Solving the mixed-model assembly line balancing problem type-I using a Hybrid Reactive GRASP

Lakhdar Belkharrroubi and Khadjidja Yahyaoui

Department of Computer Science, University Mustapha Stambouli, Mascara, Algeria

ABSTRACT

One of the most recent challenges that manufacturers confront is to respond on time to the variety of customers’ demands for different products. The Assembly line is the main element responsible for assembling products in manufacturing systems, and it requires good management to avoid several problems that could lead to production failures. The mixed-model assembly line balancing problem type-I (MiMALBP-I) occurs in the step of designing a new assembly line in which different models of one product are assembled in an intermixed sequence; it aims to optimize the number of workstations for a fixed known cycle time. In this paper, the authors propose a Hybrid Reactive Greedy Randomized Adaptive Search Procedure (HRGRASP) to solve this problem. The performance of the proposed algorithm is compared with those of the basic GRASP, heuristics based approach, and the Lingo solver using seven problems of different sizes.

1. Introduction

Balancing an assembly line is a critical problem in manufacturing systems, it aims to assign a set of assembly tasks or operations into a set of workstations in order to optimize different performance measures (Fathi et al., 2018). Workstations can be arranged in a straight assembly line, U-shaped assembly line, or parallel assembly line (see, Figure 1), and to move work-pieces between these workstations a transportation system is used, for example, a conveyor belt or any mechanical material handling system. The time taken by each workstation to accomplish all assembly tasks required by the work-piece is called cycle time. The assignment of assembly tasks into workstations is restricted by precedence relations that are presented in a diagram known as precedence graph which demonstrates all relations between tasks (Saif et al., 2014).

The first assembly line balancing problem is known in literature by SALBP for Simple Assembly Line Balancing Problem that is related to assembly lines that produce a single product in mass production. The SALBP is classified into four problems: SALBP-I aims to minimize the number of workstations for a fixed cycle time, SALBP-II aims to minimize the cycle time for a fixed number of workstations, SALBP-E aims to maximize the efficiency of the line, and SALBP-F aims to obtain a feasible balance. In the SALBP, it
is assumed that mass production of one product is planned in a paced line. The layout of
the assembly line is straight with one-sided workstations. All workstations are equally
equipped with respect to machines and workers. The nature of processing times is
deterministic. No other constraints besides the precedence relations (Becker & Scholl,
2006).

Other assembly lines in manufacturing systems are used to produce different models
of one product known as the mixed-model assembly line (MiMAL) and multi-model
assembly line (MuMAL) (see Figure 2). The MiMAL produces different models in an
arbitrary intermixed sequence while the MuMAL produces models in batches, and each
batch contains more than one unit of only one model (Liao, 2014). The main objective of
applying these types of lines in manufacturing systems is to meet the diversity of
customers’ demands (Becker & Scholl, 2006).

Balancing a mixed-model assembly line is also a critical problem that takes into
consideration all constraints of different similar products to be produced in the same
line, and it is known in literature as the mixed-model assembly line balancing problem
(MiMALBP). The MiMALBP can be transformed into SALBP using two different
approaches: combining all precedence graphs of all models in one precedence graph
called combined precedence graph or using the adjusted task processing times (Van
Zante-de Fokkert & de Kok, 1997). Like the SALBP, the MiMALBP can be classified into
different problems, for example, MiMALBP-I and MiMALBP-II. The aim in the
MiMALBP-I is to minimize the number of workstations for a fixed cycle time, whereas
in the MiMALBP-II the aim is to minimize the cycle time for a fixed number of
workstations.
In assembly lines, the nature of processing times can be deterministic or stochastic. For example, in the case of manual assembly lines where tasks are performed in the assembly line by skilled workers using sophisticated tools, the variability in processing times is less and can be neglected which makes processing times more closer to being deterministic. The nature of processing times can be stochastic, for example, in automated assembly lines where tasks are performed by machines or robots, processing times can vary due to machine breakdowns. In some cases the variations in processing times may result from non-qualification of workers, lack of training, and motivation Sivasankaran and Shahabudeen (2014) Brahim and Alain (2006).

Due to its complexity, the assembly line balancing problem is classified as NP-hard problem, and obtaining optimal solutions in a reasonable time for large-size problems using exact approaches is impossible, thus in most papers that have dealt with the ALBP, heuristic or meta-heuristic algorithms have been used to find satisfying solutions in a reasonable time (Fathi et al., 2018). The heuristic algorithms can find optimal solutions for NP-hard problems, but in some cases, they can be trapped in the local optimal solution, for this reason, meta-heuristics algorithms are used because they include some mechanisms to avoid the local optima (Rabadi, 2016).

The problem addressed in this paper is the mixed-model assembly line balancing problem type-I (MiMALBP-I), and to solve it an enhancement of the greedy randomized adaptive search procedure (GRASP) called reactive GRASP is hybridized with the shortest processing time heuristic that is used as a greedy function. In comparison with other meta-heuristics that have been used to solve the MiMALBP-I, the proposed Hybrid Reactive GRASP integrates the learning mechanism which helps the algorithm to take good decisions in the subsequent iterations and can assist programmers/manufacturers in determining the best values of the alpha parameter for a specific MiMALBP-I problem to find other best solutions. Furthermore, this work represents the first attempt of using the reactive version of the GRASP to solve an ALBP problem, and gives an idea about how classical heuristics can be used in hybridization with this algorithm to help domain researchers. Finally, the authors have generated new MiMALPB- problems of different sizes as a supplementary contribution that can help researchers in their future works concerning the addressed problem.
This paper is organized as follows. In section 2, the authors mentioned similar works in the literature that have been done to solve MiMALBP-I. In section 3, the problem and its mathematical formulation are described. Section 4 describes the basic and the reactive versions of the GRASP. The proposed algorithm is described in section 5, computational results and discussion are presented in section 6, and finally, section 7 concludes the work.

2. Literature review

Several papers can be found in literature addressing the mixed-model assembly line balancing problem with the objective of minimizing the number of workstations. Thiruvady et al. (2020) have proposed an ant colony optimization approach which is based on the learning permutations of operations to solve the mixed-model assembly line balancing problem type-I with setups. The authors have compared the proposed approach with an existing exact approach based on integer programming, and the Bender’s decomposition approach (BDA). Obtained results showed that the proposed ACO approach is better than both approaches in terms of the run time and effectiveness in finding good solutions for small and large size problems. Yuan et al. (2015) have presented an effective hybrid honey bee mating optimization algorithm for the mixed-model two-sided assembly line balancing problem with the objective of minimizing the total number of workstations for a given cycle time, and they used a simulated annealing as workers to improve broods. To test the proposed HHBMO seven sets of problems have been used and experiment results showed that the algorithm can achieve good solutions in reasonable times. Furthermore, the authors have compared their HHBMO with a mixed integer programming and the simulated annealing for small-sized problems and obtained results showed that the HHBMO achieved the same results as MIP and SA in terms of the number of mated-stations. However, in terms of smoothness index, the HHBMO outperforms both MIP and SA. For large-sized problems, the HHBMO has been compared with SA and results showed that the HHBMO outperformed the SA in terms of the number of mated-stations and the smoothness index.

A mathematical model has been proposed by Yadav et al. (2020) to solve the mixed-model two-sided assembly line balancing problem with the objective of maximizing the workload at each workstation such that the number of mated workstations is minimized. The proposed methodology has been applied to resolve a case study and authors have noticed from obtained results that there was a significant improvement in the line efficiency and also the new reduced number of workstations has reduced the length of the assembly line which increases the space in the plant. Sadeghi et al. (2018) have proposed a method based on the Ranked positional weight and the Variable Neighborhood Descent for the mixed-model assembly line balancing problem in a real industrial context, the stitching systems of a footwear company. The aim was to optimize the number of workstations in the first part and smoothing the operators’ workload in the second part, and results showed that the proposed RPW-VND is an effective method and offers better solutions than those implemented by the company’s operations managers.
Delice et al. (2017) have proposed a modified particle swarm optimization algorithm with negative knowledge to solve the mixed-model two-sided assembly line balancing problem with the objective of minimizing the number of mated-workstations and workstations for a fixed cycle time, the authors have added some procedure to the proposed algorithm such as the generation procedure which is based on the combined selection mechanism and the decoding procedure. The proposed PSO algorithm has been compared with the Simulated Annealing, and results showed that both reached optimal solutions in small CPU times for the small-sized problems and both can find the same results. But the proposed PSO outperforms the SA in finding better solutions. Kucukkoc et al. (2018) have solved the mixed-model assembly two-sided assembly lines balancing problem with underground workstations by proposing its mathematical model and using CPLEX solver for small-size and medium-size problems, and for large-size problems, they developed an ant colony optimization algorithm. Obtained results proved that the proposed ACO can find optimum solutions for small and medium-sized problems and can find near-optimum solutions for large-sized problems in comparison with the lower bounds.

Rabbani et al. (2016) have presented in their research a non-dominated bi-objective genetic algorithm (NSGA-II) and a multi-objective particle swarm optimization (MOPSO) for solving the mixed-model U-shaped assembly line balancing problem considering human-related issues with three different objective functions: minimizing the number of workstations, minimizing the cycle time, and maximizing the line efficiencies. A comparison has been made between the two proposed algorithms and results showed that the NSGA-II is better than the MOPSO in solving most problems in terms of computational time and number of solutions found. Manavizadeh et al. (2013) have proposed a simulated annealing algorithm for a mixed model U-shaped line balancing problem with the objective of reducing the number of workstations taking into consideration human efficiency and the Just-In-Time approach. Five small-sized problems have been used to compare the performance of the proposed SA with that of the GAMS software, and results demonstrated that solutions obtained by the SA algorithm are consistent with those of the GAMS software. Also seven large-sized problems have been solved and results showed that the SA still have better reliability.

Mamun et al. (2012) have proposed a heuristic procedure based on a genetic algorithm for a mixed-model assembly line balancing problem in order to minimize the total number of workstations and allows the user to control the replication process. The procedure has been used to solve a case study of a plastic bag manufacturing company and has found a minimum number of workstations. Akpinar et al. (2013) have developed a hybrid algorithm which executes in combination ant colony optimization and genetic algorithm for mixed-model assembly line balancing problem type-I with zoning constraints and sequence-dependent setup times between tasks, and in order to evaluate the performance of the proposed hybrid ACO-GA, 20 mixed-model assembly line balancing problems. The performance of the proposed hybrid ACO-GA has been compared with the performances of pure ACO, pure GA, and hGA. From obtained results, the authors concluded that the ACO-GA can improve search performance and outperforms pure ACO, pure GA, and hGA. Rabbani et al. (2012) have developed an approach based on a genetic algorithm to minimize the number of crossover workstations to balance
a mixed-model U-shaped assembly line, and the computational result showed that the proposed genetic algorithm has been able to solve large-scale problems with reasonable CPU times.

Noorul Haq et al. (2006) have proposed a hybrid genetic algorithm approach for the mixed-model assembly line balancing problem with the objective of minimizing the number of workstations. In order to reduce the search space within the global space thereby reducing the search time, a modified ranked positional weight method has been used for the initial solution. The results showed that the proposed hybrid GA has a better performance than the classical genetic algorithm. Vilarinho and Simaria (2002) have presented a mathematical programming model for the mixed-model assembly line balancing problem with parallel workstations and zoning constraints. The first goal was to minimize the number of workstations and the secondary goal was to obtain a good workload balance between and within workstations, and to tackle the problem two-stage heuristic procedure has been used which uses the simulated annealing. A comparison has been made between results obtained by the proposed approach and those obtained by the Cplex solver and the lower bound (LB) for different size problems. The proposed approach performs very well and can find good quality solutions in short running times especially for large-sized problems in comparison with the lower bound and the Cplex solver. In Table 1, a set of papers that have been done to solve different versions of type 1 are presented.

3. Problem description and mathematical formulation

3.1. Problem description

A mixed-model assembly line is designed to produce $M$ models having similar characteristics, and for each model, the demand $d_m$ ($m \in M$) is known. The total demand $D_m$ which is the sum of all demands which must be completed in a period time $PT$. Each model $m$ has a precedence graph $G$, and all graphs can be combined in a single graph to transform the MiMALBP to SALBP. Each task $i$ in the combined graph must be assigned into one workstation, and the sum of all processing times of tasks assigned in the same workstation must not exceed the cycle time which is calculated using Equation 1:

$$C = \frac{PT}{D_m}$$

The addressed MiMALBP-I have the following assumptions, these assumptions are chosen by authors based on the characteristics of the basic version of the assembly line balancing problem:

1. Processing times of tasks are deterministic.
2. The cycle time is known and fixed.
3. The assembly line is a straight line.
4. All tasks must be assigned.
5. Common tasks between models are assigned to the same workstation.
6. Processing time of common tasks may differ from model to model.
7. The assignment of tasks is restricted by precedence constraints.
### 3.2. Mathematical formulation

- \( i \) task
- \( N \) number of tasks
- \( t_i \) processing time of task \( i \)
- \( C \) Cycle time
- \( w \) workstation
- \( W_{max} \) upper bound on the number of workstation
- \( P_i \) the set of predecessors of task \( i \)
- \( X_{iw} \) equal to 1 if task \( i \) is assigned to workstation \( w \), 0 otherwise
- \( y_w \) equal to 1 if any task is assigned to workstation \( w \), 0 otherwise

From the literature, to limit the search space, the upper bound is used to determine the maximal number of workstations. In this paper the upper bound is the number of tasks, thus \( W_{max} = UB = N \), this means that in the worst case, each workstation performs only one task. The objective function of the problem is given by Equation 2:

\[
\text{Minimize } \sum_{w=1}^{W_{max}} y_w
\]

Under the following constraints:

\[
\sum_{w=1}^{W_{max}} X_{iw} = 1 \quad \text{for } i = 1, 2, \ldots, N \tag{3}
\]

\[
\sum_{i=1}^{N} t_i * X_{iw} \leq C \quad \text{for } w = 1, 2, \ldots, W_{max} \tag{4}
\]

\[
\sum_{w=1}^{W_{max}} w * X_{hw} \leq \sum_{w=1}^{W_{max}} w * X_{iw} \quad \text{where } h \in P_i \tag{5}
\]

\[
y_w \in \{0, 1\} \tag{6}
\]

\[
X_{iw} \in \{0, 1\} \tag{7}
\]

Constraint (3) ensures that each task is assigned to only one workstation, constraint (4) ensures that the sum of all processing times of tasks assigned to the same workstation does not exceed the cycle time, constraint (5) imposes the precedence relations between tasks, constraint (6) defines the possible values of \( y_w \), constraint (7) defines the possible values of \( X_{iw} \).

### 4. Basic and reactive GRASP algorithm

The greedy randomized adaptive search procedure is a multi-start meta-heuristic that has been used to solve several combinatorial problems. Two phases are applied in each iteration of the GRASP: the construction phase and local search phase. The construction phase is used to construct a feasible solution but not necessarily the best one, thus the
local search phase tries to find better solution by applying a local search procedure on the constructed solution. Two lists are used in the construction phase, the candidate list (CL) that contains all candidates that can be added to the partial solution, and from this list best candidates are selected to form another list named the restricted candidate list (RCL). In the basic GRASP, the selection of the best candidates from the CL is based on the $\alpha$ parameter which is fixed before starting the execution. The $\alpha$ parameter is used in the following equation that represents the threshold value:

$$TC_{th} = T_{min} + \alpha(T_{max} - T_{min}) \quad (8)$$

where $T_{min}$ and $T_{max}$ are the smallest and largest incremental costs, respectively. All candidates having costs smaller than or equal to the threshold value are selected to form the RCL, this is the greedy part of the GRASP. The random part lies when choosing randomly from the RCL a candidate to be added to the partial solution Marti et al. (2018) Glover and Kochenberger (2003).

The disadvantage of the basic GRASP is that it does not learn from previous iterations because all information gathered about obtained solutions are discarded, and in some cases, using a fixed value of the $\alpha$ parameter can not help the GRASP to converge toward a global optimum (Marti et al., 2018). The reactive GRASP is the first enhancement of the basic version; it has been proposed for the first time by Prais and Ribeiro (2000) for the time slot assignment problem, and has been used afterwards for solving several optimization problems, for example, the Strip Packing problem (Alvarez-Valdes et al., 2008), Capacitated Clustering problem (Deng & Bard, 2011), and Vehicle Routing problem (Jaikishan & Patil, 2019). In each reactive GRASP iteration, instead of using a fixed value, the $\alpha$ parameter takes randomly its value from a discrete set of possible values $\alpha = \{\alpha_1, \ldots, \alpha_m\}$ using probabilities $P_i, i = 1, \ldots, m$, this probabilities depend on previous obtained solutions. At the first GRASP iteration, all possible values of $\alpha$ have the same probability of selection $P_i = 1/m$ where $m$ is the number of possible values of $\alpha$. At any subsequent iteration let $\hat{Z}$ be the best solution found, and let $A_i$ be the average value of all solutions found using $\alpha = \alpha_i$ where $i = 1, \ldots, m$. The selection probabilities are updated periodically using the following equation:

$$P_i = \frac{q_i}{\sum_{j=1}^{m} q_j} \quad (9)$$

where $q_i = \hat{Z}/A_i, i = 1, \ldots, m$. The value of $q_i$ is increased if the values of $\alpha = \alpha_i$ lead to the best solutions on average. The probabilities of appropriate values will then increase when they are updated (Glover & Kochenberger, 2003).

### 5. Proposed approach

In this paper the proposed hybrid Reactive GRASP is based on the structure of the original Reactive GRASP, and to adapt it to solve the mixed-model assembly line balancing problem type-I we integrated some modifications in the construction phase and the local search phase. The execution of the Hybrid Reactive GRASP is based on four principal steps: the construction phase, the local search phase, the evaluation of the obtained solution, and finally the update of selection probabilities in each period. The Reactive GRASP is chosen
by the authors of this paper to solve the MiMALBP-I for the following reasons. First, it is easy to understand how the Reactive GRASP works in comparison with other meta-heuristics that necessitate more steps and more complicated calculations. Second, we can easily make a hybridization in any GRASP stage using another heuristic/meta-heuristic. Finally, according to the literature, the GRASP has been used to solve a variety of optimization problems and obtained results proved that is an efficient meta-heuristic.

5.1. Adopted construction phase

The initial solution is found by the construction phase and during the construction, a greedy heuristic that selects first the best elements is used, the selection of elements is based on the cost which differs from a problem to another one. In the proposed hybrid Reactive GRASP, we consider the processing time of the task as cost and we use the Shortest Processing Time heuristic as a greedy heuristic in the construction phase, thus, by using this heuristic tasks that have the shortest processing times are considered like the best elements. By Hybridizing the Reactive GRASP with the Shortest Processing Time heuristic, we can call this algorithm Hybrid Reactive GRASP-SPT. Figure 3 below shows the flowchart of the adopted construction phase.

As shown in Figure 3, the proposed construction phase is based on seven steps:

1. **Step 01:** all tasks are evaluated by their processing times.
2. **Step 02:** assignable tasks are chosen from the list of available tasks to form the candidate list. An assignable task is a task that has not a predecessor or all its predecessors are already assigned.
3. **Step 03:** the threshold value is calculated based on processing times of tasks that exist in the candidate list using Equation 8 showed in section 4.
4. **Step 04:** each task that has a processing time smaller than or equal to the threshold value is chosen to be in the restricted candidate list (RCL).
5. **Step 05:** from the restricted candidate list a task is chosen randomly.
6. **Step 06:** the chosen task is added to the partial solution.
7. **Step 07:** Update the candidate list by deleting the assigned task and adding new candidates (assignable tasks).

5.2. Adopted local search phase

Obtained solution by using the construction phase maybe not the optimal one, thus, a local search is applied starting from the obtained solution to determine if there exists a neighbor solution with a better objective value. In this proposed reactive GRASP-SPT we use a simple local search that searches for neighbor solutions for a fixed max_search number, and to search locally, two tasks that are not related by a precedence relation are swapped randomly, this means if tasks i and j are chosen randomly to be swapped, and task i is positioned before task j in the sequence, the algorithm verifies first if the new solution will respect the precedence relations.

As shown in Figure 4, the local search phase is based on four principal steps:

1. **Step 01:** After the initialization of the counter, a neighbor solution is generated using the random swap function. If the neighbor has not been found in previous iterations go to step 2, otherwise go to step 3.
Figure 3. Adopted construction phase.

Figure 4. Adopted local search phase.
(2) **Step 02**: The generated neighbor is added to a list of neighbors (LN).

(3) **Step 03**: If the generated neighbor already exists in the list of neighbors, the counter is incremented by 1. Then if the counter is less than or equal to the max_search number go to step 1 to continue the process of searching a new neighbor, otherwise if there exist in the list of neighbors a neighbor that has a better objective value go to step 4, else stop the local search.

(4) **Step 04**: Replace the initial solution by the new better neighbor solution and reset the counter to 1.

### 5.3. Evaluation of the obtained solution

The algorithm compares the new solution obtained by the local search phase with the solution that is chosen previously as the best solution found, if the new solution has a better objective value (gives a minimum number of workstations) it becomes the best solution found to be passed to the next iteration.

The evaluation of solutions is based on five steps as shown in **Figure 5**:

(1) **Step 01**: Initialization of the number of workstations with 0.

(2) **Step 02**: Open a new workstation and set its operational time with the fixed known cycle time.

(3) **Step 03**: Increment the number of workstations with 1 after each opening operation.

(4) **Step 04**: Assign the selected task to the current workstation if it has a processing time (task time) less than or equal to the operational time of the current workstation and update the operational time by decreasing from it the processing time of the assigned task, otherwise go to step 02 to open a new workstation.

![Figure 5. The proposed evaluation algorithm.](image-url)
(5) **Step 05:** If all tasks in the sequence are not assigned, select the next task and continue the process, otherwise the evaluation process stops.

### 5.4. Update of selection probabilities

In each iteration, the reactive GRASP calculates for each alpha value the number of solutions found and the sum of obtained objective values. Using this information, in each period the algorithm updates the selection probabilities for each alpha value using Equation 9 showed in section 4. This learning mechanism helps the algorithm to determine from the selection probabilities which alpha value has the highest chance to be selected and obtain good solutions as a result.

### 6. Computational results and discussion

The proposed hybrid Reactive GRASP is developed using Python on a personal computer with an Intel Core i3-4005 U CPU, 1.70 GHz, and 6GB memory, and to test its performance, 6 categories of problems have been generated randomly. Table A1 represents a small size problem with 2 models A and B, 10 tasks and 10 precedence relation in combination. The total operational time needed to accomplish model A is 14 and 12 for model B. The number of demands for model A is 24 and 22 for model B, thus, 46 in total. By dividing the given period time on the total number of demands we obtain a cycle time = 4.1. In the second small size problem Table A2 there are two models A and B to be assembled in the same line, 12 tasks in total and 14 precedence relation in combination. The operational time to accomplish model A is 17 and 14 for model B. The number of demands for model A and B is 30, thus, 60 in total. The obtained cycle time is 5 according to the total number of demands and the given period time (PT = 300). Table A3 represents a medium size MiMALB problem with two models A and B, 15 tasks and 15 precedence relation in combination. Models A need an operational time equal to 16 to be accomplish, and for model B the needs 20. In a period time = 150, 25 units of model A and 15 units of model B must be assembled which requires a cycle time = 4.3. The second medium size problem Table A4 has 3 models A, B and C, these models need respectively 21, 19 and 20 units of time to be accomplished. The number of tasks is 15 and there are 17 precedence relation in total. The number of demands is 35 for model A, 35 for model B and 38 for model C. The obtained cycle time according to the problem parameters (PT = 480, Total number of demands = 108) is 4.4. Table A5 represents a large size problem with 3 models A,B,C and 22 tasks with 25 precedence relation related in combination. Model A needs an operational time = 27, model B needs 31 and model C needs 30. The number of demands for model A is 25, for model B is 35 and 25 for model C. These demands must finished in a period time = 480 which needs a cycle time = 5.6. In the second large size problem Table A6, there are 4 models (A, B, C and D) that can be assembled in the same line. The overall number of tasks is 25 with 41 precedence relation in combination. Models A, B, D, and C need 22, 25, 26 and 26 units of time, respectively, to be accomplished. The number of demands for model A is 40, for model B is 15, for model C is 15 and 20 for model D. All these demands must be finished in a period time = 700 which needs a cycle time = 7.8. Table A7, represents a very large problem with 2 models A and B. The number of tasks is 44 and there are 45
precedence relations. To accomplish model A, 66 time units is needed, and model B requires 63 time units. The number of demands for model A is 50 while for model B is 45, thus, 95 in total. The given period time is 500 time units, and by dividing this value by the total number of demands, the cycle time will be 5.26. This problem is used to evaluate the performance of the proposed hybrid Reactive GRASP with that of the LINGO solver. tab: Table 2 shows all parameters used to solve each problem using the hybrid reactive GRASP.

Obtained results by the hybrid reactive GRASP are compared with those obtained by the basic GRASP which uses a fixed alpha value in all iterations and we propose another approach based on three heuristics. The basic GRASP has been used in (Bautista et al., 2000) and (Chica et al., 2010) to solve the assembly line balancing problem to solve the assembly line balancing problem. The proposed approach is based on three heuristics taken from (Boctor, 1995): the Ranked positional weight, the greatest number of successors, and the longest processing time, so we call it RWP-NS-LT. This proposed approach works as follows: the task that has the greatest positional weight has the highest priority to be assigned to the workstation. In the case of two tasks that have the same positional weight, we compare their numbers of successors and the task which has the greatest number of successors is the first assigned, and if tasks have the same number of successors, their processing times are compared, and which has the highest processing time has the highest priority to be assigned. Finally, if two tasks have the same processing time, the RWP-NS-LT chooses one from them at random. In the basic GRASP, for all proposed problems the fixed alpha value used is 0.

Table 3 shows the number of obtained solutions by each alpha value used to solve each problem, in most cases when alpha α=1 the search space is extended and the algorithm can find more solutions. Some solutions obtained by different alpha values are similar, and some solutions obtained when α=1 cannot be found when α=0 or α=0.5, but as we

**Table 1. Contributions of related works.**

| Paper                  | MiMALBP Type       | Line configuration          | Used approach     |
|------------------------|--------------------|-----------------------------|-------------------|
| Thiruvady et al. 2020  | Type-I             | Straight line               | ACO               |
| Yuan et al. (2015)     | Type-I             | Two-sided line              | Hybrid HBMO       |
| Yadav et al. (2020)    | Type-I             | Two-sided line              | Mathematical Model|
| Sadeghi et al. (2018)  | Type-I             | Straight line               | RPW-VND           |
| Delice et al. (2017)   | Type-I             | Two-sided line              | PSO               |
| Kucukkoc et al. (2018) | Type-I             | Two-sided line              | Mathematical model|
| Rabbani et al. (2012)  | Type-I and Type-II | U-shaped line               | Multi-objective GA|
| Manavizadeh et al. (2013)| Type-I           | U-shaped line               | ACO               |
| Mamun et al. (2012)    | Type-I             | Straight line, Parallel workstation | Heuristic based GA |
| Akpinar et al. (2013)  | Type-I             | Straight line, Parallel workstation | Hybrid ACO-GA |
| Rabbani et al. (2012)  | Type-I             | U-shaped line               | GA                |
| Noorul Haq et al. (2006)| Type-I            | Straight line               | GA                |
| Vilarinho and Simaria (2002)| Type-I          | Straight line, Parallel workstation | Mathematical model |

**Table 2. Used parameters in the hybrid reactive GRASP for all problems.**

|               | Max iterations | Max search | period | alpha values |
|---------------|----------------|------------|--------|--------------|
| small size problem 1 | 1000          | 30         | 100    | 0, 0.5, 1    |
| small size problem 2 | 1000          | 30         | 100    | 0, 0.5, 1    |
| medium size problem 3 | 1500          | 40         | 100    | 0, 0.5, 1    |
| medium size problem 4 | 1500          | 40         | 100    | 0, 0.75, 1   |
| large size problem 5  | 2000          | 40         | 100    | 0, 0.75, 1   |
| large size problem 6  | 2000          | 40         | 100    | 0, 0.45, 1   |
| very large size problem 7 | 500           | 10         | 100    | 0, 0.5, 1    |
can see, the number of obtained solutions by 0 alpha value in solving the large size problem 6 is higher than numbers obtained by α=0.45 and α=, so we can conclude that when the alpha value increases the algorithm can find more solutions.

Figure 6 shows the probability of selection for each alpha during solving the small size problem 1, and we can observe that there was not any variation from the first period to the final one, this means that all alpha values have the same chance to obtain solutions having the same objective values. Figure 7 and Figure 8 show that while solving the small size problem 2 and medium-size problem 3, the selection probabilities of α=0 and α=0.5 have the same variation during all periods in comparison with α=1, and they increase which means that when alpha takes 0 and 0.5 as values the algorithm can find good solutions which have same objective values (numbers of workstations). When α=1 the algorithm can find good and bad solutions which increases the total average of found objective values which decreases the probability of selection of α=1.

Figure 9 and Figure 10 show the variation of the selection probabilities of used alpha values while solving medium size problem 4 and large size problem 5 respectively, and we can observe that the variations of selection probabilities of α=0 and α=0.75 values are

| Table 3. Numbers of solutions found by the hybrid Reactive GRASP for each problem. |
|---------------------------------|-------------------|-----------------|-----------------|-------------------|
| small size problem 1           | α = 0             | α = 0.5         | α = 1           | total number of solutions |
|                                 | 56                | 58              | 247             | 361               |
| small size problem 2           | α = 0             | α = 0.5         | α = 1           | total number of solutions |
|                                 | 84                | 194             | 310             | 588               |
| medium size problem 3          | α = 0             | α = 0.5         | α = 1           | total number of solutions |
|                                 | 111               | 106             | 494             | 711               |
| medium size problem 4          | α = 0             | α = 0.75        | α = 1           | total number of solutions |
|                                 | 8                 | 128             | 469             | 605               |
| large size problem 5           | α = 0             | α = 0.75        | α = 1           | total number of solutions |
|                                 | 48                | 646             | 671             | 1365              |
| large size problem 6           | α = 0             | α = 0.45        | α = 1           | total number of solutions |
|                                 | 689               | 652             | 624             | 1965              |

Figure 6. Variation of selection probabilities while solving problem 1.

Figure 7. Variation of selection probabilities while solving problem 2.
similar and became better than the probability of $\alpha=1$ from the second period to the last one. Figure 11 shows that the probability of selection of $\alpha=0$ is the highest one in comparison with $\alpha=0.45$ and $\alpha=1$, and this means that the best alpha value that helps the algorithm to find optimal solutions in solving the large size problem 6 is 0.

Figure 12 and Figure 13 show six comparisons of obtained numbers of workstations by each algorithm after solving proposed problems. We can see in comparisons (a) and (f) that after solving problems 1 and 6, all algorithms obtained the same number of
Comparisons (b) and (d) show that the hybrid Reactive GRASP and the basic GRASP have an advantage in comparison with the proposed RWP-NS-LT in solving problems 2 and 4 respectively. Comparisons (c) and (e) show that best values are found only by the hybrid Reactive GRASP in comparison with the basic GRASP and the proposed RWP-NS-LT after solving problems 3 and 5. So we can conclude from all comparisons that the proposed hybrid Reactive GRASP had an advantage by obtaining optimal solutions for all proposed size problems in comparison with the basic GRASP and the proposed RWP-NS-LT. Found solutions (assignments of tasks into workstations) by the hybrid reactive GRASP for problems (Table A1, Table A2, Table A3, Table A4, Table A5, Table A6) are shown in tables (Table 4, Table 5, Table 6, Table 7, Table 8, Table 9) respectively.

**Table 4. Assignment of tasks of problem 1.**

| Wr1 | Wr2 | Wr3 | Wr4 | Wr5 |
|-----|-----|-----|-----|-----|
| 1, 3 | 5, 6, 9 | 2, 4 | 7, 8 | 10 |

**Table 5. Assignment of tasks of problem 2.**

| Wr1 | Wr2 | Wr3 | Wr4 |
|-----|-----|-----|-----|
| 3, 2, 1, 6 | 5, 9, 4 | 7, 8, 10 | 11, 12 |
Table 6. Assignment of tasks of problem 3.

| Wr1 | Wr2 | Wr3 | Wr4 | Wr5 | Wr6 |
|-----|-----|-----|-----|-----|-----|
| 3, 2, 1 | 4, 7, 9 | 5, 6 | 8, 10 | 11, 12 | 13, 14, 15 |

Table 7. Assignment of tasks of problem 4.

| Wr1 | Wr2 | Wr3 | Wr4 | Wr5 | Wr6 | Wr7 |
|-----|-----|-----|-----|-----|-----|-----|
| 8, 4, 1 | 2 | 3, 9 | 5, 6 | 7, 11 | 10, 12 | 14, 13, 15 |

Table 8. Assignment of tasks of problem 5.

| Wr1 | Wr2 | Wr3 | Wr4 | Wr5 | Wr6 | Wr7 | Wr8 |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 2,7, 5,1 | 3, 4, 9 | 13,14 | 15,10 | 16,19, 20 | 6,8 | 11,12 | 17,18, 21,22 |

Table 9. Assignment of tasks of problem 6.

| Wr1 | Wr2 | Wr3 | Wr4 | Wr5 | Wr6 | Wr7 | Wr8 |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 6,1,5, 2,4,10 | 3,12,9, 7 | 14,13,11, 8 | 16,15,19 | 17,21 | 23,18,24, 20 | 22,25 |

Table 10. Comparison of taken CPU time for solving problem 7 by the Hybrid Reactive GRASP and LINGO solver.

| Attempts | Hybrid Reactive GRASP (CPU time/Number of workstations) | LINGO solver (CPU time/Number of workstations) |
|----------|--------------------------------------------------------|---------------------------------------------|
| Attempt 1 | 10 min: 18 sec/16 | 29 min: 46 sec/16 |
| Attempt 2 | 14 min: 21 sec/16 | 21 min: 41 sec/16 |
| Attempt 3 | 16 min: 47 sec/16 | 45 min: 09 sec/16 |
| Attempt 4 | 10 min: 22 sec/16 | 28 min: 41 sec/16 |
| Attempt 5 | 10 min: 27 sec/16 | 37 min: 31 sec/16 |
| Attempt 6 | 10 min: 14 sec/16 | 23 min: 50 sec/16 |

Table 10 below shows the difference between the proposed Hybrid Reactive GRASP and the LINGO solver in terms of taken CPU time for solving the very large problem 7. By analyzing obtained results for all attempts, we can clearly conclude that the Hybrid Reactive GRASP can find the optimal number of workstations (w = 16) in shortest time in comparison with the LINGO solver.

7. Conclusion

The mixed-model assembly line is designed for producing different models of one product to meet customer’s demands in time, and finding the optimal number of workstations to maximize the workload in each workstation for each model is a complex problem. In this paper, a hybrid Reactive GRASP meta-heuristic was proposed for
solving the straight mixed-model assembly line balancing problem with the objective of minimizing the number of workstations. The Shortest processing time heuristic is used in the construction phase in order to determine the priority rule and evaluate tasks by their processing times, and a simple local search procedure that searches for a better neighbor solution is used in the local search phase of the algorithm. Results after solving all proposed problems using the hybrid Reactive GRASP, the basic GRASP and the RPW- SN-LT, show that the hybrid Reactive GRASP found optimal solutions for all proposed problems, also obtained results by the hybrid Reactive GRASP confirms that using a set of alpha values helps the algorithm to be more efficient by finding more solutions than using a fixed alpha value and can escape from the local optimum trap. Furthermore, in comparison with the LINGO solver, the proposed Reactive GRASP has spent less time for solving the proposed very large problem.

In the future, the proposed Hybrid Reactive GRASP can be used to solve other Assembly Line Balancing problems such as the Mixed Model Robotic ALBP type-I/type-II, Mixed Model Two-sided ALBP type-I/type-II, Mixed Model U-shaped ALBP type-I/type-II with other nature of the processing time (stochastic processing time), and other constraints (zoning constraints) including or not parallel workstations. The Reactive GRASP can be hybridized easily by using another classic heuristic (Ranked Positional Weight, Maximum Task Time, … etc) or meta-heuristic (Genetic Algorithm, Simulated Annealing, … etc) in any of these stages (the principal part, construction phase, local search phase).

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Lakhdar Belkharroubi https://orcid.org/0000-0002-5509-3270
Khadijja Yahyaoui https://orcid.org/0000-0001-9541-4879

References

Akpinar, S., Mirac Bayhan, G., & Baykasoglu, A. (2013). Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. Applied Soft Computing, 13(1), 574–589. https://doi.org/10.1016/j.asoc.2012.07.024
Alvarez-Valdes, R., Parreño, F., & Tamarit, J. M. (2008). Reactive GRASP for the strip-packing problem. Computers & Operations Research, 35(4), 1065–1083. https://doi.org/10.1016/j.cor.2006.07.004
Bautista, J., Suarez, R., Mateo, M., & Companys, R. (2000). Local search heuristics for the assembly line balancing problem with incompatibilities between tasks. Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia Proceedings (Cat. No. 00CH37065), 3, 2404–2409. IEEE. https://doi.org/10.1109/ROBOT.2000.846387
Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research, 168(3), 694–715. https://doi.org/10.1016/j.ejor.2004.07.023
Boctor, F. F. (1995). A multiple-rule heuristic for assembly line balancing. *The Journal of the Operational Research Society*, 46(1), 62. https://doi.org/10.1057/jors.1995.7

Brahim, R., & Alain, D. (2006). *Assembly line design*. Springer-Verlag. https://doi.org/10.1007/b138846

Chica, M., Cordón, O., Damas, S., & Bautista, J. (2010). A multiobjective GRASP for the 1/3 variant of the time and space assembly line balancing problem. In N. García-Pedrajas, F. Herrera, C. Fyfe, J. M. Benítez, & M. Ali (Eds.), *Trends in applied intelligent systems* (Vol. 6098, pp. 656–665). Springer. https://doi.org/10.1007/978-3-642-13033-5_67

Delice, Y., Kizilkaya Aydoğan, E., Özcan, U., & İlkay, M. S. (2017). A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of Intelligent Manufacturing*, 28(1), 23–36. https://doi.org/10.1007/s10845-014-0959-7

Deng, Y., & Bard, J. F. (2011). A reactive GRASP with path relinking for capacitated clustering. *Journal of Heuristics*, 17(2), 119–152. https://doi.org/10.1007/s10732-010-9129-z

Fathi, M., Fontes, D. B. M., Urenda Moris, M., & Ghoakhloo, M. (2018). Assembly line balancing problem: A comparative evaluation of heuristics and a computational assessment of objectives. *Journal of Modelling in Management*, 13(2), 455–474. https://doi.org/10.1108/JM2013-02-2017-0027

Glover, F., & Kochenberger, G. A. (Eds.). (2003). *Handbook of metaheuristics*. Kluwer Academic Publishers.

Jaikishan, T. S., & Patil, R. (2019). A reactive grasp heuristic algorithm for vehicle routing problem with release date and due date incurring inventory holding cost and tardiness cost. 2019 *IEEE international conference on industrial engineering and engineering management* (IEEM), Macao, China: IEEE, 1393–1397. https://doi.org/10.1109/IEEM4572.2019.8978851

Kucukkoc, I., Li, Z., Karaoglan, A. D., & Zhang, D. Z. (2018). Balancing of mixed-model two-sided assembly lines with underground workstations: A mathematical model and ant colony optimization algorithm. *International Journal of Production Economics*, 205, 228–243. https://doi.org/10.1016/j.ijpe.2018.08.009

Liao, L.-M. (2014). Construction and comparison of multi-model and mixed-model assembly lines balancing problems with bi-objective. *Journal of Industrial and Production Engineering*, 31(8), 483–490. https://doi.org/10.1080/21681015.2014.992984

Mamun, A. A., Khaled, A. A., Ali, S. M., & Chowdhury, M. M. (2012). A heuristic approach for balancing mixed-model assembly line of type I using genetic algorithm. *International Journal of Production Research*, 50(18), 5106–5116. https://doi.org/10.1080/00207543.2011.643830

Manavizadeh, N., Hosseini, N., Rabbani, M., & Jolai, F. (2013). A simulated annealing algorithm for a mixed model assembly U-line balancing type-1 problem considering human efficiency and Just-In-Time approach. *Computers & Industrial Engineering*, 64(2), 669–685. https://doi.org/10.1016/j.cie.2012.11.010

Martí, R., Pardalos, P. M., & Resende, M. G. C. (Eds.). (2018). *Handbook of heuristics*. Springer International Publishing. https://doi.org/10.1007/978-3-319-07124-4

Noorul Haq, A., Rengarajan, K., & Jayaprakash, J. (2006). A hybrid genetic algorithm approach to mixed-model assembly line balancing. *The International Journal of Advanced Manufacturing Technology*, 28(3–4), 337–341. https://doi.org/10.1007/s00170-004-2373-3

Prais, M., & Ribeiro, C. C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12(3), 164–176. https://doi.org/10.1287/ijoc.12.3.164.12639

Rabadi, G. (Ed.). (2016). *Heuristics, metaheuristics and approximate methods in planning and scheduling* (Vol. 236). Springer International Publishing. https://doi.org/10.1007/978-3-319-26024-2

Rabbani, M., Kazemi, S. M., & Manavizadeh, N. (2012). Mixed model U-line balancing type-1 problem: A new approach. *Journal of Manufacturing Systems*, 31(2), 131–138. https://doi.org/10.1016/j.jmsy.2012.02.002
Rabbani, M., Montazeri, M., Farrokhi-Asl, H., & Rafiei, H. (2016). A multi-objective genetic algorithm for a mixed-model assembly U-line balancing type-I problem considering human-related issues, training, and learning. *Journal of Industrial Engineering International, 12*(4), 485–497. https://doi.org/10.1007/s40092-016-0158-6

Sadeghi, P., Rebelo, R. D., & Ferreira, J. S. (2018). Balancing mixed-model assembly systems in the footwear industry with a variable neighbourhood descent method. *Computers & Industrial Engineering, 121*, 161–176. https://doi.org/10.1016/j.cie.2018.05.020

Saif, U., Guan, Z., Wang, B., Mirza, J., & Huang, S. (2014). A survey on assembly lines and its types. *Frontiers of Mechanical Engineering, 9*(2), 95–105. https://doi.org/10.1007/s11465-014-0302-1

Sivasankaran, P., & Shahabudeen, P. (2014). Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology, 73*(9–12), 1665–1694. https://doi.org/10.1007/s00170-014-5944-y

Thiruvady, D., Nazari, A., & Elmi, A. (2020). An ant colony optimisation based heuristic for mixed-model assembly line balancing with setups. *2020 IEEE Congress on evolutionary computation (CEC) Glasgow, UK* (IEEE), 1–8. https://doi.org/10.1109/CEC48606.2020.9185757

van Zante-de Fokkert, J. I., & de Kok, T. (1997). The mixed and multi model line balancing problem: A comparison. *European Journal of Operational Research, 100*(3), 399–412. https://doi.org/10.1016/S0377-2217(96)00162-2

Vilarinho, P. M., & Simaria, A. S. (2002). A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research, 40*(6), 1405–1420. https://doi.org/10.1080/00207540110116273

Yadav, A., Verma, P., & Agrawal, S. (2020). Mixed model two sided assembly line balancing problem: An exact solution approach. *International Journal of System Assurance Engineering and Management, 11*(S2), 335–348. https://doi.org/10.1007/s13198-020-00956-1

Yuan, B., Zhang, C., Shao, X., & Jiang, Z. (2015)0004. An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines. *Computers & Operations Research, 53*, 32–41. https://doi.org/10.1016/j.cor.2014.07.011

### Appendix A. Proposed problems

**Table A1. Problem 1.**

| Task | Model A | Model B | Task time | Predecessors |
|------|---------|---------|-----------|--------------|
| 1    | 1.0     | 1.0     | 1.0       | –            |
| 2    | 2.0     | 0.0     | 2.0       | –            |
| 3    | 0.0     | 2.0     | 2.0       | –            |
| 4    | 3.0     | 1.0     | 2.0       | 1, 2         |
| 5    | 2.0     | 2.0     | 2.0       | –            |
| 6    | 1.0     | 1.0     | 1.0       | 3, 5         |
| 7    | 2.0     | 0.0     | 2.0       | 4            |
| 8    | 0.0     | 2.0     | 2.0       | 4            |
| 9    | 1.0     | 1.0     | 1.0       | 6            |
| 10   | 2.0     | 2.0     | 2.0       | 7, 8, 9      |

Demands of models: DA = 24, DB = 22
Period time = 190
Cycle time = 190/46 = 4.1
Table 12. Problem 2.

| Task | Model A | Model B | Task time | Predecessors |
|------|---------|---------|-----------|--------------|
| 1    | 1.0     | 1.0     | 1.0       | –            |
| 2    | 1.0     | 1.0     | 1.0       | –            |
| 3    | 1.0     | 1.0     | 1.0       | –            |
| 4    | 2.0     | 2.0     | 2.0       | 1            |
| 5    | 3.0     | 1.0     | 2.0       | 2            |
| 6    | 1.0     | 3.0     | 2.0       | 3            |
| 7    | 0.0     | 1.0     | 1.0       | 4            |
| 8    | 2.0     | 0.0     | 2.0       | 4            |
| 9    | 0.0     | 1.0     | 1.0       | 5, 6         |
| 10   | 2.0     | 2.0     | 2.0       | 7, 8, 5      |
| 11   | 3.0     | 0.0     | 3.0       | 6            |
| 12   | 1.0     | 1.0     | 1.0       | 9, 10, 11    |

Demands of models: DA = 30, DB = 30  
Period time = 300  
Cycle time = 300/60 = 5

Table 13. Problem 3.

| Task | Model A | Model B | Task time | Predecessors |
|------|---------|---------|-----------|--------------|
| 1    | 1.0     | 2.0     | 1.0       | –            |
| 2    | 2.0     | 0.0     | 1.0       | –            |
| 3    | 2.0     | 0.0     | 1.0       | –            |
| 4    | 0.0     | 1.0     | 2.0       | –            |
| 5    | 0.0     | 3.0     | 2.0       | –            |
| 6    | 0.0     | 1.0     | 2.0       | –            |
| 7    | 2.0     | 1.0     | 1.0       | 1, 2, 4      |
| 8    | 1.0     | 2.0     | 2.0       | 3, 5, 6      |
| 9    | 3.0     | 2.0     | 1.0       | 7            |
| 10   | 1.0     | 1.0     | 2.0       | 8            |
| 11   | 0.0     | 1.0     | 3.0       | 10           |
| 12   | 1.0     | 1.0     | 1.0       | 9, 11        |
| 13   | 0.0     | 2.0     | 1.0       | 12           |
| 14   | 1.0     | 1.0     | 1.0       | 12           |
| 15   | 2.0     | 2.0     | 1.0       | 13, 14       |

Demands of models: DA = 20, DB = 15  
Period time = 150  
Cycle time = 150/35 = 4.3
Table 14. Problem 4.

Table A4. Problem 4.

| Task | Model A | Model B | Model C | Task time | Predecessors |
|------|---------|---------|---------|-----------|--------------|
| 1    | 1.0     | 1.0     | 1.0     | 1.0       | –            |
| 2    | 0.0     | 0.0     | 3.0     | 3.0       | –            |
| 3    | 0.0     | 2.0     | 0.0     | 2.0       | –            |
| 4    | 2.0     | 0.0     | 1.0     | 1.5       | –            |
| 5    | 3.0     | 3.0     | 3.0     | 3.0       | 1            |
| 6    | 1.0     | 1.0     | 1.0     | 1.0       | 2, 3, 4      |
| 7    | 2.0     | 2.0     | 2.0     | 2.0       | 5, 6         |
| 8    | 1.0     | 1.0     | 0.0     | 1.0       | –            |
| 9    | 2.0     | 2.0     | 1.0     | 1.7       | 4, 8         |
| 10   | 1.0     | 0.0     | 2.0     | 1.5       | 6, 9         |
| 11   | 2.0     | 2.0     | 2.0     | 2.0       | 7            |
| 12   | 2.0     | 2.0     | 2.0     | 2.0       | 10           |
| 13   | 3.0     | 1.0     | 1.0     | 1.3       | 11, 12       |
| 14   | 0.0     | 1.0     | 0.0     | 1.0       | 12           |
| 15   | 1.0     | 1.0     | 1.0     | 1.0       | 13, 14       |

Demands of models: DA = 35, DB = 35, DC = 38
Period time = 480
Cycle time = 480/108 = 4.4

Table A5. Problem 5.

| Task | Model A | Model B | Model C | Task time | Predecessors |
|------|---------|---------|---------|-----------|--------------|
| 1    | 1.0     | 1.0     | 2.0     | 1.3       | –            |
| 2    | 2.0     | 1.0     | 2.0     | 1.7       | –            |
| 3    | 2.0     | 1.0     | 1.0     | 1.3       | 1, 2         |
| 4    | 1.0     | 0.0     | 0.0     | 1.0       | –            |
| 5    | 0.0     | 2.0     | 1.0     | 1.5       | –            |
| 6    | 2.0     | 3.0     | 3.0     | 2.7       | –            |
| 7    | 1.0     | 1.0     | 0.0     | 1.0       | –            |
| 8    | 3.0     | 4.0     | 1.0     | 2.7       | 4, 5, 6, 7   |
| 9    | 1.0     | 0.0     | 0.0     | 1.0       | 3            |
| 10   | 0.0     | 1.0     | 4.0     | 2.5       | 3            |
| 11   | 1.0     | 2.0     | 3.0     | 2.0       | 8            |
| 12   | 2.0     | 4.0     | 4.0     | 3.3       | 9, 10, 11    |
| 13   | 3.0     | 0.0     | 2.0     | 2.5       | –            |
| 14   | 1.0     | 2.0     | 1.0     | 1.3       | 13           |
| 15   | 3.0     | 2.0     | 0.0     | 2.5       | 14           |
| 16   | 0.0     | 3.0     | 2.0     | 2.5       | 15           |
| 17   | 0.0     | 0.0     | 1.0     | 1.0       | 12           |
| 18   | 1.0     | 0.0     | 0.0     | 1.0       | 12           |
| 19   | 0.0     | 1.0     | 0.0     | 1.0       | 16           |
| 20   | 2.0     | 0.0     | 1.0     | 1.5       | 16           |
| 21   | 0.0     | 1.0     | 0.0     | 1.0       | 12           |
| 22   | 1.0     | 2.0     | 2.0     | 1.7       | 17, 18, 19, 20, 21 |

Demands of models: DA = 25, DB = 35, DC = 25
Period time = 480
Cycle time = 480/85 = 5.6
Table 16. Problem 6.

Table A6. Problem 6.

| Task | Model A | Model B | Model C | Model D | Task time | Predecessors |
|------|---------|---------|---------|---------|-----------|--------------|
| 1    | 1.0     | 1.0     | 1.0     | 1.0     | 1.0       | –            |
| 2    | 0.0     | 1.0     | 0.0     | 0.0     | 1.0       | –            |
| 3    | 0.0     | 2.0     | 2.0     | 0.0     | 2.0       | –            |
| 4    | 2.0     | 0.0     | 1.0     | 0.0     | 1.5       | –            |
| 5    | 0.0     | 0.0     | 0.0     | 1.0     | 1.0       | –            |
| 6    | 0.0     | 0.0     | 0.0     | 1.0     | 1.0       | –            |
| 7    | 3.0     | 2.0     | 0.0     | 1.0     | 2.0       | –            |
| 8    | 2.0     | 3.0     | 2.0     | 2.0     | 2.25      | 1            |
| 9    | 0.0     | 0.0     | 2.0     | 0.0     | 2.0       | 4            |
| 10   | 1.0     | 0.0     | 0.0     | 2.0     | 2.0       | 4, 5         |
| 11   | 1.0     | 0.0     | 0.0     | 2.0     | 1.5       | 4, 6, 7      |
| 12   | 0.0     | 0.0     | 1.0     | 0.0     | 1.0       | 3            |
| 13   | 1.0     | 1.0     | 0.0     | 0.0     | 1.0       | 2, 7         |
| 14   | 0.0     | 1.0     | 0.0     | 0.0     | 1.0       | 3, 7         |
| 15   | 0.0     | 0.0     | 3.0     | 0.0     | 3.0       | 3            |
| 16   | 2.0     | 0.0     | 3.0     | 0.0     | 2.5       | 8, 9, 13     |
| 17   | 2.0     | 0.0     | 5.0     | 5.0     | 4.0       | 8, 10, 11, 12, 13 |
| 18   | 4.0     | 0.0     | 0.0     | 4.0     | 4.0       | 11           |
| 19   | 0.0     | 3.0     | 1.0     | 3.0     | 2.3       | 8, 11, 12, 13, 15 |
| 20   | 1.0     | 0.0     | 1.0     | 1.0     | 1.0       | 16, 17, 18   |
| 21   | 0.0     | 4.0     | 2.0     | 0.0     | 3.0       | 14, 17, 19   |
| 22   | 0.0     | 5.0     | 0.0     | 0.0     | 5.0       | 21           |
| 23   | 1.0     | 0.0     | 0.0     | 0.0     | 1.0       | 17           |
| 24   | 0.0     | 0.0     | 0.0     | 1.0     | 1.0       | 18, 19       |
| 25   | 1.0     | 2.0     | 2.0     | 2.0     | 1.75      | 20, 22, 23, 24 |

Demands of models: DA = 40, DB = 15, DC = 15,
Period time = 700
Cycle time = 700/90 = 7.8
Table 17. Problem 7.

Table A7. Problem 7.

| Task | Model A | Model B | Task time | Predecessors |
|------|---------|---------|-----------|--------------|
| 1    | 1.0     | 1.0     | 1.0       | –            |
| 2    | 2.0     | 2.0     | 2.0       | –            |
| 3    | 1.0     | 0.0     | 1.0       | –            |
| 4    | 1.0     | 1.0     | 1.0       | –            |
| 5    | 1.0     | 1.0     | 1.0       | –            |
| 6    | 0.0     | 1.0     | 1.0       | –            |
| 7    | 2.0     | 2.0     | 2.0       | –            |
| 8    | 2.0     | 2.0     | 2.0       | 2            |
| 9    | 3.0     | 0.0     | 3.0       | 3            |
| 10   | 2.0     | 2.0     | 2.0       | 4            |
| 11   | 1.0     | 1.0     | 1.0       | 5            |
| 12   | 0.0     | 1.0     | 1.0       | 6            |
| 13   | 1.0     | 1.0     | 1.0       | 7, 8         |
| 14   | 1.0     | 0.0     | 1.0       | 9            |
| 15   | 4.0     | 2.0     | 3.0       | 10           |
| 16   | 3.0     | 3.0     | 3.0       | 11, 12       |
| 17   | 2.0     | 4.0     | 3.0       | 13           |
| 18   | 1.0     | 0.0     | 1.0       | 14           |
| 19   | 3.0     | 0.0     | 4.0       | 15           |
| 20   | 1.0     | 1.0     | 1.0       | 16           |
| 21   | 1.0     | 0.0     | 1.0       | 17           |
| 22   | 0.0     | 2.0     | 2.0       | 17           |
| 23   | 2.0     | 0.0     | 2.0       | 18           |
| 24   | 1.0     | 1.0     | 1.0       | 19, 20       |
| 25   | 1.0     | 0.0     | 1.0       | 21           |
| 26   | 0.0     | 2.0     | 2.0       | 2            |
| 27   | 1.0     | 0.0     | 1.0       | 23           |
| 28   | 1.0     | 1.0     | 1.0       | 24           |
| 29   | 1.0     | 1.0     | 1.0       | 25, 26       |
| 30   | 3.0     | 0.0     | 3.0       | 27           |
| 31   | 1.0     | 1.0     | 1.0       | 28           |
| 32   | 1.0     | 5.0     | 3.0       | 29           |
| 33   | 2.0     | 2.0     | 2.0       | 30, 31       |
| 34   | 1.0     | 1.0     | 1.0       | 32           |
| 35   | 3.0     | 1.0     | 2.0       | 33           |
| 36   | 0.0     | 4.0     | 4.0       | 34           |
| 37   | 3.0     | 0.0     | 3.0       | 4            |
| 38   | 1.0     | 1.0     | 1.0       | 35           |
| 39   | 0.0     | 1.0     | 1.0       | 36           |
| 40   | 1.0     | 0.0     | 1.0       | 37           |
| 41   | 2.0     | 2.0     | 2.0       | 38           |
| 42   | 2.0     | 2.0     | 2.0       | 39, 40       |
| 43   | 1.0     | 1.0     | 1.0       | 41           |
| 44   | 5.0     | 5.0     | 5.0       | 42, 43       |

Demands of models: DA = 50, DB = 45
Period time = 500
Cycle time = 500/95 = 5.26