*Proceedings of the 2012 International Conference on Industrial Engineering and Operations Management*
*Istanbul, Turkey, July 3 – 6, 2012*

# Solving Assembly Line Balancing Problem in the State of Multiple-Alternative Operations Sequence using GRASP Meta-Heuristic Method and Comparing with Genetic Algorithm

**Nasir Saidi Rashboogari**
**Department of Industrial Engineering**
**Payame Noor University**
**Kermanshah, Iran**

## Abstract

Assembly line balancing is a basic element of production planning systems, which assign tasks into the workstations, subject to the precedence constraints be satisfied and the number of workstations and also their idle times be minimized. In real problems, an assembly line balancing may be including of some alternatives for the assembly. Hence the evaluation of the assembly line balancing problem with multiple-alternative in order to achieve the maximum performance can be useful in practice. In this article, a generalized problem of multiple-alternative assembly line balancing is defined, which is divided into two sub-problems. Sub-problem 1 is decision making about alternatives and sub-problem 2 is an assembly line balancing problem. The complexity of the mentioned problem is NP-hard, so its solved by GRASP algorithm, a meta-heuristic method. This problem is also solved by genetic algorithm (GA) for comparing with GRASP method's capabilities.

## Keywords
Assembly line balancing, multiple-alternative, task, workstation, GRASP, GA

## 1. Introduction
The problem of the assembly line balancing is among of the problems on which, during the recent decades many research and studies have been done. These researches include the problems concerning the generalized assembly lines such as the mixed models, parallel workstations, U-shaped lines, multiple objectives models, assembly line balancing in fuzzy environments and etc. In the first assembly line balancing problem, was posed by Henry Ford in the Ford industries. This problem was analytically expressed by Helgeson and then it was published through a mathematical model by Salveson [1].

A well-known classification of ALBPs is the one proposed by Baybars [2], which differentiates between two classic problems: the Simple Assembly Line Balancing Problem (SALBP) and the General Assembly Line Balancing Problem (GALBP). The SALBP includes very simple (and therefore restricted) problems (see [2] for the assumptions of the simple case). GALBPs are those problems in which one or more assumptions of the simple case are relaxed. If one reviews the literature concerning assembly line balancing problems, one can see that a huge amount of research exists, although most authors focus on the simple case [2-6].

Numerous algorithms have been developed to solve ALBP, most of which focus on solving SALBP. Two major groups can be outlined: exact methods, which are mainly based on linear programming, dynamic programming and branch-and-bound procedures, and heuristic and meta-heuristic methods. Information concerning both types of solving procedures can be found, for example, in [2], [11], [4] and [6].

A comprehensive analysis of the prior researches on the assembly line balancing problem show that most of the prior studies deal with the simple cases, whereas most of the problem in real world deal with multiple alternatives for an assembly line, and therefore, considering the assembly line balancing problem that takes the possible alternatives of an assembly into consideration for achieving the maximum performance will be very useful and valuable. In this article, a generalized problem of multiple-alternative assembly line balancing is defined, which

is divided into two sub-problems. Sub-problem 1 is decision making about alternatives and sub-problem 2 is an assembly line balancing problem. The complexity of the mentioned problem is NP-hard, so it's solved by GRASP algorithm, a meta-heuristic method. This problem is also solved by genetic algorithm (GA) for comparing with GRASP method's capabilities.

The Problem of assembly line balancing in the state of multiple-alternative contains the following main characteristics and assumptions:

- This problem considers a serial assembly line designed for a single model of a unique product, for which all alternative assembly variants are completely known in advanced.
- None of the task processing times are larger than the cycle time.
- Tasks have to be processed completely in one workstation only, i.e., they cannot be divided between workstations.
- Workstations can process only one task at a time.
- Tasks can not be processed in an arbitrary order due to the existence of precedence constraints.
- Several sets of precedence constraints are available, instead of a unique one, which represent the precedence relations among the tasks of the available alternative.
- All tasks belonging to a particular alternative have to be performed according the specifications of the same assembly variant.
- Tasks processing times are dependent on the assembly alternative selected, but independent on the workstation where they are processed.
- Setup times are considered to be negligible.
- All workstations are equally equipped and manned; therefore, any task can be assigned to any workstation.
- Tasks are not incompatible between each other; therefore, any combination of tasks can be assigned to any of the workstations.
- Tasks must be processed at most once. Therefore, only those tasks belonging to the selected assembly alternatives (or those that do not allow alternatives) must be performed. The remaining tasks will not be considered in the assembly process and, therefore, will not be carried out.

## 2. Definitions

- **Sequence Operation:** It is related to the sequence constraints of the assembly operation and precedence constraints. It means that without having done a set of tasks completely, the rest of operation after it can not be started.
- **Precedence Graph:** This graph illustrates the form of the task on the basis of precedence constraints. A precedence graph consists of nodes to represent the tasks required by each assembly alternative and connecting arcs which indicate the corresponding task precedence relations; furthermore, task processing times are represented as node weights.
- **Sub-Graph:** According to the lack of possibility for representing the alternatives in a standard Precedence Graph, sub-graph are proposed as the means of the geometric representation of the alternatives for an assembly in a unique graph. Each alternative is represented by a separate precedence graph, in which the necessary tasks for the assembly process, determine the precedence relations among them and their process time. The capability of the Sub-graphs to depict feasible assembly variants in a unique graph may let practitioners to have a better understanding of the system as a whole.
- **Sub-Assembly:** In many of the industrial products, there may be an assembly or de-assembly process with several feasible and valid planning at hand. So, the different components of an assembly process can accept different alternatives for precedence sub-graphs. For each of the several alternatives, a set of tasks that should be assigned to a group of the workstations is available. A sub-assembly shows all the possible sub-graphs for a group of tasks that accept different alternatives.
- **Approximate Procedures:** Due to the NP-hard nature of this type of combinatorial problem, few exact methods have been developed to solve. Habitually, although guaranteeing an optimum so methods have a problem size limitation, measured in terms of computing time; therefore, they can only be applied to problem instances with small or medium number of assembly tasks. Approximate methods have been developed in order to overcome such a limitation, and aiming at providing good solutions that are as near as possible of the optimal solution. In this article, GRASP (Greedy Random Adoptive Search Procedure) and

Genetic Algorithm meta-heuristic methods are applied for achieving approximate optimized solutions that provides solutions with good quality for this problem.

## 3. Definition of the Problem

The assembly line balancing problem in the state of multiple-alternative operations sequence can be state as follows: There exists a set of tasks for which several alternative assembly variants (also called assembly routes) are available; the tasks have to be assigned to a group of workstations. Each variant for each sub-assembly is represented by an individual sub-graph, which determines the required assembly/manufacturing tasks (hence the assembly variants may be defined by different and mutually exclusive sets of tasks) and the precedence relations among them. Furthermore, task processing times are considered to be dependent on the assembly sub-graph. Therefore, total processing time may vary from one assembly alternative to another.

Tasks processing times are generally considered to be fixed, however in many real applications this is not the case. For example, task times depend on the nature of the tasks, the skills of the operators and the reliability of the machines [7]. Furthermore, the duration of a task can be determined by the complexity of performing a given task considering the current state of the system; i.e., it depends on the processing sequence. For example, it gets more difficult (it requires more time) to decorate the fairing of a motorbike after they have already been assembled onto the motorbike than when they are unassembled. Taking these assumptions into account, two problems have to be solved simultaneously: the decision problem, to select one assembly sub-graph for each subassembly that allows alternatives; and the balancing problem, to assign the tasks to the workstations.

The sub-graph of figure 1 shows an example of a medium-scale from an assembly line balancing problem in the state of multiple-alternative operations sequence. This example, which is based on the precedence diagram of Kilbrid's benchmark problem, consists of 47 tasks and seven sub-graphs that represent the assembly alternatives for three sub-assemblies.

The first sub-assembly allows two assembly variants which are represented in the sub-graph of Figure 1 by sub-graphs S1 and S2; both alternatives involve tasks 1, 3, 5, 7 and 9. The second sub-assembly allows three assembly variants: sub-graphs S3 and S4 for tasks 20, 21 and 22, and S5 for tasks 46 and 47. For the last sub-assembly there also available two assembly variants, which are represented by sub-graphs S6 and S7 both involving tasks 42, 44 and 45. In this example, the combination of the assembly sub-graphs available for each subassembly, results in a total of 12 possible assembly variants that are allowed for this assembly process.($12 = 2 \times 3 \times 2$)

In order to make a more comprehensive definition of the sub-graph as an alternative precedence diagramming tool, two aspects need to be discussed. On the one hand, it is assumed that assembly alternatives do not overlap between each other; therefore, each alternative for each available subassembly is represented by a unique and independent precedence sub-graph. On the other hand, fictitious tasks, with nil processing time, are used to facilitate the representation of two sub-assemblies with processing alternatives that are consecutive (this case is represented in Figure 1 by the fictitious task α) [12].

## 4. Approximate Methods to Solve the Problem

Exact methods have a problem size limitation and can only be applied to solve small and medium scale problems. Although, in some cases mathematical programming models can provide the optimal solution to more realistic problems, the required computation time may be too large to be of practical use [12]. As previously discussed, the assembly line balancing problem in the state of multiple-alternative operations sequence is more difficult to solve optimally, comparing with the simple case which by nature is NP-hard, since the inherent decision problem to selects the assembly sub-graphs implies an even bigger computational effort. Therefore in this article two meta-heuristics methods (GRASP and GA) are proposed to solve the assembly line balancing problem in the state of multiple-alternative operations sequence, aiming at yielding reasonable solutions in a significantly small computing time.

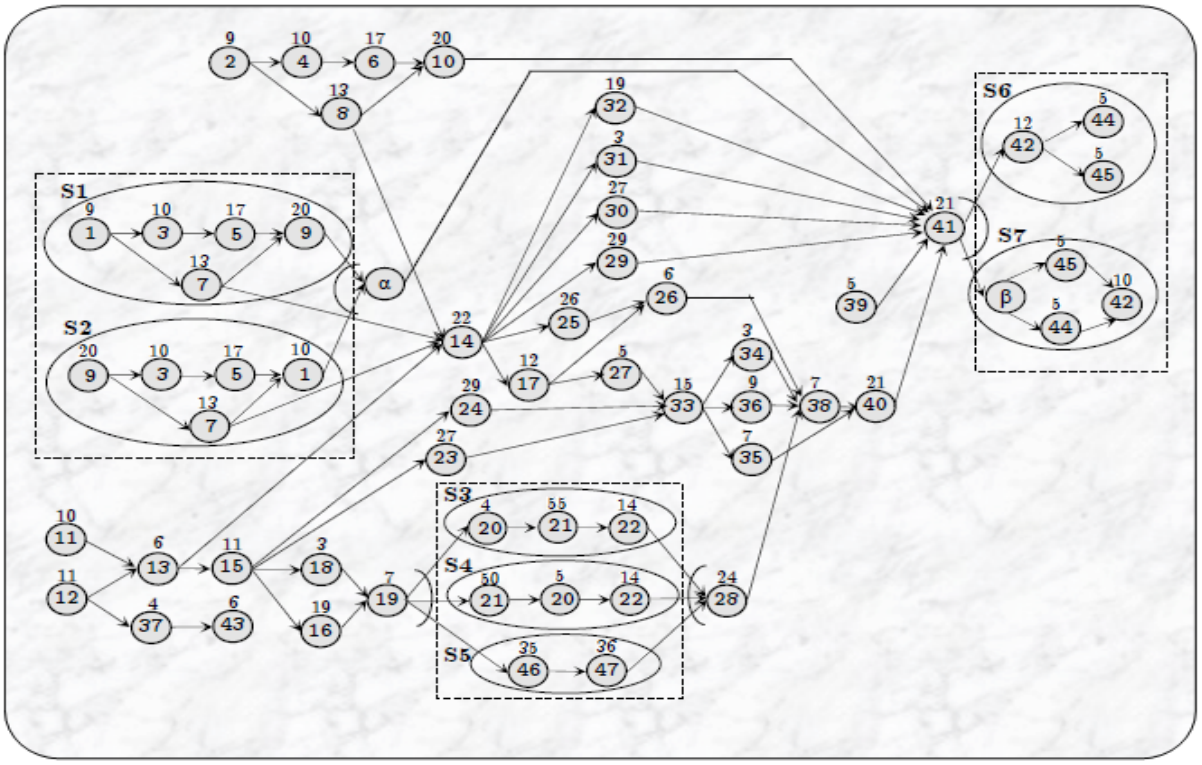


Figure 1: Precedence sub-graph for an example of 47 tasks.

### 4.1 GRASP

Is an iterative process in which each iteration consists of two phases: the construction phase, which generates an initial solution; and the improving phase, which uses a local optimization procedure to find a local optimum. The initial solution is generated by probabilistically selecting the next element to be incorporated in a partial solution from a restricted candidate list (RCL). The RCL is composed of the best elements considering a given greedy function [8]. It has been proven (e.g. [9]) that GRASP produces good quality solutions for hard combinatorial optimization problems, including line balancing problems. Andres et al. for example, proposed a GRASP procedure to solve a balancing and scheduling problem considering sequence-dependent setup times [10].

#### 4.1.1 Construction Method

The heuristic methods proposed in this article, systematically build the solution to the assembly line balancing problem in the state of multiple-alternative operations sequence by selecting the assembly sub-graphs and incrementally assigning the tasks to the workstations. Such methods use priority-rule-based and random strategies to select both the assembly sub-graphs and the next task to be assigned. In the former case, the selection is done considering a decreasing (or increasing) value of a predetermined priority rule; in the latter case, tasks and/or sub-graphs are selected following either a uniform distribution or a probability function based on weighted values of the priority rules. A solution provided by these methods consists of a set of sub-graphs (one for each subassembly that allows assembly variants), which determines the assembly tasks, the processing times, a number of required workstations and the assignment of the corresponding tasks to the workstations. In order to facilitate the evaluation of constructive methods involving most well-known priority rules, the proposed procedures aim at minimizing the number of workstations.

##### 4.1.1.1 Decision criteria for sub-graphs

In order to select the sub-graphs a priority rule are used as follows:
Minimum TT: sub-graphs are ranked according to ascending total processing time. Some of the other priority rules to select the sub-graphs are the number of precedence relations and the number of the task in each sub-graph.

#### 4.1.1.2 Decision criteria for tasks

The decision criteria used to select the next task to be assigned are presented in Table1. Priority rules values are basically determined by measuring task processing times and precedence relations, and by considering the cycle time. For instance, RPW (Rank Positional Weight) can be computed by adding to the task time the sum of the times of all its successors. Some of these criteria can be found in [11].

Table 1: Decision criteria for tasks

| No. | Name | Decision criteria | Procedure |
|---|---|---|---|
| 1 | RPW | $RPW_i = t_{\mathrm{i},\mathrm{sub(i)}} + \sum_{\mathrm{j}\in S_{\mathrm{i},\mathrm{sub(i)}}} t_{\mathrm{j},\mathrm{sub(j)}}$ | Maximum Rank Positional Weight |
| 2 | TTS | $TTS_{\mathrm{i}} = t_{\mathrm{i},\mathrm{sub(i)}} + \mathrm{TS_i}$ | Maximum Task Time plus Total Number of Successors |

The heuristic method proposed in this article, generate a single solution by exploring the solution space only via single priority rules, whereby the sub-graphs and tasks are selected according to the descendant or ascendant values of the predetermined priority rules. Tow Construction Method are obtained by combining the priority rules for tasks with the decision rules for the assembly sub-graphs; that come as follows:

- TT -RPW
- TT-TTS

TT-RPW selects a combination of sub-graphs that requires the minimum total processing time (TT) and ranks the tasks to be assigned considering descending Rank Positional Weight (RPW) values. TT-TTS selects a combination of sub-graphs that requires the minimum total processing time (TT) and ranks the tasks to be assigned considering descending the time of successors and the number of its successors.

### 4.1.2 Improving Method (Local Optimization Procedure)

A local optimization procedure based on a neighborhood search strategy, have been developed, aiming at improving the solution generated by the proposed approximate methods. At this point, it is valid to comment that a solution to the problem is represented by a sequence of tasks, which results from orderly assigning the tasks to the workstations. The local optimization procedures generate the neighborhood of the working sequence by using a transformation or exchange movement. Each exchange generates a neighbor sequence, then task are orderly assigned to the workstations resulting in a number of required workstation. If neighbor sequence improves sorted sequence, the neighbor sequence becomes the stored sequence. The local search ends when all feasible exchanges have been made for each task in working sequence, i.e., when all neighbors have been generated and evaluated. For the next iteration, the stored sequence is assigned to the working sequence. The whole procedure is repeated until a predetermined computing time has been completed. The final solution is the best of all solutions generated.
For example, consider the following initial sequence, that is obtained by applying one of the heuristic method:

$$\mathrm{IS} = 2\,,1\,,3\,,4\,,5\,,6\,,7\,,8\,,10\,,9\,,11$$

Tasks 2 and 3 since neither task 2 nor task 1 are predecessors of task 3, neither task 1 nor task 3 are successors of task 2 (i.e. precedence constraints are kept), and both tasks are assigned to different workstations (task 2 is assigned to workstation I and task 3 to workstation II). Therefore, one of the resulting neighbor sequences is illustrated in Figure 2.

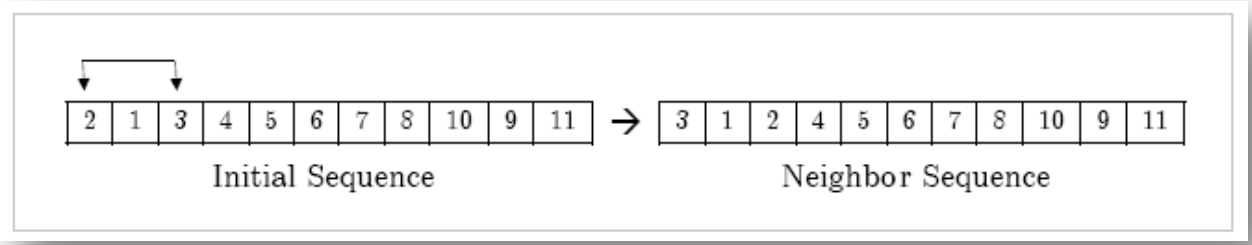


Figure2: Generation of a neighbor sequence

**4.2 Genetic Algorithm**

Genetic Algorithm; is another algorithms that has been used successfully in the problem of assembly line balancing. This algorithm is able to generate proper and good quality solutions, by using a fitness function on the basis of the cycle time.

## 5. Computational Experiment

To evaluate and compare the performance of the heuristic procedures described in the previous sections, a computational experiment was carried out.

The figure of 1 shows a numerical example of a medium-scale, which is based on the precedence diagram of Kilbrid's benchmark problem, consists of 47 tasks and the value of cycle time is 56.

**5.1 Analysis of the numerical results**

Tables 2 and 3 show the results obtained by applying Construction Method.

Table 2: The results obtained by applying TT-RPW Construction Method

| No. workstation | Assigned tasks | Idle time of workstation |
|---|---|---|
| 1 | 12, 11, 13 , 1 , 2 , 5 | 0 |
| 2 | 7 , 8 , 14 | 8 |
| 3 | 16 , 18 , 19 , 20 | 23 |
| 4 | 21 | 1 |
| 5 | 24 , 23 | 0 |
| 6 | 17 , 27 , 33 | 24 |
| 7 | 25 , 28 | 6 |
| 8 | 3 , 4 , 5 ,6 | 2 |
| 9 | 36 , 35 , 26 , 34 ,29 | 2 |
| 10 | 38 , 30 , 40 | 1 |
| 11 | 32 , 39 31 , 9 | 9 |
| 12 | 10 , 41 ,42 | 3 |
| 13 | 37 , 44 , 43 , 37 | 36 |

Table 3: The results obtained by applying TT-TTS Construction Method

| No. workstation | Assigned tasks | Idle time of workstation |
|---|---|---|
| 1 | 12, 11, 13 , 1 , 2 , 5 | 0 |
| 2 | 14 , 24 | 5 |
| 3 | 23 , 7 , 8 | 3 |
| 4 | 25 , 29 | 1 |
| 5 | 30 , 16 | 10 |
| 6 | 17 , 32 , 3 , 4 | 5 |
| 7 | 5 , 6 , 9 | 2 |
| 8 | 10 , 27 , 33 , 36 , 18 | 4 |
| 9 | 19 , 20 | 45 |
| 10 | 21 | 1 |
| 11 | 22 , 28 , 26 | 12 |
| 12 | 35 , 34 , 38 , 39 , 40 , 31 | 10 |
| 13 | 41 , 37 , 43 , 44 , 45 | 15 |

Table 4 show the results obtained by applying the proposed local optimization procedure, to improve the solutions provided by Construction Methods.

Table 4: The results obtained by applying the proposed local optimization procedures

| No. workstation | Assigned tasks | Idle time of workstation |
|---|---|---|
| 1 | 1 , 3 , 2 , 8 , 39 | 0 |
| 2 | 5 , 7 , 11 , 12 | 5 |
| 3 | 9 , 31 , 15 , 43 | 3 |
| 4 | 4 , 14 , 17 , 13 , 37 | 1 |
| 5 | 16 , 19 , 23 , 30 | 10 |
| 6 | 6 , 24 , 27 , 29 | 5 |
| 7 | 10 , 18 , 20 , 32 , 33 , 38 | 2 |
| 8 | 25 , 26 , 35 , 36 | 4 |
| 9 | 21 | 45 |
| 10 | 22 , 28 , 34 , 40 | 1 |
| 11 | 41 , 42 , 44 , 45 | 12 |

Table 5 show the results obtained by applying the Genetic Algorithm, to improve the solutions provided by Construction Methods.

Table 5: The results obtained by applying GA

| No. workstation | Assigned tasks | Idle time of workstation |
|---|---|---|
| 1 | 1 , 2 , 7 , 8 , 12 | 1 |
| 2 | 11 , 13 , 15 , 14 , 18 , 31 | 1 |
| 3 | 16 , 19 , 20 , 3 , 4 , 39 | 1 |
| 4 | 21 | 1 |
| 5 | 24 , 23 | 0 |
| 6 | 17 , 5 , 6 , 27 , 37 | 1 |
| 7 | 29 , 30 | 1 |
| 8 | 22 , 9 , 10 | 2 |
| 9 | 32 , 33 , 41 | 1 |
| 10 | 25 , 28 , 26 | 0 |
| 11 | 36 , 34 , 35 , 38 , 42 , 43 , 44 , 45 | 2 |
| 12 | 40 | 35 |

The presented results in tables 4 and 5 show that the utilization of the local optimization procedure and Genetic Algorithm cause to improvement of obtained solutions by applying construction methods. It means that fewer workstations are needed. In the local optimization procedure, two workstations and in the GA one workstation less than the construction methods are needed. This shows that the local optimization procedure is of a more adequate performance than the genetic algorithm.

## 6. Conclusions

Many real-life assembly line balancing problems involve assembly variants. Therefore, there is an increasing interest of addressing problems that consider assembly alternatives. In this article, a problem of the assembly line balancing in the state of multiple alternative operation sequence was presented. The core feature of such a problem is that it considers alternative variants for different parts of an assembly or manufacturing process. Each variant is represented by a precedence sub-graph that defines the tasks to be performed, their precedence relations and their corresponding processing times. Furthermore, mutually exclusive assembly processes, involving different sets of assembly tasks, are also considered. To solve this problem efficiently, two problems have to be solved simultaneously: (1) the decision problem to select the assembly alternative and (2) the balancing problem that assigns the tasks to the workstations. This problem implies a high level of difficulty since for the simple case it is verified the NP-hard condition. This combinatorial optimization problem thus required of the design and development of approximate methods to solve industrial-scale problems. Two heuristic methods to solve this problem were proposed in this article. Constructive methods based on priority rules have been successfully applied to assembly line balancing problems. In order to select the sub-graphs a priority rule was used (minimum TT) and two decision criteria used to select the next task to be assigned. In order to improve the solution of the proposed heuristic methods, a local optimization procedure was also proposed here, which is based on an adaptation of classical neighborhood search strategy. The analysis of the results showed that the utilization of the local optimization procedure and GA cause to improvement of obtained solutions by applying construction methods. In the local optimization procedure, two workstations and in the GA one workstation less than the construction methods are needed. This showed that the local optimization procedure is of a more adequate performance than the genetic algorithm.

## References


1. Salveson , M.E., 1954 , “Assembly Line Balancing Problem”, *Journal of Industrial Engineering* ,18-25.
2. Baybars, I. 1986, “A survey of exact algorithms for the simple assembly line balancing problem”, *Management Science*, 32, 909−932.
3. Ghosh, S., and Gagnon, R., 1989, “A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems,” *International Journal of Production Research*, 27, 637−670.
4. Erel, E. and Sarin, S.C., 1998, “A survey of the assembly line balancing procedures”, *Production Planning & Control*. 9, 414-434.
5. Rekiek, B., Dolgui., A., Delchambre., A. and Bratcu, A., 2002, “State of art of optimization methods for assembly line design”, *Annual Reviews in Control*, 26, 163-174.
6. Becker, C., and Scholl, A., 2004, “A survey on problems and methods in generalized assembly line balancing ”, *European Journal of Operational Research* , 1-11,
7. Rekiek, B., 2001, “Assembly Line Design, multiple objective grouping genetic algorithm and the balancing of mixed–model hybrid assembly line”, *Doctoral Thesis*, University Libre de Bruxelles.
8. Armentano, A., and Bassi, O., 2006, “Graph with Memory-based Mechanisms for Minimizing Total Tardiness in Single Machine Scheduling with Setup Times,” *Journal of Heuristics*, 12, 427−446.
9. Feo, T., Resende, M. and Smith, S., 1994, “A greedy randomized adaptive search procedure for maximum independent set,” *Operations Research*, 42, 860−878.
10. Andres, C., Miralles, C., and Pastor, R., 2006, “Balancing and sequencing tasks in assembly lines with sequence–dependent setup times”, *European Journal of Operational Research*.
11. Talbot, F.B, Patterson, J.H., and Gehrlein, W.V., 1986, “A comparative evaluation of heuristic line balancing techniques,” *Management Science*, 32, 431−453.
12. Capacho, L., and Pastor, R., 2005, “The Alternative Sub-graphs Assembly Line Balancing Problem”, *International Journal of Production Research*, 11, 1-8.