5, M11, M12, M18	M3, M13	
11	40/20/6	M2, M5, M9, M15	M1, M12, M13	M4, M10, M16	M6, M7, M17	M3, M20	M8, M11, M14, M18, M19
12	45/22/6	M9, M12, M16, M21	M5, M6, M13, M17, M19	M3, M10, M15	M1, M2, M14, M20	M4, M8, M11	M7, M18, M22
13	50/24/6	M5, M6, M13, M16	M1, M4, M7, M9	M2, M8, M11, M12, M14, M24	M3, M10, M19, M22, M23	M15, M20, M21	M17, M18

Table 4 Comparison between scatter search procedures

	Scatter Search	RGM	PGM	IM1	IM2
1	SSRI	√		√	
2	SSRM	√			√
3	SSGI		√	√	
4	SSGM		√		√

strategies are compared with CPLEX and the results for each instance with optimal solutions are summarized in Table 6 wherein the gap (in percentage) and execution time taken over the test problems. The gap in Table 6 is the percentage deviation of the total weighted tardiness obtained by each scatter search strategy from the optimal value obtained by CPLEX. It is calculated as follows:

$$gap = (tardiness - opt)/opt \times 100\% \quad (9)$$

where $tardiness$ is the total weighted tardiness, and opt denotes the optimal value.

To clearly present the gaps between the tardiness and optimal values in Table 6, comparison results for the former ten problems with optimal solutions is given in Fig. 8.

As it is shown in Table 6 and Fig. 8, for small-medium size problems, optimal solutions can be obtained by CPLEX 10.1 within 10 h running time, even despite the fact that CPLEX tests have to be repeated as the memory space is exceeded occasionally. The gap between the total weighted tardiness obtained by the proposed scatter search and the optimal solution is about less than 7.1%. Average CPU time for CPLEX is too long, which validates the solution time superiority of the scatter search strategies on solution time over the commercial solver. For large size problems, CPLEX failed to provide the optimal solution for the test problems within tolerance time, 10 h of computation time. Moreover, the above four scatter search strategies were able to successfully solve all the test problems with significantly improved solutions and less computational effort. In conclusion, the proposed scatter search approach proved to be a very useful method for tackling the small-medium size problems of this difficult combinatorial optimization problem. As for the large size

Table 5 Results for the problems solved by scatter search procedures

Problem	Parts/Machines/Cells	SSRI		SSRM		SSGI		SSGM	
		Tardiness	Time (s)	Tardiness	Time (s)	Tardiness	Time (s)	Tardiness	Time (s)
1	5/4/2	1.0728	0.3055	1.0728	0.3090	1.0728	0.4251	1.0728	0.4474
2	8/5/2	6.4581	0.3085	6.4581	0.3006	6.4581	0.3833	6.4581	0.4170
3	10/6/3	17.0803	0.6931	15.7261	0.7395	15.1497	0.8626	14.6304	0.8144
4	12/7/3	23.1187	6.0049	22.0689	8.0699	20.6402	8.0546	19.3708	9.3383
5	16/8/3	33.5475	7.8763	35.4221	8.6738	29.9728	8.4479	26.1383	10.5631
6	20/10/4	38.3991	11.4015	36.8542	18.8816	32.3686	17.6364	31.8786	19.1677
7	22/12/4	41.1316	27.2016	40.0641	30.2255	36.3043	24.0704	35.7638	28.6265
8	26/14/5	48.4849	28.9090	46.0863	36.1863	38.9693	33.2122	38.7486	32.6530
9	30/16/5	58.2176	44.2349	57.4624	40.1969	43.4865	40.8756	43.7680	37.5094
10	35/18/5	62.5500	59.6200	59.3574	56.4119	58.2176	53.9935	58.5500	56.3234
11	40/20/6	105.8912	74.8091	103.1550	82.7822	98.7624	69.3110	97.6474	74.2974
12	45/22/6	186.1222	70.3788	184.9834	104.2073	183.7631	94.3987	176.3521	106.8106
13	50/24/6	345.7539	101.7820	335.4643	116.7797	324.8497	140.3802	322.0795	181.7602
	Mean	74.4483	33.3481	72.6289	38.7511	68.4627	37.8501	67.1122	42.9791

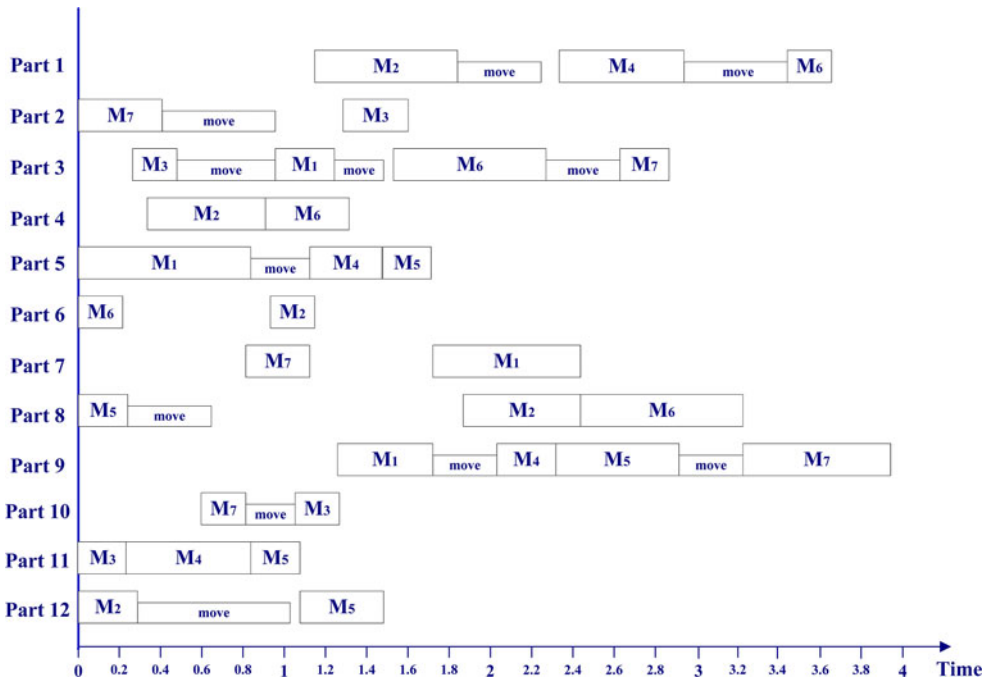


Fig. 7 Gantt chart of one solution obtained by SSGM for test problem 4

problems, although the quality of respective solutions is not known, the proposed scatter search approach is, at least, an easy tool to consistently generate feasible solutions, at a low computing time, even when the standard software behaved very poorly.

Conclusions

This paper studied the problem of parts scheduling in a CMS by considering exceptional parts processed on machines in multiple cells and proposed a nonlinear mathematical pro-

Table 6 Comparison of results for all the problems with optimal solutions

No.	CPLEX		SSRI		SSRM		SSGI		SSGM	
	Optimal value	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
1	1.0728	2.813	0	0.3055	0	0.3090	0	0.4251	0	0.4474
2	6.4581	7.234	0	0.3085	0	0.3006	0	0.3833	0	0.4170
3	14.35	10.844	19.03	0.6931	9.59	0.7395	5.57	0.8626	1.95	0.8144
4	18.464	3521.279	25.21	6.0049	19.52	8.0699	11.79	8.0546	4.91	9.3383
5	24.414	3791.813	37.41	7.8763	45.09	8.6738	22.77	8.4479	7.06	10.5631
6	30.641	4906.344	25.32	11.4015	20.28	18.8816	5.64	17.6364	4.04	19.1677
7	33.426	4485.734	23.05	27.2016	19.86	30.2255	8.61	24.0704	6.99	28.6265
8	36.225	5663.406	33.84	28.9090	27.22	36.1863	7.58	33.2122	6.94	32.6530
9	41.135	6515.484	41.53	44.2349	39.69	40.1969	5.72	40.8756	6.40	37.5094
10	55.462	7773.719	12.78	59.6200	7.02	56.4119	4.97	53.9935	5.57	56.3234
11	–	36000	–	74.8091	–	82.7822	–	69.3110	–	74.2974
12	–	36000	–	70.3788	–	104.2073	–	94.3987	–	106.8106
13	–	36000	–	101.7820	–	116.7797	–	140.3802	–	181.7602

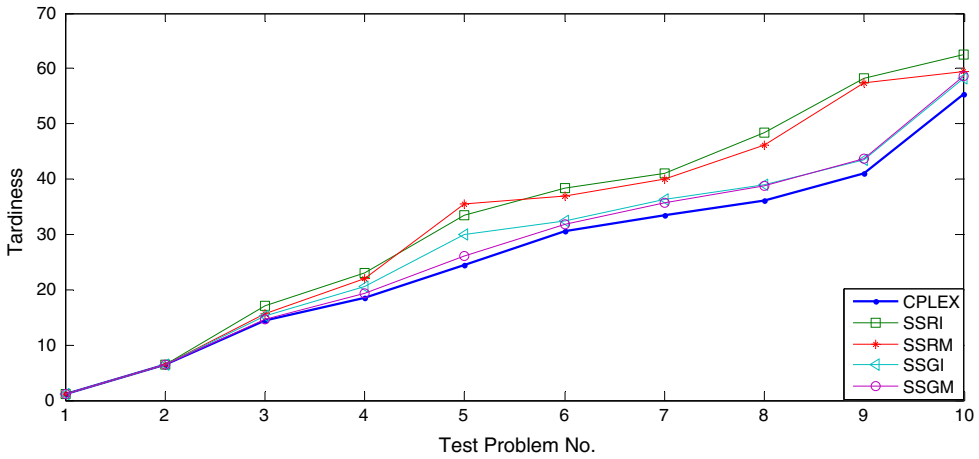


Fig. 8 Total weighted tardiness of the four scatter search strategies and CPLEX

gramming model of the problem by minimizing the total weighted tardiness in a CMS. Since CPS problem is a hard problem to solve, a scatter search approach was developed which redesigns the common components of scatter search and incorporates diversification generation method, local search method and other improvement mechanisms. In order to check the efficiency of the proposed scatter search, a series of test problems were generated and solved by using SSRI, SSRM, SSGI, SSGM, and CPLEX 10.1 respectively.

The immediate future extension of the study could search for tight lower bounds as it is essential, especially for large instances, and focus on the development of different meta-heuristic procedures to try to improve the quality of the feasible solutions.

Acknowledgements This research was financially supported by the National Science Fund (70625001, 70721001), and the funding of PhD program from Ministry of Education of China (20060145009).

References

Arikan, F., & Gungor, Z. (2005). A parametric model for cell formation and exceptional elements' problems with fuzzy parameters. *Journal of Intelligent Manufacturing, 16*(1), 103–114. doi:10.1007/s10845-005-4827-3.

Askin, R. G., Selim, H. M., & Vakharia, A. (1997). A methodology for designing flexible cellular manufacturing systems. *IIE Transactions, 29,* 599–610.

Baker, K. R. (1990). Scheduling groups of jobs in the two-machine flow shop. *Mathematical and Computer Modelling, 13*(3), 29–36. doi:10.1016/0895-7177(90)90368-W.

Bazargan-lari, M., Kaebernick, H., & Harraf, A. (2000). Cell formation and layout designs in a cellular manufacturing environment: A case study. *International Journal of Production Research, 38*(7), 1689–1709. doi:10.1080/002075400188807.

Billo, R. E., Bidanda, B., & Tate, D. (1996). A genetic cluster algorithm for the machine-component grouping problem. *Journal of Intelligent Manufacturing, 7*(3), 229–241. doi:10.1007/BF00118082.

Burcu, B. K., & Halit, Ü. (2006). A scatter search-based heuristic to locate capacitated transshipment points. *Computers & Operations Research, 34*(10), 3112–3125.

Cheng, R., Gen, M., & Tsujimura, Y. (1996). A tutorial survey of job-shop scheduling problems using genetic algorithms. *Computers & Industrial Engineering, 30*(4), 983–997. doi:10.1016/0360-8352(96)00047-2.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences, 8,* 156–166. doi:10.1111/j.1540-5915.1977.tb01074.x.

Glover, F. (1998). A template for scatter search and path relinking. *Artificial Evolution, Lecture Notes in Computer Science 1363*. Berlin: Springer, pp. 13–54.

Gupta, J. N. D., & Schaller, J. E. (2006). Minimizing flow time in a flow-line manufacturing cell with family setup times. *The Journal of the Operational Research Society, 57*(2), 163–176.

Hsu, C. M., & Su, C. (1998). Multi-objective machine-component grouping in cellular manufacturing, a genetic algorithm. *Production Planning and Control, 9*(2), 155–166. doi:10.1080/095372898234370.

Krishnamoorthy, B., & Kamath, M. (2000). Scheduling in a cellular manufacturing environment: A review of recent research. *Journal of Engineering Valuation and Cost Analysis, 2*(5), 409–423.

Laguna, M., & Marti, R. (2003). *Scatter search. Methodology and implementations in C*. Boston: Kluwer.

Liang, M., & Zolfaghari, S. (1999). Machine cell formation considering processing times and machine capacities: An ortho-synapse Hopfield neural network approach. *Journal of Intelligent Manufacturing, 10*(5), 437–447. doi:10.1023/A:1008923114466.

Logendran, R., Mai, L., & Talkington, D. (1995). Combined heuristics for bi-level group scheduling problems. *International Journal of Production Economics, 38*(2–3), 133–145. doi:10.1016/0925-5273(94)00083-M.

Mahmood, F., Dooley, K. J., & Starr, P. J. (1990). An investigation of dynamic group scheduling heuristics in a job shop manufacturing cell. *International Journal of Production Research, 28*(9), 1695–1711. doi:10.1080/00207549008942824.

Marti, R., Laguna, M., & Glover, F. (2006). Principles of scatter search. *European Journal of Operational Research, 169*(2), 359–372. doi:10.1016/j.ejor.2004.08.004.

Reddy, V., & Narendran, T. T. (2003). Heuristics for scheduling sequence-dependent set-up jobs in flow line cells. *International Journal of Production Research, 41*(1), 193–206. doi:10.1080/00207540210163973.

Ruben, R. A., Mosier, C. T., & Mahmoodi, F. (1993). Comprehensive analysis of group scheduling heuristics in a job shop cell. *International Journal of Production Research, 31*(6), 1343–1369. doi:10.1080/00207549308956795.

Safaei, N., Saidi-Mehrabad, M., & Babakhani, M. (2007). Designing cellular manufacturing systems under dynamic and uncertain conditions. *Journal of Intelligent Manufacturing, 18*(3), 383–399. doi:10.1007/s10845-007-0029-5.

Schaller, J. (2000). A comparison of heuristics for family and job scheduling in a flow-line manufacturing cell. *International Journal of Production Research, 38*(2), 287–308. doi:10.1080/002075400189419.

Selim, H. M., Askin, R. G., & Vakharia, A. J. (1998). Cell formation in group technology: Review, evaluation and direction for future research. *Computers & Industrial Engineering, 34,* 2–30. doi:10.1016/S0360-8352(97)00147-2.

Skorin-Kapov, J., & Vakharia, A. J. (1993). Scheduling a flow-line manufacturing cell: A tabu search approach. *International Journal of Production Research, 31*(7), 1721–1734. doi:10.1080/00207549308956819.

Solimanpur, M., Vrat, P., & Shankar, R. (2004). A heuristic to minimize makespan of cell scheduling problem. *International Journal of Production Economics, 88*(3), 231–241. doi:10.1016/S0925-5273(03)00196-8.

Sridhar, J., & Rajendran, C. (1994). A genetic algorithm for family and job scheduling in a flowline-based manufacturing cell. *Computers & Industrial Engineering, 27*, 469–472. doi:10.1016/0360-8352(94)90336-0.

Tsai, C. H., & Li, R.-K. (2000). Due-date oriented scheduling heuristic for job shop cellular manufacturing system. *International Journal of Industrial Engineering: Theory Applications and Practice, 7*(1), 76–88.

Venkataramanaiah, S. (2008). Scheduling in cellular manufacturing systems: A heuristic approach. *International Journal of Production Research, 46*(2), 429–449. doi:10.1080/00207540601138577.

Vin, E., De Lit, P., & Delchambre, A. (2005). A multiple-objective grouping genetic algorithm for the cell formation problem with alternative routings. *Journal of Intelligent Manufacturing, 16*(2), 189–205. doi:10.1007/s10845-004-5888-4.

Wang, T. Y., Wu, K. B., & Liu, Y. W. (2001). A simulated annealing algorithm for facility layout problems under variable demand in cellular manufacturing systems. *Computers in Industry, 46*(2), 181–188. doi:10.1016/S0166-3615(01)00107-5.

Wei, N. C., & Mejabi, O. O. (2008). A clustering approach for minimizing intercell trips in cell formation. *Journal of Intelligent Manufacturing, 19*(1), 13–20. doi:10.1007/s10845-007-0042-8.

Wemmerlov, U., & Johnson, D. J. (1997). Cellular manufacturing at 46 user plants: Implementation experiences and performance improvements. *International Journal of Production Research, 35*(1), 29–49. doi:10.1080/002075497195966.

Wemmerlov, U., & Vakharia, A. J. (1991). Job and family scheduling of a flow-line manufacturing cell: A simulation study. *IIE Transactions, 23*(4), 383–393. doi:10.1080/07408179108963871.

Willow, C. C. (2002). A feedforward multi-layer neural network for machine cell formation in computer integrated manufacturing. *Journal of Intelligent Manufacturing, 13*(2), 75–87. doi:10.1023/A:1014524611895.

Wu, X., Chao-Hsien, C., Yunfeng, W., & Weili, Y. (2007). A genetic algorithm for cellular manufacturing design and layout. *European Journal of Operational Research, 181*(1), 156–167. doi:10.1016/j.ejor.2006.05.035.

Yang, W. H., & Liao, C. J. (1996). Group scheduling on two cells with intercell movement. *Computers & Operations Research, 23*(10), 997–1006. doi:10.1016/0305-0548(96)00003-2.