Genetic and Memetic Algorithms for Sequencing a New JIT Mixed-Model Assembly Line

R. Tavakkoli-Moghaddam[i*]; Y. Gholipour-Kanani[ii] and R. Cheraghalizadeh[iii]

Received 22 Nov 2010; received in revised 03 Sep 2012; accepted 23 Sep 2012

ABSTRACT

This paper presents a new mathematical programming model for the bi-criteria mixed-model assembly line balancing problem in a just-in-time (JIT) production system. There is a set of criteria to judge sequences of the product mix in terms of the effective utilization of the system. The primary goal of this model is to minimize the setup cost and the stoppage assembly line cost, simultaneously. Because of its complexity to be optimally solved in a reasonable time, we propose and develop two evolutionary meta-heuristics based on a genetic algorithm (GA) and a memetic algorithm (MA). The proposed heuristics are evaluated by the use of random iterations, and the related results obtained confirm their efficiency and effectiveness in order to provide good solutions for medium and large-scale problems.

KEYWORDS

JIT mixed-model assembly line balancing, Setup cost, Stoppage cost, Genetic algorithm, Memetic algorithm

1. INTRODUCTION

An assembly line is a set of sequential workstations linked by a material handling system. In each workstation, a set of tasks is performed using a pre-defined assembly process, in which the following issues are defined: 1) task times, i.e. the time required to perform each task, 2) a set of precedence relationships, which determine the sequence where the tasks can be performed, and 3) a set of zoning constraints, which force or forbid the assignment of different tasks to the same workstation [1].

The assembly line can be dedicated to produce for a single product model or multiple product models. In the second configuration, many items of a product model can be processed at a time in batches, or all product models are handled simultaneously on the same assembly line with the lot sizes of one item for each product model [2]. These configurations of an assembly line represent for classes of line balancing problems:

- Single-model assembly line balancing problem with deterministic/stochastic/fuzzy processing times.
- Batch-model assembly line balancing problem with deterministic/stochastic/fuzzy processing times.
- Mixed-model assembly line balancing problem with deterministic/stochastic/fuzzy processing times.

In these line balancing problems, the requirement is often to distribute the tasks to workstations such that a certain objective (e.g., number of workstations, total cost, production rate, etc.) is optimized and the precedence relationship is not violated. The workstation time, which is the sum of times of all tasks assigned to that workstation, must not exceed the given cycle time. The processing time of tasks are also given. In general, the line balancing problem has several variances coming from the requirement, objective, or the form of processing time or the structure of the lines. The requirement of the problem is not only to allocate tasks to workstations, but also to sequence product models to be assembled in the designing batch/mixed-model lines or determine optimal batch sizes for batch-model configuration [3].

Determining the sequence of introducing models to the mixed-model assembly lines in a just-in-time (JIT) production system is of particular importance considering the goals crucial for efficient implementation of the JIT system. These goals are basically: 1) leveling the load on each process within the line; and 2) keeping a constant speed in consuming each part on the line [4]. Toyota Corporation developed the Goal Chasing I and II (GC-I and GC-II) methods to handle these problems [4].

[i]* Corresponding Author, R. Tavakkoli-Moghaddam is a professor in Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran (e-mail: tavakoli@ut.ac.ir)

ii Y. Gholipour-Kanani is a faculty member in Department of Management, Qaemshahr Branch, Islamic Azad University, Qaemshahr, Iran (e-mail: gholipourkanani@yahoo.com)

iii R. Cheraghalizadeh has received her M.Sc. degree from Mazandaran University of Science & Technology, Babol, Iran (e-mail: r_cheraghalizadeh@yahoo.com)

GC-I selects a model that minimizes the one-stage variation at each stage, and GC-II simplifies GC-I under special assumptions regarding product structures. Here a ‘‘stage’’ represents a position in the order of a sequence. Miltenburg [5] developed a non-linear programming model for the second above-mentioned goal. The time complexity function of the proposed program was exponential. Thus, he developed and solved the problem by applying two heuristic procedures. Miltenburg et al. [6] solved the same problem with a dynamic programming algorithm. Inman and Bulfin [4] solved the problem introduced by Miltenburg [5] by converting it to a new mathematical model. Other objectives were also considered by a number of researchers. Yano and Rachamdugu [8] minimized the total utility work. Bard et al. [9] considered an objective to minimize the overall line length. Okamura and Yamashina [10] developed a heuristic algorithm to minimize the risk of conveyor stoppage.

In the mathematical complexity, the line balancing problem is NP-complete in strong sense because the NP-complete bin-packing problem can be transformed easily to the line balancing problem in polynomial time [11]. Therefore, several heuristic procedures have been proposed to solve different versions of the line balancing problem. Some comprehensive analyses and reviews of the line balancing problems can be found in Baybars [11], Ghosh and Gagnon [12], and Scholl [13].

According to Fokkert and de Kok [14], most of studies on assembly line balancing problems are dated back to before 1980s. Most of approaches focus on solving a single-model line balancing problem with deterministic processing times [15-17]. A few efforts have been made for the batch-model and mixed-model line balancing problem. Recently, attention on the batch/mixed-model assembly line balancing is paid back due to the requirement of mass assembly with the support of advanced technologies, which help the assembly lines higher flexibility and faster processing speed.

Kabir and Tabucanon [18] studied a batch-model assembly line balancing problem using a multi-attribute decision making (MADM) approach. They generated a set of feasible number of workstations, which were balanced for each product model. Then, they selected the best configuration (i.e., number of workstations) considering multiple criteria, such as production rate, variety, minimum distance moved, quality, etc. The survey carried out by Fokkert and de Kok [14] also shows that there are two approaches in the literature transforming the mixed-model line balancing problem into a single-model line balancing problem. These approaches use combined precedence diagrams and adjusted task processing times. The experiment results also indicate that the position of common tasks in the precedence diagram of the different models has a significant effect on both the computation time and the unequal distribution of the total work content of single models among workstations. The extension to the approaches of a single-model for the mixed-model has been utilized by Gokcen and Erel in their different formulations [19-21]. The binary goal programming model of Gokcen and Erel [19] is the first multi-criteria decision making (MCDM) approach to the mixed-model version based on the model of Deckro and Rangachari [22] for a single model assembly line balancing problem. In the next version, Gokcen and Erel [20] formulate a binary integer programming model for the mixed-model assembly line balancing problem. The size of the model has been reduced significantly to an applicable problem with up to 40 tasks by using a combined precedence relationship and some variables that limit the increase in the number of decision variables and constraints.

Erel and Gokcen [21] also developed a shortest route formulation for MALB. This model was also based on the shortest-route model developed by Gutjahr and Nemhauser [23] for SMALB. Their proposed model was better than the shortest-route model proposed by Robert and Villa [24] in terms of the size of the formulated network as the number of tasks increases because they assigned common tasks of different product models to the same stations as well as the constraints that could be imposed by increasing the designers’ limit.

Tavakkoli-Moghaddam et al. [25 and 26] presented the optimal schedule and sequence of a set of jobs for a single machine with idle insert, in which the objective function is to minimize the sum of maximum earliness and tardiness ($n/1/I/ET_{max}$). Sequencing mixed-model assembly lines have also been studied as a multi-criteria problem. Bard et al. [24] developed a model involving two objectives as follows: (1) minimizing the overall line length; and (2) keeping a constant rate of part usage. They solved the problem by using the weighted sum and they proposed a tabu search (TS) method to solve such a problem. Hyun et al. [28] addressed three objectives as follows: (1) minimizing total utility work, (2) keeping a constant rate of part usage, and (3) minimizing total setup cost. This problem was solved by proposing a new genetic evaluation and selection mechanism. Korkmazel and Meral [29] developed a weighted sum approach for two goals introduced by Monden [30].

McMullen and Fraizer [31] developed a simulated annealing (SA) method for the model used by McMullen [32] and they compared this SA against the TS method. McMullen [33–35] also solved the same problem by using genetic algorithms (GA), Kohonen self-organizing map (SOM), and ant colony optimization, respectively. He also compared the performance of these three methods with SA and TS methods. Mansouri [36] also solved the same problem with GA, in which a new selection mechanism was introduced. A number of other metaheuristic methods can be applied to any other combinatorial optimization

problem. Tavakkoli-Moghaddam et al. [37] proposed an efficient memetic algorithm (MA) with a simulated annealing-based local search engine in order to solve a new model of a cell formation problem (CFP) for a multi-period planning horizon.

Emde and boysen [38] introduced an exact solution procedure that solves both problems simultaneously in a polynomial runtime. In addition, management implications regarding the trade-off between the number and capacity of two trains and in-process inventory near the line are investigated within a comprehensive computational study. Furthermore, Emde and boysen [39] discussed the general pros and cons of the supermarket concept and treats the decision problem of determining the optimal number and placement of supermarkets on the shop floor. A mathematical model is proposed, an exact dynamic programming algorithm presented, and the validity of the proposed approach for practical purposes as well as the trade-off resulting from fixed installation and maintenance cost is investigated in a comprehensive computational study.

Dong and Gui [40] solved the separation in the traditional serial production planning and scheduling in mixed-model assembly line, the integrated optimization complete model of production planning and scheduling based on multiple objectives and constraints was constructed.

Hamzadayi and Yildiz [41] presented a priority-based genetic algorithm (PGA) for the simultaneously tackling of the mixed-model U-shape assembly line (MMUL) balancing/model sequencing problems (MMUL/BS) with parallel workstations and zoning constraints, which allows the decision maker to control the process to create parallel workstations and to work in different scenarios.

Zenga et al [42] investigated the operator allocation problems (OAP) with jobs sharing and operator revisiting for balance control of a complicated hybrid assembly line, which appears in the apparel sewing manufacturing system. Multiple objectives and constraints for the problem are formulated.

Bautista et al [43] presented an extension to the mixed-model sequencing problem with work overload minimisation (MMSP-W) for production lines with serial workstations and parallel homogeneous processors. Boysena and Bock [44] proposed a new approach for a scheduling JIT part supply from a central storage center. Materials are usually stored in boxes allotted to the consumptive stations of the line by a forklift. For such a real-world problem, a new model, a complexity proof and different exact/heuristic solution procedures are provided.

Zhenga et al [45] presented a mathematical model and an ant colony optimization (ACO) algorithm for assembly line balancing (ALB). The mathematical model can be used to formally describe the problem. Akpınar and Bayhan [46] proposed a hybrid genetic algorithm (GA) to solve a mixed-model assembly line balancing problem of type I (MMALBP-I). There are three objectives to be achieved, namely minimizing the number of workstations, maximizing the workload smoothness between workstations, and maximizing the workload smoothness within workstations.

Hua et al [47] first reviewed the state of the art research in the areas of the assembly system design, planning and operations in the presence of product varieties. Methods for assembly representation, sequence generation and assembly line balancing are reviewed and summarized. The operational complexity and the role of human operators in assembly systems are then discussed in the context of product varieties. Challenges in disassembly and remanufacturing in the presence of high variety are presented.

Özcan [48] considered the problem of balancing two-sided assembly lines with stochastic task times (STALBP). A chance-constrained, piecewise-linear, mixed-integer programming (CPMIP) model is proposed to formulate and then solved by a simulated annealing (SA) algorithm.

Giard and Jeunet [49] presented an integer programming formulation for the sequencing problem in mixed-model assembly lines, in which the number of temporarily hired utility workers and the number of sequence-dependent setups are to be optimized through a cost function simultaneously. The resultant model offers an operational way to implement the utility work needed to avoid line stoppages, unlike previous papers addressing the goal of smoothing the workload.

Yang and Zhang [50] studied the parameter design and the performance optimization of a Kanban system without stockouts in a multi-stage, mixed-model assembly line.

Boysena et al [51] aimed at evenly smoothing the part consumption induced by the production sequence over time. Among these approaches, the popular product rate variation (PRV) problem is considered to be an appropriate approximate model, if either all products require approximately the same number and mix of parts or part usages of all products are almost completely distinct. These statements are further specified by analytical findings, which prove the equivalence of product and material oriented level scheduling under certain conditions.

Main difference between this work and other studies is that we consider simultaneously two objectives; namely the total stoppages assembly line cost and the total setup cost. The rest of this paper is organized as follows. In Section 2, we present the detailed description of the mixed-model assembly line (MMAL). In Section 3, we discuss about the complexity of the proposed model and propose GA and MA to solve such a hard model. Section 4 provides experimental results where a number of test problems are solved to show the efficiency of the proposed GA and MA and we present our conclusions in

this section.

2. MATHEMATICAL MODEL

A. Mixed-model assembly line sequencing problem

In this paper, the mixed-model assembly line (MMAL) is a conveyor system moving at a constant speed (v_c). Similar products are launched onto the conveyor at a fixed rate. The line is partitioned into *J* stations. It is assumed that the stations are all closed types. A closed station has boundaries that cannot be crossed by workers. Such a closed station is often found in reality where the use of facilities is restricted within a certain boundary. The tasks allocated to each station are properly balanced and their operating times are deterministic. The worker moves downstream on the conveyor while performing his/her tasks to assemble a product. To complete the job, the worker moves upstream to the next product. Suppose that the worker's moving time is ignored.

The design of the MMAL involves several issues, such as determining operator schedules, product mix, and launch intervals. Two types of operator schedules (i.e., early start schedule and late start schedule) are found in [52]. An early start schedule is more common in practice and is used in this paper. Second, a minimum part set (MPS) production, which is a strategy widely accepted in the mixed-model assembly lines, is also used in this paper.

B. Notations and problem description

The following notation is first given:

I. Indices

M Number of station

I Number of model for

T Maximum available time in each station

II. Parameters

a_m Difference coefficient of station *m* when line stops

s_m Risk cost of line stop in station *m*

$t_{[k]}$ Proceeding time of product that assign to sequence *k*

t_{mir} Proceeding time in station *m* when model *i* change to model *r*

T_m Average proceeding time of each station

S_{mir} Setup cost in station *m* when model *i* change to model *r* in this station

d_i Demand of model *i*

III. Decision variables

$$x_{kir} = \begin{cases} 1 & \text{if in sequence } k \text{ the model change from } i \text{ to } r \\ 0 & \text{Otherwise} \end{cases};$$

T_m Average proceeding time of each station

$$\left(T_m = \frac{\sum_{r=1}^{I}\sum_{i=1}^{I}\sum_{k=1}^{Q} t_{mir} x_{kir}}{Q}\right)$$

Q Total demand (i.e., $Q = \sum_{i=1}^{I} d_i$)

C. Mathematical model

I. Minimizing the risk of line stoppage

To minimize the risk of line stoppage, we should position a job with small proceeding time after a job with big proceeding time and also take equal between difference of these two times and average proceeding time of that station.

$$\left|(t_{[k]} - T_m) + (t_{k\_1} - T_m)\right| = \left|t_{[k]} + t_{[k-1]} - 2T_m\right| \qquad (1).$$

Because the line stoppage is related with all the products, we minimize the deviations of product proceeding times from the average proceeding time of that station as:

$$\sum_{m=1}^{M} a_m \sum_{k=2}^{Q} \left|t_{[k]} + t_{[k-1]} - 2T_m\right| \qquad (2).$$

where $t_{[k]}$ is the proceeding time in station *m* for the part positioned in sequence *k*; therefore, we have:

$$t_{[k]} = \sum_{i=1}^{I}\sum_{r=1}^{I} t_{mir} x_{kir} \qquad (3)$$

$$\sum_{m=1}^{M} a_m . s_m \sum_{k=2}^{Q} \left|t_{[k]} + t_{[k-1]} - 2T_m\right| =$$

$$\sum_{m=1}^{M} a_m . s_m \sum_{k=2}^{Q} \left|\sum_{i=1}^{I}\sum_{r=1}^{I} t_{mir} x_{kir} + \sum_{i=1}^{I}\sum_{r=1}^{I} t_{mir} x_{(k-1)ir} - 2T_m\right| \qquad (4)$$

$$= \sum_{m=1}^{M} a_m s_m \sum_{k=2}^{Q} \left|\sum_{i=1}^{I}\sum_{r=1}^{I} t_{mir}(x_{kir} + x_{(k-1)ir}) - 2T_m\right|$$

By the above consideration, we explain the first objective function minimizing the stoppages assembly line cost. The model is presented below.

$$\min \sum_{m=1}^{M} a_m . s_m \sum_{k=2}^{Q} \left|\sum_{i=1}^{I}\sum_{r=1}^{I} t_{mir}\left(x_{kir} + x_{(k-1)ir}\right) - 2T_m\right| \qquad (5)$$

s.t.

$$\sum_{i=1}^{I}\sum_{r=1}^{I} x_{kir} = 1, \forall k \qquad (6)$$

$$\sum_{i=1}^{I} x_{kir} = \sum_{p=1}^{I} x_{(k+1)rp}, k = 1,...,Q-1, \forall r \quad (7)$$

$$\sum_{i=1}^{I} x_{Qir} = \sum_{p=1}^{I} x_{1rp}, \forall r \quad (8)$$

$$\sum_{k=1}^{Q}\sum_{i=1}^{I} x_{kir} = d_r, \forall r \quad (9)$$

$$\sum_{k=1}^{Q}\sum_{i=1}^{I}\sum_{r=1}^{I} t_{mir} x_{kir} \le T, \forall m \quad (10)$$

$$x_{kir} \in \{0,1\}, \forall k, i, r \quad (11)$$

Eq. (6) is a set of position constraints indicating that every position in a sequence is occupied by exactly one product. Eqs. (7) and (8) ensure that the sequence of products is maintained while repeating the cyclic production. Eq. (9) imposes the restriction that all the demands should be satisfied in terms of MPS. Eq. (10) ensure that the total time not be exceeded in each station.

II. Minimizing the setup cost

In many industries, sequence-dependent setups are considered as an important item in assembly operations. The model considering sequence-dependent setups developed by Hyun et al. [53] is considered in this paper.

$$\min \sum_{m=1}^{M}\sum_{k=1}^{Q}\sum_{i=1}^{I}\sum_{r=1}^{I} x_{kir} s_{mir} \quad (12)$$

s.t.

Eqs. (6) to (9) and (11)

where S_{mir} is the setup cost required when the model type is changed from *i* to *r* at station *m*. X_{kir} is 1 if model type *i* and *r* are assigned, respectively, at position *k* and *k*+1 in a sequence; otherwise X_{kir} is 0.

III. Final objective function

The main objective function is the sum of each function (Equations 2 and 12) when multiplied by their weighting coefficients (i.e., λ_1 and λ_2 for the first and second objective functions, respectively). Note that these coefficients are in [0, 1] and the sum of them is 1.

The final objective function of this model is give below:

$$\min Z = \lambda_1 \sum_{m=1}^{M} a_m * s_m \sum_{k=2}^{Q} \left| \sum_{i=1}^{I}\sum_{r=1}^{I} t_{mir} \left(x_{kir} + x_{(k-1)ir} \right) - 2T_m \right| + \lambda_2 \sum_{m=1}^{M}\sum_{k=1}^{Q}\sum_{i=1}^{I}\sum_{r=1}^{I} x_{kir} s_{mir}$$

3. HEURISTIC METHODS AND PROPOSED ALGORITHMS

A. Heuristic Methods

There are several models used in the literature when considering different objective functions. The first type of the model is a mathematic model combined with a heuristic method. For examples, Hoffmann [54] combined a branch-and-bound algorithm with a heuristic method to approach the optimal solution, and Gokcen and Erel [19] modified the Patterson et al. algorithm with a binary goal-programming model to achieve the satisfactory results when facing conflicting goals. The second type of model is a heuristic search that is based on the existing solution. Malakooti [55] used existing heuristic balancing approaches to generate a set of efficient alternatives to minimize the buffer size in the case of multiple criteria.

Sonekar et al. [56] proposed a multiple criteria decision-making (MCDM) approach to minimize the number of subassemblies. This system was able to generate an entire set of efficient alternatives and used an interactive paired comparison of alternatives to solve the problem. The third type of model is a heuristic search that concentrates on less processing time. For instance, Boctor [57] proposed a heuristic that utilized a single-pass and composite method consisting of general assignment with priority ordering. Kabir and Tabucanon [18] used an analytic hierarchy process (AHP) and simulation (known as the Ignall algorithm) to determine the number of workstations in a multi-attribute problem. As an example reported by Enmer et al. [58], a model was introduced to balance an industrial truck engine assembly line. The heuristic generated all the feasible alternative sets of tasks for a single station, selecting the one with the least slack and then moved to the next station. Unfortunately, due to the simplified assumptions found in most heuristic methods and complexity of the real system, many line balancing tasks are still performed manually. This problem leads to another research direction, the intelligent system. In the intelligent line balancing system approaches, the intelligent component is implemented by using an expert system.

Roy and Allchurch [59] proposed a Prolog expert system to perform a mixed-model assembly line balancing. Oh [60] presented an expert line balancing system (ELBS). The ELBS applied a heuristic method and computerized into expert system shell that performs as an expert in an interactive mode. This system produced the number of substations in each major operational station, system cycle time, total number of stations in the system, total number of hours required, and overall efficiency. Kumar and Malakooti [61] reported an expert system model, which was constructed in C programming. The purpose of this model is to assist the practitioner in making decisions when there are many conflicting

objectives. Arinze and Partori [62] were the first to report a system that was able to produce the precedence network from a knowledge-based system to perform all the job allocations. Sudhir and Rajagopalan [63] also presented a frame based model, the ANGEL prototype system, to generate the precedence network through an artificial intelligence approach for assembly line balancing.

B. Evolutionary algorithms

In this section, we briefly describe GA and MA, and then we discuss the manner in which they were implemented to solve the flowshop scheduling problem with family setups.

I. Proposed genetic algorithm

Genetic algorithm (GA), which is a population-based algorithm, uses analogies to natural, biological, and genetic concepts including chromosome, mutation, crossover, and natural selection. Basically, it consists of making a population of solutions evolve by mutation and reproduction processes. The best fitted solutions of the population shall survive while the worse fitted will be replaced. After a large number of generations, it is expected that the final population is composed of highly adaptable individuals, or in an optimization application, high-quality solutions of the problem at hand. The basic steps of a canonical GA are as follows.

Step 1. Initialize the population.
Step 2. Select individuals for recombination and.
Step 3. Recombine individuals generating new ones.
Step 4. Mutate the new individuals.
Step 5. If the stopping criterion is satisfied, Stop the algorithm; otherwise, replace old individuals with the new ones restructure the population tree and return to Step 2.

In Step 1, the initial population is created. In our study, this population is composed of randomly generated solutions. Step 2 consists of selecting individuals among the population to recombine. This selection normally takes into account the fitness of the individuals or the quality of the solutions (regarding the objective function which in this case is the makespan). As our algorithm works with a hierarchically structured population, the selection is somewhat different. In Step 3, the selected individuals recombine and generate new individuals. This means that new information is being added to the population. In this step, we use a crossover oerator. In Step 4, some individuals are submitted to a mutation process in order to preserve the diversity of the whole population. The mutation should be very light or important information can be lost. Finally, in Step 5, the search stops if previously determined stopping criterion is satisfied. Otherwise, the new individuals generated in Steps 3 and 4 replace some individuals of the population. In general, the solutions to be replaced are chosen accordingly to their quality, and the worst fitted will give their place to the new ones [64].

II. Proposed Memetic Algorithm

In the case of a memetic algorithm (MA), after Step 4 in the above-mentioned GA, all new individuals go through a local search procedure before Step 5. Details of MA are described in Moscato [65] and Corne et al. [66].

The implementations of GA and MA are similar to previous work by Mendes, Muller, Franc¸a, and Moscato [67]. They share some main characteristics with the versions implemented for the single and parallel machine scheduling problems [67] although some improvements have been introduced in this adaptation to the flowshop problem.

Memetic algorithms can be viewed as a marriage between a population-based global technique and a local search made by each of the individuals. They are a special kind of genetic algorithms with a local hill climbing. Like genetic algorithms, memetic algorithms are a population-based approach. They have shown that they are orders of magnitude faster than traditional genetic algorithms for some problem domains. In a memetic algorithm the population is initialized at random or using a heuristic method. Then, each individual makes local search to improve its fitness. To form a new population for the next generation, higher quality individuals are selected. The selection phase used in the MA is the same as used in the classical genetic algorithm. Once two parents are selected, their chromosomes are combined and the classical operators of crossover are applied to generate new individuals. The latter are enhanced using a local search technique. The role of local search in memetic algorithms is to locate the local optimum more efficiently then genetic algorithms.

The basic steps of MA used here are as follows.

Step 1. Initialize the population.
Step 2. Select individuals for recombination.
Step 3. Recombine individuals generating new ones.
Step 4. Mutate the new individuals.
Step 5. Apply the local search on the new individuals.
Step 6. If the stopping criterion is satisfied, Stop the algoithm; otherwise, replace old individuals with the new ones, restructure the population tree and return to Step 2. Fig. 1 provides a general template for the memetic algorithm.

C. Implementation of evolutionary algorithms

I. Solution representation

Each solution is represented with a matrix. This matrix size is $1\times Q$ that Q is the total demand of parts. A solution can be feasible or infeasible. For example, a solution is shown in Fig. 2 for a problem with 3 models and 6 parts, in which the demand of each model 1, 2 and 3 is 2, 3 and 1 respectively.

II. Fitness

Each solution has a fitness function value, which is

related to the objective function value of the solution. However, the population can have feasible and infeasible solutions. An option to manage the infeasibility is to use both cost and feasibility. This can be written as fitness cost feasibility; where *s* is the solution, cost of the objective function value, and feasibility=1 if the solution is feasible; and 0, otherwise. Therefore, the fitness is not one value; however, it is two (i.e, the cost and the feasibility of the solution).

III. Initialization

The initialization can be executed with either a randomly created population or a well-selected population. In this study, an initial population of the desired size is generated randomly. For example, when there are five parts, the algorithm generates 10 chromosomes at random, depending on the problem size.

```
Procedure Memetic Algorithm;
Begin
initialize population;
foreach individual do local-search individual;
repeat
for individual =1 to #crossovers do
select two parent individual1, individual2 \in population
      with use parent selection strategy;
individual3:=crossover(individual1, individual2);
individual3:=local-search(individual3);
add individual3 to population;
end for;
for individual=1 to #mutations do
select an individual of population randomly;
individual{m} := mutate (individual);
individual{m} := local-search (individual{m});
add individual{m} to population;
end for;
population:=select(population);
if population converged then
for each individual of best populations do individual :=local-
      search(mutate(individual);
end if
until terminate=true;
```

Figure 1: General outline of the MA in a pseudo code.

2	3	2	1	1	2

Figure 2:. Example of a feasible solution.

IV. Parent selection

Reproduction selects good strings in a population and forms a mating pool. That is why the reproduction operator is sometimes called the selection operator. Methods of selection mechanisms are as follows:

- Rank order selection method
- Roulette wheel selection method
- Tournament selection method

Following is a breif description of the roulette wheel selection method. For crossover and mutation operators, the strings of higher fitness values are selected. The probability of selecting each string is calculated by:

$$SI = \frac{F(t)}{F_{total}} \tag{13}$$

where $F(t)$ = Fitness value of the *i*th string, F_{total} = total fitness value of all strings.

The total fitness of the population is computed by:

$$F(t) = \sum_{i=1}^{n} F(t) \tag{14}$$

To select *n* strings, random numbers between 0 and 1 are generated.

V. Crossover

The chromosome to be crossed and the crossing points are to be selected randomly. Crossover is carried out with a probability called the crossover probability (CPROB). For the above random numbers (i.e., initial population) are generated for each chromosome and compared with the crossover probability values. The chromosomes with higher values of CPROB are chosen for crossover. The crossover probability is taken as 0.90. There are several types of crossover operators available and some of them are single point, two points, and uniform and arithmetic crossover operators. In this study, we use single point crossover as shown in Fig. 3.

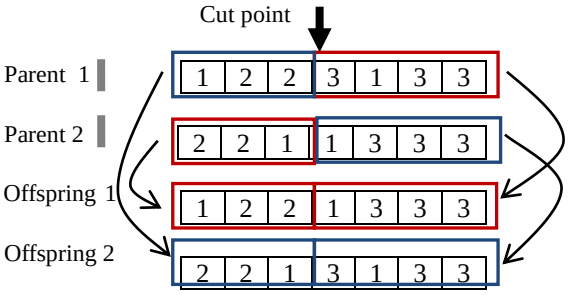

Figure 3. Crossover operation

VI. Mutation

Mutation is the process by which a randomly chosen bit in a chromosome is flipped. It is employed to introduce new information into the population and also to prevent the population from becoming saturated with similar chromosomes (i.e., premature convergence). Large mutation rates increase the probability that good schemata be destroyed, but increase population diversity. A schema is a subset of chromosomes which are identical in certain fixed positions [68, 69].

Mutation is carried out with mutation probability of 0.05. At first with use parent selection strategy choose one chromosome and then two random integers' r_1 and r_2 are generated.

$1<r_1<=Q$ (i.e., total demand) $1<r_2<=Q$. The algorithms then removes gene number r_1 from the current string and assigns it to gene r_2 and also assign gene number r_2 to

gene r_1, as shown below. Take random numbers r_1 =3 and r_2 =4, then we have:

Before mutation:	1	2	3	1	2	3	3

After mutation:	1	2	1	3	2	3	3

VII. Reproduction

We choose a chromosome with the best fitness and then this mechanism just copies the chosen chromosome to the next generation. Reproduction is carried out with the reproduction probability of 0.05.

VIII. Local search algorithms

Our local search algorithm for the multi-objective case uses the same notion of neighborhood as in the single-objective case. However, the acceptance criterion of the single-objective local search algorithms needs to be changed to take into account several objectives. While in iterative improvement local search for the single-objective case a solution is accepted if it is better than the current one, for multi-objective problems an extension of this acceptance criterion should take into account the concept of Pareto optimality. For the sake of simplicity, a first approach for an acceptance criterion may be to accept a neighboring solution, if it is *not worse* than the current solution. However, a solution accepted by such a criterion may be dominated by other solutions seen previously in the local search. To avoid this, we maintain an archive of non-dominated solutions. The final acceptance criterion used for the local search is as follows. Each new solution is compared with the current one. If the new solution is *not worse* in the Pareto sense, it is compared in the next step to all solutions in the archive. Only if the new solution is not dominated by any solution of the archive, it is finally accepted and included into the archive. In fact, during this process, some solutions from the archive can become dominated by some of the recently introduced ones. Such solutions are eliminated continuously from the archive.

The local search algorithm starts from a randomly generated initial solution that is put into the archive. It then works as follows: First, it picks a solution randomly from the archive and iteratively explores the neighborhood of this solution. If a *not worse* solution is found, it is compared with the solutions in the archive and the local search continues. If at some point all neighboring solutions were explored and none is accepted any more, the solution is flagged as *visited*, *i.e.*, this solution is a Pareto local optima solution and it will not be chosen again. We terminate the local search procedure if all the neighborhoods of all solutions in the archive were explored, *i.e.* every solution in the Pareto local optimum set is flagged as *visited*. In this case, a Pareto local optimum set is found. It should be remarked that this local search is similar to PAES [70], although we stress the importance of neighborhood and use a simpler acceptance criteria for comparing and accepting non-dominated solutions.

D. GA and MA parameters

The values of a variety of parameters and policies, such as crossover rate, mutation rate, population size, number of generations, and the like, are to be chosen and there should be a balance between the exploitation and the exploration aspects of GA and MA operators. Reproduction is responsible for the exploration of the current population by making many duplicates of good strings and cross over and mutation is responsible for exploring a set of good strings for better strings. The success of GA and MA depends on a balance between the two. If too many copies of the good strings are allocated in the mating pool then the diversity of the mating pool reduces, which in turn reduces the extent of the search that can be accomplished using crossover and mutation operators. Therefore, we use 10 individuals which correspond to a ternary tree with three levels. The crossover rate should be set at high levels, since our insertion policy works as a filter on the new individuals, accepting only those that improve the overall fitness. In our tests we tried crossover rates of 0.75, 0.8 and 0.9. In other words we tested situations where we create from 7 to 26 new individuals every generation. The best number was 20 individuals which correspond to a crossover rate of 0.9.

With less individuals being created every generation, we have heavy mutations procedures being applied too often. That occurred because it was very difficult to have a new individual replacing an old one after the population had already evolved for, say, 10 or 15 iterations. With 26 individuals being created, the number of generations executed during the 30-s time span dropped considerably, worsening the results. The mutation rate was set at 0.05 for the MA that means 5% of MA new individuals go through a light mutation process every generation and then reproduction is carried out with reproduction probability of 0.05.

4. COMPUTATIONAL RESULTS

In this section, we evaluate the quality of the GA and MA solutions by means of a computational study. We teste the GA and MA at random problems, and compared GA and MA solutions with solutions obtained using a branch-and-bound (B&B) procedure with the Lingo 8 software. The B&B returns an optimal solution if given enough time. The heuristic was coded in Matlab 7.0 and run on a 2.4 GHz Pentium IV computer. So, the heuristic is tested on a set of mixed-model assembly line balancing problems developed by the authors and whose main characteristics are shown in Table 1.

In this paper, we consider eight small and 14 large-scale problems generated from uniform distributions. Considering the results of Table 1 and Fig. 4, the proposed

GA and MA are able to find and report the optimal or near-optimal solutions in a reasonable computational time. This indicates the success of our proposed algorithms.

Fig. 5 illustrates the diagram of the CPU time related to the solutions reported by Lingo 8.0, GA and MA in respect to the number of problems for those instances given in Table 1. This figure indicates that the computational time increases quickly, when the number of stations, the total part and the number of models increases.

Table 2 illustrates the computational results of large-scale problems. These results reveal that the proposed GA and MA have the ability to compete with the Lingo software from the quality point of view. That is very clearly true for large-scale problems, in which Lingo cannot produce any feasible solution.

In GAs and MAs, various procreation parameters are used. We carry out experiments with different combinations of parameters to recognize the suitable set of parameters and their effect on solutions. The selection with a greater pressure on better individual (i.e., with a higher rate of crossover) reduce the diversification; therefore, the solutions are premature.

In the mixed-model assembly line balancing problem, as a big partition of the population come together to special solutions, the likelihood of solution improvement decreases because the rate of selecting the same solution increases. Thus, it is important to find a fitting rate for crossover and mutation. An experiment is performed by considering different crossover rates from 0.55 to 0.95 with an increase of 0.05, and different mutation rates from 0.40 to 0.00 with a reduction of 0.05. However, the crossover rate plus the mutation rate should be equal to 0.95. The result of this experiment shows that our algorithm provides the best solution when the crossover and mutation rates are 0.80 and 0.15, respectively.

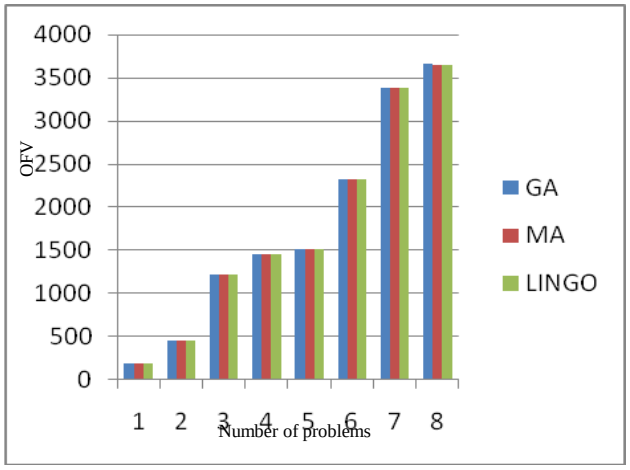

Figure 4: Objective function values.

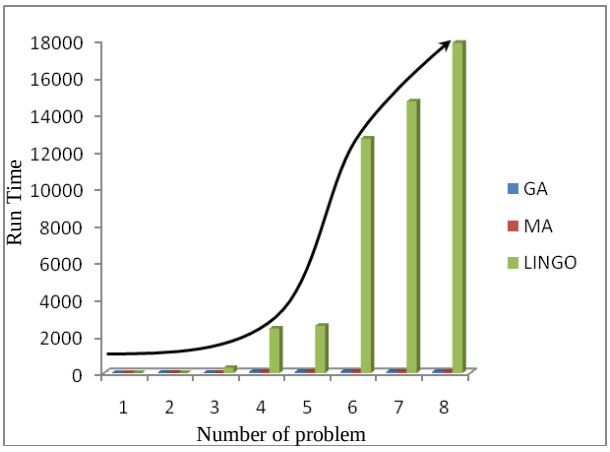

Figure 5: Computational time obtained by Lingo and the proposed GA and MA for small-scale problems.

TABLE 1
COMPARISON BETWEEN LINGO 8 AND THE PROPOSED GA AND MA FOR SMALL-SCALE PROBLEMS

Problem	No. of Stations	Total part	No. of models	Algorithm	Run time	Objective function value
1	2	3	3	GA	5	185
				MA	7	185
				LINGO	2	185
2	2	5	4	GA	10	458
				MA	10	458
				LINGO	3	458
3	4	6	3	GA	15	1220
				MA	18	1220
				LINGO	289	1220
4	5	7	4	GA	61	1458
				MA	63	1450
				LINGO	2410	1450
5	5	8	4	GA	65	1520
				MA	69	1520
				LINGO	2556	1520
6	6	8	4	GA	70	2320
				MA	70	2320
				LINGO	12687	2320
7	7	8	4	GA	72	3380
				MA	75	3380
				LINGO	14692	3380
8	7	8	5	GA	61	3660
				MA	63	3640
				LINGO	17867	3640

TABLE 2
COMPARISON BETWEEN LINGO 8 AND THE PROPOSED GA AND MA FOR LARGE-SCALE PROBLEMS

Problem	No. of Stations	Total part	No. of models	Algorithm	Run time	Objective function value
1	8	10	5	GA	75	5470
				MA	79	5556
				LINGO	21600	N/A
2	8	11	8	GA	145	8639
				MA	143	8751
				LINGO	21600	N/A
3	8	12	10	GA	210	12377
				MA	215	12377
				LINGO	21600	N/A
4	10	15	12	GA	405	28956
				MA	400	28956
				LINGO	21600	N/A
5	12	21	15	GA	856	66951
				MA	880	66951
				LINGO	21600	N/A
6	18	25	20	GA	2097	207385
				MA	2452	207385
				LINGO	21600	N/A
7	19	25	20	GA	2097	207385
				MA	2452	207385
				LINGO	21600	N/A
8	20	25	20	GA	2163	265152
				MA	2339	265152
				LINGO	21600	N/A
9	20	26	21	GA	2270	27320
				MA	2520	27305
				LINGO	21600	N/A
10	20	28	22	GA	2897	285870
				MA	3020	279870
				LINGO	21600	N/A
11	22	28	24	GA	2956	305829
				MA	3185	298256
				LINGO	21600	N/A
12	22	30	26	GA	3215	311956
				MA	3305	308950
				LINGO	21600	N/A
13	22	30	28	GA	3185	317562
				MA	3365	312850
				LINGO	21600	N/A
14	24	30	30	GA	3185	317562
				MA	3365	312850
				LINGO	21600	N/A

5. CONCLUSION

This paper has presented a new mathematical programming model for the mixed-model assembly line balancing problem that minimizes the total conveyor stoppage time and setup cost. Since the conveyor stoppages are frequently caused in many actual mixed-model assembly lines, the objective of minimizing the total conveyor stoppage time becomes more and more important. An excellent sequence for mixed models to be assembled on the conveyor can help to improve the performance of the assembly line. The properties characterized in this paper would be useful and powerful in developing efficient algorithms and evaluating the quality of solutions. In this sense, the considered problem is more general than the conventional flow shop group scheduling problem. In addition, we have proposed a genetic algorithm (GA) and a memetic algorithm (MA) to solve the foregoing problems. The Lingo software and our proposed GA and MA have been used to solve 21 different test problems. The obtained results have indicated that the proposed GA and MA have been able to reduce the time as compared to the Lingo software.

6. REFERENCES

[1] P. M. Vilarinho and A.S. Simaria, "A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations", International Journal of Production Research, Vol. 40, pp. 1405-1420, 2002.

[2] Y. Zhang and P.B. Luh, K.Yoneda, T. Kano and Y. Kyoya, "Mixed-Model Assembly Line Scheduling Using the Lagrangian Relaxation Technique", Institute of Industrial Engineering, Vol. 32, 125-134, 2000.

[3] N. V. Hop, "A heuristic solution for fuzzy mixed-model line balancing problem", European Journal of Operational Research , Vol. 168 (3), pp. 798–810, 2006.

[4] Y. Monden, Toyota Production System, second Ed. The Institute of Industrial Engineers, Norcross, GA, 1993.

[5] J. Miltenburg, "Level schedules for mixed-model assembly lines in just-in-time production systems", Management Science, Vol. 35, pp. 192–207, 1989.

[6] J. Miltenburg, G. Steiner and S. Yeomans, "A dynamic programming algorithm for scheduling mixed-model just-in-time production systems", Mathematical Computation Modeling, Vol. 13, pp. 57–66, 1990.

[7] P.R. Inman and R. L. Bulfin, "Note on sequencing JIT mixed-model assembly lines", Management Science, Vol. 37, pp. 904–910, 1991.

[8] C.A. Yano, and R. Rachamadugu, "Sequencing to minimize work overload in assembly lines with product options", Management Science, Vol. 37, pp. 572–586, 1991.

[9] J.F. Bard, E.M. Dar-El, and A. Shtub, "An analytic framework for sequencing mixed model", International Journal of Production Research, Vol. 30, pp. 35–48, 1992.

[10] K. Okamura, and H. Yamshina, "A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the conveyor", International Journal of Production Research, Vol. 17, 233–247, 1979.

[11] I. Baybars, "A survey of exact algorithms for the simple assembly line balancing problem", Management Science, Vol. 2, pp. 909–932, 1986.

[12] S. Ghosh, and R. J. Gagnon, "A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems". International Journal of Production Research, Vol. 27, pp. 637-670, 1989.

[13] A. Scholl, Balancing and sequencing of assembly lines, Physica, Heidelberg, second Ed., 1999.

[14] Fokkert, J.I.V.Z.D. and de Kok, T.G., "The mixed and multimodel line balancing problem: A comparison", European Journal of Operational Research, Vol. 100, pp. 399–412, 1997.

[15] W.B. Helgeson, and D.P. Birnie, "Assembly line balancing using the ranked positional weight technique", Journal of Industrial Engineering, Vol. 12, pp. 394–398, 1961.
[16] A.L. Gutjahr and G.L. Nemhauser, "An algorithm for the line balancing problem", Management Science, Vol. 11, pp. 308–315, 1964.
[17] E. M. Mansoor, "Assembly Line Balancing – An Improvement on the Ranked Positional Weight Technique", Journal of Industrial Engineering, Vol. 15, pp. 73-78, 1964.
[18] A. Kabir, and M. Tabucanon, "Batch-Model Assembly Line Balancing: A Multi- Attribute Decision Making Approach", Int. J. of Production Economics, Vol. 41, pp. 193-201, 1995.
[19] H. Gokcen and E. Erel, "A Goal Programming Approach to Mixed-Model Assembly Line Balancing Problem", Int. J. of Production Economics, Vol. 48, pp. 177-185, 1997.
[20] H. Gokcen and E. Erel, "Binary integer formulation for mixed-model assembly line balancing problem", Computers and Industrial Engineering, Vol. 34, pp. 451–461, 1998.
[21] E. Erel and H. Gokcen, "Shortest-route formulation of mixed-model assembly line balancing problem", European Journal of Operational Research, Vol. 116, pp. 194–204, 1999.
[22] R.F. Deckro and S. Rangachari, "A goal approach to assembly line balancing", Computers and Operations Research, Vol. 17, pp. 509–521, 1990.
[23] A.L. Gutjahr and G.L. Nemhauser, "An algorithm for the line balancing problem", Management Science, Vol. 11, 1964, pp. 308– 315, 1964.
[24] S.D. Roberts and C.D. Villa, "On a multiproduct assembly line-balancing problem", AIIE Transactions, Vol. 2, pp. 361–364, 1970.
[25] R. Tavakkoli-Moghaddam, G. Moslehi, M. Vasei and A. Azaron, "Optimal scheduling for a single machine to minimize the sum of maximum earliness and tardiness considering idle insert", Applied Mathematics and Computation, Vol. 167, pp. 1430–1450, 2005.
[26] R. Tavakkoli-Moghaddam, G. Moslehi, M. Vasei and A. Azaron, "A branch-and-bound algorithm for a single machine sequencing to minimize the sum of maximum earliness and tardiness with idle insert", Applied Mathematics and Computation, Vol. 17, pp. 388–408, 2006.
[27] J.F. Bard, A. Shtub and S.B. Joshi, "Sequencing mixed-model assembly lines to level parts usage and minimize the length", International Journal of Production Research, Vol. 32, pp. 2431–2454, 1994.
[28] C.J. Hyun, Y. Kim and Y.K. Kim, "A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines", Computers and Operations Research, Vol. 25, pp. 675–690, 1998.
[29] T. Korkmazel and S. Meral, "Bi-criteria sequencing methods for the mixed-model assembly line in just-in-time production systems", European Journal of Operational Research, Vol. 131, pp. 188–207, 2001.
[30] Y. Monden, Toyota Production System, Institute of Industrial Engineers Press, Atlanta, 1983.
[31] P.R. McMullen and G.V. Frazier, "A simulated annealing approach to mixed-model sequencing with multiple objectives on a JIT line", IIE Transactions, Vol. 3, pp. 679–686, 2000.
[32] P.R. McMullen, "JIT sequencing for mixed-model assembly lines with setups using tabu search", Production Planning and Control, Vol. 9, pp. 504–510, 1998.
[33] P.R. McMullen, "An efficient frontier approach to addressing JIT sequencing problems with setups via search heuristics", Computers and Industrial Engineering, Vol. 41, pp.335–353, 2001.
[34] P.R. McMullen, "A Kohonen self-organizing map approach to addressing a multiple objective, mixed-model JIT sequencing problem", International Journal of Production Economics, Vol. 72, pp. 59–71, 2001.
[35] P.R. McMullen, "An ant colony optimization approach to addressing a JIT sequencing problem with multiple objectives", Artificial Intelligence in Engineering, Vol. 15, pp. 309–317, 2001.
[36] S.A. Mansouri, "A multi-objective genetic algorithm for mixed-model sequencing on JIT assembly lines", European Journal of Operational Research, Vol. 167, pp. 696–716, 2005.
[37] R. Tavakkoli-Moghaddam, N. Safaei and M. Babakhani, "Solving a dynamic cell formation problem with machine cost and alternative process plan by memetic algorithms", in: O.B. Lupanov, O.M. Kasim-Zade, A.V. Chaskin, K. Steinhofel (Eds.), Stochastic Algorithms: Foundation and Applications, Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 3777, pp. 213–227, 2005.
[38] S. Emde and N. Boysen, "Optimally routing and scheduling tow trains for JIT-supply of mixed-model assembly lines", European Journal of Operational Research, Vol. 217, 287–299, 2012.
[39] S. Emde and N. Boysen, "Optimally locating in-house logistics areas to facilitate JIT-supply of mixed-model assembly lines", International Journal of Production Economics, Vo. 135, 393–402, 2012.
[40] Q.Y. Dong, J. Lu, and Y. Gui, "Integrated Optimization of Production Planning and Scheduling in Mixed Model Assembly Line", Procedia Engineering, Vol. 29, 3340–3347, 2012.
[41] A. Hamzadayi and G. Yildiz, "A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints", Computers & Industrial Engineering Volume 62, 206– 215, 2012.
[42] X. Zenga, W. K. Wonga and S. Y. Leung, "An operator allocation optimization model for balancing control of the hybrid assembly lines using Pareto utility discrete differential evolution algorithm", Computers & Operations Research, Vol. 39, 1145–1159, 2012.
[43] J. Bautista, A. Cano and R. Alfaro, "Modeling and solving a variant of the mixed-model sequencing problem with work overload minimization and regularity constraints. An application in Nissan's Barcelona Plant", Expert Systems with Applications, Available online 14 March 2012.
[44] N. Boysena and S. Bock, "Scheduling just-in-time part supply for mixed-model assembly lines", European Journal of Operational Research, Vol. 211, 15-25, 2011.
[45] Q. Zhenga,Y. Lia and M. Li, "Assembly Line Balancing Model Based on Ant Colony Optimization Algorithm", Energy Procedia, Vol. 13, 5366–5372, 2011.
[46] S. Akpınar and G. M. Bayhan, "A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints", Engineering Applications of Artificial Intelligence, Vol. 24, 449–457, 2011.
[47] S.J. Hua, J. Kob, L. Weyandc, H.A. ElMaraghyd, T.K. Liene, Y. Korena, H. Bleyc, G. Chryssolourisf, N. Nasrg and M. Shpitalnih, "Assembly system design and operations for product variety", CIRP Annals - Manufacturing Technology, Vol. 60, 715–733, 2011.
[48] U, Özcan, "Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm", European Journal of Operational Research, Vol. 205, 81–97, 2010.
[49] V. Giard and J. Jeunet, "Optimal sequencing of mixed models with sequence-dependent setups and utility workers on an assembly line", International Journal of Production Economics, Vol. 123, 290–300, 2010.
[50] L. Yang and X. Zhang, "Design and Application of Kanban Control System in a Multi-Stage, Mixed-Model Assembly Line", Systems Engineering - Theory & Practice, Vol. 29, 64-72, 2009.
[51] N. Boysena, M. Fliednerb and A. Scholl, "The product rate variation problem and its relevance in real world mixed-model assembly lines", European Journal of Operational Research, Vol. 197, 818–824, 2009.
[52] J.F. Bard, E.M. Dar-El and A. Shtub, , "An analytic framework for sequencing mixed model", International Journal of Production Research, Vol. 30, pp. 35–48, 1992.
[53] C.J. Hyun, Y. Kim and Y.K. Kim, "A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines", Computers and Operations Research, Vol. 25, pp. 675–690, 1998.
[54] T. Hoffmann, "Eureka: A hybrid system for assembly line balancing", Int. J. of Management Science, Vol. 38, pp. 39-47, 1992.
[55] B.B. Malakooti, "Assembly line balancing with buffers by multiple criteria optimization", Int. J. of Production Research, Vol. 32,

pp. 2159-2178, 1994.
[56] P.P Sonekar, S.M. Sindhi, J.V.L. Venkatesh, B.M. Dabade and S.P. Kallurkar, "A multiple criterion heuristic software for the practical assembly line balancing problem", Stochastic Models Optimization Techniques and Computer Applications, pp. 303-313, 1994.
[57] F. Boctor, "A multiple-rule heuristic for assembly line balancing", Int. J. of Operational Research Society, Vol. 46, pp. 62-69, 1995.
[58] A. Enmer, J. Favrel and J. Gauthie, "Balancing an assembly line for industrial truck engines", Proceedings for IFAC Intelligent Manufacturing System, Bucharest, Romania, pp. 163-165, 1995.
[59] R. Roy and M.J. Allchurch, "Development of a knowledge-based system for balancing complex mixed model assembly lines", International Journal of Computer Integrated Manufacturing, Vol. 9, pp. 205-216, 1996.
[60] K. Oh, "Expert line balancing system (ELBS)", Computer & Industry Engineering, Vol. 33, pp. 303-306, 1997.
[61] A. Kumar and B. Malakooti, "A knowledge-based system for solving multi-objective assembly line balancing problems", International Journal of Production Research, Vol. 34, pp. 2533-2552, 1996.
[62] B. Azinze and F. Partovi, "A knowledge based method for designing precedence networks and performing job allocation in line balancing", Compute Industry Engineering, Vol. 18, pp. 351-364, 1990.
[63] K. Sudhir and K. Rajagopalan, "An artificial approach to precedence network generation for assembly line balancing", Computers in Industry, Vol. 18, pp. 177-191, 1992.
[64] R. Tavakkoli-Moghaddam, Y. Gholipour-Kanani, and R. Cheraghalizadeh, "A genetic algorithm and memetic algorithm to sequencing and scheduling of cellular manufacturing systems", International Journal of Management Science and Engineering Management, Vol. 3, pp. 119-130, 2008.
[65] P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Technical Report C3P 826. Caltech Concurrent Computation Program, California Institute of Technology, Pasadena: CA, 1989.
[66] P. Moscato, "Memetic algorithms: A short introduction, In: D. Corne, M. Dorigo, F. Glover (Eds.), "New ideas in optimization", McGraw-Hill, London, pp. 219-213, 1999.
[67] A. S. Mendes, F. M. Muller, a.P.M. Franc and P. Moscato, Comparing meta-heuristic approaches for parallel machine scheduling problems with sequence-dependent setup times, Proceedings of the 15th Int. Conf. on CAD/CAM Robotics and Factories of the Future, A ` guas de Lindo` ia, SP, Brazil, 1999.
[68] D.E. Goldberg, Genetic algorithms in search, optimization and machines learning, Addison-Wesley, Reading, MA, 1989.
[69] J. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.
[70] J. Knowles and D. Corne, "The pareto archived evolution strategy: A new baseline algorithm for pareto multi-objective optimization", The Proc. of CEC'99, 98–105, 1999.