Available online at www.sciencedirect.com
ScienceDirect
Expert Systems with Applications 33 (2007) 451–467
Expert Systems with Applications
www.elsevier.com/locate/eswa

Using memetic algorithms with guided local search to solve assembly sequence planning

Hwai-En Tseng *, Wen-Pai Wang, Hsun-Yi Shih

Department of Industrial Engineering and Management, National Chin-Yi Institute of Technology, 35, Lane 215, Section 1, Chung-Shan Road, Taiping City, Taichung 411, Taiwan, ROC

Abstract

The goal of assembly planning consists in generating feasible sequences to assemble a product and selecting an efficient assembly sequence from which related constraint factors such as geometric features, assembly time, tools, and machines are considered to arrange a feasible assembly sequence based on planner's individual heuristics. Suchlike planning may implement genetic algorithms to go towards the assembly sequence features of speed and flexibility. As regards the large constraint assembly problems, however, traditional genetic algorithms will generate a great deal of infeasible solutions in the evolution process which results in inefficiency of the solution-searching process. Guided genetic algorithms proposed by Tseng, then, got over the restrictions of traditional GAs by means of a new evolution procedure. However, Guided-GAs dealt with the assembly sequence problem in the feasible solution range simply. They were consequently inclined to lapse into the local optimal situation and fall short of the expectations. This paper attempts to add global search algorithms not only based on GAs but also treated of the Guided-GAs as the local search mechanism. The proposed novel method under the name of memetic algorithms for assembly sequence planning is possessed of the competence for detecting the optimal/near-optimal solution with respect to large constraint assembly perplexity. Also, actual examples are presented to illustrate the feasibility and potential of the proposed MAs approach. It has been confirmed that MAs satisfactorily provide superior solutions for assembly sequence problems on the strength of comparison with Guided-GAs.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Memetic algorithms; Assembly sequence planning; Guided genetic algorithms; Guided local search

1. Introduction

Assembly sequence planning (ASP) refers to a task for which planners, on the basis of relationships among components in respect of the geometric limitation factors such as assembly time, geometric features, tools, and machines, arrange a specific assembly sequence according to the product design description. An ASP plays an important role on the fields of CAD/CAM design issues, the cost of assembly/manufacturing, as well as the selection of equipment. In the past, a significant amount of publications in this area has been implemented from various viewpoints related to the fact that there is an enormous number of different characteristics and objectives according to the planning environment (Abdullah, Popplewhell, & Page, 2003; Zha, Lim, & Fok, 1998).

In solving the assembly planning problem, most researchers made their proposal on the basis of De Fazio and Whitney's (1987) liaison graph, thereby combining graph theory and an exhaustive search method to approach the problem (Baldwin, Abel, Lui, De Fazio, & Whitney, 1991; Homem De Mello & Sanderson, 1991; Gottipolu & Ghosh, 1997). Although it is possible to find feasible solutions or even optimal solutions by means of graph theory, there exist difficulties to identify the global-optimal solution in a short time period. Moreover, the scale of the problem space is extremely restricted. Genetic Algorithms (GAs) have recently become popular global optimization

* Corresponding author. Tel.: +886 4 23924505x6012; fax: +886 4 23921742.
E-mail address: hwai_en@seed.net.tw (H.-E. Tseng).

0957-4174/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2006.05.025

techniques, specifically in combinatorial optimization. A GA investigates the feasible space by subsidizing populations of solutions, which evolve by reproduction, crossover and mutation operators. One of the GA's characteristics is the multiple points' search, which discriminates the GA from the other random search methods. As a result, many researchers have tried to apply GAs to assembly planning problems lately (Dini, Failli, Lazzerini, & Marcelloni, 1999; Fujimoto & Sebaaly, 2000; Guan, Liu, & Zhong, 2002; Smith & Smith, 2003). Most importantly, the results of the aforementioned studies demonstrate that using GAs is evidently better than the traditional graph-searching method.

The foregoing searching methods are mainly based on De Fazio and Whitney's liaison graph for product description. Applying dissimilar approach from the liaison graph, Tseng and Li (1999) proposed a connector-based assembly planning model. In their study, connectors functioned as assembly elements in product description and served as concept product building blocks in the design stage. Accordingly, more distinguishing engineering features can be included, and the degree of complexity in assembly planning can be effectively reduced. In addition, Yin, Ding, Li, and Xiong (2003) tried to expand the application of connectors and thereby took into consideration the reuse context of assembly planning. Tseng, Li, and Chang (2004) utilized GAs in a connector-based environment, attempting to incorporate the advantages of connectors and GA concepts. Subsequently, Tseng and Tang (2006) sequentially integrated the problem between assembly sequence planning and assembly line balancing. In addition, manufacturers typically divide assembly planning into three phases: selecting an assembly method, assembly sequence planning, and assembly operations planning (Smith & Smith, 2003). The connector-based approach addressed in this paper is in point of assembly sequence planning.

The model proposed by Tseng et al. (2004) was executed by traditional GA procedure which was termed TGAs. Nevertheless, the TGAs approach was a mere random and blind-searching procedure in which had a tendency to generate a great deal of infeasible solutions during the evolution procedure, especially arduous to find feasible solution associated with large constraint assembly problems. Afterward, Tseng (2006) developed a new method defined as guided genetic algorithms (Guided-GAs) which succeed in overcoming the assembly planning problems with regard to large-scale constraints. Implementing Guided-GAs, initial feasible solutions are setting through the binary-tree algorithms before the evolution process starts. Furthermore, an improvement in the crossover and mutation mechanism will be conducive to solve the enormous infeasible solutions in the evolution process.

However, this mechanism, Guided-GA, deals with the assembly sequence problem in the feasible region, and maybe results in insufficient chromosome changes, but inclines towards lapsed into the local optimum searching. Accordingly, the goal of this study attempts to provide a higher-quality solution method than Guided-GAs. Moscato (1992) introduced the term "memetic algorithms" (MAs) which combines evolutionary algorithms with the intensification power of a local search, and has a pragmatic perspective for better effects than GAs. As such MAs, a local optimizer is applied to each offspring before it is inserted into the population in order to make it towards optimum and then GAs platform as a means to accomplish global exploration within a population. MAs are being used for several NP optimization problem such as scheduling problem (Fransca, Mendes, & Moscato, 2001; Maheswaran, Ponnambalam, & Aravindan, 2005), cell formation problem (Muruganandam, Prabhaharan, Asokan, & Baskaran, 2005), multistage capacitated lot sizing problem (Berretta & Rodrigues, 2004) and many others, respectively.

Incorporing Guided-GAs proposed by Tseng (2006) as a local constraint solver into the MA mechanism, the authors employ a traditional genetic algorithm as the random permutation method. The remainder of the paper is structured as follows. In Section 2, the contents of connectors are discussed. Section 3 reports the details of the procedure of memetic algorithms for assembly sequence planning. Section 4 verifies and exemplifies the proposed algorithm through various down-to-earth examples and a comparison of the performance of our algorithm with earlier proposed heuristics. Finally, Section 5 will present some final comments on the presented approach.

2. Concepts and engineering data of connectors

2.1. Content of connectors

This paper deliberates upon the engineering information of connectors like combination, tool, direction and connector-based precedence graph (Tseng et al., 2004). They are described as follows:

(1) *Combination*. The classifications of connectors presented by Akagi, Osaki, and Kikuci (1980) are employed. Assembly parts are divided into four types according to the connector's combination property (Table 1).

Table 1
Classification of fastener types according to Akagi et al. (1980)

Type	Code	Example
Fixed fastener		
Disassembled	FD	Screw, bolted joint, key, spline, wedge
Not disassembled	FND	Pressing fits, riveted joints, welding
Movable fastener		
Disassembled	MD	Snap ring, bearing, spring
Not disassembled	MND	Races and ball-bearing balls

(1) FD, fixed fastener disassembled, (2) FND, fixed fastener not disassembled, (3) MD, movable fastener disassembled, (4) MND, movable fastener not disassembled.

Table 2
Classification of assembly tools

Level	Force magnitude	Tool name	Details on assembly operation
T_1	None	Hand	No tools are needed; i.e., the assembly is manual.
T_2	Small	Work-bench, handgun, screw-driver, spanner, pliers	Use a simple hand tool to assemble, no strict interference occurs between components.
T_3	Medium	Screw driver, spanner, racket spanner	Use simple hand tool to assemble; other tools are needed to support the assembly work.
T_4	Large	Hacksaw, heavy sledgehammer, crusher, torsional twister, chassis	Use a special tool to assemble the product; the operation may cause a destructive result.

(2) *Assembly tools*. Even as expounded by Tseng et al. (2004), assembly tools are classified into four categories according to the degree of difficulty of assembly tasks (Table 2).
(3) *Direction*. In this study, six directions are took account of connectors: $+x$, $-x$, $+y$, $-y$, $+z$ and $-z$, respectively.
(4) *Connector-based precedence graph*. The features of connectors or the geometric information of related parts conclude the precedence graph among connectors. In this study, the precedence relationship is assumed to be predetermined.

A stapler in Fig. 1(a) is an example for illustration. The stapler amounted to eighteen components. According to the combination types, assembly tools and directions for these parts, nine connectors can be defined (Table 3). For example, three parts comprise connector 8: the steel cover, the bracket spring, and rivet 1, individually. Its combination property belongs to the fixed-not-disassembled combination (FND) shown in Table 1; the assembly direction is y, and a hand vice (T_3) should be used as the assembly tool. Fig. 1(b) depicts the connector-based precedence graph for the stapler's connectors.

2.2. Connector-precedence-matrix *P*

For the input information of MAs, a connector-precedence graph was utilized to establish the connector-precedence-matrix P. $p_{i,j}$ indicates the element of the ith row and the jth column in the matrix P. When the ith connector should be assembled after the jth connector, $p_{i,j} = 1$; otherwise, $p_{i,j} = 0$. In the stapler example, the precedence-relationship graph (Fig. 1(b)) apparently demonstrates that C_2 should be assembled before C_3 and C_4, and C_8 should be assembled after all other connectors. Parts of the stapler's connector-precedence-matrix P can be known: $p_{3,2} = p_{42} = 1$ and $p_{8,0} = p_{8,1} = p_{8,2} = p_{8,3} = p_{8,4} = p_{8.5} = p_{8.6} = p_{8,7} = 1$ (Fig. 2(a)).

2.3. Connector-engineering-data-similarity-matrix *S*

The design of the fitness function is determined by the similarity of the engineering data of the connector. Therefore, connector-engineering-data-similarity-matrix S is created first of all. $S_{i,j}$ represents the element of the ith row and the jth column in the matrix S, specifically, the engineering-data-similarity of the ith and the jth connectors. The value of $S_{i,j}$ can be calculated by formula (1):

$$S_{i,j} = W_c \times C_{i,j} + W_d \times D_{i,j} + W_t \times T_{i,j} \tag{1}$$

where

$S_{i,j}$ represents the similarity between connector i and connector j; if $i = j$, then $S_{i,j} = 0$; $i,j = 1, 2, 3, \ldots, m$; m is the number of connectors;
W_c is the weight of the combination property;
W_d is the weight of the direction property;
W_t is the weight of the tool property;
$C_{i,j}$ is the combination condition between the connectors. When the combination property of connector C_i and C_j is the same, $C_{i,j} = 1$; otherwise, $C_{i,j} = 0$;
$D_{i,j}$ is the direction condition between the connectors. When the direction property of connector D_i and D_j is the same, $D_{i,j} = 1$; otherwise, $D_{i,j} = 0$;
$T_{i,j}$ is the tool condition between the connectors. When the tool property of connector T_i and T_j is the same, $T_{i,j} = 1$; otherwise, $T_{i,j} = 0$.

In contrast with the Guided-GAs (Tseng, 2006), the weight of the engineering data of the connector in formula (1) is all settled on 1.

2.4. Coding for chromosomes

The usage of decimal numbers is towards the genetic coding of chromosomes in this paper. The chromosome length equals the number of connectors; the numeral stands for the connector number; the location of the numeral indicates the assembly sequence of connectors. Fig. 3 illustrates a chromosome of the stapler. The assembly sequence of the connectors is accordingly $3 \rightarrow 2 \rightarrow 0 \rightarrow 1 \rightarrow 4 \rightarrow 8 \rightarrow 5 \rightarrow 6 \rightarrow 7$. The mark '$\rightarrow$' denotes the precedence order of the connectors. For example, $1 \rightarrow 4$ signifies that connector C_1 should be assembled prior to connector C_4.

2.5. Fitness function

The fitness function in this paper is determined by the similarity of the engineering data of the connectors because the arrangement of similar connectors can reduce the number of changes in the assembly tools and the assembly direction, thus indicating the assembly task time is also low. This viewpoint can be inferred from Göngör and Gupta's research (1997). Therefore, the calculation of the fitness function is based upon connector-similarity-matrix

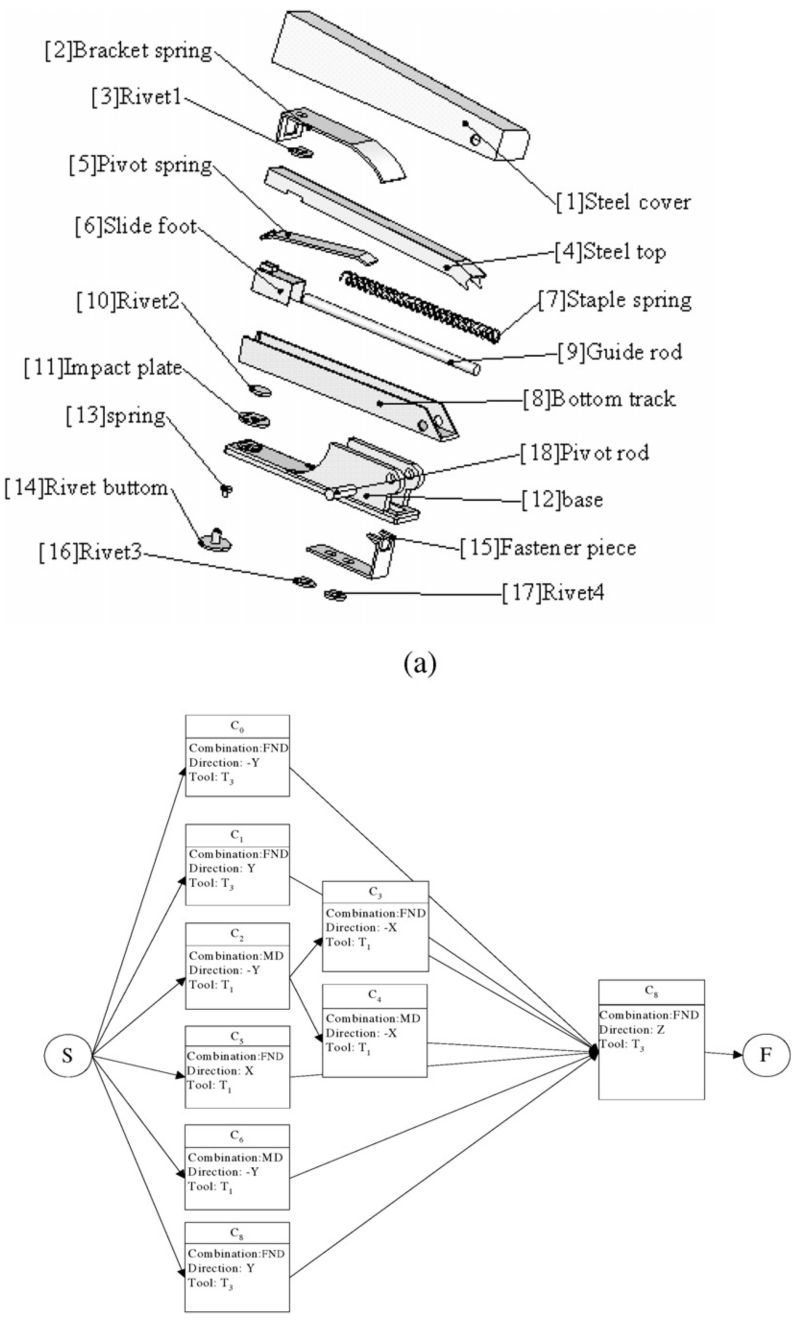

[2]Bracket spring
[3]Rivet1
[5]Pivot spring
[6]Slide foot
[10]Rivet2
[11]Impact plate
[13]spring
[14]Rivet buttom
[16]Rivet3
[1]Steel cover
[4]Steel top
[7]Staple spring
[9]Guide rod
[8]Bottom track
[18]Pivot rod
[12]base
[15]Fastener piece
[17]Rivet4
(a)
S
C0 Combination:FND Direction: -Y Tool: T3
C1 Combination:FND Direction: Y Tool: T3
C2 Combination:MD Direction: -Y Tool: T1
C5 Combination:FND Direction: X Tool: T1
C6 Combination:MD Direction: -Y Tool: T1
C8 Combination:FND Direction: Y Tool: T3
C3 Combination:FND Direction: -X Tool: T1
C4 Combination:MD Direction: -X Tool: T1
C8 Combination:FND Direction: Z Tool: T3
F
(b)

Fig. 1. Stapler: (a) graph of parts and (b) connector-based precedence graph.

S, as shown in Fig. 2(b). Then the sum of the similarity of the engineering data of two neighboring connectors in a chromosome is calculated as follows.

$$\text{Fitness value } (F) = \sum_{h=1}^{m} S^G_{h,h+1} \tag{2}$$

$S^G_{h,h+1}$ denotes the information similarity of the connector in the hth position and the connector in the $(h+1)$th position; $h = 1, 2, 3, \ldots, m$; m is number of connectors.

Take the chromosome of the stapler in Fig. 3 as an example. According to formula (2) and Fig. 2(b), similarity matrix S can be obtained. Thus, the value of the fitness function of this chromosome, $(F) = S^G_{0,1} + S^G_{1,2} + S^G_{2,3} + S^G_{3,4} + S^G_{4,5} + S^G_{5,6} + S^G_{6,7} + S^G_{7,8} = 2.16$.

The term $S^G_{h,h+1}$ indicates the similarity in engineering data of the connector corresponding to the hth location and the connector corresponding to the $(h+1)$th location in chromosome G. In the chromosome example shown in Fig. 3, the connectors stored in the first and second locations are connectors C_3 and C_2. From Fig. 2(b), $S^G_{1,2} = 0.5$ is known. If connector C_5 comes immediately after connector C_3, then they have the highest degree of

Table 3
Connector information for stapler

No.	Connector name	Combination type	Direction	Tool	Component owned by connector
C_0	Interference fit	FND	$-y$	T_3	10, 11, 12, 13, 14
C_1	Interference fit	FND	y	T_3	12, 15, 16, 17
C_2	Spring	MD	$-x$	T_1	7, 9
C_3	Insert	FND	$-x$	T_1	6, 9
C_4	Spring	MD	$-x$	T_1	6, 7
C_5	Insert	FND	x	T_1	8, 9
C_6	Snap fit	MD	$-y$	T_1	6, 5, 4
C_7	Interference	FND	y	T_3	1, 2, 3
C_8	Interference	FND	z	T_3	1, 4, 8, 12, 18

(1) FD, fixed fastener disassembled, (2) FND, fixed fastener not disassembled, (3) MD, movable fastener disassembled, (4) MND, movable fastener not disassembled.

$$\begin{array}{c|ccccccccc|}
 & C_0 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 \\
\hline
C_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_4 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C_8 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
\hline
\end{array}$$

(a)

$$\begin{array}{c|ccccccccc|}
 & C_0 & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 \\
\hline
C_0 & 0 & 0.83 & 0 & 0.5 & 0 & 0.5 & 0.33 & 0.83 & 0.83 \\
C_1 & 0.83 & 0 & 0 & 0.5 & 0 & 0.5 & 0 & 1 & 0.83 \\
C_2 & 0 & 0 & 0 & 0.5 & 1 & 0.33 & 0.83 & 0 & 0 \\
C_3 & 0.5 & 0.5 & 0.5 & 0 & 0.5 & 0.83 & 0.33 & 0.5 & 0.5 \\
C_4 & 0 & 0 & 1 & 0.5 & 0 & 0.33 & 0.83 & 0 & 0 \\
C_5 & 0.5 & 0.5 & 0.33 & 0.83 & 0.33 & 0 & 0.33 & 0.5 & 0.5 \\
C_6 & 0.33 & 0 & 0.83 & 0.33 & 0.83 & 0.33 & 0 & 0 & 0 \\
C_7 & 0.83 & 1 & 0 & 0.5 & 0 & 0.5 & 0 & 0 & 0.83 \\
C_8 & 0.83 & 0.83 & 0 & 0.5 & 0 & 0.5 & 0 & 0.83 & 0 \\
\hline
\end{array}$$

(b)

Fig. 2. Initial connector-based information for stapler: (a) precedence-relationship-matrix P and (b) similarity matrix S for engineering information.

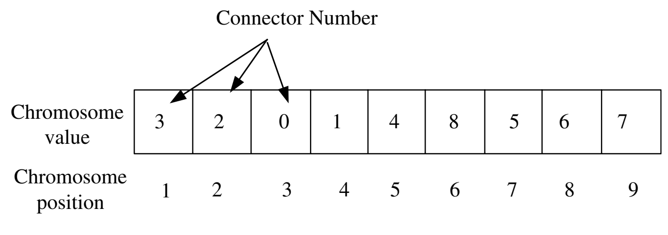


Fig. 3. Coding of chromosome for stapler.

similarity, which is 0.83. For each location h, an ideal similarity coefficient, $\max_i(S^G_{h,i})$, is defined as the maximum degree of similarity between those two adjacent connectors. In this example, $\max_7(S^G_{1,7}) = 0.83$. The ratio of fitness is therefore used as the criterion for the generation of the genetic crossover block in this paper, and as illustrated below:

$$R_h = \frac{S^G_{h,h+1}}{\max\limits_i(S^G_{h,i})} \tag{3}$$

where

> R_h stands for the fitness ratio of the h position of chromosome G; $h = 1, 2, 3, \ldots, m$; m is number of connectors; $\max_i(S^G_{h,i})$ is defined as the maximum degree of similarity between the connector of the hth position and the adjacent connectors.

In the foregoing example, where individual values are $S^G_{1,2} = 0.5$, $\max_7(S^G_{1,7}) = 0.83$, and the ratio $R_h = 0.602$. The more the value of R_h, the greater the probability, that the genetic coding in the hth location of the chromosome is blocked. For any two connectors, the ideal situation exists when $R_h = 1$. On the contrary, when the value of R_h is smaller, the probability that the genetic coding in the hth location of the chromosome is blocked is smaller.

3. Memetic algorithms with guided local search

3.1. Procedure for memetic-oriented algorithms

The main procedure for MAs is shown in Fig. 4. The authors take advantage of the binary-tree algorithms to generate the initial population (*Step* 1). A feasible solution can be found through the binary-tree algorithms. For the sake of better quality solution, the crossover and mutation mechanism of traditional GAs are employed to disturb the initial feasible solution (*Steps* 3 and 4). The disturbed infeasible chromosome array can be transferred into feasible solution again through the same binary-tree algorithms (*Step* 5). Guided-GAs play the roles of local search method are proposed by *Steps* 6–8. The guided crossover and guided mutation are aimed for solving the constraint problem in ASP.

Through the connector-precedence-matrix P and connector-engineering-data-similarity- matrix S, the procedure for the algorithms of the MAs is as follows:

Step 1: Through the binary-tree concepts, six steps are utilized to generate N feasible chromosomes for the initial population (Section 3.2).
Step 2: Calculate the fitness value of each chromosome, which will serve as the criterion for the evaluation of chromosomes (Section 2.5).

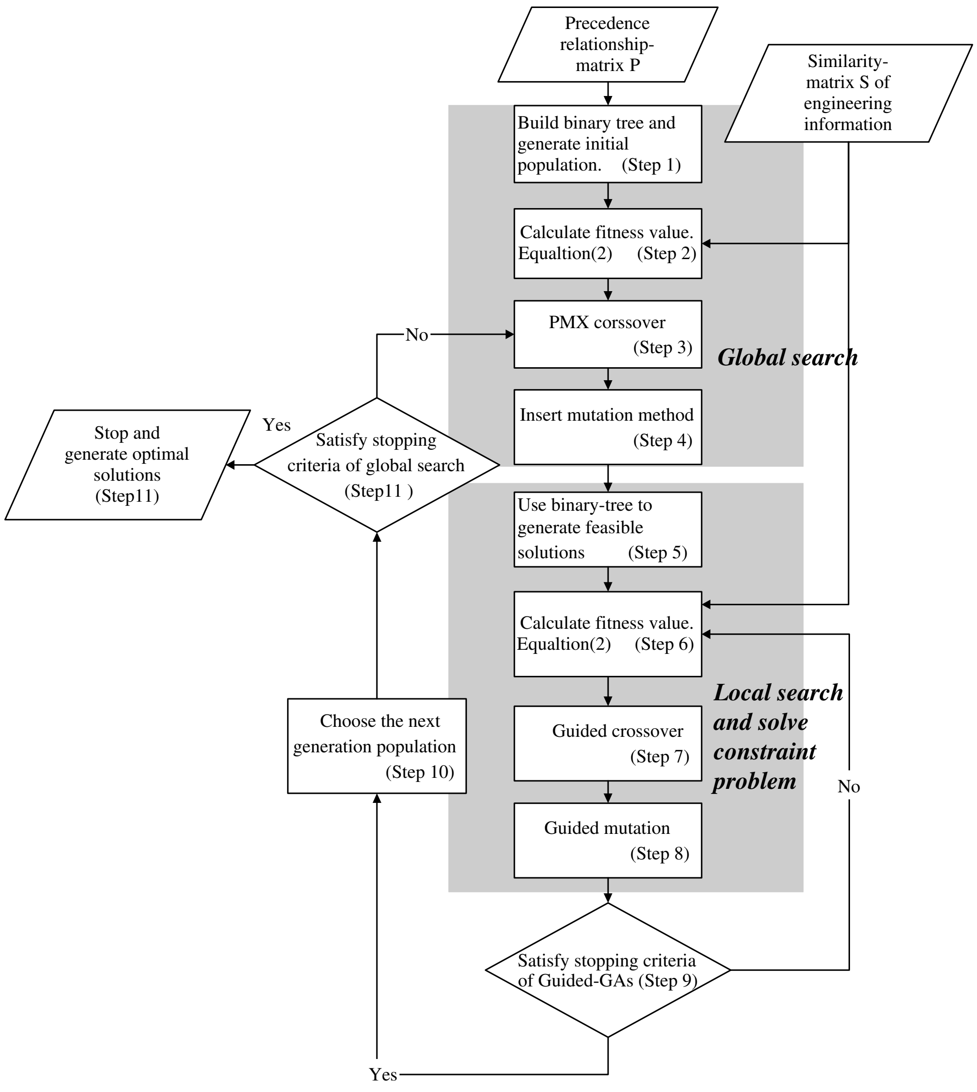

Precedence relationship-matrix P
Similarity-matrix S of engineering information
Build binary tree and generate initial population. (Step 1)
Calculate fitness value. Equaltion(2) (Step 2)
PMX corssover (Step 3)
Global search
Insert mutation method (Step 4)
No
Yes
Stop and generate optimal solutions (Step11)
Satisfy stopping criteria of global search (Step11)
Use binary-tree to generate feasible solutions (Step 5)
Calculate fitness value. Equaltion(2) (Step 6)
Local search and solve constraint problem
Choose the next generation population (Step 10)
Guided crossover (Step 7)
Guided mutation (Step 8)
No
Satisfy stopping criteria of Guided-GAs (Step 9)
Yes

Fig. 4. Flowchart for memetic algorithms.

Step 3: Process crossover according to the PMX method (Section 3.3).

Step 4: Process mutation according the insert mutation method (Section 3.4).

Step 5: Convert the infeasible solution to feasible solution through the binary-tree algorithms (Section 3.2).

Step 6: Calculate the fitness value of each chromosome (Section 2.5).

Step 7: Process the guided-crossover method, twelve major steps for adjustment are necessary (see Section 3.5).

Step 8: Process the guided mutation method, there are six steps for the guided mutation operation (see Section 3.6).

Step 9: Determine if the evaluation for local search should be terminated. The number of generations is the termination index in this paper.

(a) If the output does not meet the termination condition, then execute *Step* 6.

(b) If the output meets the termination condition, then execute *Step* 10.

Step 10: Choose chromosomes from the parent and offspring generation.

Step 11: Determine whether the evaluation for global search should be terminated or not. The maximum number of generations is the termination index in this paper.

(a) If the output does not meet the termination condition, then execute *Step* 3.

(b) If the output meets the termination condition, stops the algorithm and generates the near-optimal assembly sequence.

3.2. Combining the binary-tree concept to generate initial population

The authors introduced a connector-based binary-tree structure into the basis of the connector precedence relationships. A binary tree is a tree diagram illustrating the hierarchical data structure for the purpose of data storage, organization, and retrieval (Wess, 1999). Furthermore, the feasible assembly sequence solutions are listed according to the inorder traversal rank. These feasible sequence solutions will serve as the initial populations of the MAs. In creating a connector-based binary-tree data structure, the requirements that the connector on the left child node point should be assembled before the connector on the root point and that the connector on the right child node point has the lowest priority of assembly should be satisfied. Related terminologies are defined as follows:

G represents a chromosome; a chromosome of the stapler example is shown in Fig. 3;
g_h signifies the corresponding connector of the hth location of chromosome G; $h = 1, 2, 3, \ldots, m$, and m is the number of connectors. In Fig. 3, the corresponding connector of g_1 is C_3;
r stands for the root node point;
l denotes the leaf node point.

The pertinent steps of the connector-based binary-tree are listed below:

```
Step 1.1: Randomly generate a chromosome, $G$.
Step 1.2: Set $h = 2$.
Step 1.3: Set $g_1$'s corresponding connector at root node point $r$.
Step 1.4: Set $g_h$'s corresponding connector at leaf node point $l$, and decide the precedence relationship of $r$ and $l$.
    (1) If $p_{r,l} = 1$, the priority of assembly sequence of node point $l$'s corresponding connector is higher than that of node point $r$'s corresponding connector.
        (a) If $r$'s left child node point is not empty, then set $r$'s left node point at the new root node point $r$ and repeat Step 1.4.
        (b) If $r$'s left child node point is empty, then insert $l$ at $r$'s left node point; set $h = h + 1$, and execute Step 1.5.
    (2) If $p_{r,l} = 0$, there is no limit in the assembly precedence between $l$'s corresponding connector and $r$'s corresponding connector.
        (a) If $r$'s right child node point is not empty, then set $r$'s right node point at the new root node point $r$, and execute Step 1.4.
        (b) If $r$'s right child node point is empty, then insert $l$ at $r$'s right node point; set $h = h + 1$, and execute Step 1.5.
Step 1.5: Check whether $h = m$. If $h = m$, then execute Step 1.6; otherwise, execute Step 1.3.
Step 1.6: List feasible solutions according to the inorder traversal rank and stop the algorithm.
```

3.3. PMX crossover method

In the *Step* 3, the PMS (partial mapped crossover) crossover is used (Gen & Cheng, 2000). The core steps of PMX are as follows:

```
Step 3.1: Randomly choose two parents.
Step 3.2: Randomly select two crossover points and generate a crossover region in the two chromosomes.
Step 3.3: Transform connectors for the crossover region between the two parents and generate two new offsprings.
Step 3.4: Generate exchange rules according to relationship in the crossover region of parents.
Step 3.5: Refer to the exchange rules in Step 3.4 and adjust unreasonable econnectors in the new offsprings.
```

3.4. Insert mutation method

In the *Step* 4, the insert mutation method is utilized to deal with the mutation mechanism. Two cutting points are randomly selected from the array of individual. The array behind the point replaces the array in front of the point, and the other arrays move one location backwards accordingly.

3.5. Guided-crossover method

Guided-crossover is the *Step* 7 in Fig. 4. The guided-crossover method is primarily composed of two stages: (1) generate the crossover block of the chromosome at stage 1, and (2) exchange the chromosome's genetic coding at stage 2.

3.5.1. Generate crossover block

If *block-start* indicates the initial location of the crossover block of the chromosome, and *block-size* records the size of the crossover block, the pertinent algorithms can be described as follows:

```
Step 7.1: Set $h = 1$.
Step 7.2: Randomly generate an integral, $n$, for the judgment of block size; $n = 2, \ldots, m$, and $m$ is the number of connectors;
Step 7.3: Set block-start $= h$, and block-size $= 1$.
Step 7.4: Randomly generate a floating point number, $p$; $0 < p < 1$.
Step 7.5: According to formula (3), calculate the fitness ratio $R_h$ of the connector in the $h$th location of the chromosome.
Step 7.6: Compare the size of $p$ and $R_h$.
    (1) If $R_h \geqq p$, the genetic coding in the $h$th location of chromosome $G$ will be blocked; then, set $h = h + 1$, block-size $=$ block-size $+ 1$, and execute Step 7.4.
    (2) If $R_h < p$, the genetic coding in the $h$th location of chromosome $G$ will not be blocked.
```

(a) If *block-size* $< n$, then set $h = h + 1$, and execute *Step* 7.3.
(b) If *block-size* $\geqq n$, then execute *Step* 7.7.

Step 7.7: Stop searching and set *block*-start at the initial location and *block-size* as the size of the crossover block.

3.5.2. Exchange genetic coding of chromosome

The purpose of adopting the guided-crossover method is to ensure that, after the crossover procedure, the chromosome in the offspring generation meets with the constraints of the assembly problem. The constraints, therefore, should be considered during the duplication and exchange of the genetic coding of the chromosome. There are five steps in the exchange of the genetic coding of the chromosome.

Step 7.8: Randomly choose two chromosomes from the initial population, *parent1* and *parent2*.
Step 7.9: Generate the crossover blocks of these two chromosomes, and divide the genetic block into three sections: *block-front*, *block*, and *block-rear*.
Step 7.10: Duplicate the connectors in the block of chromosomes *parent1* and *parent2* and place them on the corresponding locations of the offspring chromosomes *offspring1* and *offspring2*.
Step 7.11: According to the precedence relationships of the connectors in chromosome *parent2*, duplicate the connectors in the *block-front* area of chromosome *parent1* and place them on the corresponding locations of chromosome *offspring1*. In the same way, duplicate the connectors in the *block-front* area of chromosome *parent2* and place them on the corresponding locations of chromosome *offspring2*, according to the precedence relationships of the connectors in chromosome *parent1*.
Step 7.12: According to the precedence relationships of the connectors in chromosome *parent2*, duplicate the connectors in the *block-rear* area of chromosome *parent1* and place them on the corresponding locations of chromosome *offspring1*. Similarly, duplicate the connectors in the *block-rear* area of chromosome *parent2* and place them on the corresponding locations of chromosome *offspring2*, according to the precedence relationships of the connectors in chromosome *parent1*.

3.6. Guided mutation

Afterward a guided mutation method is proposed to solve the problem so that the offspring generations will be feasible chromosomes in the mutation procedure. If *cut-point* is the node point in the chromosome that will be cut for mutation, and *combine-point* is the node point in the chromosome that will be combined for mutation, then the algorithm for the guided mutation can be described as follows:

Step 8.1: Randomly select a chromosome from the initial population.
Step 8.2: Randomly generate an integer n, n is between 1 and m. m is the number of connectors.
Step 8.3: Randomly generate a *cut-point*, and set *combine-point* = *cut-point* + 1.
Step 8.4: Check the assembly precedence relationship between the connector on the *cut-point* and that on the *combine-point* connector.
(a) If the connector on the *cut-point* has a higher priority of assembly sequence than that on the *combine-point*, then execute *Step* 8.5.
(b) If the connector on the *cut-point* does not have a higher priority of assembly sequence than that on the *combine-point*, then add 1 to the *combine-point*, and execute *Step* 8.4 again.
Step 8.5: Insert the connector on the *cut-point* to the location of *combine-point*-1.
Step 8.6: Repeat the procedure form *Steps* 8.1–8.5 m/n times.

3.7. Exemplification

In this article, stapler example is utilized to illustrate the aforementioned algorithm shown as follows:

Step 1: Generate feasible chromosomes for the initial population.
Step 1.1: Randomly generate two chromosomes, G_1 and G_2, as shown in Fig. 5(a). We use the chromosome G_1 to illustrate the binary-tree concept.
Step 1.2: Set $h = 2$.

3	2	0	1	4	8	5	6	7

G_1

1	2	5	3	4	8	7	6	0

G_2

(a) Chromosomes before binary-tree transforming

2	3	0	1	4	5	6	7	8

G_1

1	2	5	3	4	7	6	0	8

G_2

(b) Chromosomes after binary-tree transforming

Fig. 5. Transformation for the binary-tree algorithms: (a) chromosomes before binary-tree transforming and (b) chromosomes after binary-tree transforming.

Step 1.3: Set g_1's corresponding connector C_3 at root node point r.

Step 1.4: Set g_2's corresponding connector C_2 at the leaf node point l. Because $P_{3,2} = 1$ and since r's left node point is an empty one, insert l at r's left node point (Fig. 6(a)), and set $h = 2 + 1 = 3$. Execute *Step* 1.5.

Step 1.5: The length of the stapler's genetic chromosome is 9. Because h does not equal to 9, repeat *Step* 1.3.

Step 1.3: Set g_1's corresponding connector C_3 at the root node point r.

Step 1.4: Set g_3's corresponding connector C_0 at the leaf node point l. Because $P_{3,0} = 0$ and since r's right node point is empty, insert l at r's right node point (Fig. 6(b)), and set $h = 3 + 1 = 4$. Execute *Step* 1.5.

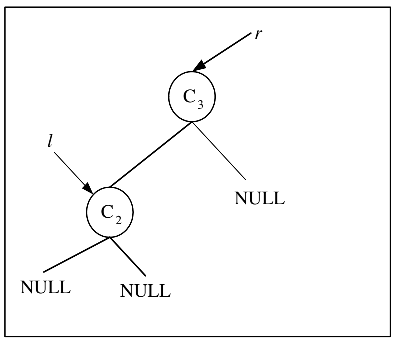
Step 1.5: Because i does not equal to 9, repeat *Step* 1.3. One by one, in the same way, insert all connectors into the connector-based binary-tree diagram, as shown in Fig. 6(c).

Step 1.6: According to the inorder traversal rank, list the order of the feasible assembly sequence for chromosome G_1: $2 \to 3 \to 0 \to 1 \to 4 \to 5 \to 6 \to 7 \to 8$. In the same way, we can get the same feasible solution $1 \to 2 \to 5 \to 3 \to 4 \to 7 \to 6 \to 0 \to 8$ for G_2 (Fig. 5(b)).

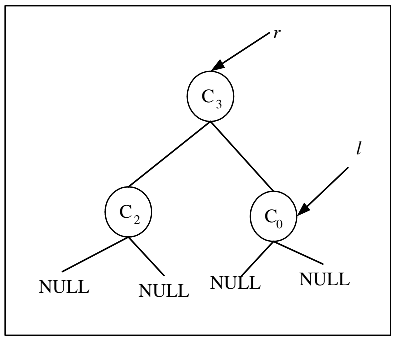
Step 2: Calculate the fitness value of each chromosome. The fitness value F for $G_1 = S^G_{0,1} + S^G_{1,2} + S^G_{2,3} + S^G_{3,4} + S^G_{4,5} + S^G_{5,6} + S^G_{6,7} + S^G_{7,8} = 3.32$, The fitness value F for $G_2 = 2.82$.

Step 3: Process the PMX crossover method.

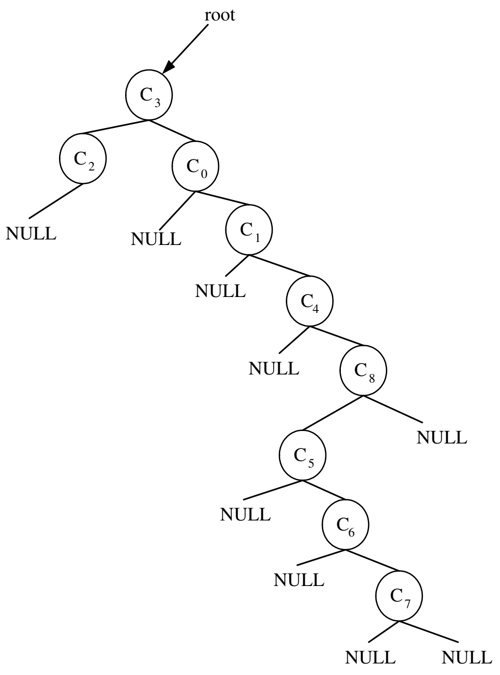
Step 3.1: G_1 and G_2 are hypothesized as chromosomes for the crossover process in Fig. 5(b).



(a)



(b)



(c)

Fig. 6. Connector-based binary tree: (a) insert connector C_3 into binary tree, (b) insert connector C_1 into binary tree and (c) build binary tree for stapler example.

Step 3.2: Randomly generate the crossover region in the chromosomes of G_1 and G_2, as shown in Fig. 7(a).

Step 3.3: Exchange the connectors in the crossover region between the G_1 and G_2. Generate initial offsprings termed *child1* and *child2*. The duplicated connector C_3 and C_7 for *child1* can be found in Fig. 7(b). Furthermore, the duplicated connector C_0 and C_1 for *child2* can be located.

Step 3.4: Create the exchange rule from the duplicated connectors in *Step* 3.3, as shown in Fig. 7(c). Two rules $C_0 \leftrightarrow C_5 \leftrightarrow C_7$C and $C_1 \leftrightarrow C_3$ can be found.

Step 3.5: Adjust the connectors in *child1*and *child2* according the exchange rule in *Step* 3.4. Generate the new chromosomes *offspring1* and *offspring2* (Fig. 7(d)).

Step 4: Process the insert mutation method. Take the *offspring1* in *Step* 3.5 as an example. Location 2 (C_3) and 5 (C_4) are first randomly selected from the *offspring1*. Location 5 is removed to location 2. Then C_5, C_1, C_0, C_6, C_7, C_8 are removed one unit backwards, as shown in Fig. 8. The sequence in *offspring1* is $2 \to 4 \to 3 \to 5 \to 1 \to 0 \to 6 \to 7 \to 8$. In the same way, the sequence in *offspring2* is $1 \to 4 \to 2 \to 7 \to 3 \to 5 \to 6 \to 0 \to 8$.

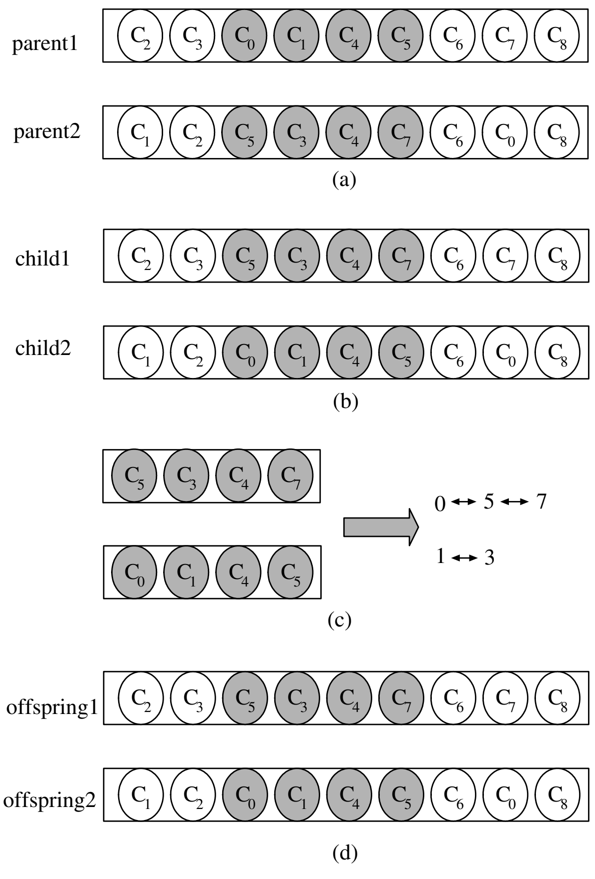


Fig. 7. PMX crossover mechanism for stapler.

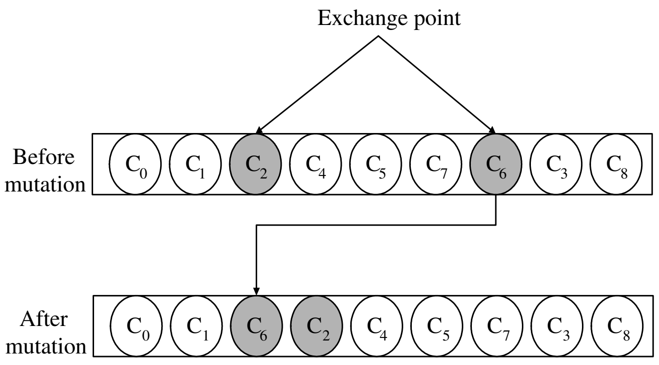


Fig. 8. Insert mutation method for *offspring1* example.

Step 5: Through the binary-tree algorithms *Step* 1, we can transform the *offspring1* to $2 \to 4 \to 3 \to 5 \to 1 \to 0 \to 6 \to 7 \to 8$, in the same way, a feasible solution for *offspring2* can be solved for $1 \to 2 \to 4 \to 7 \to 3 \to 5 \to 6 \to 0 \to 8$.

Step 6: Fitness vale in *offspring1* can be calculated by formula (3) and Fig. 3(b). At last, $F = 4.82$. In the same way, the fitness value $F = 3.82$ in *offspring*.

Step 7: Process the guided-crossover method.

Step 7.1: Chromosome *offspring1* is introduced here; Set $h = 1$.

Step 7.2: Randomly generate an integral $n = 3$; $n = 2, \ldots, 9$.

Step 7.3: Set *block*-*start* = 1, and *block*-*size* = 1.

Step 7.4: Randomly generate a floating-point number, $p = 0.5$.

Step 7.5: According to formula (3), calculate the fitness ratio R_h of the connector in the first location of the chromosome: $R_1 = 1$.

Step 7.6: Compare the size of p and R_h. Because $R_h > p = 0.5$, set *block*-*size* $= 1 + 1 = 2$ and $h = 1+1 = 2$. Execute *Step* 7.4 again.

Step 7.4: Again, randomly generate $p = 0.63$.

Step 7.5: Calculate the fitness ratio R_h of the connector in the second location of chromosome: $R_2 = 0.5$.

Step 7.6: Compare the size of p and R_2. Because $R_2 < p = 0.63$ and *block*-*size* $< n$, set $h = 2 + 1 = 3$, and execute *Step* 3 again. In the same way, the crossover block of the chromosome for *offspring1* can be calculated as shown in Fig. 9.

Step 7.7: The crossover block of the chromosome for *offspring2* can be found in the same method.

Step 7.8: Randomly choose two chromosomes from the initial population, *parent1* and *parent2*.

Step 7.9: Generate the crossover blocks of these two chromosomes, and divide the genetic block

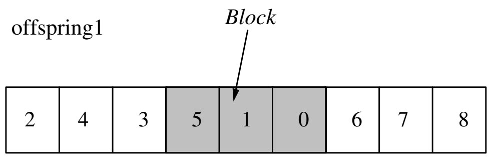


Fig. 9. Block area of *offspring1* chromosome.

into three sections: *block-front*, *block*, and *block-rear*, as shown in Fig. 10(a).

Step 7.10: Only the chromosome of the new offspring generation, *Q1*, to *offspring1* is discussed here. Duplicate connectors C_5, C_1, and C_0 in the block area of *offspring1* for *Q1* directly (Fig. 10(b)).

Step 7.11: According to the precedence relationships of the connectors in chromosome *offspring2*, $C_2 \rightarrow C_4 \rightarrow C_3$, duplicate connectors C_2, C_4, and C_3 in the *block-front* area of chromosome *parent1* and place them on the corresponding locations of chromosome *offspring1* Q_1 (see Fig. 10(c)).

Step 7.12: Duplicate connectors C_6, C_3, and C_9 in the *block-rear* area of chromosome *parent1* and place them on the corresponding locations of chromosome *offspring1* according to the precedence relationships of the connectors in chromosome parent2, $C_7 \rightarrow C_6 \rightarrow C_8$, as shown in Fig. 10(d).

Step 8: Process the guided mutation method.

Step 8.1: Randomly select a chromosome from the initial population. Only the chromosome of *offspring1* is discussed here.

Step 8.2: Randomly generate an integer $n = 3$,the values of n are between 1 and 9.

Step 8.3: Randomly generate *cut-point* = 4, which is the genetic coding on the 4th location of the chromosome (see Fig. 11(a)) and set *combine-point* = *cut-point* = 4+1 = 5.

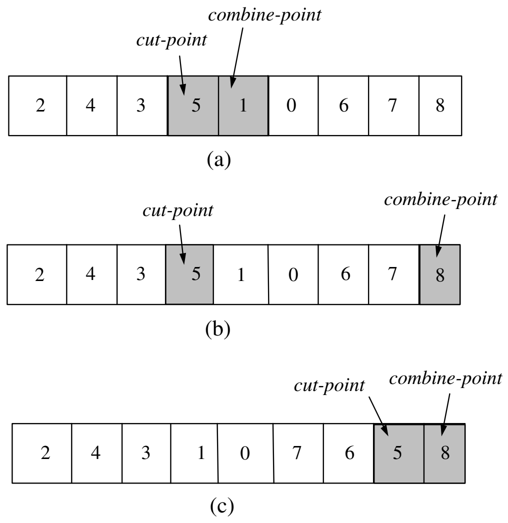


Fig. 11. Operation of guided mutation.

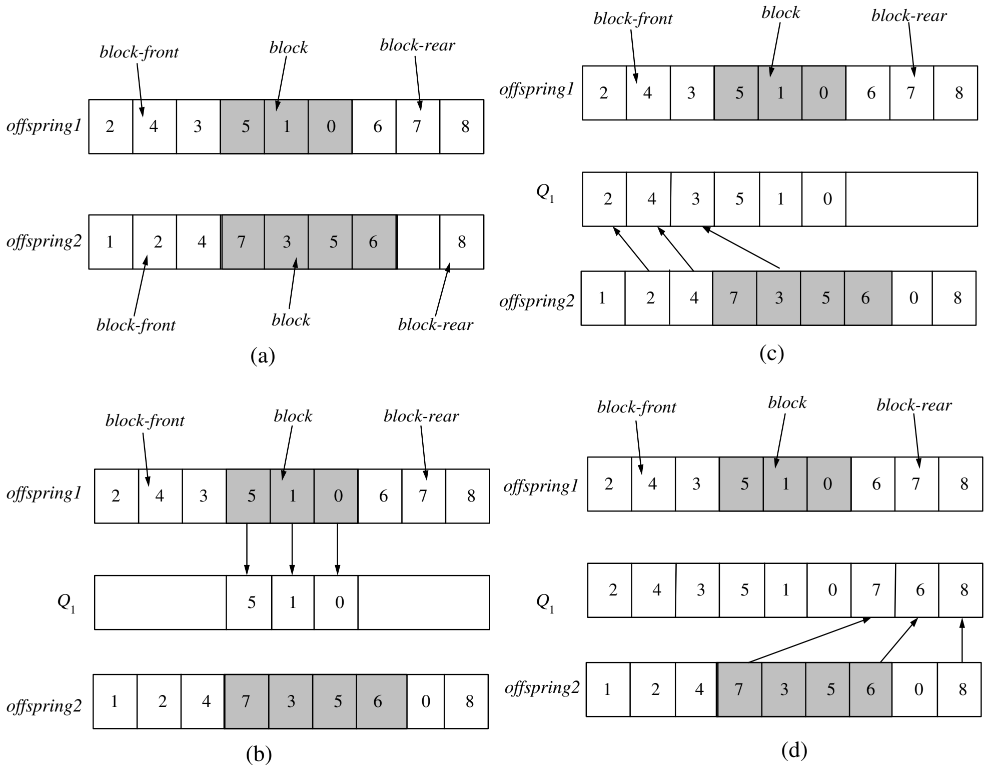


Fig. 10. Operation of guided crossover.

Step 8.4: Check the assembly precedence relationship between the connector on the *cut-point* and that on the *combine-point* connector. Because the connector on the *cut-point* does not have a higher priority of assembly sequence than that on the *combine-point*, set *combine-point* + 1 = 6, and execute *Step* 8.3 again. In the same way, it can be specified *combine-point* = 9,as shown in Fig. 11(b).

Step 8.5: Insert connector C_5 on the *cut-point* to the location of *combine-point*-1, as shown in Fig. 11(c).

Step 8.6: Repeat the procedure from *Steps* 8.2–8.5 three times and stop the mutation mechanism.

Step 9: Repeat the procedure of MAs until stopping criteria is satisfied.

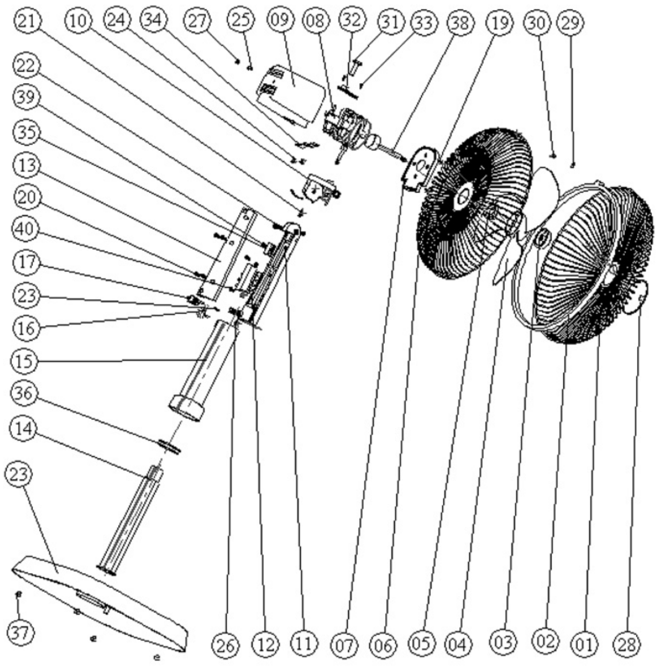
Step 10: Generate new population.

Steps 11, 12: Repeat the procedure until stopping criteria is satisfied.

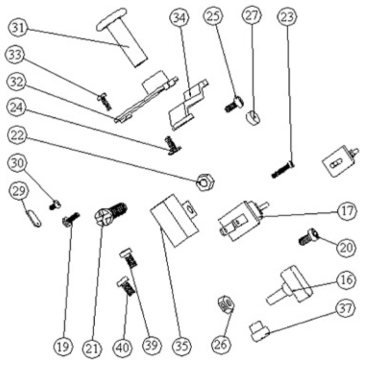
4. Practical examples

The algorithms utilized in Fig. 4 is written in Boland C++6.0. The test environment is that of a Pentium 2.4 GMHz PC at 512 MB RAM. A stapler, an electric fan and a laser printer were exemplified to compare the feasibility of MAs and Guided-GAs. The local search loop is predetermined 10 generations which have been verified by many tests. On the basis of the predetermined local 10 generations, we can group the verification viewpoints into two aspects: (1) preset the maximum number of generations and search the average fitness and the maximum fitness value, and (2) preset the fitness value and determine the computation time and average generation number, respectively.

In the first scenario, the crossover rate is set at 70%; the mutation rate, 30%; the population size, 51; and the maximum number of generations, 1500. In terms of the engineering data, the combination type, assembly tools, and assembly direction are equally important. In accordance with the principle of connector rule (Tseng & Li, 1999), 25 connectors are determined in associated with the electric fan example which consists of 40 components (Fig. 12). The precedence graph for the fan connectors, thereupon, can be ascertained (Fig. 13). Under suchlike condition, the Guided-GAs and MAs implement individually 10 tests for the sake of the required comparisons. From the results of Guided-GAs in Table 4, the average computation time of the overall 10 trials took up 2.808 s, the average fitness value was 16.133, and the maximum fitness value was 16.667. In addition, the average computation time of MAs engaged 3.951 s, the average fitness value was 18.285, and the maximum fitness value was 18.333. Fig. 14 depicts the convergence diagrams of the two algorithms. In the second example, the printer consists of 92 parts, from which 91 connectors are assigned. Figs. 15 and 16 display the part drawing and the connector-precedence graph, respectively. The results of comparison for Guided-GAs and MAs can be found in Table 5. From the results of Guided-GAs in Table 5, the average computation took up 18.5427 s, the average fitness value was 75.595, and the maximum fitness value was 76.67. Furthermore, the average computation time of MAs engaged 25.97 s, the average fitness value was 79.096, and the maximum fitness value was 80. Fig. 17 presents the convergence diagrams of these two algorithms. The critical point here indicates that if we sacrifice a little computation time for achieving more superior solution quality, the MAs is undoubtedly a preferable selection for assembly planning decision.



(a)



(b)

Fig. 12. Illustration of electric-fan: (a) exploded drawing and (b) enlarged scale drawing for small parts in (a).

In the second scenario, the crossover rate is designated at 70%, the mutation rate is 30%, and the population size is 51. In the electric fan's example, the fitness value is objectively predetermined to 16. In 2 of 10 tests, feasible

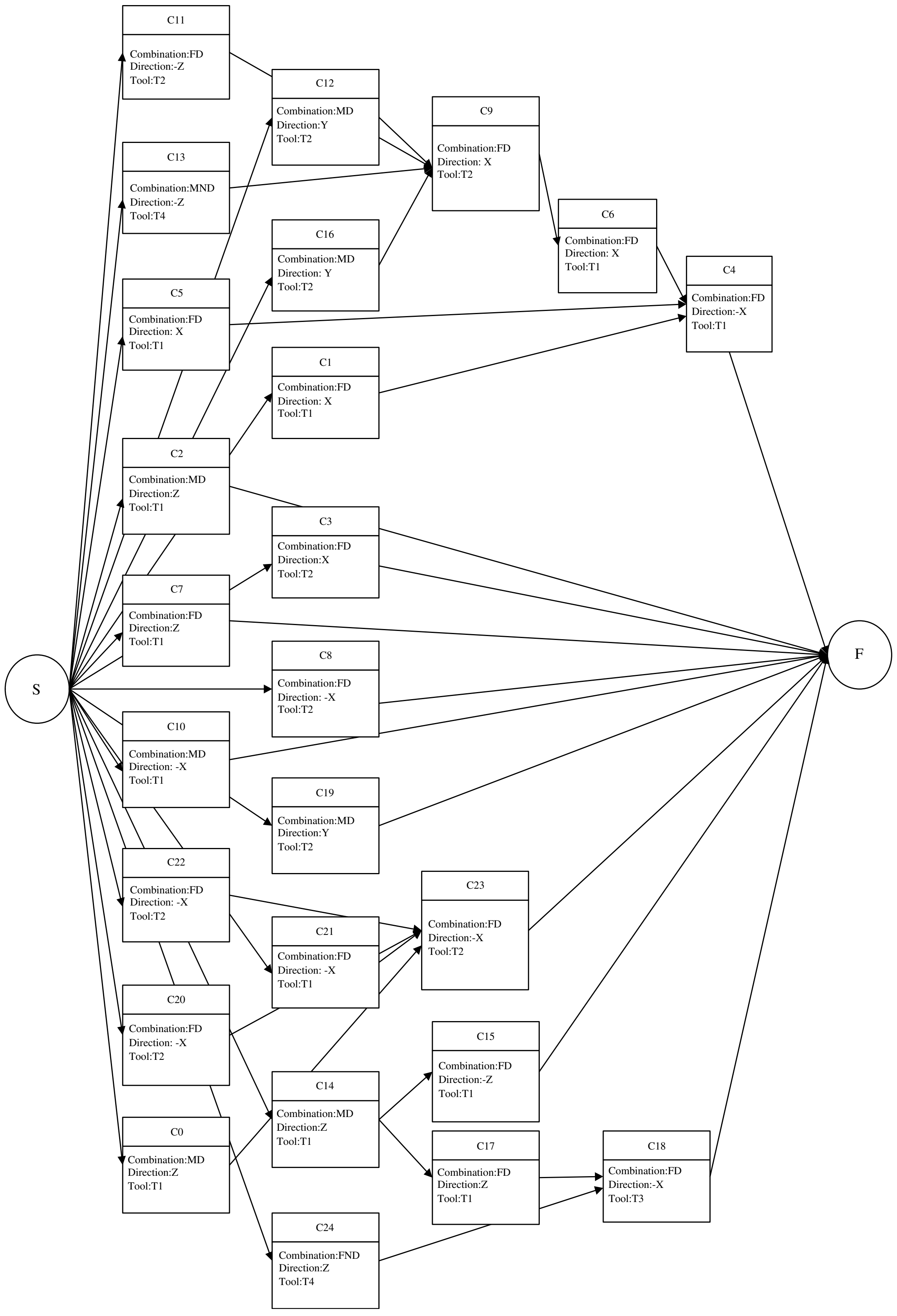

S
F
C11 Combination:FD Direction:-Z Tool:T2
C12 Combination:MD Direction:Y Tool:T2
C9 Combination:FD Direction: X Tool:T2
C13 Combination:MND Direction:-Z Tool:T4
C16 Combination:MD Direction: Y Tool:T2
C6 Combination:FD Direction: X Tool:T1
C4 Combination:FD Direction:-X Tool:T1
C5 Combination:FD Direction: X Tool:T1
C1 Combination:FD Direction: X Tool:T1
C2 Combination:MD Direction:Z Tool:T1
C3 Combination:FD Direction:X Tool:T2
C7 Combination:FD Direction:Z Tool:T1
C8 Combination:FD Direction: -X Tool:T2
C10 Combination:MD Direction: -X Tool:T1
C19 Combination:MD Direction:Y Tool:T2
C22 Combination:FD Direction: -X Tool:T2
C23 Combination:FD Direction:-X Tool:T2
C21 Combination:FD Direction: -X Tool:T1
C20 Combination:FD Direction: -X Tool:T2
C15 Combination:FD Direction:-Z Tool:T1
C14 Combination:MD Direction:Z Tool:T1
C17 Combination:FD Direction:Z Tool:T1
C18 Combination:FD Direction:-X Tool:T3
C0 Combination:MD Direction:Z Tool:T1
C24 Combination:FND Direction:Z Tool:T4

Fig. 13. Precedence graph of connectors for electric fan.

Table 4
Comparison between guided-GAs and memetic algorithms for electric fan

Method	Average run time	Average fitness value	Max fitness value
Guided-GAs	2.808	16.133	16.667
Memetic algorithms	3.951	18.285	18.333

solutions could not be found, whereas, the predetermined value could be found in every course of the proposed MA procedure. The computation time for Guided-GAs took up 0.26275 s, and the average generation is 75.13. In addition, the computation time for MAs engaged 0.1493 s, the global search loop is 4.3 generation. In the printer's example, the fitness value is objectively predetermined to 76. In 3 of 10 tests, feasible solutions could not be found, whereas, the predetermined value could be found in every course of the MA procedure. The computation time for Guided-GAs took up 11.5465 s, and the average generation is 911.29. Furthermore, the computation time for MAs engaged 3.0483 s, the global search loop is 14.9 generation. The results of the illustrated examples demon-

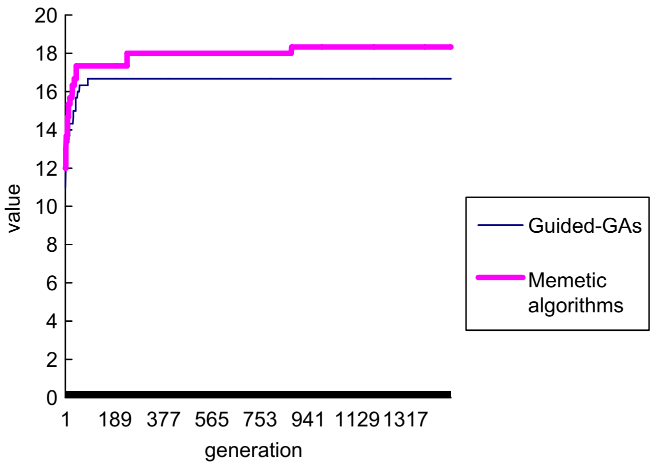


Fig. 14. Convergence plot of electrical fan.

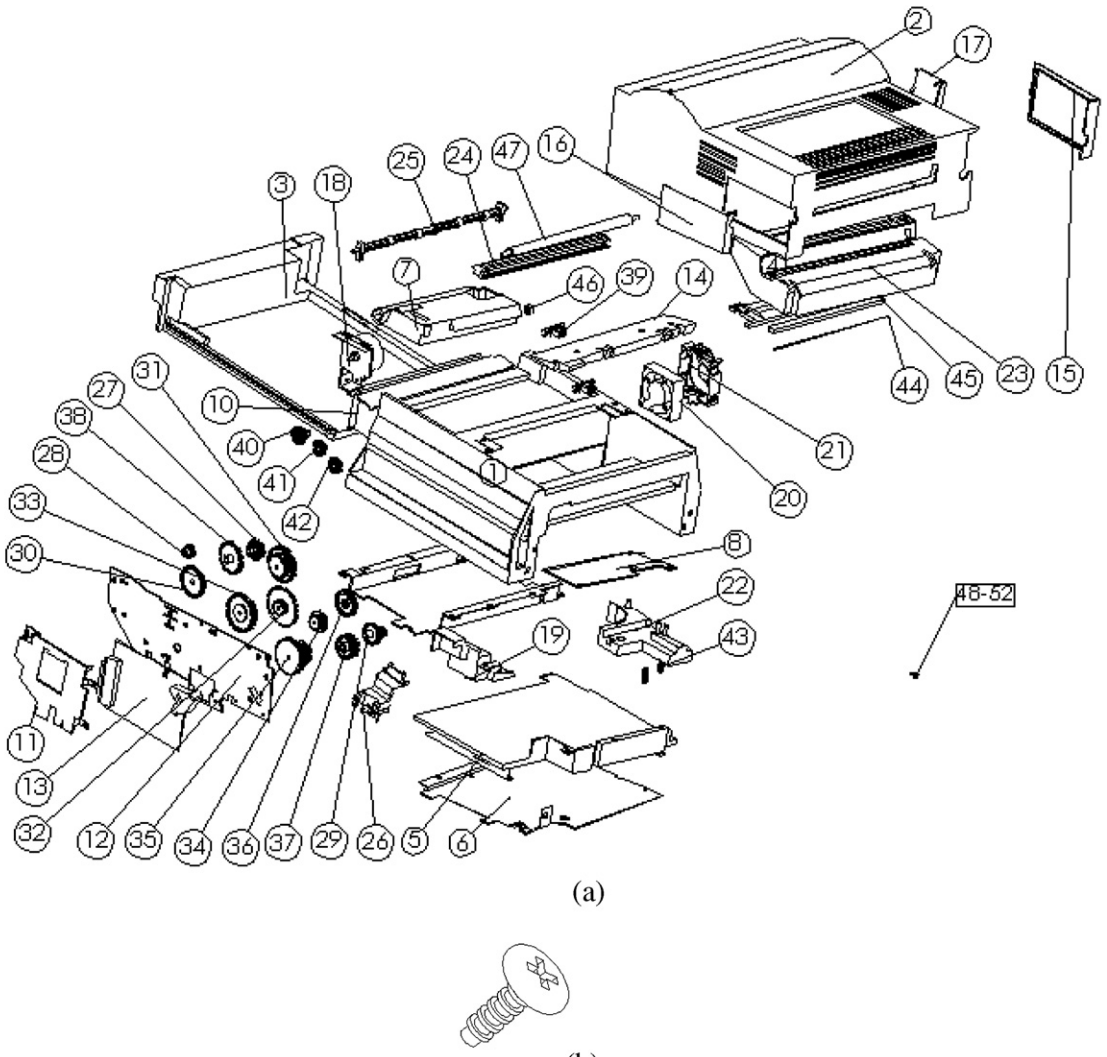


Fig. 15. Laser-printer parts. Parts numbered 48–92 are screws: (a) exploded drawing and (b) enlarged scale drawing of screw parts in (a).

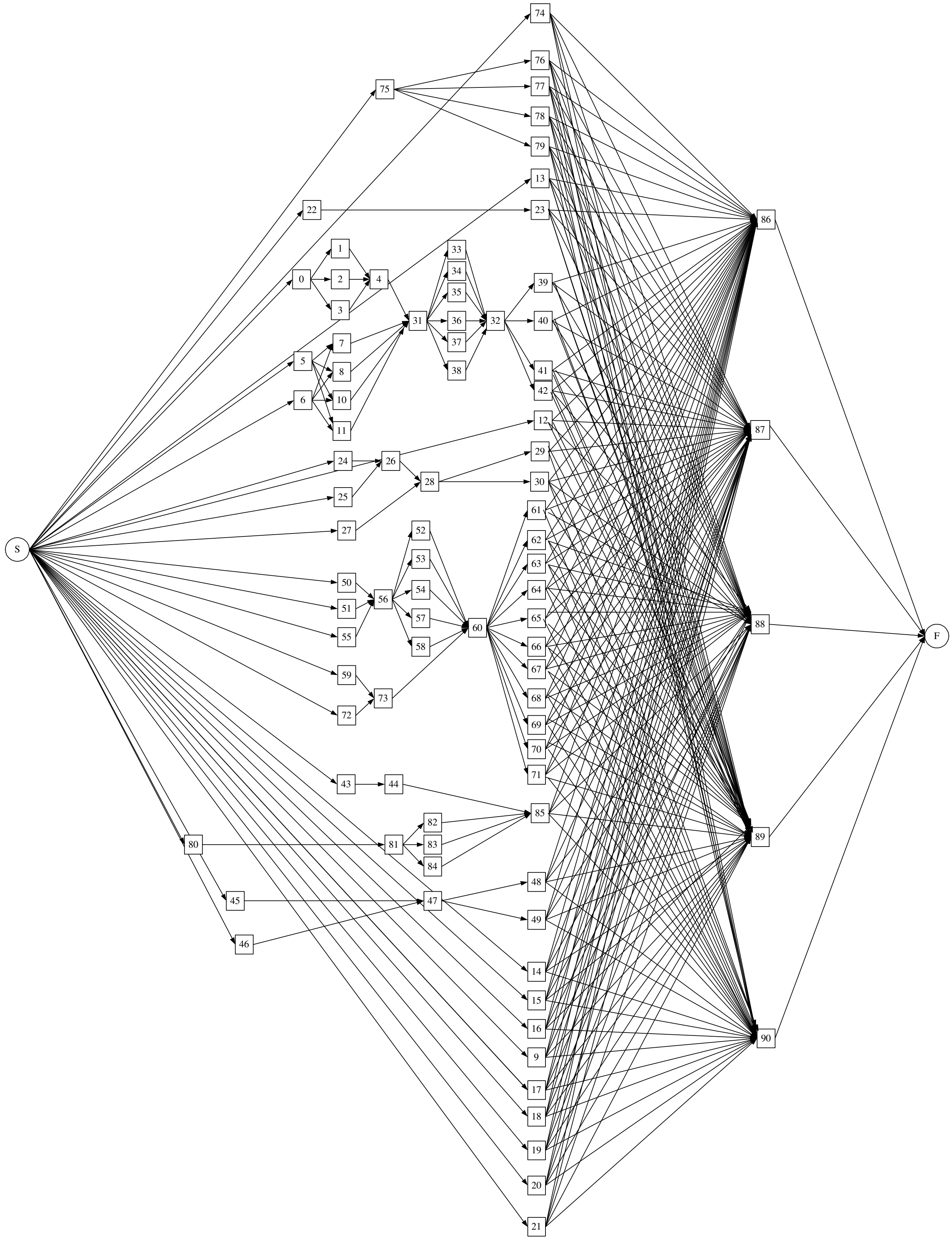


Fig. 16. Precedence graph of connectors for laser printer.

strate that, in terms of solution quality and efficiency, MAs are better than Guided-Gas indeed. By means of the proposed tests, it can be observed that MAs can rise above the local optimal problem owing to the global mechanism

Table 5
Comparison between Guided-GAs and memetic algorithms for laser printer

Method	Average run time	Average fitness value	Max fitness value
Guided-GAs	18.5427	75.595	76.67
Memetic algorithms	25.97	79.096	80

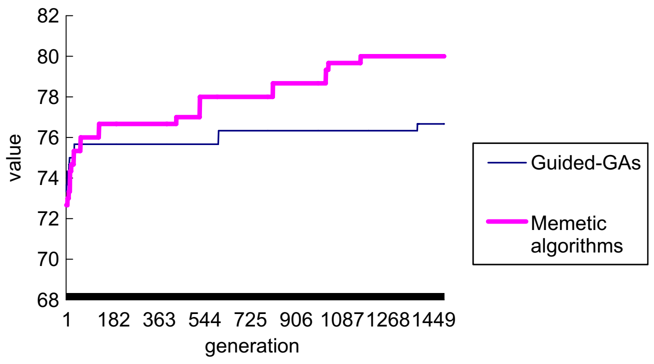


Fig. 17. Convergence plot of laser printer.

consisted of the PMX crossover and insert mutation accordingly.

5. Conclusions

In the prior research, Tseng et al. (2004) takes the lead in advocacy to associate with genetic algorithms and connector-based assembly planning. Using genetic algorithms features of speed and flexibility can conform to the various applications even as the extension to assembly line balancing. Concerning the situation with large-scale constraint assembly problems, however, GAs will generate quantities of infeasible solutions during the evolution procedure, and then failed in searching the feasible solution on occasion. Accordingly, a Guided-GA approach proposed by Tseng (2006) copes with the assembly planning problems that consist of large-scale constraints. In general, Guided-GAs can search out high-quality solutions faster than traditional genetic algorithms, but short on reliability to find the global-optimal solutions. As a result, in this paper we focus on improving the performance of Guided-GAs, and memetic algorithms are proposed further to resolve all perplexities occurred in the foregoing methods. The Guided-GAs in consequence are used as the local search mechanism to work out the extensive constraints in assembly planning, and in the appearance of Guided-GAs, the global search engine comprised by PMX crossover and inserted mutation is employed to enhance the quality of Guided-GAs. In contrast with Guided-GAs, the entirety of solution quality of MAs enhances around 4.6–13.3%. Although the computation time of MAs increased around 40% as compared with Guided-GAs, in the future it seems meaningless due to a rapid expansion of computer technology. Suchlike objectives eventually carry off the validity through exemplified pilot schemes.

Randomly local search can be delineated as one of many aspects in the future. The proposed approach can improve the computation time of MAs. In addition, it is hypothesized in the current study that the connector-based precedence relationships among connectors graph are predetermined. The precedence relationships could be automatically generated from the recognition of CAD data structure, and can be pay close attention in the future.

Acknowledgement

Research was supported by the National Science Council of the Republic of China under Grant No. NSC 94-2213-E-212-017.

References

Abdullah, T. A., Popplewhell, K., & Page, C. J. (2003). A review of support tools for the process of assembly method selection and assembly planning. *International Journal of Production Research, 41*(11), 2391–2410.

Akagi, F., Osaki, H., & Kikuci, S. (1980). The method of analysis of assembly work based on the fastener method. *Bulletin of the JSME, 23*(184), 1670–1675.

Baldwin, D. F., Abel, T. E., Lui, M. C. M., De Fazio, T. L., & Whitney, D. E. (1991). An integrated computer aids for generating and evaluating assembly sequences for mechanical products. *IEEE Transaction on Robotics and Automation, 7*(1), 78–94.

Berretta, R., & Rodrigues, L. F. (2004). A memetic algorithm for a multistage capacitated lot sizing problem. *International Journal of Production Economics, 87*, 67–81.

De Fazio, T. L., & Whitney, D. E. (1987). Simplified generation of all mechanical assembly sequence. *IEEE Journal of Robotics and Automations, 3*(6), 640–658.

Dini, G., Failli, F., Lazzerini, B., & Marcelloni, F. (1999). Generation of optimized assembly sequences using genetic algorithms. *Annals of the CIRP, 48*, 17–20.

Fransca, P. M., Mendes, A., & Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. *European Journal of Operation Research, 132*, 224–242.

Fujimoto, H., & Sebaaly, M. F. (2000). A new sequence evolution approach to assembly planning. *Transaction of the ASME Journal of Manufacturing Science and Engineering, 122*, 198–205.

Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization*. New York: John Wiley & Sons.

Göngör, A., & Gupta, S. M. (1997). An evaluation methodology for disassembly processes. *Computer & Industrial Engineering, 33*, 329–332.

Gottipolu, R. B., & Ghosh, K. (1997). Representation and selection of assembly sequences in computer-aided assembly process planning. *International Journal of Production Research, 35*(12), 3447–3465.

Guan, Q., Liu, J. H., & Zhong, Y. F. (2002). A concurrent hierarchical evolution approach to assembly process planning. *International Journal of Production Research, 40*, 3357–3374.

Homem De Mello, L. S., & Sanderson, A. C. (1991). A correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transaction on Robotics and Automation, 7*(2), 228–240.

Maheswaran, R., Ponnambalam, S. G., & Aravindan, C. (2005). A meta-heuristic approach to single machine scheduling problems. *International Journal of Advanced Manufacturing Technology, 25*, 772–776.

Moscato, P. (1992). A memetic algorithm approach for the traveling salesman problem: implementation of a computational ecology for combinatorial optimization on message-passing system. In *Parallel computing and transportation applications* (pp. 187–194). Amsterdam, The Netherlands: IOS Press.

Muruganandam, A., Prabhaharan, G., Asokan, P., & Baskaran, V. (2005). A memetic algorithms approach to the cell formation problem. *International Journal of Advanced Manufacturing Technology, 25*, 988–997.

Smith, G. C., & Smith, S. S. F. (2003). Automated initial population generation for genetic assembly planning. *International Journal of Computer Integrated Manufacturing, 16*, 219–228.

Tseng, H. E. (2006). Guided genetic algorithms for solving larger constraint assembly problem. *International Journal Production Research, 44*(3), 601–625.

Tseng, H. E., & Li, R. K. (1999). A novel means of generating assembly sequences using the connector concept. *Journal of Intelligent Manufacturing, 10*, 423–435.

Tseng, H. E., Li, J. D., & Chang, Y. H. (2004). Connector-based approach to assembly planning using genetic algorithms. *International Journal Production Research, 42*(11), 2243–2261.

Tseng, H. E., & Tang, C. E. (2006). A sequential consideration for assembly sequence planning and assembly line balancing using the connector concept. *International Journal of Production Research, 44*(1), 97–116.

Wess, M. L. (1999). *Data structures and algorithms in C++* (2/E ed.). London: Addition Wesley.

Yin, Z. P., Ding, H., Li, H. X., & Xiong, Y. L. (2003). A connector-based hierarchical approach to assembly sequence planning for mechanical assemblies. *Computer-Aided Design, 35*, 37–56.

Zha, X. F., Lim, S. Y. E., & Fok, S. C. (1998). Integrated intelligent design and assembly planning: a survey. *International Journal of Advanced Manufacturing Technology, 14*, 664–685.