Int J Adv Manuf Technol (2011) 55:1111–1122
DOI 10.1007/s00170-010-3120-6

ORIGINAL ARTICLE

A novel two-stage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks

Seyed Mahmood Kazemi · Reza Ghodsi · Masoud Rabbani · Reza Tavakkoli-Moghaddam

Received: 7 February 2010 / Accepted: 13 December 2010 / Published online: 7 January 2011

Abstract A widespread supposition on mixed-model assembly line-balancing problems assigns a task, which is shared between two or more models to a single station. Bukchin and Rabinowitch (European Journal of Operational Research, 174:492–508, 2006) relaxed the restriction for mixed-model straight-line assembly line problems and allowed tasks common to multiple models to be assigned to different stations, called task duplication. In this paper, considering the same relaxation but for mixed-model U-shaped assembly lines, a novel genetic algorithm (GA) approach for solving large-scale problems is developed. Although superiorities of U-shaped assembly lines over straight lines have been discussed in several articles, this paper makes the advantage more tangible by providing a quantitative example. This paper also presents a novel two-stage genetic algorithm which is fittingly devised for solving the new proposed model. In order to evaluate the effectiveness of the GA, one small-scale and one medium-scale problem are solved using both the proposed GA and Lingo 8.0 software, and the obtained outcomes are compared. The computational results indicate that the GA is capable of providing high-quality solutions for small- and medium-scale problems in negligible central processing unit (CPU) times. It is worth mentioning that, for large-scale problems, such as Kim and Arcus test problems, no analogous results for those obtained by our proposed GA exist. To conclude, it can be said that the proposed GA performs well and is able to solve large-scale problems within acceptable CPU times.

S. M. Kazemi · R. Ghodsi (✉) · M. Rabbani · R. Tavakkoli-Moghaddam
Industrial Engineering Department, University of Tehran, Tehran, Iran
e-mail: ghodsi@ut.ac.ir

Keywords Assembly line-balancing · Mixed-model U-line balancing · Task duplication costs · Two-stage genetic algorithm

1 Introduction

An assembly line is a productive line in which units move continuously through a number of successive stations by some kinds of a transportation system. These stations are usually aligned in a serial manner. An assembly line-balancing problem (ALBP) consists of finding a feasible line balance (i.e., assigning tasks to stations) with respect to some objectives such that the precedence constraints along with additional restrictions are satisfied. Figure 1 shows a precedence diagram with 22 tasks. A rectangle represents a task; the numbers within the rectangles are the task numbers, and an edge connecting two tasks represents a precedence relation between the tasks. Since the installation of an assembly line is a long- to mid-term planning problem and usually needs considerable capital investments, it is necessary to design and balance such a system so that it performs at the highest possible efficiency. According to variety of products, different types of assembly lines are recognized as shown in Fig. 2.

1.1 Single-model line

If only one product or several products with insignificant variations in setup times and performance task times (e.g., compact disks or drinking cans) are assembled, the assembly system can be considered like a single-model line. Bowman [4] developed two different linear-programming approaches to the assembly line-balancing problem. Gutjahr and Nemhauser [15] developed an

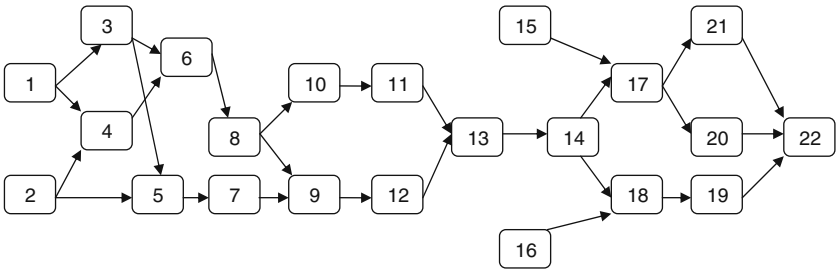

Fig. 1 An example of a precedence graph

algorithm based on finding the shortest route in a finite directed network for solving the single-model ALBP. The optimal solution can be obtained by finding any path from the starting to terminating node containing a minimal number of arcs. Patterson and Albracht [24] presented a zero–one integer programming formulation with the Fibonacci search. Betts and Mahmoud [3] developed a branch-and-bound algorithm to obtain an optimal solution for the ALBP. A novel tabu search algorithm for solving a simple assembly line-balancing type-1 problem and also a real-life version of the problem is presented by Lapierre et al. [20]. Kim et al. [19] considered five various objectives and presented genetic algorithms to solve ALBPs with five objectives, namely (1) minimizing number of workstations, (2) minimizing cycle time, (3) maximizing workload smoothness, (4) maximizing work relatedness, and (5) a multiple objective with the foregoing objectives 3 and 4. Rekiek et al. [25] focused on the line-balancing and resource planning. In Section 2, we present a simple ALBP formulation and a review of existing methods. We also investigate a number of optimization methods, such as linear programming, dynamic programming, branch-and-bound, ranked positional weight, reversed-ranked positional weight, COMSOAL, and genetic algorithms.

1.2 Mixed-model line

In a mixed-model system, setup times between models can be ignored because, in general, it is assumed that all models are variations of the same base product. The mixed-model ALBP involves the assignment of tasks of all models to the workstations in a way that some performance measures are optimized. Avoiding work overload in mixed-model production systems entails an investigation into both balancing and sequencing problems at the same time and that is why some authors have considered both planning problems simultaneously. However, because of the existing differences between planning horizons of balancing and sequencing problems (the former is a long- to mid-term planning problem whereas the latter has a short-term planning horizon), this simultaneous approach is only practical under very special conditions.

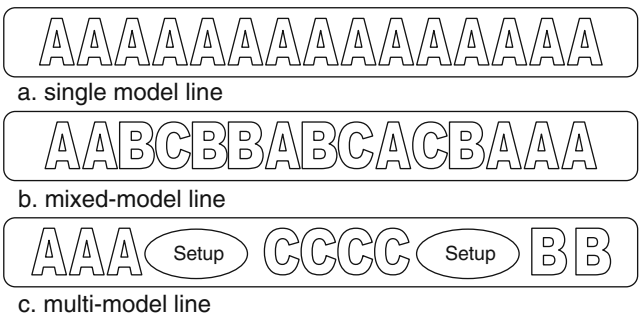

Fig. 2 Different types of assembly lines

Thomopoulos [30] focused on smoothing all the workstations, so that each station has an equal amount of work on a daily or shift basis. This paper shows how mixed-model assembly line-balancing algorithms can be adapted for obtaining smoother model assignments and how the procedure applies to assembly lines operated on a batched basis. Gokcen and Erel [12] developed a binary integer programming model for mixed-model assembly line-balancing problem. The model employs some properties that prevent the fast increase in the number of variables. Erel and Gokcen [11] used a shortest route formulation for a mixed-model ALBP. This formulation was based on the algorithm proposed by Gutjahr and Nemhauser [15]. Bukchin et al. [6] concentrated on mixed-model assembly line design in a make-to-order (MTO) environment and presented a mathematical formulation considering the differences between that model and traditional models. A heuristic is developed to minimize the number of stations for a fixed cycle time. Noorul Haq et al. [23] used a hybrid genetic algorithm (GA) approach to mixed-model assembly line-balancing type-1 problem (i.e., minimizing a number of workstations). In this paper, two different models of a base single product are considered, and the precedence relations of the models use a combined precedence diagram. In the proposed GA, a modified ranked positional weight (MRPW) method is used to generate the initial solutions for GA.

An ordinary assumption on a mixed-model ALBP is to assign a task that is common to multiple models to a single station. Bukchin and Rabinowitch [7] relaxed the restriction and allowed the assignment of tasks that are common to different models to different stations. They called the

situation as task duplication. They considered two cost elements in the objective function, namely (1) the station cost that is the cost proportional to the number of stations in the line and (2) the task cost that captures the additional cost of duplication.

1.3 Multi-model line

In multi-model production systems, the degree of similarities between products is not high enough to allow setup times to be ignored. In such kind of assembly lines, the analysis of setup times and costs is to be considered, and a short-term lot-sizing problem arises which groups models to batches and determines their assembly sequence [5].

Recently, as a consequence of implementing just-in-time (JIT) production principles, many factories are switching their assembly lines from traditional straight lines to U-shaped production lines. In the U-line balancing problem, tasks can be assigned to stations either after all its predecessors or all of its successors have been assigned previously. Miltenburg and Wijngaard [22] introduced and modeled the U-shaped ALBP and developed a dynamic programming procedure for solving the problem. They also addressed some advantages of U-lines by comparison with traditional straight lines, such as (1) improvement of visibility and communications, (2) multi-skilled operators, (3) simplicity of rebalancing the line, and (4) reduction in number of stations. Ajenblit and Wainwright [1] applied an order-based (GA) to the U-line balancing type-1 problem, in which a chromosome of length n is a permutation of 1,2,...,n and each chromosome of length n shows a specific task-ordering of n tasks. Scholl and Klein [27] proposed the branch-and-bound procedure ULINO for solving different versions of the U-line ALBP. Gokcen et al. [13] presented a shortest route formulation of a simple U-type ALBP. The node generation process used in this paper is similar to the node generation process developed by Gutjahr and Nemhauser [15] for the traditional single-model ALBP. Gokcen and Agpak [14] developed a goal programming model for the simple U-line balancing problem. They claimed that their approach was the first multi-criteria decision making (MCDM) approach to the U-line version.

Chiang and Urban [10] considered the stochastic U-line balancing problem and presented a hybrid heuristic composed of an initial feasible solution module and a solution improvement module. Baykasoglu and Özbakir [2] assumed that each task time has a normal distribution with mean μ and standard deviation σ. To balance the stochastic U-line, they developed a new algorithm integrating the COMSOAL method, ten different task assignment heuristics, and a GA. The algorithm is able to quickly search effective solutions for U-type ALBPs. In their proposed GA, the genes of chromosomes are integers. Each integer represents a task assignment rule. Sabuncuoglu et al. [26] proposed the ant colony optimization (ACO) algorithm to solve a deterministic single-model U-type ALBP. Hwang et al. [17] developed a multi-objective genetic algorithm using the priority-based coding method for solving the U-shaped ALBP.

Sparling and Miltenburg [28] introduced and modeled a mixed-model U-line balancing problem. Kara and Tekin [18] presented a mixed-integer programming formulation for optimal balancing of mixed-model U-shaped assembly lines. Their proposed approach minimizes the number of required stations for a given model sequence. Hwang and Katayama [16] proposed a multi-decision genetic approach for workload balancing of mixed-model U-type lines. Tasan and Tunali [29] presented a review of the current applications of genetic algorithms in ALBPs.

In this paper, the model introduced by Bukchin and Rabinowitch [7] is modified to be used in a mixed-model U-shaped assembly line system, and a novel two-stage GA , which is fittingly devised for solving the obtained new model, is developed. It should be pointed out that the modified model is a mixed-model U-line balancing type-1 problem. Furthermore, the task assignment rules proposed by Baykasoglu and Özbakir [2] are employed in our proposed GA. It is worth noting that applying these rules increases the flexibility of the chromosomes and eases the crossover and mutation operations as well as eliminates repairing procedures. Employing two-stage techniques or a combination of several methods for solving complicated problems is frequent in the literature. Liang et al. [21], for instance, proposed a novel two-stage noise removal algorithm to deal with impulse noise. Cheng-Wu Chen [8] presented an effective neural network-based approach to be used in nonlinear structural systems. The proposed approach is a combination of H^{∞} control performance with Takagi–Sugeno fuzzy control. Chen et al. [9] also combined a Takagi–Sugeno fuzzy model approach with a parallel distributed compensation (PDC) scheme to cope with their work under study.

The rest of the paper is organized as follows. In section 2, the model is described and the problem formulation is presented. In section 3, the proposed GA is explained. In section 4, an illustrative example is presented and some numerical experiments based on the standard benchmark data sets with the computational results are given. Finally, in section 5, conclusions and a review of the previous sections are presented.

2 Model description

The model presented in this paper is a modification of the model introduced by Bukchin and Rabinowitch [7], in

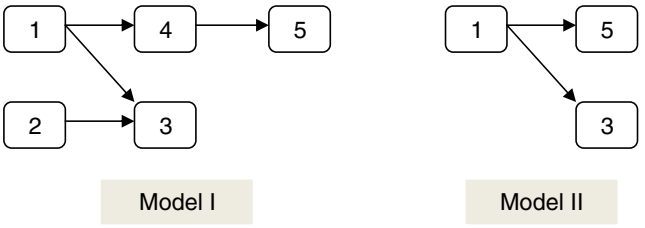

Fig. 3 Precedence diagrams taken from Bukchin and Rabinowitch [7]

which the assumptions of the model and further explanations were given. The difference between their model and our presented model stems from the existing difference between U-line and traditional straight-line modeling. Following, we present our mixed-model U-line balancing (MMULB) model.

2.1 Integer programming formulation for the MMULBP

Parameters:

- n Total number of different assembly tasks
- m Number of models to be assembled on the line
- t_{ij} Processing time of task i performed on model j
- IP_{ij} Set of immediate predecessors of task i in model j
- IS_{ij} Set of immediate successors of task i in model j
- c_j Required cycle time for model j
- SC Station or fixed cost associated with each station
- TC_i Task cost associated with each station to which task i is assigned

Decision variables:

- z Number of stations to be used in the assembly line

$$x_{ijk} = \begin{cases} 1 \text{ if task } i \text{ of model } j \text{ is assigned to station } k; \\ 0 \text{ otherwise.} \end{cases}$$

$$\tau_{ik} = \begin{cases} 1 \text{ if task } i \text{ of any of the models is assingned to station } k; \\ 0 \text{ otherwise.} \end{cases}$$

$$y_{ij} = \begin{cases} 1 \text{ if immediate successors of task i in model j have been previously assigned;} \\ 0 \text{ if immediate predecessors of task i in model j have been previously assigned.} \end{cases}$$

Table 1 Performance times and task costs for the example

Task	Model I	Model II	TC_i
1	6	2	11
2	4	–	5
3	6	4	8
4	4	–	3
5	2	3	2

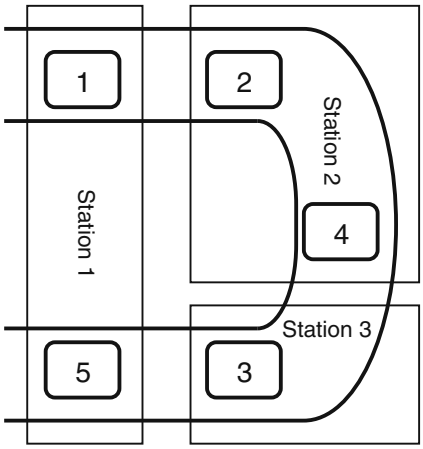

Fig. 4 Optimal balance for the mixed-model U-line production system

where, the number of tasks n is assumed as an upper bound on the number of stations.

The problem formulation is as follows.

$$\min\left\{SC \times z + \sum_{i=1}^{n} TC_i \sum_{k=1}^{n} \tau_{ik}\right\} \tag{1}$$

Subject to:

$$\sum_{k=1}^{n} x_{ijk} = 1 \quad \forall i,j \tag{2}$$

$$\sum_{k=1}^{n} k \times \left(x_{gjk} - x_{hjk}\right) \le M y_{hj} \quad \forall j,g,h \text{ and } g \in IP_{hj} \tag{3}$$

$$\sum_{k=1}^{n} k \times \left(x_{ljk} - x_{hjk}\right) \le M\left(1 - y_{hj}\right) \quad \forall j,h,l \text{ and } l \in IS_{hj} \tag{4}$$

$$\sum_{i=1}^{n} x_{ijk} \times t_{ij} \le c_j \quad \forall j,k \tag{5}$$

$$z \ge \sum_{k=1}^{n} k \times x_{ijk} \quad \forall i,j \tag{6}$$

$$\tau_{ik} \ge \frac{1}{m} \times \sum_{j=1}^{m} x_{ijk} \quad \forall i,k \tag{7}$$

$$x_{ijk}, \tau_{ik}, y_{ij} \in \{0,1\} \quad z \ge 0 \text{ and integer} \tag{8}$$

The objective function given in Eq. 1 minimizes the total costs associated with the number of stations and task duplication costs. In view of the fact that tasks are inseparable units in which each task of each model must be assigned to exactly one station. Constraint (2) guarantees

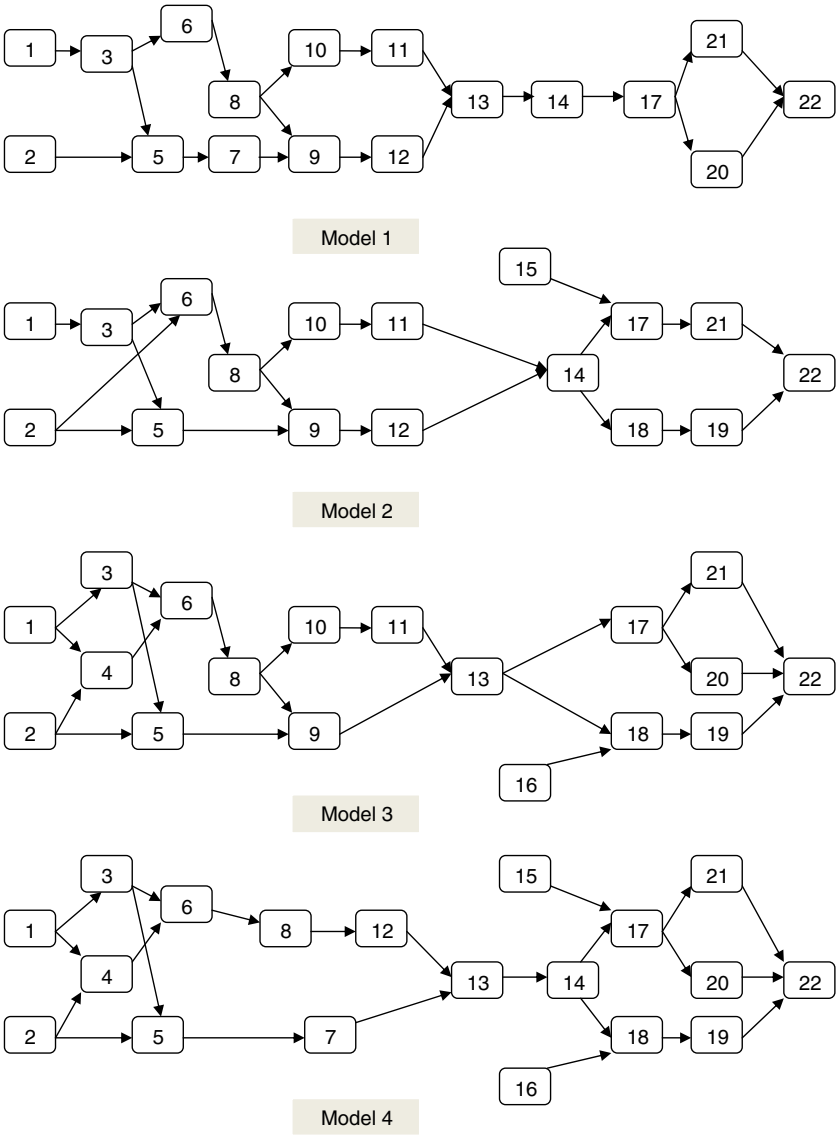

Fig. 5 Precedence diagrams for models 1 to 4

the restriction. Constraints (3) and (4) satisfy precedence relationships between tasks. Constraint (3) ensures that task h of model j can be assigned to station k if its immediate predecessor tasks have been already assigned to that station or to one of the previous stations. Constraint (4), which originates as a result of being U-shaped line, denotes task h of model j can be assigned to station k only if all of its immediate successors previously have been assigned to that station or to an upstream station. These two constraints work together as an either-or constraint and a binary variable, y_{hj}, is added to produce the practical either-or constraint set. In other words, task h of model j is assigned to station k if either all of its immediate predecessors or its immediate successors are assigned before. Constraint (5) is the cycle time constraint and assures that the total performance time of each model at each station cannot exceed the model's cycle time. In order to handle complexities of U-shaped systems, in our proposed model, it is assumed that *the model that arrives to each crossover workstation in back of the line is just like the one arrives to that workstation in front of the line.* This assumption is employed in the cycle time constraint. Since in mixed-model production systems, different models are variations of a same base product, this assumption seems to be justifiable. The total number of workstations is equal to the greatest workstation index number of all tasks and models and is shown in Constraint (6). Constraint (7) checks whether task i of any model is assigned to workstation k. Constraint (8) denotes that x_{ijk}, τ_{ik}, and y_{ij} are binary variables.

2.2 Illustrative example

In this section, a simple example of a mixed-model system presented by Bukchin and Rabinowitch [7] is solved by our

Table 2 List of task assignment rules employed in the genetic algorithm

Rule number	Task assignment rules	
1	Shortest processing time	SPT
2	Longest processing time	LPT
3	Minimum total number of successor tasks	MiTNST
4	Maximum total number of successor tasks	MaTNST
5	Maximum total time of successor tasks	MaTTST
6	Minimum total time of successor tasks	MiTTST
7	Maximum total number of predecessor tasks	MaTNPT
8	Minimum total number of predecessor tasks	MiTNPT
9	Maximum total time of predecessor tasks	MaTTPT
10	Minimum total time of predecessor tasks	MiTTPT

proposed model using Lingo 8.0 software to show how changing traditional straight lines to U-shaped assembly lines can results in better solutions and more flexibility. Figure 3 depicts two models in a form of precedence diagrams. Table 1 show the performance times and task costs for this given example. In addition to the given data, we also consider the parameter sets for SC=10, c_1=8, and c_2=5.

We solved this example by our integer programming for the MMULBP using Lingo 8.0 software and the optimal objective function value (OFV) found is equal to 59. It shows that switching from traditional straight lines to the U-shaped lines can result in reducing the total costs because the optimal OFV found by Bukchin and Rabinowitch [7] is equal to 61. The optimal solution for our model is depicted in Fig. 4.

In this figure, the total number of workstations in the solution is equal to 3; so the first part of the objective function is equal to $SC \times z = 10 \times 3 = 30$. The rounded rectangles in the figure can be considered as machines that are required to perform each task. For example, a rounded rectangle containing number 2 stands for the machine that is responsible for performing task 2. This point is noted to have a better sense about task duplication costs. In addition, each machine can perform only the related task and cannot perform other tasks. In a duplication situation, for example, machine number 5 can be assigned to more than one station, as occurred in the optimal solution reported by Bukchin and Rabinowitch [7]. While no task duplication occurred in our optimal solution, the second part of the objective function is equal to: $\sum_{i=1}^{5} TC_i \sum_{k=1}^{3} \tau_{ik} = 29$. So, the minimum value of the objective function is equal to 30+29=59.

3 The proposed algorithm

In this section, the proposed genetic algorithm (GA) is introduced. This approach is a novel GA because it is fittingly devised for solving our new proposed model. The GA includes two parts seeing that the objective function includes two parts: the first part, $SC \times z$, minimizes costs associated with the number of stations and the second part, $\sum_{i=1}^{n} TC_i \sum_{k=1}^{n} \tau_{ik}$, minimizes task duplication costs. Accordingly, the first part of the GA, which is referred to hereafter as *the first GA*, explores for solutions with minimum number of stations and the second part of the GA, which is called hereafter *the main GA,* tries to find solutions with minimum task duplication costs among those solutions found by the first part.

The first GA task is to provide initial population for the main GA while the main GA starts with this initial population and solves the proposed MMULB problem. In the first stage, each of individual models is solved separately by the first GA as an independent problem for solving a single-model U-line balancing type-1 problem. It should be noted that the initial populations of the first GA are randomly generated. In the first GA, total number of iterations for each model is set to 100 iterations with the same objective function value. At the end of solving each model, the last population is kept. Finally, m populations achieved for m models are combined together, and the initial population for the main GA is created. For example, consider the models presented in Fig. 5. The chromosome length used in the first GA is equal to 22 for the reason that the total number of different assembly tasks is equal to 22. When four populations are achieved, these populations are combined together in a way that the produced united population is a 100×88 matrix, in which 100 is the number of solutions in each population and 88 is the length of each chromosome. The first 22 genes of these 88 genes are corresponding to the first chromosome of the first model; the next 22 genes from gene 23 to gene 44 are related to the first chromosome of the second model, and so on. In fact, four 100×22 matrices are augmented in order to create a 100×88 matrix.

Since the first part of the objective function minimizes the total number of stations and the objective function for the SULB type-1 problem also minimizes the number of workstations, this augmentation, instead of using a random initial solution for the main GA, increases the convergence rate of the GA notably. Since the number of 100 iterations as stopping criterion for each model is not a large number (e.g., diversity of chromosomes in a population is not reduced drastically) and chromosomes with the minimum

10	3	7	5	9	8	5	1	9	5	7	8	10	8	2	5	10	10	5	9	1	4

Fig. 6 A chromosome with 22 genes used in the first GA

1	22	19	2	20	15	3	16	21	4	17	5	7	6	18	8	10	11	9	14	12	13

Fig. 7 Task-ordering selection produced by the deduction procedure

number of stations (i.e., minimum number of stations found during 100 iterations) are not necessarily identical (e.g., different chromosomes can result in the same value for number of stations), the variety needed to form an initial population for the main GA is slightly stained.

3.1 Solution representation

As mentioned before, we use the task assignment rules proposed by Baykasoglu and Özbakir [2] in our proposed GA because applying these rules increases the flexibility of the chromosomes and eases the crossover and mutation operations and eliminates repairing procedures. A chromosome length is equal to the number of different tasks, and the value of each gene denotes the task assignment rule. A list of task assignment rules used in Baykasoglu and Özbakir [2] is shown in Table 2.

Suppose that we want to solve a single-model U-line balancing type-1 problem based on a precedence diagram represented in Fig. 1, by using the first GA where the cycle time is equal to 19. A chromosome constituted by 22 genes is created, and 22 random integers between 1 and 10 are incorporated into genes of the chromosome. Figure 6 shows an example of the chromosome for the given problem.

Applying these rules in chromosomes compelled us to insert a deduction procedure for deducing the related balance from a chromosome. We explain the procedure by the chromosome represented in Fig. 6.

Definition Assignable tasks are those whose predecessor or successor tasks have been assigned already.

Table 3 Balance of the assembly line derived from the chromosome shown in Fig. 6 using the deduction procedure

Stations	1	1,19,22 station time=13
	2	2 station time=11
	3	3,15,20 station time=16
	4	16,21 station time=10
	5	4 station time=11
	6	5,17 station time=15
	7	6,7 station time=17
	8	18 station time=12
	9	8,10 station time=14
	10	9,11 station time=14
	11	12,13,14 station time=17

Deduction procedure:

Step 1: Determine set of assignable tasks and incorporate them into the *Ablet* set.
Set i=1, k=1; n=*length of the chromosome*;

Step 2: Go to the ith gene and read the value. Select a task from the *Ablet* set using the task assignment rule equal to the value. If more than one task has the same situation to be selected considering the task assignment rule, the task with the minimum number is chosen to be assigned.
Insert the selected task in a.
If the idle time of station k is equal to or greater than performance time of task a,
Assign the selected task to station k;
Else

$k = k + 1$;
Assign the selected task to station k;

Step 3: Update *ablet* set.
$i = i + 1$;
If $i \leq n$

Go to step 2;
Else

Go to Step 4;

Step 4: Terminate the procedure and display the assembly line configuration.

Using the above procedure for the chromosome shown in Fig. 6, the task-ordering selection depicted in Fig. 7 is deduced. The assembly line balance for the chromosome shown in Fig. 6 is represented in Table 3.

3.2 Crossover operator

We use a two-point crossover method as shown in Fig. 8 in order to generate offspring from parents, in which these two points are randomly generated. Two points are assumed to be 12 and 18.

3.3 Mutation operator

Two numbers between 1 and the chromosome length are arbitrarily generated. Let the smallest and largest numbers be set to a and b, respectively. Then, $b-a+1$ integers

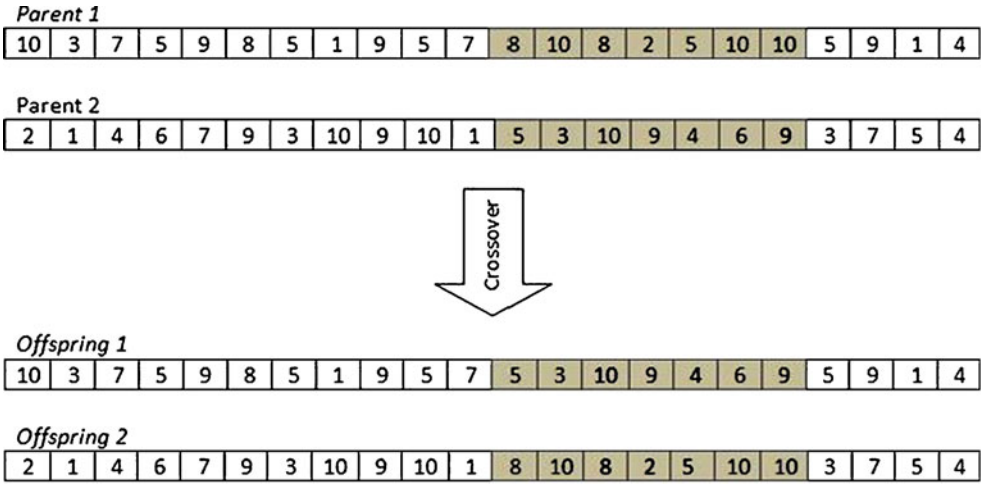

Fig. 8 Two-point crossover method

between 1 and 10 are produced randomly and inserted in the chromosome.

3.4 Fitness function and selection

The evaluation function used in the first GA is total number of stations, and the evaluation function used in the main GA is the objective function computed by:

$$\text{Eval}(\text{solution}(j)) = SC \times z + \sum_{i=1}^{n} TC_i \sum_{k=1}^{n} \tau_{ik}$$

$$1 \le j \le \text{population size}$$

In selection, the individuals producing offspring are chosen. Two well-known selection methods are examined in the proposed GA: "Tournament selection" method and "Roulette wheel selection" method. In tournament selection, a number *Tour* of individuals is chosen randomly from the population, and the best individual from this group is selected as parent. This process is repeated as often as individuals to choose. In roulette-wheel selection a probability is assigned to each chromosome proportional to its fitness value. The higher the probability, the more chance to be selected. In tournament selection, the fitness function is calculated by the evaluation function defined above. But, in roulette-wheel selection method, we are compelled to determine a fitness function such that better solutions with lower evaluation function values have higher fitness function values. So, we define *maxfit* that is equal to the maximum amount of the evaluation function in each generation and use this variable for determining the fitness function.

$$\text{Fitness}(\text{solution}(j)) = maxfit - 0.9 \times \text{Eval}(\text{solution}(j))$$

$$1 \le j \le \text{population size}$$

The probability of selecting a solution (j) is computed by:

$$\text{Prob}(\text{solution}(j)) = \text{Fitness}(\text{solution}(j)) / \sum_{j=1}^{\text{popsize}} \text{Fitness}(\text{solution}(j))$$

Table 4 Performance times and task costs for the numerical problem

Task number	Task time	TC_i	Task number	Task time	TC_i
1	4	11	12	2	2
2	11	10	13	7	3
3	1	9	14	8	4
4	11	8	15	4	5
5	6	7	16	5	6
6	10	6	17	9	7
7	7	5	18	12	8
8	11	4	19	7	9
9	6	3	20	11	8
10	3	2	21	5	7
11	8	1	22	2	6

Table 5 Solution found by the proposed GA for the numerical problem

		Models			
		1	2	3	4
Stations	1	1,2	1,2	1,2,22	1,2,22
	2	3,20,22	3,15,19,22	3,19,20	3,19,20
	3	6,21	6,21	4,21	4,21
	4	8	18	16,18	16,18
	5	5,17	5,17	5,17	5,15,17
	6	13,14	11,14	9,13	13,14
	7	9,11,12	9,12	6,11	6,12
	8	7,10	8,10	8,10	7,8

Table 6 Task duplication costs

Task number	Number of duplications	Duplication cost	Task number	Number of duplications	Duplication cost
1	1	11	12	1	2
2	1	10	13	1	3
3	1	9	14	1	4
4	1	8	15	2	10
5	1	7	16	1	6
6	2	12	17	1	7
7	1	5	18	1	8
8	2	8	19	1	9
9	2	6	20	1	8
10	1	2	21	1	7
11	2	2	22	2	12
					Sum=156

4 Computational results

In this section, a numerical problem based on the models shown in Fig. 5 is developed and solved by the proposed GA, and the related calculations are explained. Afterwards, a number of well-known test problems are solved by the novel GA, and computational results are given. The proposed algorithm is programmed in the MATLAB 7.2.0.232 (R2006a) software.

4.1 Numerical problem

Consider Fig. 5. The combined diagram of these four models is shown in Fig. 1. In fact, we first design the combined diagram, and then the models are produced. The tasks removed from each model are selected arbitrarily. For example, in the first model, tasks 4,15,16,18, and 19 are randomly chosen to be eliminated. The performance task times are also selected randomly from the numbers between one and 12. The related values for required parameters illustrated in Table 4. In addition to the given data, we also consider the parameter sets for SC=10, c_1=16, c_2=16, c_3=19, and c_4=19.

Table 7 Task duplication costs for the Thomopoulos data set

Task number	TC_i	Task number	TC_i
1	11	11	1
2	10	12	2
3	9	13	3
4	8	14	4
5	7	15	5
6	6	16	6
7	5	17	7
8	4	18	8
9	3	19	9
10	2		

The GA finds the solution illustrated in Table 5 after 1,080 iterations in 226 s CPU time from a PC with 2.2 GHz CPU and 2 GB RAM.

In this table, the number of stations is eight. So, the first part of the objective function is equal to $SC \times z = 80$. Now, we need to calculate task duplication costs in order to find the objective function value. By considering task 1, we can see that task 1 of all models is assigned to station 1 so the second part of the objective function for task 1 is equal to $TC_1 \times \sum_{k=1}^{8} \tau_{1k} = 11 \times 1 = 11$. Now, consider task 6; task 6 of models 1 and 2 is assigned to station 3, and consequently, τ_{63}=1. In the other side, task 6 of models 3 and 4 is assigned to station 7, and for that reason, τ_{67}=1. Accordingly, the duplication cost for task 6 is equal to $TC_6 \times (\tau_{63} + \tau_{67}) = 6 \times 2 = 12$. The above calculations are done for each of individual tasks, and the related results are shown in Table 6.

According to Table 6, total task duplication costs is equal to 156, and therefore, the objective function value is equal to 80+156=236. It is worth noting that, because this solution is derived from a meta-heuristic algorithm, there is no guarantee for the answer to be an optimal solution. Figure 9 shows the solution found by our proposed GA for the given example.

As stated before, the rounded rectangles are a symbol of machines rather than a symbol of tasks. It represents that

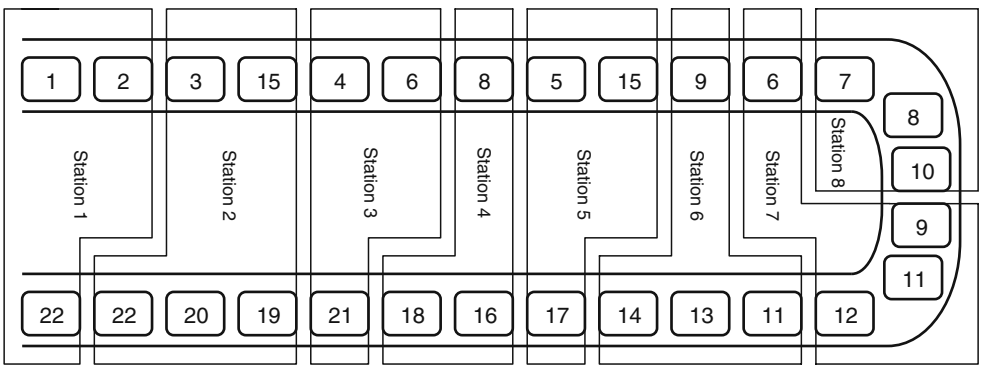

Fig. 9 Balance found by GA for the numerical problem

Table 8 Task duplication costs for the Kim data set

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

machines needed for operating tasks 6, 8, 9, 11, 15, and 22 are installed in more than one station; thus, the money is paid for operation of the related tasks is duplicated. When model 1 arrives to station 2 in front of the U-line, task 3 is done on the model, and then the model leaves the station until arrives to station 2 for the second time in back of the line, and tasks 20 and 22 are operated on the model; after that, model 1 leaves the line. Although task 15 is common to models 2 and 4, the station that performs task 15 is different for the two models. Machine number 15 installed in station 2 is planned for performing task 15 on model 2 whereas task 15 of model 4 is done in station 5. Considering the mentioned statements, Fig. 9 becomes meaningful.

4.2 Test problems

A number of well-known test problems obtained from the Website http://www.assembly-line-balancing.de/files/uploads/data-mmulbs.doc are solved using our novel GA. Since task duplication costs are not considered in the test problems, we develop arbitrary values for the missed data. Table 7 shows the new data set taken from Thomopoulos [30] with 19 tasks

Table 9 Task duplication costs for the Arcus data set

Task number	TC_i	Task number	TC_i	Task number	TC_i	Task number	TC_i	Task number	TC_i	Task number	TC_i
1	20	21	20	41	20	61	20	81	20	101	20
2	19	22	19	42	19	62	19	82	19	102	19
3	18	23	18	43	18	63	18	83	18	103	18
4	17	24	17	44	17	64	17	84	17	104	17
5	16	25	16	45	16	65	16	85	16	105	16
6	15	26	15	46	15	66	15	86	15	106	15
7	14	27	14	47	14	67	14	87	14	107	14
8	13	28	13	48	13	68	13	88	13	108	13
9	12	29	12	49	12	69	12	89	12	109	12
10	11	30	11	50	11	70	11	90	11	110	11
11	10	31	10	51	10	71	10	91	10	111	10
12	9	32	9	52	9	72	9	92	9		
13	8	33	8	53	8	73	8	93	8		
14	7	34	7	54	7	74	7	94	7		
15	6	35	6	55	6	75	6	95	6		
16	5	36	5	56	5	76	5	96	5		
17	4	37	4	57	4	77	4	97	4		
18	3	38	3	58	3	78	3	98	3		
19	2	39	2	59	2	79	2	99	2		
20	1	40	1	60	1	80	1	100	1		

Table 10 Solutions for the given test problems if roulette-wheel selection method is employed

Problem set	Objective function value	Total number of iterations	CPU time, s
Thomopoulos	176	875	139
Kim	536	1,858	1,454
Arcus problem 1	3,043	2,833	5,644
Arcus problem 2	2,256	3,265	6,760
Arcus problem 3	2,220	1,891	3,565
Arcus problem 4	2,072	2,618	5,317
Arcus problem 5	1,881	2,008	3,923

and three models considering SC=10, c_1=0.9, c_2=1, and c_3=1.5. Table 8 shows the new data set taken from Kim et al. [19] with 61 tasks and four models considering SC=10, c_1=10, c_2=9, c_3=12, and c_3=12. Table 9 shows the new data set taken from Arcus with 111 tasks and five models considering SC=25 and the following parameters.

Problem 1: c_1=6615, c_2=6615, c_3=6615, c_4=6615, c_5=6615.
Problem 2: c_1=10027, c_2=10027, c_3=10027, c_4=10027, c_5=10027.
Problem 3: c_1=10743, c_2=10743, c_3=10743, c_4=10743, c_5=10743.
Problem 4: c_1=11378, c_2=11378, c_3=11378, c_4=11378, c_5=11378.
Problem 5: c_1=17067, c_2=17067, c_3=17067, c_4=17067, c_5=17067.

The values of the first GA parameters are as follows.

Number of chromosomes in each population=100.
Percentage of the best current chromosomes copied to the next generation=10.
Crossover probability (or rate)=0.8.
Mutation probability (or rate)=0.1.
Stopping condition=100 iterations with the same OFV.
Tournament size (Tour)=4 (if tournament selection is employed).

The values of the main GA parameters are as follows.

Number of solutions in each population=100.
Percentage of the best current chromosomes copied to the next generation=10.
Crossover probability (or rate)=0.6.
Mutation probability (or rate)=0.3.
Stopping condition=500 iterations with the same OFV.
Tournament size (Tour)=4 (if tournament selection is employed).

The computational results of the mixed-model U-line balancing problem, which minimizes the task duplication costs and number of stations simultaneously, are presented in Tables 10 and 11. Table 10 shows the results if roulette-wheel selection method is employed while Table 11 shows the results if tournament selection method is applied.

Comparing results of Table 10 and Table 11, it can be concluded that employing tournament selection method in the proposed GA generally improves objective function values while makes CPU times worse.

To conclude the effectiveness of the GA, the model has been solved once for the small-scale example shown in section 2.2 and once for the medium-scale problem of Thomopoulos [30] using both Lingo 8.0 software and the proposed GA. For the former problem, the obtained results are the same, and the GA provides the global optimum solution. In the latter case, the novel GA provides a solution with an objective function value of 176 during 2–3 min. However, having interrupted the Lingo 8.0 solver, which was being run for about 49 h, provided a solution with an objective function value of 170. Although this solution was not reported as a global optimum, simple calculations indicate that it is a global optimum solution. As can be seen, for the medium-scale problem the GA presents a solution with 3.53% error from the global optimum solution in negligible CPU time. It is worth noting that for large-scale problems, such as Kim and Arcus test problems, no comparable results for those obtained via our proposed GA exist. Thus, at this time, it can be said that the proposed GA performs well and is able to solve large-scale problems within acceptable CPU times.

5 Conclusion

A widespread supposition on mixed-model assembly line-balancing problems is to allow the assignment of a task which is shared between two or more models to a single station. Bukchin and Rabinowitch [7] relaxed the limitation and allowed tasks common to multiple models to be assigned to different stations. They also proposed an integer programming formulation for mixed-model straight line-

Table 11 Solutions for the given test problems if tournament selection method is employed

Problem set	Objective function value	Total number of iterations	CPU time, s
Thomopoulos	176	1,191	197
Kim	528	2,714	2,141
Arcus problem 1	2,860	3,222	6,501
Arcus problem 2	2,463	3,920	8,417
Arcus problem 3	2,029	4,501	9,316
Arcus problem 4	2,211	4,881	10,229
Arcus problem 5	1,878	3,466	7,460

balancing problem. However, we modified their model in a way that our presented model could be applied in U-shaped assembly systems. A comparison was made between U-lines and traditional straight lines by solving a simple example with both models carried out. This comparison gave you an idea about how switching from traditional straight lines to U-lines reduces costs of a typical production system. Subsequently, in order to cope with medium- and large-scale problems of real world, a novel two-stage genetic algorithm was proposed.

In an attempt to fully explain the proposed model and the novel GA a medium-scale problem with 22 tasks and four models was developed and solved. The duplication situation was elaborated, and the related balance for the solution of the medium-scale problem on a U-line layout was illustrated. To assess the performance of the proposed GA, some standard data sets were used while overlooked data were generated randomly. Furthermore, effectiveness of the proposed GA was evaluated by solving the modified model using both the novel GA and Lingo 8.0 software for a small- and a medium-scale problem. The result of this comparison is that the GA can solve these types of problems in negligible CPU times. It is worth mentioning that, for large-scale problems, such as Kim and Arcus test problems, no analogous results for those obtained by our proposed GA exist. Thus, at this time, it can be said that the proposed GA performs well and is able to solve large-scale problems within acceptable CPU times. In addition to what was stated before, a comparison between roulette-wheel selection method and tournament selection method was also provided in this paper.

References

1. Ajenblit DA, Wainwright RL (1998) Applying genetic algorithms to the U-shaped assembly line balancing problem. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation (ICEC'98), part of WCCI, Anchorage, Alaska, May 4–9, pp. 96–101
2. Baykasoglu A, Özbakir L (2007) Stochastic U-line balancing using genetic algorithms. Int J Adv Manuf Technol 32(1–2):139–147
3. Betts J, Mahmoud KI (1989) A method for assembly line balancing. Eng Costs Prod Econ 18:55–64
4. Bowman EH (1960) Assembly-line balancing by linear programming. Oper Res 8(3):385–389
5. Boysen N, Fliedner M, Scholl A (2008) Assembly line balancing: which model to use when? Int J Prod Econ 111:509–528
6. Bukchin J, Dar-El EM, Rubinovitz J (2002) Mixed-model assembly line design in a make-to-order environment. Comput Ind Eng 41:405–421
7. Bukchin Y, Rabinowitch I (2006) A branch-and-bound based solution approach for the mixed-model assembly line balancing problem for minimizing stations and task duplication costs. Eur J Oper Res 174:492–508
8. Chen Cheng-Wu (2009) Modeling and control for nonlinear structural systems via a NN-based approach. Expert Syst Appl 36:4765–4772. doi:10.1016/j.eswa.2008.06.0622009
9. Chen C, Lin J, Lee W, Chen C (2010) Fuzzy control for an oceanic structure: a case study in time-delay TLP system. J Vib Control 16:147–160. doi:10.1177/1077546309339424
10. Chiang WC, Urban TL (2006) The stochastic U-line balancing problem: a heuristic procedure. Eur J Oper Res 175:1767–1781
11. Erel E, Gokcen H (1999) Shortest route formulation of mixed-model assembly line balancing problem. Eur J Oper Res 116:194–204
12. Gokcen H, Erel E (1998) Binary integer formulation for mixed-model assembly line balancing problem. Comput Ind Eng 34:451–461
13. Gokcen H, Agpak K, Gencer C, Kizilkaya E (2005) A shortest route formulation of simple U-type assembly line balancing problem. Appl Math Model 29:373–380
14. Gokcen H, Agpak K (2006) A goal programming approach to simple U-line balancing problem. Eur J Oper Res 171:577–585
15. Gutjahr AL, Nemhauser GL (1964) An algorithm for the line balancing problem. Manage Sci 11:308–315
16. Hwang R, Katayama H (2009) A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems. Int J Prod Res 47:3797–3822
17. Hwang R, Katayama H, Gen M (2008) U-shaped assembly line balancing problem with genetic algorithm. Int J Prod Res 46:4637–4649
18. Kara Y, Tekin M (2009) A mixed integer linear programming formulation for optimal balancing of mixed-model U-lines. Int J Prod Res 47:4201–4233
19. Kim YK, Kim YJ, Kim YH (1996) Genetic algorithms for assembly line balancing with various objectives. Comput Ind Eng 30(3):397–409
20. Lapierre SD, Ruiz A, Soriano P (2006) Balancing assembly lines with Tabu search. Eur J Oper Res 168:826–837
21. Liang S, Lu S, Chang J (2008) A novel two-stage impulse noise removal technique based on neural networks and fuzzy decision. IEEE Trans Fuzzy Syst 16(4):863–873
22. Miltenburg J, Wijngaard J (1994) The U-line line balancing problem. Manage Sci 40:1378–1388
23. Noorul Haq A, Jayaprakash J, Rengarajan K (2006) A hybrid genetic algorithm approach to mixed-model assembly line balancing. Int J Adv Manuf Technol 28(3–4):337–341
24. Patterson JH, Albracht JJ (1975) Assembly line balancing: zero-one programming with Fibonacci search. Oper Res 23:166–172
25. Rekiek B, Dolgui A, Delchambre A, Bratcu A (2002) State of art of optimization methods for assembly line design. Annu Rev Control 26:163–174
26. Sabuncuoglu I et al (2009) Ant colony optimization for the single model U-type assembly line balancing problem. Int J Prod Econ 120(2):287–300
27. Scholl A, Klein R (1999) ULINO: optimally balancing U-shaped JIT assembly lines. Int J Prod Res 37:721–736
28. Sparling D, Miltenburg J (1998) The mixed-model U-line balancing problem. Int J Prod Res 36:485–501
29. Tasan SO, Tunali S (2008) A review of the current applications of genetic algorithms in assembly line balancing. J Intell Manuf 19 (1):49–69
30. Thomopoulos NT (1970) Mixed model line balancing with smoothed station assignments. Manage Sci 16(9):593–603