European Journal of Academic Essays 1(4): 68-75, 2014
ISSN: 2183-1904
www.euroessays.org

Multi-manned Assembly Line Balancing Problem with Variable Task Times

Abdollatif Sepahi,[1], Seyed Gholamreza Jalali Naini[2]

[1,2]Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
[1]*Latif.sepahi@gmail.com*
[2]*sgjalalin@yahoo.com*

Abstract: In many real world assembly lines the product is of large size and more than one worker operates on the same work-piece in each station. In this paper the multi-manned assembly line (*MAL*) balancing problem is considered. Typically in the literature of *MAL* it is assumed that the task times are deterministic and independent of other factors. However in real world assembly lines it is common to expect a greater task time when the number of workers in a station increases. This situation hasn't been considered in the previous studies on *MAL*. In this paper it is assumed that task times are dependent on the concentration of workers in the station. A mathematical formulation is presented to solve this problem with the objective of minimizing the number of stations. Since the problem is NP-hard, four heuristic procedures are developed to solve this problem. Computational results show the performance of these heuristics.

Keywords: Assembly line Balancing, Straight line, multi manned, optimization

1. Introduction

Assembly lines are flow based production systems used to produce standardized commodities in high volume. These systems even gain importance in producing low volume customized products.

An assembly line consists of m workstations arranged along a material handling equipment. Beginning from the first station to the last station, work-pieces are moved through the line. In each station a set of operations are performed on the work-pieces regarding the cycle time (maximum available time in each work-cycle). The decision problem of assigning tasks to stations with the aim of optimizing some objective functions is called assembly line balancing (*ALB*) problem. The most studied problem in the field of *ALB* is called simple assembly line balancing problem (*SALBP*) and has the following assumptions [1]-[3]:

- Mass production of one homogenous goods
- Given production process
- Paced line with a fixed cycle time c.
- Every task j has a deterministic and numeral operation time t_j
- No assignment limitations besides precedence constraints
- Serial line layout with m stations
- Wholly stations are similarly equipped with respect to machines and workers
- Maximize the line efficiency: $Eff = \frac{t_{sum}}{m \times c}$, in which m is the number of stations and $t_{sum} = \sum_{j=1}^{n} t_j$ is the sum of processing time of all tasks.

These assumptions are very restricting with respect to real assembly line production systems. Therefore many researchers have recently focused on identifying and modeling more realistic situations in assembly lines. The resulting problems are called generalized assembly line balancing problems (*GALBP*).

Several generalizations have been studied for the *ALBP*. Several cases of these generalizations are U-shaped assembly lines balancing [4], considering parallel workstations [5], considering process alternatives [6] and two sided assembly lines [7]. Some recent surveys of generalized assembly line models are [2], [3], [8]-[10].

[1] Corresponding Author: Abdollatif Sepahi, Department of Industrial Engineering, Iran university of Science and Technology, Tehran, Iran,

Corresponding Author: Abdollatif Sepahi

Multi-manned assembly line balancing problems (*MALBP*) are a new type of *GALBP*s in which more than one worker can be assigned to each station. This situation frequently occurs in industries with large-sized products such as automotive industries in which the product is of reasonable large size to use multi-manned assembly line configuration [11].

Some advantages of multi-manned assembly lines over simple assembly lines are reducing the length of assembly line, the cycle time, the charge of tools and fixtures, material handling and setup times [12]. These advantages have enough motivation to use *MAL* configuration in producing large-sized commodities.

Although *MAL*s are very common in real world assembly line systems, few studies have considered this problem. The most similar problem considered in the literature of *ALB* is proposed by Bartholdi, in which the two-sided line is considered [7]. In a two-sided line there are two serial lines in parallel. Instead of single stations, couples of adverse stations on either side of the line work in parallel on the same work-piece. The difference between this problem and the problem considered in this paper is that in a two-sided line there are two workers in each station and each worker is constrained to work on only one part of the station. While in the problem considered in this paper there can be more than two workers in each station and these worker can perform tasks on either sides of the station. The problem addressed in this study is different from problems that consider cooperation of several workers on the same task and the same product to reduce the cycle time. There are some studies considering parallel stations in which several workers perform the same tasks on different work-pieces [5], [13], [14]. This situation is different from the *MAL*s in which several workers perform different tasks on the same work-piece. Dimitriadis, considered the *MALBP* for the first time. He also developed a heuristic assembly line balancing procedure to solve the problem [11]. Cevikcan et al, proposed a mathematical programming model for creating assembly physical multi-manned stations in mixed model assembly lines. Since the model proposed by them is NP-hard, they proposed a scheduling-based heuristic to solve the problem [15]. Chang & Chang, discussed a mixed-model assembly line balancing problem with multi-manned workstations and developed a mathematical model for the mixed-model assembly line balancing problem with simultaneous production (*MALBPS*) to decide the optimal number of workstations. They also proposed a coding system, Four-Position Code (*FPC*), to re-code the tasks to tackle this issue, and provided a computerized coding program written in C++ to generate those *FPCs* [16]. Fattahi et al, presented a mixed integer programming model for *MALBP*. They also developed an ant colony meta-heuristic approach to efficiently solve the medium- and large-size scales of this problem [12].

In this paper the *MALBP* is extended in such a way that task time are dependent on the number of workers in the station. In the previous studies on *MALBP* it is assumed that the number of workers that can be assigned to a station is restricted by the maximum feasible 'worker concentration'. This quantity is provided by the system modeler according to the product size. But in many realistic situations the number of workers in the station has a direct impact on the processing time of the tasks assigned to the station. For example there may be a certain number of a specific tool in a station, if the number of workers exceed the number of tools, waiting time for the tool to be released by other operators may be incurred. Another factor is the required space to perform the tasks i.e. the task time may increase if there is not enough space to perform the task. These examples and many other realistic situations highlight the importance of considering task times which are dependent on the number of workers in the station. To the greatest of our knowledge this kind of task times hasn't been considered in the literature of *MALBP* so far. To illustrate the model an example is offered, this example is created from the well-known example of Mertens. The precedence graph for this instance is presented in fig.1 and the task times for each task and number of workers in are presented in table 1. Cycle time is assumed to be 6. In table 1 some task times for some number of workers are greater than cycle time. This means that the task cannot be accomplished with that number of worker concentration in a station.

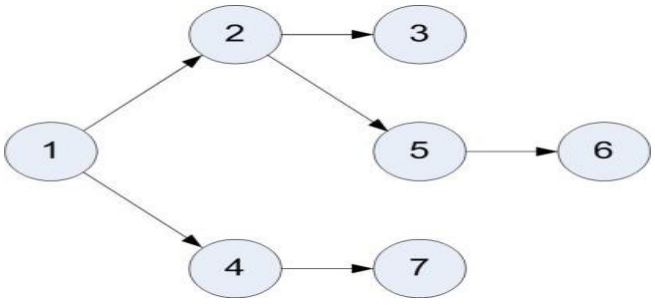

Fig. 1 precedence graph of Mertens instance

Table 1 task times

task	Number of workers in the station		
	1	2	3
1	1	1	2
2	5	5	6
3	4	5	6
4	3	4	5
5	5	6	7
6	6	6	7
7	5	5	6

An optimum solution for this example is presented in fig. 2. For each task, starting time and finishing time are shown alongside its bar. Shaded rectangles indicate unavoidable delay between two consecutive tasks, or idle time at the end of the cycle time. Unavoidable delays occur when a worker must wait until other work elements assigned to some other workers are complete. For example in fig.2 starting time of task 2 is delayed until task 1, which is predecessor of task 2, is complete. As it can be seen from fig. 2 three stations and 6 workers are needed to perform the tasks in a multi-manned assembly line system.

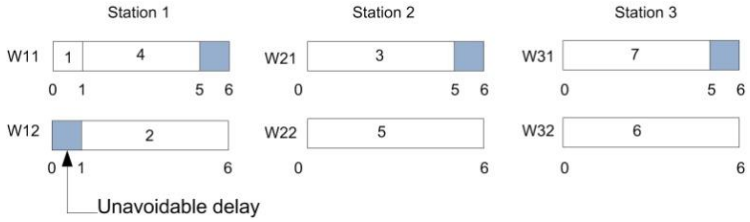

Fig. 2 Assignment of tasks to workers and stations in an optimum solution

The remainder of this paper is organized as follows: in section 1 a mathematical formulation is proposed to solve the problem. In section 2 heuristic algorithms are developed to solve the problem. Computational results are presented in section 4. The main conclusions of the paper and suggestions for future research are presented in section 5.

2. Mathematical formulation

The notations used to formulate the problem are presented in table 2.

Table 2 Notations used for the mathematical model

i, h	task
J	station
K	worker
I	Set of tasks
K	Set of workers
J	Set of workstations
P_i (P_i^*)	Set of direct (all) predecessors of task i
F_i (F_i^*)	Set of direct (all) successors of task i
C	Cycle time
M	Number of stations
M	A big positive number
MC	Maximum concentration of workers in a station
N	Number of tasks
p_{ik}	Processing time of task i when there are k workers in the station
t_{ij}	Processing time of task i in station j.

$x_{ijk} \in\{0,1\}$	Equals 1 if task i is assigned to worker k in station j.
$y_{ih} \in\{0,1\}$	Equals to 1 if task i and h is assigned to the same worker and task i is performed earlier than task h.
$w_{jk} \in\{0,1\}$	Equals to 1 if k th worker is used in station j.
w_{sjk}	Equals to 1 if k workers are used in station j.
st_i	Start time of task i

The problem is formulated as follows:

$$min \sum_{j\in J}\sum_{k\in K} j\times x_{mjk} \quad (1)$$

$$\sum_{j\in J}\sum_{k\in K} x_{ijk} = 1 \qquad \forall i \in I \quad (2)$$

$$\sum_{j\in J}\sum_{k\in K} j\times x_{hjk} \le \sum_{j\in J}\sum_{k\in K} j\times x_{ijk} \qquad \forall i \in I, h \in P_i \quad (3)$$

$$st_i + t_{ij} \le C + M\times(1-\sum_{k\in K} x_{ijk}) \qquad \forall i \in I, j \in J \quad (4)$$

$$st_i - st_h + M\times\left(1-\sum_{k\in K} x_{hjk}\right) + M\times\left(1-\sum_{k\in K} x_{ijk}\right) \ge t_{hj} \qquad \forall i \in I, h \in P_i, j \in J \quad (5)$$

$$st_h - st_i + M\times(1-x_{hjk}) + M\times(1-x_{ijk}) + M\times(1-y_{ih}) \ge t_{ij} \qquad \forall i \in I, j \in J, k \in K,\ h \in\{r \mid r \in I-(P_i^* \cup F_i^*) \wedge i<r\} \quad (6)$$

$$st_i - st_h + M\times(1-x_{hjk}) + M\times(1-x_{ijk}) + M\times(y_{ih}) \ge t_{hj} \qquad \forall i \in I, j \in J, k \in K,\ h \in\{r \mid r \in I-(P_i^* \cup F_i^*) \wedge i<r\} \quad (7)$$

$$\sum_{i\in I} x_{ijk} - N\times w_{jk} \le 0 \qquad \forall j \in J, k \in K \quad (8)$$

$$\sum_{k} k\times ws_{jk} = \sum_{k} w_{jk} \qquad \forall j \in J \quad (9)$$

$$w_{j(k+1)} \le w_{jk} \qquad \forall j \in J, k \in K \quad (10)$$

$$t_{ij} = \sum_{k\in K} p_{ik}\times ws_{jk} \qquad \forall i \in I, j \in J \quad (11)$$

$$st_i \ge 0 \qquad \forall i \in I \quad (12)$$

$$x_{ijk} \in\{0,1\} \qquad \forall i \in I, j \in J, k \in K \quad (13)$$

$$y_{ih} \in\{0,1\} \qquad \forall i \in I,\ h \in\{r \mid r \in I-(P_i^* \cup F_i^*) \wedge i<r\} \quad (14)$$

$$w_{jk} \in\{0,1\} \qquad \forall j \in J, k \in K \quad (15)$$

$$ws_{jk} \in\{0,1\} \qquad \forall j \in J, k \in K \quad (16)$$

In this formulation (1) is the objective function to be minimized, which is the number of stations. *m* is a fictitious task that is successor of all tasks, therefore it is always assigned to the last station. Therefore to minimize the number of stations it is sufficient to minimize the index of the station to which task *m* is assigned. Constraints (2) ensure that each task *i* is assigned to only one worker and one station. Equations (3) ensure that precedence constraints are observed. Constraints (4) imply that all tasks must be finished before the end of the cycle time. Equations (5) imply that if task *h* is a direct predecessor of task *i* and they both are assigned to the same station, then task *i* must be started after task *h* is finished. If these tasks are not in the same station, equation (5) becomes redundant. If tasks *i* and *h* don't have any direct precedence relations and are assigned to the same worker constraints (6) and (7) become active. If *i* is assigned earlier than task *h* then

$y_{ih}=1$ so the equation (7) becomes redundant and equation (6) leads to $st_h - st_i \geq t_{ij}$ this means that task *h* must start after task *i* is finished. On the other hand if *h* is assigned earlier than *i* then $y_{ih}=0$ and equation (6) becomes redundant and equation (7) leads to: $st_h - st_i \leq t_{ij}$. The constraints (8) are the worker constraints and imply that if the *k* th worker hasn't been assigned to the station *j*, no task can be assigned to it. With constraints (9) number of ws_{jk} is set to be 1 if the number of workers in station *j* equal to *k*, otherwise it is set to be 0. Constraints (10) ensure that the workers are loaded in increasing order of their indexes. With (11) t_{ij} assumes the processing time of task *i* in station *j* according to the number of workers in the station. Constraints (12) ensure that starting times are non-negative. Constraints (13) through (16) indicate that variables x_{ijk}, y_{ih}, w_{jk} and ws_{jk} are binary variables.

3. Heuristic procedures developed

Since *SALBP* is known to be NP-hard [17], [18] and the problem considered here is a generalization of *SALBP*. Therefore the problem under consideration is also NP-hard. So it is fully justified to develop heuristic algorithms in order to obtain good solutions in a computational time short enough to be applied in industrial real instances. In this section, four heuristic procedures are developed to solve the problem introduced in previous sections. All of these procedures are based on priority rules. The general procedure for these heuristics is as follows:

Step 1: Set station counter *Sc=0*, and available tasks *Avail_task= {1, 2… N}*. Available tasks are the tasks that haven't been assigned to any worker in any station.

Step 2: Set the number of workers in the station *Wn*=1 and previous number of tasks *Temp_Tc*=0.

Step 3: Set the current worker *k=1* and task counter *Tc*=0. Consider two temporary sets of tasks *Temp_task1* and *Temp_task2* and set *Temp_task1=Avail_task.* Set all of the workers as empty i.e. no task has been assigned to them.

Step 4: Among tasks of *Temp_task1*, ones that are assignable to worker *k*, select the task with the highest priority, according to one of the priority rules which will be presented later in this section. Assign it to worker *k* and delete it from *Temp_task1* then set *Tc=Tc*+1. Repeat this process until there is no task assignable to worker *k*. Then go to step 5.

Step 5: Set *k=k*+1. If *k<=Wn* then go to step 4, if not go to step 6.

Step 6: A trial station with *Wn* number of workers is complete. If the number of tasks in this station, *Tc*, is more than *Temp_Tc* and *Wn*+1 is not more than *MC*, maximum concentration of workers in a station, then set *Wn=Wn*+1, *Temp_task2=Temp_task1* and *Temp_Tc=Tc* then go to step 3. If *Tc* is more than *Temp_Tc* and *Wn+1* is more than *MC* then set *Avail_task=Temp_task1* and go to step 7. Finally if *Tc* is not more than *Temp_Tc* set *Avail_task=Temp_task2* and go to step 7.

Step 7: A station is completed and the number of workers and the tasks assigned to each worker must be saved. If *Avail_task* is empty return *Sc* and end the procedure, otherwise set *Sc=Sc*+1 and go to step 2.

All of the heuristics presented in this paper use this procedure. Four priority rules are used in this paper:

- Maximum task time
- Minimum task time
- Maximum value of $\frac{F_j^*}{Max_{i \in V}(F_i^*)}$ where F_j^* is the set of all successors of task *j*.
- Maximum value of $\frac{t_j}{c} + \frac{F_j^*}{Max_{i \in V}(F_i^*)}$

Therefore four heuristic procedures are to be considered. For the rest of this paper *Max_t* and *Min_t* are used to refer to the heuristics that use the maximum and minimum task time priority rules respectively. Also *Max_s* and *Max_ts* are used for the heuristics that use the third and fourth priority rules.

4. Computational results

In this section performance of proposed heuristic procedures is illustrated. Due to the innovative nature of the problem, there are no existing methods to compare the results of the proposed algorithms with. Therefore in this section, the results of the proposed algorithms are compared with the exact approach and since in most cases no exact solution is found, a lower bound approach is implemented to be able to test the efficiency of the algorithm. The tests are implemented in C++ language and run on a PC with 2.4 GHz Intel Core i3 and 4 GB of RAM memory.

In this section, at first the lower bound structure is explained. It is assumed that the first task in the precedence graph is predecessor of all other tasks. Similarly it is assumed that the last task in the graph is successor of all of the other tasks. If there is no such tasks, fictitious tasks is to be considered. To obtain a lower bound on the number of stations, the longest path, also called critical path, from the first task to the last task is considered. In this paper critical path is computed by considering the minimum value of processing time for each task. The length of this path is a lower bound on the time needed to produce one commodity, lessening or increasing the number of workers in each station does not change this value. Thus, the formulation for lower bound is:

$$LB_{number\ of\ stations} = \left\lceil \frac{\sum_{j \in critical\ path} t_{j1}}{c} \right\rceil \qquad (1)$$

To compare the heuristic algorithms different instances are randomly generated from 25 different well-known precedence networks available at www.assembly-line-balancing.de .These instances are solved and their relative deviation from the lower bound is computed using the following formula:

$$Relative\ Deviation = \frac{Algorithm_{sol} - LB}{LB} \qquad (2)$$

The results are presented in fig. 3. As it can be seen from this figure *Max_s* has a better overall performance, in comparison to other heuristics.

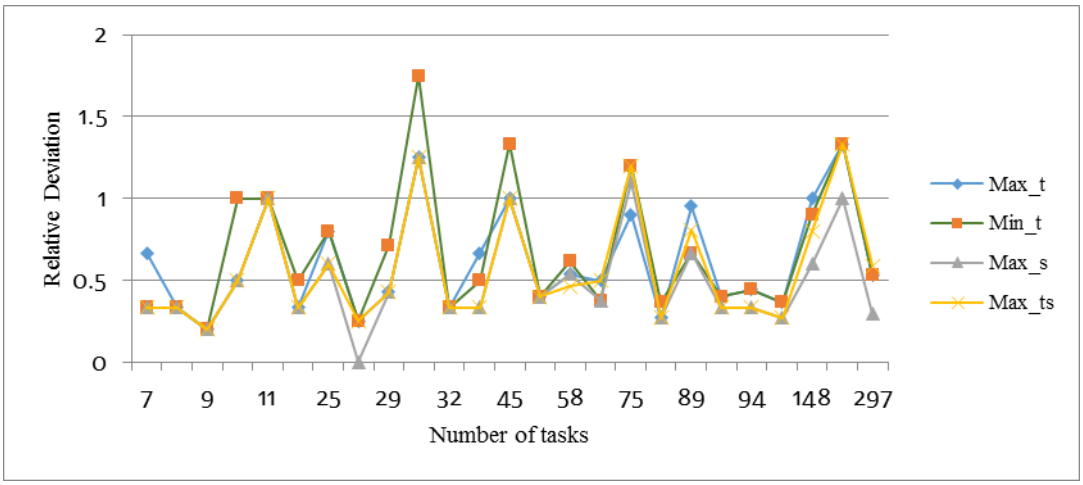

Fig. 3 Performance of heuristic algorithms with respect to the number of tasks

Therefore the *Max_s* algorithm finds better solutions than the other heuristics. This encourages checking the performance of *Max_s* more precisely. Specifically it is interesting to illustrate the performance of the algorithm for different problem characteristics such as order strength[2] (*OS*) of the precedence network or time variability[3] (*TV*) of the instance. The performance of *Max_s* with respect to these two characteristics is presented in figures 4 and 5. As it can be from these figures, there is no significant trend for the performance of *Max_s* with increasing these characteristics. In fig. 4, the values of relative deviation for values of *OS* between 0.22 and 0.50 are generally higher than other values of *OS*. For other values of *OS*, *Max_s* performs more efficiently. As it can be seen from fig. 5 values of relative deviation for *TV* between 13 and 25 are higher than other values of *TV*. This implies that for vary low or very high values of *TV*; *Max_s* performs better than average values of *TV*.

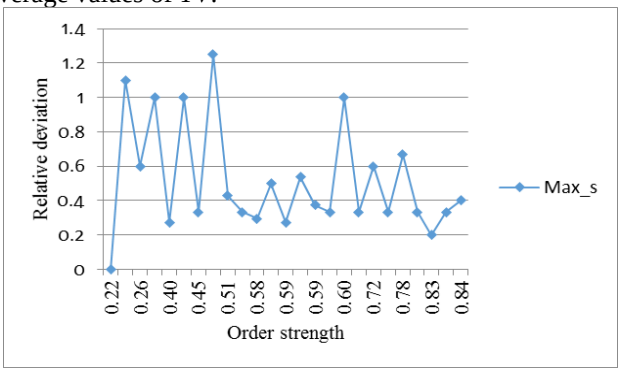

Fig. 4 Performance of *Max_s* with respect to *OS*

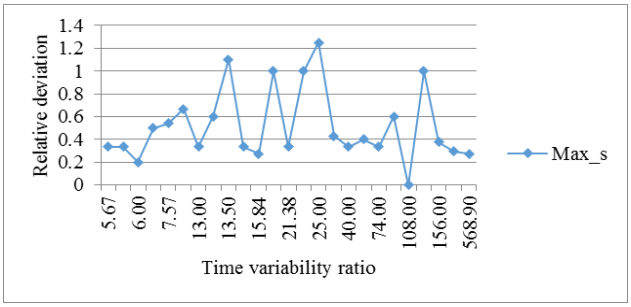

Fig. 5 Performance of *Max_s* with respect to *TV*

A final experiment is designed to illustrate the performance of *Max_s*. Another data set is generated using a selection of well-known instances for *SALBP*-1. In order to facilitate comparison of the proposed algorithm with other future algorithms, the task times are assumed to be equal to the corresponding *SALBP* instance if there is one worker in the station and $p_{i,k+1}=p_{ik}+1; for\ k>1$. The results are presented in Table 3. The optimum numbers of stations for Mertens and Bowman examples are obtained through by solving the mathematical model using Lingo 11. For other examples a lower bound is computed using equation (1).

[2] Order strength is the ratio of the number of all precedence relations to its maximum possible number

[3] Time variability is the ratio between the maximum and minimum task time. in this study, the original TV of the instance is used.

Table 3 Results of *Max_s* for a selection of well-known instances

Author	Tasks	Cycle time	Opti*MAL* number of stations for *SALBP*	Opti*MAL* number of stations (or a lower bound of it)	Obtained number of stations	Obtained number of workers	Maximum concentration of worker in each station	CPU time (s)
MERTENS	7	6	6	4	4	6	4	0.000
		7	5	3	4	5	4	0.000
		8	5	3	3	6	4	0.000
		10	3	3	3	3	4	0.000
		15	2	2	2	3	3	0.000
BOWMAN8	8	20	5	4	4	6	4	0.000
JAESCHKE	9	6	8	5	6	8	4	0.000
		7	7	4	6	7	4	0.000
		8	6	4	6	6	4	0.000
		10	4	3	4	5	4	0.000
		18	3	2	3	3	4	0.000
JACKSON	11	7	8	4	6	9	4	0.000
		9	6	3	5	7	4	0.000
		10	5	3	4	7	4	0.000
		13	4	2	4	6	4	0.000
		14	4	2	3	4	4	0.000
MANSOOR	11	48	4	2	4	5	4	0.000
		62	3	2	3	4	4	0.000
		94	2	1	2	4	4	0.000
MITCHELL	21	14	8	6	7	10	4	0.000
		15	8	5	7	10	4	0.000
		21	5	4	5	6	4	0.000
		26	5	3	4	6	4	0.000
		35	3	3	3	4	3	0.000
HESKIA	28	138	8	4	4	10	4	0.000
		205	5	3	3	7	4	0.001
		216	5	3	3	7	4	0.000
		256	4	2	3	6	4	0.001
		324	4	2	2	6	4	0.000
SAWYER30	30	25	14	4	9	17	6	0.000
		27	13	4	8	17	5	0.000
		30	12	4	8	15	5	0.000
		33	11	3	7	15	5	0.000
		36	10	3	7	13	5	0.001
KILBRID	45	56	10	3	6	16	6	0.001
		57	10	3	6	15	6	0.000
		62	9	3	5	15	5	0.000
		69	8	2	5	12	5	0.001
		79	7	2	4	11	5	0.000
TONGE70	70	160	23	8	11	29	5	0.001
		168	22	8	11	27	5	0.000
		176	21	7	11	28	5	0.001
		185	20	7	11	26	5	0.001
		195	19	7	12	28	5	0.000
ARC83	83	3786	21	11	14	27	4	0.001
		3985	20	11	14	25	4	0.002
		4206	19	10	12	23	4	0.001
		4454	18	10	12	24	4	0.001
		4732	17	9	11	23	4	0.002
ARC111	111	5755	27	11	14	34	5	0.002
		5785	27	11	14	35	5	0.002
		6016	26	11	13	37	5	0.003
		6267	25	10	13	34	5	0.003
		6540	24	10	13	33	5	0.003

As it can be seen from this table the CPU times are very low and for instance for less than 28 tasks the CPU time is less than one millisecond. This implies the time efficiency of the proposed algorithm. Another observation is the significant improvement in the number of stations in comparison to *SALBP*-1, despite the fact that there is a penalty on the task times when the number of workers increases in each station.

5. Conclusions and future research

In this paper the *MAL* model presented by Dimitriadis [11], is extended in such a way that task times are assumed to be dependent on the number of workers in the station. A mathematical formulation is presented to solve this problem with the objective of minimizing the number of stations. Since the problem is NP-hard, four heuristic procedures are developed to solve this problem. These heuristics are based on the priority rules and the computational results show that the *Max s* heuristic performs better than other heuristics. Developing other heuristic or meta-heuristics such as genetic algorithms to solve the introduced model and considering multiple-objective optimization problem by taking into account several other criteria, such as load balancing and smoothing are recommended for future research in this area.

References

[1] Baybars, I. (1986). A survey of exact algorithms for the simple assembly line balancing problem. Management Science, 32, 909–932.

[2] Scholl, A., Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research, 168, 666-693.

[3] Scholl, A. (1999). Balancing and sequencing assembly lines, 2nd edn. Physica, Heidelberg.

[4] Miltenburg, J., Wijngaard, J. (1994). The U-line line balancing problem. Management Science, 40, 1378–1388.

[5] Buxey GM. (1974). Assembly line balancing with multiple stations. Management Science, 20, 1010–21.

[6] Pinto, P.A., Dannenbring, D.G., Khumawala, B.M. (1983). Assembly line balancing with processing alternatives: an application. Management Science, 29, 817–830.

[7] Bartholdi, J.J. (1993). Balancing two-sided assembly lines: A case study. International Journal of Production Research, 31, 2447–2461.

[8] Erel, E., Sarin, S.(1998). A survey of the assembly line balancing procedures. Production Planning and Control, 9, 414–434.

[9] Rekiek, B., Dolgui, A., Delchambre, A., Bratcu, A. (2002). State of art of optimization methods for assembly line design. Annual Reviews in Control, 26, 163–174.

[10]Boysen N., Fliedner M., Scholl A. (2007). A classification of assembly line balancing problems. Eur J Oper Res 183:674–693

[11]Dimitriadis, SG. (2006) Assembly line balancing and group working: a heuristic procedure for workers' groups operating on the same product and workstation. Comput Oper Res 33:2757–2774.

[12]Fattahi, P., Roshani, A., Roshani, A. (2011). A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem. Int J Adv Manuf Technol, 53, 363–378.

[13]Pinto, PA., Dannenbring, DG., Khumawala, BM. (1981) Branch and bound and heuristic procedures for assembly line balancing with paralleling of stations. International Journal of Production Research,19, 565–76.

[14]Akagi, F., Osaki, H., Kikuchi, S. (1983). A method for assembly line balancing with more than one worker in each station. International Journal of Production Research, 21, 755–70.

[15]Cevikcan E., Durmusoglu BM., Unal ME. (2009). A team-oriented design methodology for mixed model assembly systems. Comput Ind Eng, 56,576–599.

[16]Chang H. J., Chang T. M. (2010). Simultaneous Perspective-Based Mixed-Model Assembly Line Balancing Problem. Tamkang Journal of Science and Engineering, 13 (3), 327_336.

[17]Ege, Y., Azizoglu, M., Ozdemirel, NE. (2009). Assembly line balancing with station paralleling. Comput Ind Eng. doi:10.1016/j.cie.2009.05.014.

[18]Yeh, DH., Kao, HH. (2009). A new bidirectional heuristic for the assembly line balancing problem. Comput Ind Eng. doi:10.1016/j. cie.2009.05.004.