Available online at www.sciencedirect.com
SCIENCE DIRECT®
PERGAMON
Engineering Applications of Artificial Intelligence 15 (2002) 447–457
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

A hybrid genetic algorithm approach on multi-objective of assembly planning problem

Ruey-Shun Chen*, Kun-Yung Lu, Shien-Chiang Yu

Institute of Information Management, National Chiao Tung University, 1001 Ta Hsueh Road, Hsin-Chu, Taiwan

Abstract

In practice, modeling an assembly system often requires assigning a set of operations to a set of workstations. The aim is to optimize some performance indices of an assembly line. This assignation is usually a tedious design procedure so a significant amount of manpower is required to obtain a good work plan. Poor assembly planning may significantly increase the cost of products and reduce productivity. However, these optimization problems fall into the class of NP-hard problems. Finding an optimal solution in an acceptable time is difficult, even using a powerful computer. This study presents a hybrid genetic algorithm approach to the problems of assembly planning with various objectives, including minimizing cycle time, maximizing workload smoothness, minimizing the frequency of tool change, minimizing the number of tools and machines used, and minimizing the complexity of assembly sequences. A self-tuning method was developed to correct infeasible chromosomes. Several examples were employed to illustrate the proposed approach. Experimental results indicated that the proposed method can efficiently yield many alternative assembly plans to support the design and operation of a flexible assembly system.

Keywords: Assembly planning; Genetic algorithm; Assembly line balancing; Multi-objective; Design for assembly

1. Introduction

Fig. 1 illustrates how the production of a product or a system involves many parts or operations. This dependence raises an assembly line modeling problem that requires a set of tasks to be assigned to a set of workstations and establishing workstations with a set of tools and machines, such that constraints are satisfied and some performance indices are optimized. Usually, the number of workstations, cycle time, workload balance, and other factors, are used to measure the performance of an assembly line. Indeed, a feasible assembly plan can increase production efficiency and reduce the cost of a product. Furthermore, assembly planning can facilitate concurrent engineering in product development, operations and system analysis, to enhance the quality of a particular system. Such a problem is a multi-objective problem.

A flexible assembly system is usually employed to fabricate the products in small batches and meet many specific orders, to serve quickly and responsively service customers in a competitive environment. Recently, assembly line planning problems have received considerable attention. These studies can be classified as addressing one of a number of production optimization problems, including the well-known assembly line balancing problem (ALB) (Ponnambalam et al., 2000; Kim et al., 1996), the tooling problem (Lazzerini and Marcelloni, 2000), and many routing and scheduling problems (Leu et al., 1996; Song et al., 2001). Although many heuristic algorithms have been employed to solve these problems, a problem that involves m workstations with n tasks requires time $O(m^2n)$ to find the optimal solution (Lucertini et al., 1998). A typical NP-hard optimization problem cannot be solved in polynomial time. Finding an optimal solution in an acceptable time is thus difficult, even using a powerful computer.

Although the above problems have been extensively studied, as surveyed by Tai (1997), only a few companies utilize published techniques to balance their lines, because, models usually only consider "contact-base precedence relationships" when assigning operations to the workstations. The "contact-base precedence relationships" are generated from the connection relation-

*Corresponding author. Tel.: +886-3-571-2121x57428; fax: +886-3-573-3792.
E-mail address: rschen@bis03.iim.nctu.edu.tw (R.-S. Chen).

0952-1976/02/$ - see front matter
PII: S0952-1976(02)00073-8

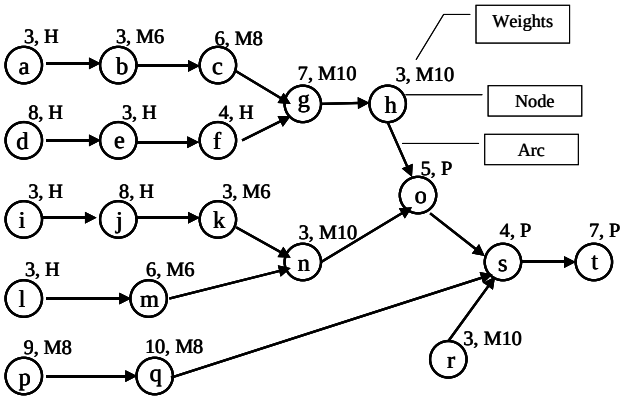

Fig. 1. Precedence diagram of the assembly tree. The upper symbol of node means the process time and tooling.

ships of two parts—no contact (translatable), attached and fastened (non-translatable). Only the number of workstations and the cycle time are considered. Ponnambalam et al. (2000), while also considering multi-objective assembly planning, focused on the objectives of line balance, line efficiency, and optimization of smoothness index, according to contact-base precedence relationships. Importantly, an optimal assembly plan must simultaneously consider crucial factors, including physical and geometrical constraints, similarities among assembly operations, the frequency of tool changes, and others. These factors are frequently considered to be marginal in an assembly system and may strongly influence the performance of the system.

According to Lucertini et al. (1998), effective assembly line modeling can reduce the number of tools or machines used and can eliminate a bottleneck in a production line. In a flexible assembly system, bottlenecks are often caused by the transportation network. Part transfer minimization must be considered when addressing traffic problems and considering set-up costs. Although Lucertini et al. pointed out several important facts to evaluate the performance of the production line, they lacked a clear approach to finding the optimal solutions.

Based on evolutionary computation techniques, genetic algorithm (GA) methods have received much attention and have been applied successfully in many research fields over the last decade (Fogel, 1995; Gen and Chen, 1997; Runarsson and Jonsson, 1999). Holland initially introduced the GA, which is a global search technique (Holland, 1975). GA explores the solution space using concepts taken from natural genetics and evolution theory. During the search, candidate solutions in the solution space are encoded as symbolic strings, known as chromosomes. The search algorithm analyzes and extracts superior evolving information from the search space, and guides the search in a pre-specified direction. Indeed, the GA method is robust and effective in various task domains (Mak et al., 1998; Onwubolu and Mutingi, 2001).

This study presents a hybrid GA approach combined with a self-tuning method to solve the problem of assembly line planning. Initially, several well-known heuristic methods are used to generate some feasible solutions. These solutions are then included into the randomly derived population of evolving pool. The goal of including heuristics solutions in this population is to reduce the search space from the size of the global space, thereby reducing the searching time. The proposed tuning mechanism can maintain effective schemata of chromosomes to prevent the generation of incorrect precedence relationships in the assembly. Based on the "*Schema Theorem*" of Goldberg (1989), effective schemata can enhance the GA process. Exhaustive search or mathematical programming methods theoretically guarantee optimal solutions, but are infeasible for solving complex problems due to their unacceptable computation. This study aims to reduce evolution times and improve the quality of the solutions obtained.

An example including 20 operations and six workstations was presented to elucidate the effects of the proposed approach. The experiments were run in three phases. Firstly, the experiments were tested by the pure GA method. Secondly, the problem was solved by conventional heuristic methods. Finally, the solutions generated by the heuristic methods were introduced into the evolving pools used by the pure GA method and then retested by the proposed method. Experimental results showed that the proposed method can significantly improve the quality of solutions obtained by conventional heuristic methods. The proposed method can also efficiently yield many alternatives of assembly plan to support the design and operation of a flexible assembly system.

The rest of this paper is organized as follows: Section 2 defines the problem of assembly line modeling. Section 3 describes the GA. Section 4 considers the numerical experiments involving the proposed method. Conclusions and areas for future research are finally discussed in Section 5.

2. Problem statement and method of solution

A product's assembly relationships can commonly be represented as a weighted acyclic graph $G(N, A)$ (Fig. 1), where the node (N) weights represent the time, tools and others, required to perform the tasks of assembly and the arcs (A) represent the precedence constraints of assembly. In the assembly tree of $G(N, A)$, if arc $(i,j) \in A$, then a workstation w may perform task j only when the sub-assembly that results from task i is available on workstation w. This study assumes that if no precedence relationship exists between i and j, then these tasks can

be executed in parallel and performed on different sub-assemblies. Consider a set (W) of w flexible workstations, $W = 1, 2, \ldots, w$. A workstation $w \in W$ can execute only one task at a time, and preemption is not permitted.

In practice, the contact-base feature is usually employed to represent the precedence relationships of the product. A planner can successively assign tasks to workstations according to the precedence diagram. However, the contact-base precedence diagram cannot effectively express the complexity of the assigned assembly relations. A poor assignment may increase the operational time. For example, the assembly drawings shown in Fig. 2 can be transformed into the precedence diagram in Fig. 3. If the assignment of Part b precedes that of Part e, then the difficulty of putting Part e into Part f is increased.

This paper employs the idea of penalties to express the complexity of assembly relations to prevent the poor assignment. The use of penalties allows many criteria to be considered simultaneously in assembly planning. The penalty represents the cost incurred (for example, by increasing operational time) by following an unsuitable or infeasible assembly sequence. For example, the penalty index is set to zero if part i is not violated by part j; the penalty index is set to another number if the assignment is poor, such that the operational time increases, and is set to a large number when the assembly relation is prohibited. Other crucial factors in assembly planning, such as the frequency at which tools are changed, the similarity of assembly operations, the quality of the assembly, and others, can be evaluated using the penalty index. A small penalty index is generally assigned to a beneficial assignment. Table 1 presents an example of penalty weightings of some crucial factors.

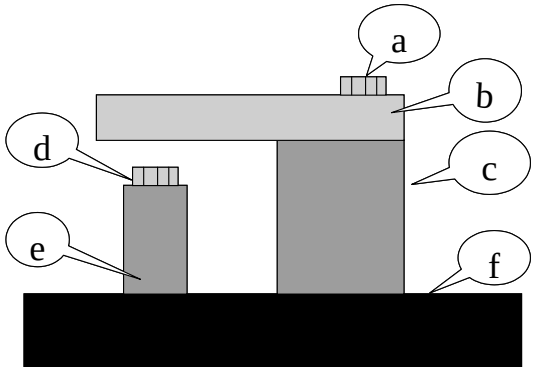

Fig. 2. An example of assembly diagram.

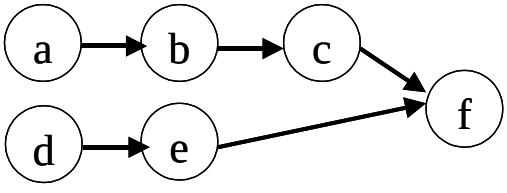

Fig. 3. An example of contact-base precedence diagram.

The penalty matrix of an assembly plan can be synthesized from the penalty index of respective crucial factors, using Eq. (1). p_{ijk} represents the penalty index of crucial factors between parts i and j; m is the number of crucial factors evaluated, and w_k is the weighting of those crucial factors. In practice, the planner can decide to set the weightings to different values

$$P_{ij} = \sum_{k=1}^{m} w_k \times p_{ijk}. \tag{1}$$

Assembly line modeling problems have been conventionally classified into two types—Type I and Type II (Hackman et al., 1989). In Type I problems, the required production rate (cycle time), assembly tasks, tasks times, and precedence requirements are given. Designing a new assembly line is generally such a problem whose objective is to minimize the number of workstations. In Type II problems, the number of workstations or production employees is fixed. The aim is to minimize the cycle time. Type I problems often occur when a factory wants to produce the optimum number of items using a fixed number of workstations without adding new machines. This study focuses on the Type II problem.

Optimal solutions of the ALB problems have typically been obtained by following a heuristic approach, such as those of maximum ranked positional weight (Helgeson and Birnie, 1961), maximum total number of follower tasks (Brian and Patterson, 1984), minimum total number of predecessor tasks (Elsayed and Boucher, 1994), maximum task time (Kilbridge and Wester, 1961), minimum reverse positional weight (Elsayed and Boucher, 1994), and others. However, these approaches can only solve the problem of a single model and deterministic task time. This study seeks various Pareto optimal solutions to the problem of multiple objective assembly line modeling. A Pareto optimal solution is one where no other solutions are superior to the solution in all objectives. This study employed a GA approach to search for Pareto optimal solutions in parallel by maintaining a population of solutions. The solution procedure is summarized as follows.

Step 1: Employ heuristic methods to obtain feasible solutions.

Step 2: Randomly generate an initial population.

Step 3: Insert the solutions derived from heuristic methods into the initial pools.

Step 4: Validate and adjust the chromosomes of the population.

Table 1
An example of penalty index in different assembly relations

Penalty index	Description
0	Simple work, forced precedent sequence, direct or absolute above relation
1–5	A little difficulty, need careful operation, tools changing infrequently, close above relation
6–9	Very difficult, easily damage the component, tools changing frequently, loose above relation
999	Prohibit, no any relation

Step 5: Employ GA to evolve the population.

Step 6: Check the stop criteria. If a match exists, then stop; otherwise, go to Step 4.

The system objectives are expressed as follows:

$$\text{Minimize } f1 = ct \text{ where } ct \text{ is cycle time,} \quad (2)$$

$$\text{Maximize } f2 = cs \text{ where } cs \text{ is workload smoothness,} \quad (3)$$

$$\text{Minimize } f3 = ft \text{ where } ft \text{ is the frequency of tool changes,} \quad (4)$$

$$\text{Minimize } f4 = tn \text{ where } tn \text{ is the number of tools,} \quad (5)$$

$$\text{Minimize } f5 = tp \text{ where } tp \text{ is the total penalty of assembly relations,} \quad (6)$$

$$\text{Subject to : the workstations and the precedence relationships are given.} \quad (7)$$

In practice, minimizing cycle time can increase yields. Maximizing the workload smoothness can balance the loads of respective workstations. The objectives shown in Eqs. (4) and (5) can reduce the cycle time and the required capital investment, respectively. Minimizing the total penalty can lower the complexity of assembly relations.

3. Genetic algorithm and application

LibGa (Corcoran and Wainwright, 1993), coded in C language, was employed to simulate the given problems. LibGa provides complete source codes for GA and a friendly interface for users to define both the objective function and the GA parameters. This study uses Turbo C++ to code the objective function and the tuning function, based on the presented algorithm, and then links these functions to LibGa to generate the execution program. The GA parameters are stored in a standard text file, which can be modified by any text editor. While the LibGa is running, the GA parameters are automatically loaded into the LibGa.

3.1. Genetic algorithm

The genetic search process used herein is outlined below:

Step 1: Generate a random initial population of chromosomes of size P.

Step 2: Decode all chromosomes and evaluate the objective function of their corresponding candidate solutions.

Step 3: Determine the fitness values of the chromosomes using the predetermined objective function.

Step 4: If the elitism policy is employed, insert the best chromosomes into the new generation pool.

Step 5: Choose a pair of parent chromosomes from the current population without replacement. Apply the crossover and mutation operators to yield a pair of new chromosomes.

Step 6: Insert the new chromosomes into the new population. If the new population is smaller than P, return to Step 5.

Step 7: If the pre-specified stopping criterion has been met, then stop the search process. Select and decode the overall best chromosome to be the final solution. Otherwise, proceed to the next generation and replace the population with the new one, and return to Step 2.

3.2. Chromosome representation

A string scheme is required to encode the candidate solutions in the form of symbolic strings, called chromosomes, to solve the studied problems by the GA technique. Integers are employed to represent the operation of assigning task to workstation in successive number of $1, 2, \ldots, N$, respectively. Fig. 4 presents an example of encoded strings that involves three tasks and three workstations.

Tasks / Workstations	a	b	c
W1	1	2	3
W2	4	5	6
W3	7	8	9

Fig. 4. An example of the encoded strings for assigning tasks to workstations.

Encoding: A candidate solution can be encoded as a string by a permutation of $\{1, 2, \ldots, N\}$ to form a chromosome. For example, "2-1-4-3-5-6-8-7-9" is one possible chromosome in the example shown in Fig. 4.

Decoding: A feasible solution can be decoded from the given chromosome. For example, string of "1-4-5-2-9-6-8-7-3" can be decoded as, Task a is assigned to Workstation W1 (String 1), b to W2 (String 5), and c toW3 (String 9), according to the encoded string in Fig. 4. Strings 4 (Task a), 2 (Task b), 6 (Task c), 8 (Task b), 7 (Task a), and 3 (Task c) are dummies.

Self-tuning: Occasionally, the sequence of genes in a given chromosome may violate the precedence relationships. A self-tuning mechanism is used here to alter an infeasible chromosome, as follows:

Step 1: Input a gene in sequence from the chromosome processed.

Step 2: Validate the sequence of the given gene.

(1) If the gene satisfies the precedence relationships, then assign it to the corresponding workstation and store it in the correct gene record; if the gene is a dummy string (the corresponding task has been assigned) then store it in the correct gene record; otherwise,

(2) Put the gene into the backlog of bad genes.

Step 3: Inspect the backlog of bad genes.

(1) If the backlog of bad genes is empty, then go Step 4,

(2) Inspect the precedence relationships of every gene in the backlog record:

(a) If the gene is a dummy, then remove it from the backlog record and put it into the correct gene record,
(b) If the gene satisfies the precedence relationships, assign it to the corresponding workstation and remove it from the backlog record; then, put it into the correct gene record; otherwise,
(c) Reserve the gene in the backlog record.

Step 4: Check the primary chromosome. If it is empty, then go to Step 6.

Step 5: Go to Step 1.

Step 6: Return the correct gene to the population.

3.3. Selection operation

The selection policy employed herein is a combination of the rotating roulette wheel strategy and the elite strategy. The elite strategy can force the best surviving chromosomes, based on the fitness values of the current generation, into the next generation. The chromosomes with higher fitness are more likely to become parents of new chromosomes. The reproduction gap defined in the GA configuration file determines the proportion of elite chromosomes. The best chromosomes in the last generation are copied directly into the next generation. Rotating the roulette wheel to determine which one will be chosen from the old pool as the parent, facilitates the selection of the remaining chromosomes that form these new generations. Notably, two chromosomes are selected in each cycle. Then, crossovers and mutations are employed to process the pair of parent chromosomes, and thereby breed a pair of offspring.

3.4. Crossover operation

Although LibGA provides 10 crossover methods, only "ORDER1", "ORDER2", "PMX" and "CYCLE," which are suited to permutation processing, are employed here. The procedures of each method are briefly described below.

*ORDER*1: Initially, select two random crossover points. Then, the offspring inherits the elements between the two crossover points from the selected parent, in the order and position in which the parent appeared. The remaining elements are inherited from the other parent in the order in which they appeared in that parent, beginning with the first position, followed by the second crossover point, and skipping over all elements that are already present in the offspring.

*ORDER*2: Select four random crossover points as key positions to determine the order in which these genes appear in one parent and are subsequently imposed on the other parent to produce two offspring.

PMX: Randomly select two crossover points. Then, the offspring inherits the elements between the two starting positions in one of the parents. Each element between the two crossover points of the other parent is mapped to the position of this element in the first parent. Then, the remaining elements are inherited from the other parent.

CYCLE: Randomly select a crossover point as a cycle starting point. The offspring inherits the element at the starting point of the selected parent. The element that is located in the corresponding position in the other parent cannot then be placed in this position, and thus it takes a position in the selected parent from which the offspring then inherits it. This procedure is repeated until the initial item in the unselected parent is encountered, completing the cycle. The offspring inherits from the unselected parent any elements that are not yet present.

The crossover operation is processed whether or not it is determined by a value generated randomly. If the value exceeds the crossover rate that was established in the configuration file of GA parameters, then the clone operation, rather than the crossover operation, is performed.

3.5. Mutation operation

The swap method is used herein as the mutation operation. The swap procedure randomly selects two

points to determine the two changing points of the selected parent. The mutation operation is processed whether or not it is determined by a value generated randomly. If the value exceeds the mutation rate established in the configuration file of GA parameters, no mutation is performed and the selected parent is passed directly into the new generation.

3.6. Fitness function

A weighting function (Eq. (8)) is derived from the five objective functions specified in Section 2 to evaluate the fitness values (FV) of a given chromosome and thereby simultaneously consider the many criteria of the given problem. In Eq. (8), w_i represents the weighting values of objectives. A smaller FV means that the chromosome is more likely to survive to the next generation. In this study, several combination values of w_i were used to test for finding the optimal solutions

$$FV = \sum_{i=1}^{5} w_i f_i. \tag{8}$$

4. Numerical example

4.1. Experimental example

An example, shown in Fig. 1, is used to demonstrate the proposed approach. It involves six workstations and 20 tasks. Table 2 lists the process times and tools used for each task. One tool change was assumed to take 3 s.

Table 2
Process time and tool of the given example

Task	Process time	Tooling
a	3	H
b	3	M6
c	6	M8
d	8	H
e	3	H
f	4	H
g	7	M10
h	3	M10
i	3	H
j	8	H
k	3	M6
l	3	H
m	6	M6
n	3	M10
o	5	P
p	9	M8
q	10	M8
r	3	M10
s	4	P
t	7	P

Workstations / Tasks	W1	W2	W3	W4	W5	W6
a	1	21	41	61	81	101
b	2	22	42	62	82	102
c	3	23	43	63	83	103
d	4	24	44	64	84	104
e	5	25	45	65	85	105
f	6	26	46	66	86	106
g	7	27	47	67	87	107
h	8	28	48	68	88	108
I	9	29	49	69	89	109
j	10	30	50	70	90	110
k	11	31	51	71	91	111
l	12	32	52	72	92	112
m	13	33	53	73	93	113
n	14	34	54	74	94	114
o	15	35	55	75	95	115
p	16	36	56	76	96	116
q	17	37	57	77	97	117
r	18	38	58	78	98	118
s	19	39	59	79	99	119
t	20	40	60	80	100	120

Fig. 5. The encoded strings of assigning relationships.

Encoding: Fig. 5 shows the assigned relationships among tasks and workstations, in string form. Although there are 120 relationships, there are only 20 effective ones. For example, assigning the task, f, to a different workstation can be encoded as 6, 26, 46, 66, 86, and 106. If string 26 is in front of the other five strings in a given chromosome, then the other five strings become dummies.

System objective evaluation: The system objectives were measured by the performance indices of cycle time, workload smoothness, number of tool changes, number of tools and machines used, and complexity of assembly relationships. The respective objects were evaluated as described as follows:

Cycle time: The cycle time equals the longest process time on any workstation.

Workload smoothness: The workload smoothness can be calculated as below

$$\text{Workload deviation} = \frac{\sum_{i=1}^{n}(CT - PT_i)}{n}, \tag{9}$$

where CT is the cycle time, PT_i is the process time of respective workstation, and n is the number of workstations.

Minimizing the workload deviation can maximize the workload smoothness.

Number of tool changes: It is determined from the frequency of tool changes in successive operations.

Number of tools and machines: It is the total number of tools and machines used in respective workstations.

Tasks	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
a	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
b	999	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
c	999	999	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
d	3	3	3	0	0	0	0	0	2	2	2	0	0	0	0	4	4	4	0	0
e	3	3	3	999	0	0	0	0	2	2	2	0	0	0	0	4	4	4	0	0
f	3	3	3	999	999	0	0	0	2	2	2	0	0	0	0	4	4	4	0	0
g	999	999	999	999	999	999	0	0	3	3	3	2	2	2	0	2	2	2	0	0
h	999	999	999	999	999	999	999	0	3	3	3	2	2	2	0	2	2	2	0	0
i	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0
j	2	2	2	2	2	2	2	2	999	0	0	0	0	0	0	0	0	0	0	0
k	2	2	2	3	3	3	3	3	999	999	0	5	5	0	0	1	1	4	0	0
l	9	9	9	9	9	9	9	9	2	2	2	0	0	0	0	1	1	4	0	0
m	9	9	9	9	9	9	9	9	2	2	2	999	0	0	0	1	1	4	0	0
n	9	9	9	9	9	9	9	9	999	999	999	999	999	0	0	1	1	4	0	0
o	999	999	999	999	999	999	999	999	999	999	999	999	999	999	0	8	8	8	0	0
p	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	0	0	4	0	0
q	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	999	0	4	0	0
r	2	2	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0
s	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	0	0
t	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	0

Fig. 6. The penalty index matrix of the given problem.

Complexity: It is determined from the penalty index, shown below

$$TP = \sum X_{ij} P_{ij}, \tag{10}$$

where $X_{ij} = 1$, if the assembly sequence i to j is selected, and is zero otherwise. P_{ij} is the penalty index of the assembly sequence i to j. Fig. 6 shows the penalty indices of assembly relations in the example problem.

4.2. Experiments and results

The experiments were performed in three phases. Phase 1 used the pure GA method to generate feasible solutions. Phase 2 employed four conventional heuristic methods to compare the results generated by the proposed method. Phase 3 insert the solutions derived from heuristic methods into the initial pools of Phase 1 and retested them using the proposed method. All experiments were run on a PC with a Pentium III processor at 600 MHz.

Phase 1: The experiments were run 4800 times with various combinations of GA parameters, as shown in Table 3. Table 4 summarizes the experimental results, which show that a larger pool can improve the quality of the solution. However, increasing the pool size significantly increases computation times. The results also show that the GA approach can yield multiple feasible solutions (for example, matching the criteria of 20-1.2-4-14-32, 21-1.2-6-11-2, 21-2.2-4-10-0, etc.). Hereafter, the symbol, "x-x-x-x-x" represents the criteria concerning cycle time, workload deviation, number of tool changes, number of tools and machines used, and the complexity of the assigned assembly sequences, in order.

Phase 2: The four popular heuristic rules used for comparison are as follows:

Table 3
GA parameters of the experiments

Parameter	Value
Reproduction gap	0.1, 0.2, 0.3, 0.4, 0.5
Crossover rate	0.6, 0.5, 0.4, 0.3
Mutation rate	0.05, 0.1, 0.15, 0.2
Pool size	30, 60, 100
$w1$-$w2$-$w3$-$w4$-$w5$ (weighting value)	1-0-0-0-0, 1-1-1-1-1, 3-1-1-1-1, 3-1-1-1-3, 3-1-3-1-3

$w1$: weight of cycle time, $w2$: weight of workload smoothness, $w3$: weight of number of tooling change, $w4$: weight of number tooling and machine, $w5$: weight of assembly complexity.

H1: Maximum ranked positional weight (Helgeson and Birnie, 1961). Assign the tasks based on the positional weights of $W_{H1} = \text{Max}\,(t_i + \sum_{j \in S_i} t_j)$, where t is the process time of the assigned task, S_i is the set of tasks which must succeed Task i.

H2: Maximum total number of follower tasks (Brian and Patterson, 1984). Assign the tasks based on the positional weights of $W_{H2} = \text{Max}\,(Ns_i)$, where S_i is the set of tasks which must succeed Task i.

H3: Minimum total number of predecessor tasks (Elsayed and Boucher, 1994). Assign the tasks based on the positional weights of $W_{H3} = \text{Min}\,(N_{P_j})$, where P_j is the set of tasks which must precede Task j.

H4: Maximum task time (Kilbridge and Wester, 1961). Assign the tasks based on the positional weights of $W_{H4} = \text{Max}\,(t_i)$, where t is the process time of the assigned task.

Tables 5 and 6 list the positional weights of tasks of the four heuristic methods and the generated solutions, respectively. A comparison with the results summarized

Table 4
The solutions of pure GA approach

Weights (w_i)	Criteria	Pool size		
		30	60	100
1-0-0-0-0	Avg. value	27-5.8-9-14-23[a]	26-4.9-9-14-24	25-3-10-15-18
	Min. value	23-2.2-8-14-27	21-0.7-7-13-25	20-1.2-4-14-32
	Times (s)	78.7	247.7	646.9
1-1-1-1-1	Avg. value	27-5.6-10-15-11	26-5.2-9-14-9	25-3.6-9-14-7
	Min. value	22-0.7-9-15-12	22-2.2-6-12-2	21-1.2-6-11-2
	Times (s)	148.8	406.3	1093.6
3-1-1-1-1	*Avg. value*	*27-5.4-9-14-15*	*25-4.4-9-14-11*	*24-3.2-9-14-9*
	Min. value	*22-1.7-7-13-8*	*22-2.2-6-12-4*	*21-2.2-4-10-0*
	Times (ls)	*139*	*463.4*	*1067.3*
3-1-1-1-3	Avg. value	28-5.8-11-16-9	27-5.4-10-15-7	25-3.7-9-15-7
	Min. value	22-1.2-8-13-7	22-2.2-6-12-4	21-1.2-6-11-2
	Times (s)	133.5	463.6	972.8
3-1-3-1-3	Avg. value	28-6-10-15-10	27-5.7-9-14-8	25-3.9-8-14-6
	Min. value	22-1.2-8-13-7	22-2.7-5-10-9	21-1.2-6-11-2
	Times (s)	144	432.2	1116.9

[a] 27-5.8-9-14-23 represents the objectives of cycle time, workload deviation, number of tool change, number of tooling, and complexity of assembly relations, respectively.

Table 5
Positional weights of tasks for the four heuristic methods

Method	Positional weight of tasks (W_{Hi})																			
	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
H1	38	35	32	41	33	30	26	19	33	30	22	28	25	19	16	30	21	14	11	7
H2	7	6	5	7	6	5	4	3	6	5	4	5	4	3	2	3	2	2	1	0
H3	0	1	2	0	1	2	6	7	0	1	2	0	1	5	14	0	1	0	18	19
H4	3	3	6	8	3	4	7	3	3	8	3	3	6	3	5	9	10	3	4	7

Table 6
The solutions of the four heuristic methods

Method	Assigning sequences	Criteria	Running times (s)
H1	d,a,b,e| i,c,f| j,p| l,g,m| k,q,h,n| o,r,s,t	25-2.7-11-14-30	0.9775
H2	a,d,b,e| i,c,f,l| j,g| k,m,h,n,r| p,q| o,s,t	23-3.2-6-10-18	1.0925
H3	a,d,i,l| p,r,b| e,m,j| q,c| f,k,n,g| h,o,s,t	23-2.7-7-12-24	1.145
H4	p,d| q,a,b| c,e,f,i| j,g,h| k,l,m,n| o,r,s,t	25-3.7-9-14-23	1.21

in Table 6 reveals that the GA method yielded solutions (Table 4) superior to the best non-GA heuristic techniques. However, the execution time for the GA method is much greater than non-GA methods because GA searches for global optimal solutions and thus requires more iterations.

Phase 3: The experiments were also run 4800 times with various combinations of GA parameters, as shown in Table 3. Table 7 lists the experimental results, which show that the proposed method can significantly enhance the quality of the solution and can reduce computation times. The experimental results also show that a small pool can generate a good solution, like a larger one (for example, with a size of 30). Although the small pool size cannot yield a minimal solution (one of which is "20-1.7-3-9-14"), it significantly reduces the computational overhead.

The feasible solutions were extracted and decoded from the final evolving pool, according to the minimal total fitness. Occasionally, these solutions must be

Table 7
The solutions of the hybrid GA approach

Weighting (w_i)	Criteria	Pool size		
		30 H	60 H	100 H
1-0-0-0-0	Avg. value	21-2.2-4-10-36	21-2.2-4-10-36	20-1.2-4-10-36
	Min. value	21-2.2-4-10-36	21-2.2-4-10-36	20-1.7-3-9-24
	Times (s)	72.1	166.9	331.4
1-1-1-1-1	Avg. value	22-1.7-8-14-15	22-2.2-7-13-13	22-2.3-7-13-10
	Min. value	22-3.2-4-10-9	22-1.2-8-14-9	21-2.2-4-10-16
	Times (s)	103.7	290.8	587.2
3-1-1-1-1	*Avg. value*	*22-1.3-8-14-17*	*22-1.3-8-14-16*	*22-1.3-8-14-14*
	Min. value	*21-2.2-4-10-16*	*21-2.2-4-10-25*	*20-1.7-3-9-14*
	Times (s)	*80.4*	*207.8*	*570.9*
3-1-1-1-3	Avg. value	22-1.7-8-14-14	22-2.5-8-14-12	22-2.7-8-13-8
	Min. value	22-2.7-5-11-9	21-2.7-7-13-9	21-1.7-5-11-11
	Times (s)	105.7	317.5	540.5
3-1-3-1-3	Avg. value	23-1.9-8-14-14	23-2.7-7-13-11	23-3.1-7-13-9
	Min. value	22-1.2-8-14-9	21-0.7-7-13-9	21-1.7-5-10-10
	Times (s)	107.9	349.5	613.5

filtered by an experienced planner for determining which solutions absolutely dominate the others.

4.3. Cross analysis by respective GA parameters and weighting value

The results in Tables 4 and 7 reveal that the weights, "3-1-1-1-1", generated the highest quality solution. Generally, the cycle time is a key factor in enhancing the performance of a production system. A large w_1 can reinforce the minimizing effect of cycle time. The cycle time and workload deviation can be minimized simultaneously. In so doing, the w_2 can be set to a small value. In this paper, tool changes are assumed to take extra time, slightly increasing the cycle time; the weighting value can be set low. The number of tools and machines is a key factor that determines capital investment, and must be considered along with other factors. Finally, the complexity of the assembly relationships may affect the working speed and the quality of the components: a higher complexity may lower the working speed and increase the rework rate. In practice, a planner can choose one suitable set of weighting values, according to demand at the plant. The more important objective must be the more heavily weighted. The experimental results revealed that the weights, "3-1-1-1-1" seem to be a good combination.

The experimental results were forwardly analyzed by cross analysis with different gaps, crossover rates, and mutation rates, to verify the effects of using various GA parameters with weights, "3-1-1-1-1" in Phase 3. The generated results were not sensitive to various GA parameters. The comparison is omitted to save space.

4.4. Advanced experiments

Several experiments, involving 20 tasks-6 workstations, 20 tasks-4 workstations, 30 tasks-6 workstations, and 50 tasks-6 workstations, were advance-tested to verify the robustness of the proposed method. The weights, "3-1-1-1-1" were used in applying the latter two methods. The fitness values and running times were used to compare the improved effects. Table 8 summarizes the experimental results. The proposed approach significantly outperforms the traditional GA method, even using a small population. Fig. 7 shows that the comparison of converged iterations of two GAs, based on the experiment of 20 Tasks-6 workstations. The results also indicated that the proposed method could yield feasible solutions within an acceptable computation time.

4.5. Discussion

An assembly line modeling problem generalizes a wide set of decision problems at different levels of aggregation, and with diverse business cultures. Employing different decision variables may yield many diverse solutions. Clearly, minimizing the cycle time corresponds to maximizing the production rate of the plant, whose objective is often the system objective. However, in practice, the conditions applied to yield a solution are seldom verified, such as the complexity of

Table 8
The results of advanced experiments

Experiments	Heuristic method	Pure GA	Hybrid GA	
20 Tasks	Avg. fitness	119.6	107.2	103.3
6 Workstations	Min. fitness	106.2	79.2	76.6
	Times (s)	1.1063	368.0	201
20 Tasks	Avg. fitness	156.6	131	128.1
4 Workstations	Min. fitness	140	112	109
	Times (s)	0.9376	208.9	134
30 Tasks	Avg. fitness	213.2	175.4	167.3
6 Workstations	Min. fitness	196	153.1	148
	Times (s)	2.095	408.7	274.5
50 Tasks	Avg. fitness	327.6	292.4	276.8
6 Workstations	Min. fitness	304.9	258.3	249.7
	Times (s)	3.435	592.8	366.6

The two GA approaches are tested by ORDER1 crossover method, based on the weights of '3-1-1-1-1' and pool size 100.

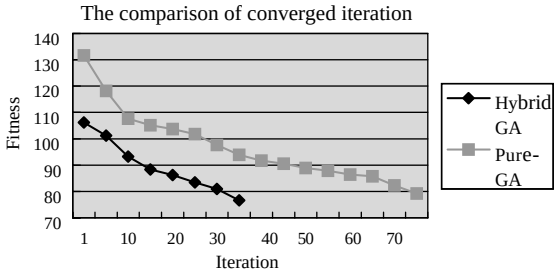

Fig. 7. The comparison of converged iterations of two GAs. (crossover method: ORDER1; pool size: 100; weightings: 3-1-1-1-1).

Table 9
Four Pareto optimal solutions of the given example

Solution	Criteria	Assigning sequences
1	20-1.7-3-9-4	i, j, r, n| p, q| o, s, t| l, k, b, m| a, d, e, f| c, g, h
2	20-2.2-2-8-7	a, d, e, b| f, l, i, j| g, h, r, n| c, m, k| p, q| o, s, t
3	21-2.2-4-10-0	i, j, r, n| p, q| k, o, t| b, m, s| a, d, e, f, l| c, g, h
4	21-3.2-2-8-2	i, j, r, n| p, q| o, s, t| b, m, k| a, d, e, f, l| c, g, h

assembly relations, the number of tools used, and others. In reality, many solutions can yield approximately the same cycle time. Consequently, other goals should be simultaneously considered.

The proposed GA approach can quickly find many feasible solutions for a given set of criteria. For example, running one round of the proposed approach yields four feasible solutions (Table 9). These solutions all meet the given criteria. However, no one solution absolutely dominates. For example, Solutions 1 and 2 minimize the cycle time to 20 but increase the complexity of the assembly relations; Solution 3 minimizes the complexity to zero; however, the cycle time is not minimal. The feasible solutions can be extracted from the final evolving pool, according to minimal total fitness. Sometimes, these extracted solutions must be filtered by an experienced planner who can justify which solutions absolutely dominate. The planner can choose one suitable solution in designing the assembly line, based on the plant conditions. The designer can better design the product by reducing the complexity of the assembly relations by reevaluating the penalty indices thereof, potentially facilitating the design for assembly.

The initial population, including the results generated by the heuristic rules, may be "a bad initial solution". However, it can reduce the time to converge to solutions from the global searching space. In such cases, the mutation mechanism of the GA can prevent the search direction from falling into pre-matured sub-optimal points. Experimental results also indicate that slightly increasing the mutation rate can guarantee convergence to a result that jumps out from the local optimization.

5. Conclusion

This study presented a hybrid GA approach to solve the assembly line planning problem. A self-tuning method was developed to enhance the effective schemata of chromosome during GA processing. The proposed method significantly increases the solution quality and reduces the computation times by combining solutions of heuristic methods. The proposed approach can more quickly solve the multi-objective problem while meeting the criteria of cycle time, number of tool changes,

number of tool and machine used, and complexity of assembly relations than conventional heuristic methods can. Moreover, the proposed approach can find many feasible solutions in one round of GA testing, helping the planner to choose a suitable alternative of assembly plan for modeling a flexible assembly system.

Acknowledgements

This paper was partially supported by the National Science Council, Taiwan, under Grant No. NSC89-2213-E009-023. Mr. Corcoran is appreciated for allowing us to use the LibGa source codes. The authors would like to thank Professor De Silva, and anonymous referees whose insights improved the content of the manuscript.

References

Brian, T.F., Patterson, J.H., 1984. An integer programming algorithm with network cuts for solving the assembly line balancing problem. Management Science 30 (1), 85–99.

Corcoran, A.L., Wainwright, R.L., 1993. LibGa: source codes and technical notes. Department of Mathematical and Computer Sciences, The University of Tulsa.

Elsayed, E.A., Boucher, T.O., 1994. Analysis and Control of Production Systems. Pretice Hall International Series in Industrial and Systems Engineering, New Jersey.

Fogel, D., 1995. Evolution Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, Piscataway, NJ.

Gen, M., Chen, R., 1997. Genetic Algorithm and Engineering Design. Wiley, New York.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York.

Hackman, S.T., Magazine, M.J., Wee, T.S., 1989. Fast, effective algorithms for assembly line balancing problems. Journal of Operational Research 37 (6), 916–924.

Helgeson, W.P., Birnie, D.P., 1961. Assembly line balancing using the ranked positional weight technique. Journal of Industrial Engineering 12 (6), 394–398.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems.. The University of Michigan Press, Ann Arbor, MI.

Kilbridge, M.D., Wester, L., 1961. A heuristic method of assembly line balancing. Journal of Industrial Engineering 12 (4), 292–298.

Kim, Y.K., Kim, Y.J., Kim, Y.H., 1996. Genetic algorithm for assembly line balancing with various objectives. Computers and Industrial Engineering 30 (3), 397–409.

Lazzerini, B., Marcelloni, F., 2000. A genetic algorithm for generating optimal assembly plans. Artificial Intelligence in Engineering 14, 319–329.

Leu, Y.Y., Matheson, L.A., Rees, L.P., 1996. Sequencing mixed-model assembly lines with genetic algorithms. Computers and Industrial Engineering 30 (4), 1027–1036.

Lucertini, M., Pacciarelli, D., Pacifici, A., 1998. Modeling an assembly line for configuration and flow management. Computer Integrated Manufacturing Systems 11 (1–2), 15–24.

Mak, K.L., Wong, Y.S., Chan, F.T.S., 1998. A genetic algorithm for facility layout. Computer Integrated Manufacturing Systems 11 (1–2), 113–127.

Onwubolu, G.C., Mutingi, M., 2001. Optimizing the multiple constrained resources product mix problem using genetic algorithms. International Journal of Production Research 39 (9), 1897–1910.

Ponnambalam, S.G., Aravindan, P., Naidu, G.M., 2000. A multi-objective genetic algorithm for solving assembly line balancing problem. Advanced Manufacturing Technology 16, 341–352.

Runarsson, T.P., Jonsson, M.T., 1999. Genetic production systems for intelligent problem solving. Journal of Intelligent Manufacturing 10, 181–186.

Song, D.P., Earl, C.F., Hicks, C., 2001. Stage due date planning for multistage assembly systems. International Journal of Production Research 39 (9), 1943–1954.

Tai, P.H., 1997. Feature-based assembly modeling for assembly sequence planning of three-dimensional products. M.Sc. Thesis. Cranfield University, UK.