3

A Metaheuristic Approach to Solve the Alternative Subgraphs Assembly Line Balancing Problem

Liliana Capacho[1] and Rafael Pastor[2]
[1]*University of Los Andes*
[2]*Technical University of Catalonia*
[1]*Venezuela*
[2]*Spain*

1. Introduction

Nowadays, assembly line balancing problems are commonly found in most manufacturing and production systems. In its basic form, an assembly line balancing problem consists of finding an assignment of tasks to a group of workstations in such a way that the precedence constraints among the tasks are maintained and the sum of the times of the task assigned to each workstation does not exceed the maximum workstation time (i.e. the cycle time). According to the objective considered, two variants of the problem are distinguished: (1) the problem aims at minimizing the number of workstations for a given cycle time and (2), given the number of workstations, the problem seeks to minimize the cycle time. Over the last years a significant amount of research work has been done towards solving assembly line balancing problems efficiently. Finding the best solution is a crucial task for maintaining the competitive advantage of industries and, in some cases, for their survival. Falkenauer (2005), pg. 360, argues that the efficiency difference between an optimal and a sub-optimal assignment can yield economies reaching millions of dollars per year. However, solving real life problems is a very difficult task for decision makers and practitioners since even the simple case is NP-hard by nature. For this reason, most assembly line balancing problems involve only a few aspects of the real systems (see, for example, (Becker & Scholl, 2006)).

In order to deal with the complexity of industrial problems, a great variety of problem definitions (i.e. generalized assembly line balancing problems) have arisen, which consider other restrictions apart from the technological ones. Most common, these include mixed models, multiple products, different line layouts, parallel workstations and multiple objectives. However, real problems require tackling many of those generalizations simultaneously (Falkenauer, 2005). Such a consideration must also be taken into account when alternatives processes are involved. Alternatives may appear when, for example, new technologies are taking place in a production system, in which different procedures are available to complete a production unit, or when the processing order affects the processing times of certain tasks; i.e., the realization of one task facilitates, or makes more difficult, the completion of other tasks (see, for example, (Scholl *et al*., 2008) and (Das & Nagendra, 1997)).

A novel generalized assembly line balancing problem, entitled ASALBP: the Alternative Subgraphs Assembly Line Balancing Problem, is addressed here. In this problem alternative variants for different parts of an assembly or manufacturing process are considered. Each variant is represented by a subgraph that determines the tasks required to process a particular product and the task precedence relations. Thus, alternative assembly sub-processes for a sub-assembly may involve completely different set of tasks. Consequently, in addition to cycle time or workstations requirements, subgraph constraints must also be taken into account to ensure that tasks belonging to a particular subassembly are processed considering its corresponding assembly subgraph. Furthermore, it is also considered that task processing times are not fixed, but instead are dependent on the assembly subgraph. Therefore, total processing time may vary from one processing alternative to another (even when the alternatives involve the same set of tasks). Similarly to the simple case, the ASALB problem aims at minimizing the number of required workstations for a given bound on the cycle time (i.e., ASALBP-1), or minimizing the cycle time for a given number of workstations (i.e., ASALBP-2). This work focuses on ASALBP-1. As previously discussed, to solve the ASALBP efficiently, two problems must be solved simultaneously: the decision problem, which involves selecting a single assembly subgraph for each subassembly that allows alternatives, and the balancing problem to assign the tasks to the workstations.

In practice, due to the complexity of assembly problems, a two-stage approach is usually used to solve assembly balancing problems that involve alternatives. In a first stage, an assembly variant is selected considering a given criterion, such as, for example, shortest total processing time. When the production alternative has been selected (i.e., the complete assembly process has been defined) and a single precedence graph is available, the problem is then balanced in a second stage. (Capacho & Pastor, 2008) illustrated, by means of numerical examples, how by selecting a priory an alternative, it cannot be guaranteed that an optimal solution of the global problem will be obtained, because the best solution of a problem can be discarded if it does not meet the selection criteria. For instance, it was shown that the alternative assembly variant with the longest processing time required the smallest number of workstations for a given cycle time.

The Alternative Subgraphs Assembly Line Balancing Problem was introduced by (Capacho & Pastor, 2006) and was optimally solved by means of two mathematical programming models. The computational experiment carried out with the models showed that only small- and medium-scale problems could be solved optimally in significantly small computing times. Other attempts have also been done to solve the ASALBP optimally, see for example (Scholl *et al*., 2006). In order to solve ASALBP for practical sizes, (Capacho *et al*., 2006, 2009) proposed a group of heuristic methods based on adapting well-known priority rules (e.g. (Talbot *et al.,* 1986), most of which are based on the precedence relations and the cycle time. Solutions of good quality were obtained for problems involving up to 305 tasks and 11 assembly subgraphs. In this work the use of weighted, rather than nominal, values for the priority rules is explored. In particular, it is considered an adaptation of a class of metaheuristic methods, namely GRASP (Greedy Randomized Adaptive Search Procedure), which has been successfully applied to hard combinatorial problems (Delorme *et al*., 2004).

A GRASP overview and literature reviews can be found in (Festa & Resende, 2008a, 2008b). The former research work discusses the algorithmic aspects of this type of procedures; the second one presents GRASP applications covering a wide range of optimization problems, including scheduling, routing, graph theory, partitioning, location, assignment, and manufacturing. Other examples of GRASP can be found in (Dolgui *et al*., 2010), which

propose an algorithm for balancing transfer lines with multi-spindle machines; (Andrés *et al*., 2008) and (Martino & Pastor, 2007), both of which tackle problems involving setup times.
Basically, a GRASP is an iterative process in which each iteration consists of two phases: the construction phase, which generates a feasible solution, and the search phase that uses a local optimization procedure to find a local optimum in the neighbourhood of the constructed solution (Resende & Ribeiro, 2003). Feasible solutions are generated by iteratively selecting the next element to be incorporated to a partial solution. The best element is selected by ordering all candidate elements according to a greedy function. In this work, the adaptation to the ASALBP of thirteen well-known priority rules are used for selecting tasks and three priority rules are use for selecting the subgraphs (see (Capacho *et al*., 2009)). This resulted in a total of 39 construction methods, which are used to generate the initial feasible solutions. Furthermore, the application of two neighbourhood search strategies produces a total of 78 GRASP algorithms that are proposed, implemented and tested here. The performance of such algorithms is evaluated by considering a set of 150 medium- and large-scale problem instances; and compared with the results obtained in (Capacho *et al*., 2009) and with known optimal solutions (see (Capacho & Pastor, 2006, 2008)).
The remainder of this chapter is organized as follows: Section 2 describes the metaheuristic procedures (i.e., GRASP) that are designed and implemented here; Section 3 presents the computational experiment carried out to evaluate and compare the proposed algorithms; Section 4 presents the conclusions and proposes future research work; and, finally, Section 5 lists the References.

2. Grasp procedures for solving the ASALBP

As mentioned above, a GRASP involves a construction phase and a local search phase. In the proposed procedures (see (Capacho, 2007)), the construction phase generates a feasible solution by applying a construction method in which both the subgraphs and the assembly tasks are randomly selected following probability distributions based on weighted priority rule values. Weighted values are proportional or inversely proportional (when using a maximizing or minimizing criterion, respectively) to the values obtained considering a given priority rule. The local search phase generates and evaluates all neighbours of the current solution and maintains the best neighbour solution (i.e., the one that requires the fewest number of workstations). This process is repeated for a given length of time. The best overall solution generated is the final solution.
A solution for the Alternative Subgraphs Assembly Line Balancing Problem consists of a task sequence, a number of required workstations and a set of assembly subgraphs (i.e., one subgraph for each subassembly that allows alternatives).

2.1 The construction phase

To construct an initial solution, one subgraph is randomly selected for each available subassembly following a distribution (also referred to as an assembly route), following a probability function based on a priority rule for subgraphs. Once the subgraphs have been chosen, the set of available assembly tasks (AVT) is defined. The set of available tasks is formed with the tasks that belong to the selected subgraphs and those tasks that do not allow assembly variants. The assignable tasks are determined to form the list of candidate tasks (LCT). A task is assignable if all of its predecessors have already been assigned and the sum of its time and the time of the tasks assigned to the current workstation does not exceed

the cycle time. For each task in the candidate list LCT, the priority rule value is computed to construct a probability distribution, which is then used to randomly select the next task to be assigned to the current workstation. Once a task has been assigned it is removed from AVT. New lists LCT are generated and tasks are systematically assigned until all assembly tasks have been assigned (i.e., the sets AVT and LCT are empty) and the solution has been completed. The probability distributions used for selecting subgraphs and tasks are built using weighted values of the following priority rules.

2.1.1 Priority rules for subgraphs

The priority rules used for selecting subgraphs are the following:

a. **Minimum NP**: this rule ranks the subgraphs of the same subassembly according to ascending number of precedence relations involved in each subgraph, which is the total number of arcs entering into and within the subgraph.
b. **Minimum TT**: subgraphs are ranked according to ascending total processing time (i.e., the sum of the times of all tasks belonging to the subgraph).
c. **Minimum NT:** subgraphs are ranked according to ascending number of tasks.

Let consider, for example, a subassembly of a given manufacturing process that allows three alternative subgraphs, S1, S2 and S3, with total processing time of 30, 35 and 35 time units, respectively. Considering the priority rule TT, the cumulative probability distribution for selecting a subgraph *s* is as follows (r ∈ [0, 1) is a random value):

$$s = \begin{cases} S1 \text{ if } 0 \le r < 0.30 \\ S2 \text{ if } 0.30 \le r < 0.65 \\ S3 \text{ if } 0.65 \le r < 1 \end{cases} \quad (1)$$

2.1.2 Priority rules for tasks

Table 1 lists the priority rules considered to build the probability distribution to select the next tasks to be assigned. It is valid to mention that priority rules 3, 4, 5, 6 and 13 of Table 1 are minimization rules while all others are maximization ones. These rules are thirteen well-known priority rules that have been adapted to the ASALBP (see (Capacho *et al.*, 2009)) and random choice assignment. Basically, these rules are determined by considering the cycle time and task precedence relations and by measuring tasks processing times.

The following notation is considered:

n Number of tasks
ct Cycle time
m_{max} Upper bound on the number of workstations
t_{ir} Duration of task i when processed through route r (i = 1,…,n ; $r \in R_i$)
P_{ir} Set of immediate predecessors of task i, if processed through route r (i=1,…,n ; $r \in R_i$)
S_{ir} Set of all successors of task i, if it is processed through route r (i=1,…,n; $r \in R_i$)

Once the set of selected routes SR is known, the following values can be defined:

$sub(i)$ Subgraph chosen for task i ($\forall i \in AVT$); in this way it is possible to know the duration of task i ($t_{i,sub(i)}$).
Ei, Li Earliest and latest workstation to which task i can be assigned, respectively ($\forall i \in AVT$).
SIi, Si Set of immediate and total successors of task i, respectively ($\forall i \in AVT$).

No.	Label	Priority rule	Computation procedure
1	RPW	Rank Positional Weight	$RPW_i = t_{i,sub(i)} + \sum_{j \in S_{i,sub(i)}} t_{j,sub(j)}$ (2)
2	T	Task Time	$t_{i,sub(i)}$
3	EW	Earliest Workstation	$EW_i = \left\lceil (t_{i,sub(i)} + \sum_{j \in P_{i,sub(i)}} t_{j,sub(j)}) / ct \right\rceil$ (3)
4	LW	Latest Workstation	$LW_i = m_{max} + 1 - \left\lceil (t_{i,sub(i)} + \sum_{j \in S_{i,sub(i)}} t_{j,sub(j)}) / ct \right\rceil$ (4)
5	N	Task Number	i
6	Sk	Slack	$Sk_i = LW_i - EW_i$ (5)
7	TLW	Task Time divided by Latest Workstation	$TL_i = t_{i,sub(i)} / LW_i$ (6)
8	IS	Number of Immediate Successors	$IS_i = \lvert SI_i \rvert$ (7)
9	TS	Number of Total Successors	$TS_i = \lvert S_i \rvert$ (8)
10	TTS	Task Time plus Total Number of Successors	$TTS_i = t_{i,sub(i)} + TS_i$ (9)
11	STS	Average Time of Successors	$STS_i = (\sum_{j \in S_{i,sub(i)}} t_{j,sub(j)}) / TS_i$ (10)
12	TSSk	Number of Total Successors divided by the Slack	$TSSk_i = TS_i / (Sk_i + 1)$ (11)
13	LWTS	Average Latest Workstation	$LWTS_i = LW_i / (TS_i + 1)$ (12)

Table 1. Priority rules used to form the probability distributions for tasks

The combination of the resulting probability distributions, based on the various priority rules used for tasks and subgraphs, produced 39 constructive methods, which are presented in Table 2.

2.2 The local search phase

Two different neighbourhood search strategies based on task exchange movements are considered. At this point, it is valid to recall that a solution to the problem is represented by a sequence of tasks.

The following notation is used to describe such strategies:

m_k Number of workstations required for a given sequence (solution) k
IS Initial tasks sequence generated in the construction phase
WS Working sequence (the first *WS* is *IS*)
SS Stored sequence (the first *SS* is *IS*)
NS Neighbour sequence
Slk_j Slack of workstation j (i.e., cycle time minus workstation time)
α Weight parameter

Weighted rules for tasks	Weighted rules for subgraphs					
	NP		*TT*		*NT*	
	No.	Heuristic Name	No.	Heuristic Name	No.	Heuristic Name
RPW	1	W-NP_RPW	14	W-TT_RPW	27	W-NT_RPW
T	2	W-NP_T	15	W-TT_T	28	W-NT_T
EW	3	W-NP_EW	16	W-TT_EW	29	W-NT_EW
LW	4	W-NP_LW	17	W-TT_LW	30	W-NT_LW
N	5	W-NP_N	18	W-TT_N	31	W-NT_N
Sk	6	W-NP_Sk	19	W-TT_Sk	32	W-NT_Sk
TLW	7	W-NP_TLW	20	W-TT_TLW	33	W-NT_TLW
IS	8	W-NP_IS	21	W-TT_IS	34	W-NT_IS
TS	9	W-NP_TS	22	W-TT_TS	35	W-NT_TS
TTS	10	W-NP_TTS	23	W-TT_TTS	36	W-NT_TTS
STS	11	W-NP_STS	24	W-TT_STS	37	W-NT_STS
TSSk	12	W-NP_TSSk	25	W-TT_TSSk	38	W-NT_TSSk
LWTS	13	W-NP_LWTS	26	W-TT_LWTS	39	W-NT_LWTS

Table 2. Heuristic methods used in the construction phase

The proposed local optimization procedures generate the neighbourhood of the working sequence *WS* by using a transformation or exchange movement (which are described in what follows). Each exchange generates a neighbour sequence *NS,* which is evaluated and compared with the stored sequence *SS*. If *NS* improves the stored sequence *SS* (i.e., it requires fewer workstations) the neighbour sequence becomes the stored sequence *SS*.

When a neighbour sequence requires the same number of workstations as the store sequence (situation that may frequently occur in line balancing problems), a secondary objective function (13) is used as tie-breaker. This function creates solutions by attempting to load the first workstations at maximum capacity and the last ones at minimum capacity. To achieve this objective, the weight parameter α of *f* has been set to 10. It was verified that equivalent results can be obtained by using $\alpha=10^e$, being *e* an integer greater than 1.

$$min\ f = \sum_{j=1}^{m_k} \alpha^j \cdot Slk_j \tag{13}$$

The local search ends when, for each task in *WS*, all feasible exchanges have been made; i.e., all neighbours have been generated and evaluated. For the next iteration, the stored sequence *SS* is assigned to the working sequence *WS*. The entire procedure is repeated until a predetermined computing time has been completed. The final solution is the best of all solutions that have been generated.

Exchange Movements

An adaptation of two classical transformations (see (Armentano & Basi, 2006)) has been considered to generate the neighbourhood of a given solution, as follows.

a. The exchange of the positions in WS of a pair of tasks

In this case, the exchange movement tries to exchange the position, in the working sequence *WS,* of any two tasks *i* and *k*, provided it is feasible; i.e., the precedence relations amongst

the tasks are maintained. Furthermore, both task *i* and task *k* should have been assigned to different workstations. When task *i* and task *k* belong to the same subgraph *s*, new neighbour sequences are searched by interchanging *s* with each one of the remaining subgraphs available for such tasks.

b. The movement of task *i* to another position of the working sequence WS

In this type of movement a task is yielded to a different workstation. A task *i* can be moved to the position of task *k* when the tasks precedence relations are maintained and when task *k* and task *i* have been assigned to different workstations. In this case, all tasks between the positions of task *i* and *k,* including task *k* itself, are moved in the sequence one position backwards. Furthermore, for each movement, neighbour sequences are generated by interchanging the alternative subgraphs available for the moved task.

When a movement exchange type *a* is used in the local search phase, the local optimization procedure is regarded as LOP-1; otherwise, it is regarded as LOP-2.

Examples of exchange movements

Let consider an example (see (Capacho *et al*., 2009)) of an ASALB problem involving 11 tasks and 7 alternative subgraphs, which represent the assembly variants that are allowed for three parts of a manufacturing system: alternative S1 and S2, for subassembly one; S3 and S4, for the second part; and S5, S6 and S7, for the third subassembly. Then, if the *NR* (i.e., minimum number of tasks) is the rule applied, subgraphs S1, S3 and S5 are selected for subassembly 1, 2 and 3, respectively. By choosing such subgraphs, the precedence relations presented in Table 3 are determined.

task	1	2	3	4	5	6	7	8	9	10	11
Immediate predecessors	-	-	-	1, 2	3	2, 5	4	6	7	8	9

Table 3. Resulting precedence relations by considering rule *NT* for selecting subgraphs.

Then, by applying rule *RPW* for selecting the next task to be assigned, the solution presented in Table 4 is obtained. Table 4 includes the number of required workstations *m*, the tasks assigned to each workstation, and the corresponding workstation time *wt*.

m=6	workstation					
	I	II	III	IV	V	VI
Tasks	2, 1	3	4, 5	6, 7	8, 10	9, 11
wt	11	17	16	16	15	15

Table 4. Task assignment to the workstations by applying rule *RPW* .

As can be observed in Table 4, tasks are assigned to the workstations in a particular order, which defines the tasks sequence that is used as the initial sequence *ISq*, as follows:

ISq = 2, 1, 3, 4, 5, 6, 7, 8, 10, 9, 11.

If a transformation type *a* is applied, the neighbour solution of Figure 1 is obtained by interchanging the positions of task 2 and task 3. This movement is possible since, as can be seen in Table 3, neither task 2 nor task 1 are predecessors of task 3, and neither task 1 nor task 3 are successors of task 2 (i.e., the precedence constraints are maintained). Moreover, as shown in Table 4, the tasks are assigned to different workstations: task 2 is assigned to workstation I and task 3 is assigned to workstation II.

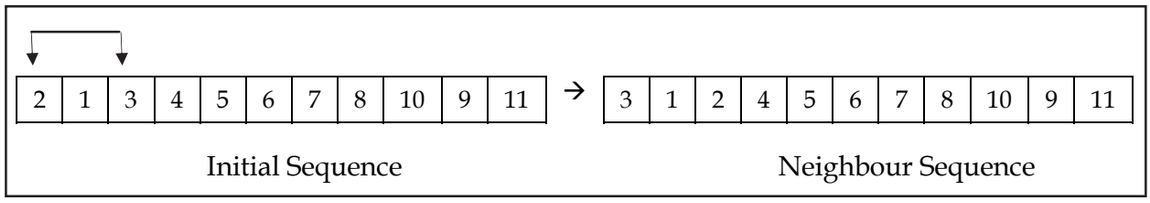


Fig. 1. Generation of a neighbour sequence applying transformation *a*.

If transformation *b* is considered, the neighbour sequence is generated by moving task 2 to the position of task 3; and by moving task 1 and task 3 one position backwards in the sequence. The resulting neighbour sequence is as follows.

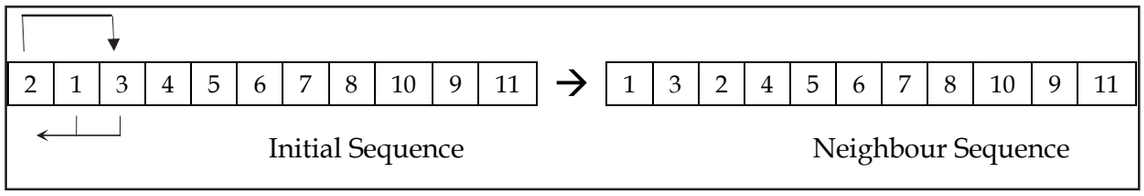


Fig. 2. Generation of a neighbour sequence applying transformation *b*.

3. Computational experiment

To evaluate and compare the performance of the proposed GRASP procedures described in the previous section, a computational experiment was carried out considering medium- and large-scale ASALBP. The data sets (see Table 5) used in the computational experiment are base on the following 10 benchmark problems: Gunther, Kilbrid, Hann, Warnecke, Tongue, Wee-Mag, Lutz3, Arcus2, Bartholdi and Scholl; with 35, 45, 53, 58, 70, 75, 89, 111, 149 and 297 tasks, respectively. Each benchmark problem is subdivided into two, three and four subassemblies; involving five, eight and eleven subgraphs, respectively. For each problem instance five cycle time values were considered. Benchmarks are available online at the web page for assembly line balancing research: www.assembly-line-balancing.de.
A total of 150 problem instances, involving from 37 to 305 tasks, were solved considering a computation time of 0.1 second on a Pentium IV, 3GHZ CPU with 512 Mb of RAM with each of the 78 proposed algorithms. All heuristic methods were implemented using C++.

4. Analysis of the results

To present the results obtained in the computational experiment, the following notation is used: *NI*, number of the tested instances; *CT*, computation time; *NBS*, number of best solutions obtained; *PBS*, percentage of best solutions; Δ_{max} , Δ_{av} , Δ_{min}, maximal, average and minimal deviation from the best solution, respectively. For each problem instance, the relative deviation from the best solution *BS*, of each heuristic solution *HS*, is computed as follows:

$$\Delta = \frac{HS - BS}{BS} \times 100 \tag{14}$$

Problem	Cycle time values					Number of subgraphs		
	ct_1	ct_2	ct_3	ct_4	ct_5	5	8	11
						Number of tasks		
Bowman	20	-	-	-	-	10	-	-
Mansor	48	62	94	-	-	11	-	-
Mitchell	14	21	35	-	-	21	21	-
Buxey	30	36	54	-	-	29	29	-
Gunther	41	44	49	61	81	37	37	37
Kilbrid	57	79	92	138	184	45	46	48
Hahn	2004	2338	2806	3507	4676	56	56	63
Warnecke	54	62	74	92	111	63	63	67
Tonge	160	176	207	251	320	73	75	75
Wee-Mag	28	33	39	46	56	77	81	83
Lutz3	75	83	97	118	150	93	98	101
Arcus2	5785	6540	7916	9400	11570	115	121	125
Bartholdi	403	470	564	705	805	151	157	160
Scholl	75	83	97	118	150	299	302	305

Table 5. Data Sets

The overall performance of all methods is evaluated by considering the number of best solutions provided by the methods. The best solution for a problem instance, the basis for the comparative analysis, is the best of all solutions found by the proposed heuristic methods. The results are also compared with previous results obtained by methods using nominal, rather than weighted, values of the priority rules (e.g., Capacho *et al.*, 2009).

Table 6 shows the results obtained by applying all proposed procedures to solve all data sets. As observed in Table 6, better results were obtained by methods using local procedure LOP-2, which in most cases outperformed methods applying LOP-1; i.e., 24 methods performed better, in two cases both performed the same, and for 3 procedures it behave worse that LOP-1. On the other hand, all methods using LOP-2 provided best solutions in more than 54% of the cases.

The best performance was recorded for the method that employed W-NT_TSSk in the construction phase and applied LOP-2; which generated best solutions in 85,3% of the cases (this represents a 3.3% of improvement comparing with the same method when applying LOP-1). Furthermore, method W-NT_TSSk yielded a comparatively small value of Δ_{max} (16.7%), and the smallest value of Δ_{av} (1%).

Regarding LOP-1, the method that performed the best was W-NT_TSSk, which generated best solutions in 82,7% of the cases. Method W-NT_TSSk performed the same, generating best solutions in 82.7% of the cases regardless of the local optimization procedure applied.

Other methods that also performed well are those that employed W-TT_LWTS, W-TT_LWTS, W-NT_TS, W-TT_TS and applied LOP-2, all of which generated the best solutions in 78.7% of cases. Table 6 also shows that for most methods Δ_{max} is significantly high.

As it can be observed in Table 6, priority rule *EW* has a very poor performance, regardless of the rule used for subgraphs and the local optimization procedure applied; generating best solutions, at best, in 56 % of the cases.

Method		LOP-1				LOP-2			
		NBS	*PBS*	Δ_{max}	Δ_{av}	*NBS*	*PBS*	Δ_{max}	Δ_{av}
1	W-NP_RPW	112	74.7	16.7	1.9	114	76.0	16.7	1.8
2	W-NP_T	92	61.3	33.3	3.1	94	62.7	33.3	2.9
3	W-NP_EW	75	50.0	33.3	4.0	85	56.7	33.3	3.7
4	W-NP_LW	74	49.3	33.3	3.8	85	56.7	33.3	3.6
5	W-NP_N	85	56.7	25.0	3.2	90	60.0	25.0	2.9
6	W-NP_Sk	74	49.3	33.3	3.7	83	55.3	33.3	3.4
7	W-NP_TLW	96	64.0	33.3	3.0	98	65.3	33.3	2.8
8	W-NP_IS	87	58.0	33.3	3.3	90	60.0	33.3	3.1
9	W-NP_TS	111	74.0	16.7	1.7	113	75.3	16.7	1.6
10	W-NP_TTS	93	62.0	25.0	2.7	94	62.7	25.0	2.6
11	W-NP_STS	76	50.7	33.3	3.8	81	54.0	33.3	3.5
12	W-NP_TSSk	109	72.7	16.7	1.8	109	72.7	16.7	1.7
13	W-NP_LWTS	111	74.0	16.7	1.9	112	74.7	16.7	1.7
14	W-TT_RPW	113	75.3	16.7	1.4	116	77.3	16.7	1.4
15	W-TT_T	99	66.0	33.3	2.5	100	66.7	33.3	2.3
16	W-TT_EW	80	53.3	33.3	3.3	84	56.0	33.3	3.0
17	W-TT_LW	85	56.7	33.3	3.2	88	58.7	33.3	3.0
18	W-TT_N	94	62,7	25.0	2.6	96	64.0	25.0	2.3
19	W-TT_Sk	83	55.3	33.3	3.3	89	59.3	33.3	3.0
20	W-TT_TLW	102	68.0	33.3	2.3	104	69.3	33.3	2.1
21	W-TT_IS	97	64.7	25.0	2.3	100	66.7	25.0	2.0
22	W-TT_TS	116	77.3	16.7	1.3	118	78.7	16.7	1.1
23	W-TT_TTS	105	70.0	25.0	2.0	106	70.7	25.0	1.8
24	W-TT_STS	88	58.7	33.3	2.8	90	60.0	33.3	2.6
25	W-TT_TSSk	124	82.7	16.7	1.0	124	82.7	16.7	1.0
26	W-TT_LWTS	119	79.3	16.7	1.3	118	78.7	16.7	1.3
27	W-NT_RPW	116	77.3	16.7	1.4	113	75.3	16.7	1.4
28	W-NT_T	99	66.0	33.3	2.5	100	66.7	33.3	2.3
29	W-NT_EW	79	52.7	33.3	3.4	83	55.3	33.3	3.2
30	W-NT_LW	85	56.7	33.3	3.2	88	58.7	33.3	3.0
31	W-NT_N	93	62.0	25.0	2.6	95	63.3	25.0	2.4
32	W-NT_Sk	83	55.3	33.3	3.3	89	59.3	33.3	3.0
33	W-NT_TLW	102	68.0	33.3	2.3	104	69.3	33.3	2.1
34	W-NT_IS	96	64.0	33.3	2.5	99	66.0	33.3	2.2
35	W-NT_TS	116	77.3	16.7	1.3	118	78.7	16.7	1.1
36	W-NT_TTS	105	70.0	25.0	2.0	106	70.7	25.0	1.8
37	W-NT_STS	88	58.7	33.3	2.8	90	60.0	33.3	2.6
38	W-NT_TSSk	123	82.0	16.7	1.0	128	85.3	16.7	1.0
39	W-NT_LWTS	119	79.3	16.7	1.3	118	78.7	16.7	1.3
Δ_{min}= 0 in all cases									

Table 6. Results of applying the GRASP procedures (*NI* = 150)

On the other hand, Table 6 also shows that methods employing different local search procedures behave very similarly when the same heuristic method is used to build the initial solution. This means that when a constructive method performs well when applying LOP-1, it also performs well (or better) when applying LOP-2. This behaviour can also be observed in Figure 3, which shows the percentage of improvement (in one workstation) of the proposed procedures comparing with other multi-pass methods.

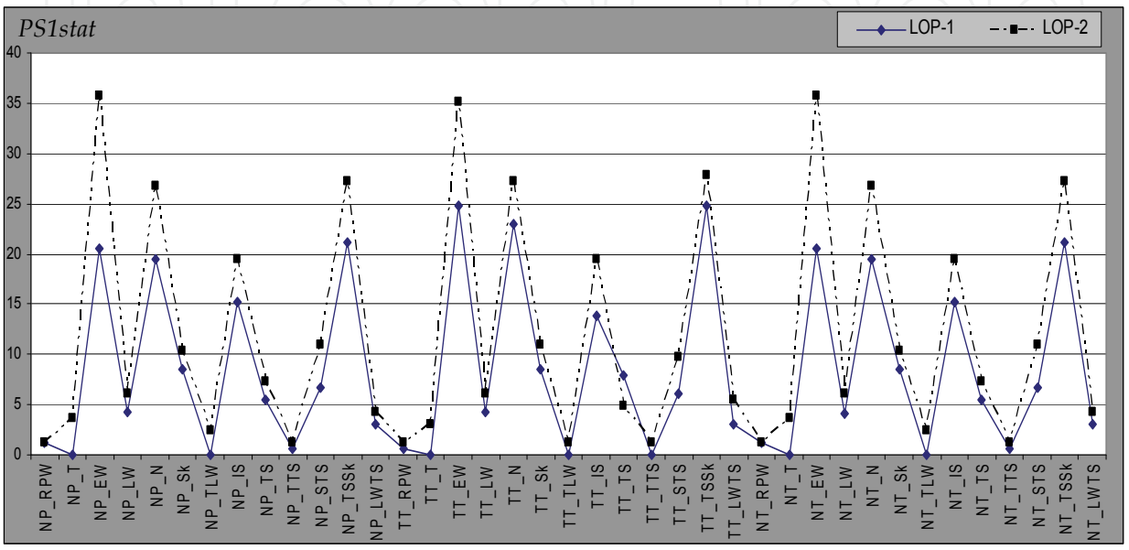


Fig. 3. Performance of GRASP procedures

The results obtained with the proposed GRASP methods were also compared with the results obtained by multi-pass methods, in which the assembly subgraphs are randomly selected and tasks are assigned according to nominal values of the priority rules of Table 1. Table 7 shows the percentages of improvement in the solution provided by the proposed GRASP methods comparing with multi-pass ones (e.g., Capacho *et al.*, 2009), considering a CT=0.1 seconds. Data in Table 7 includes Imp_{max}, Imp_{av}, Imp_{min}, maximal, average, and minimum percentage of improvement, respectively. It also shows the percentage of cases in which the best solution found requires 1, 2 or 3 workstations less (%1*ws*, %2*ws*, %3*ws*), respectively, than the best result provided by the corresponding multi-pass methods. As can be seen in Table 7, the best results were obtained when LOP-2 was applied. This strategy provided an additional 12.9% of best solutions, getting in some cases up to 35.6%. Table 7 also reveals that both types of procedures were able to generate improved solutions in which up to three fewer workstations were required; i.e., in 0.1% and 0.3% of the cases with LOP-1 and LOP-2, respectively.

5. Conclusions and further research work

In this chapter, the metaheuristic approach GRASP was used to solve the Alternative Subgraphs Assembly Line Balancing Problem (ASALBP). Thirty-nine construction methods,

based on weighted priority rule values, and two local search strategies (LOP-1 and LOP-2) were proposed, implemented and evaluated.

Local method	Imp_{max}	Imp_{av}	Imp_{min}	% 1 *ws*	% 2 *ws*	% 3 *ws*
LOP-1	24.9	6.8	0.2	2.2	0.3	0.1
LOP-2	35.6	12.9	2.7	3.6	0.6	0.3

Table 7. Comparison of nominal- and weighted- rule based methods

The results obtained showed that methods that used LOP-2 performed better than those that used LOP-1, achieving best solutions in up to 85.3% of all cases, considering all the proposed construction methods. Nevertheless, some methods applying LOP-1 (i.e., W-NT_TSSk and W-TT_TSSk) also performed well, providing best solutions in up to 82.7%.

The results also showed that a significant improvement can be obtained in comparison to previous results obtained using multi-pass methods based on single priority rule values and using random choice for subgraphs. Improved solutions were obtained in which the number of workstations required was reduced by one, two or even three, which represent the best results obtained with any method developed up to now to solve the ASALBP. Thus, all proposed methods that used LOP-2 could be applied to solve an Alternative Subgraphs Assembly Line Balancing Problem to select the best overall solution.

Further research involves exploring other methods to solve the ASALBP. The growing interest on using Evolutionary Algorithms to solve optimization problems in industry makes the use of such procedures an attractive approach, which, in addition, has been successfully applied to complex assembly line balancing problems. On the other hand, in order to increase the practicality of the problem, its definition can be extended by including new features such as, for example, stochastic processing times, setups, and different line layouts.

6. Acknowledgment

Supported by the University of Los Andes, Mérida – Venezuela.

7. References

Andrés, C., Miralles, C. & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, Vol. 187, No. 3, pp. 1212-1223.

Armentano, A. & Bassi, O. (2006). Graph with memory-based mechanisms for minimizing total tardiness in single machine scheduling with setup Times. *Journal of Heuristics*, Vol. 12, pp. 427-446.

Becker, C. & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, Vol. 168, pp. 694-715.

Capacho, L. (2007). *ASALBP: the Alternative Subgraphs Assembly Line Balancing Problem. Formalization and resolution procedures*. PHD thesis. Tecnical Universtity of Catalunya. Barcelona, Spain.

Capacho L. & Pastor, R. (2006). The ASALB problem with processing alternatives involving different tasks: definition, formalization and resolution. *Lecture Notes in Computer Science*, Vol. 3982, pp. 554-563.

Capacho, L. & Pastor, R. (2008). ASALBP: The alternative subgraphs assembly line balancing problem. *International Journal of Production Research*, Vol. 46, pp. 3503-3516.

Capacho, L., Pastor, R., Dolgui, A. & Guschinskaya, O. (2009). An evaluation of constructive heuristic methods to solve the alternative subgraphs assembly line balancing problem. *Journal of Heuristics*, Vol. 15, pp. 109-132.

Capacho, L., Pastor, R., Guschinskaya, O. and Dolgui, A. (2006). Heuristic Methods to Solve the Alternative Subgraphs Assembly Line Balancing Problem. *Proceedings of the IEEE Conference on Automation Science and Engineering - CASE 2006*, pp. 501-506, ISBN: 1-4244-0310-38-11, Shanghai-China. October 7-10, 2006.

Das, S. & Nagendra, P. (1997). Selection of routes in a flexible manufacturing facility. *International Journal of Production Economics*, Vol. 48, pp. 237–247.

Delorme, X., Gandibleux, X. & Rodriguez, J. (2004). GRASP for set packing problems, *European Journal of Operational*, Vol. 153, pp. 564-580.

Dolgui, A., Eremeev, O. & Guschinskaya, O. (2010). MIP-based GRASP and genetic algorithm for balancing transfer lines. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*, Maniezzo, V., Stutzle, T., Voss, S. (eds.), Annals of Information Systems, Springer, 10, 189-208.

Falkenauer, E. (2005). Line Balancing in the Real World. *Proceedings of the International Conference on Product Lifecycle Management* PLM´05, 360 - 370.

Festa, P. & Resende, M. (2008a). An annotated bibliografy of GRASP. Part I: Algoritms. Technical Report. AT&T Labs Research. Available online at www.optimization-online.org/DB_FILE/2008/06/2011.pdf. Visited: 01.03.11.

Festa, P. & Resende, M. (2008b). A*n annotated bibliografy of GRASP. Part II: Aplications. Technical Report*. AT&T Labs Research. 01. 03.11. Available online at: www2.research.att.com/~mgcr/doc/gabib-appl.pdf. Visited: 01.03.11.

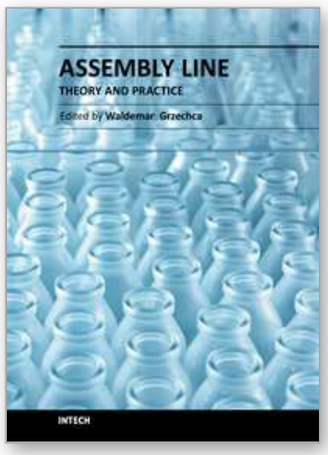
Martino, L. & Pastor, R. (2007). *Heuristic procedures for solving the General Assembly Balancing Problem with Setups* (*GALBPS*). Tecnical Report, IOC-DT-P-2007-15. Tecnical Universtity of Catalunya.

Resende, M. & Ribeiro, C. (2003). Greedy Randomized Adaptive Search Procedures. Handbook of Metaheuristics. *International Series in Operations Research & Management Science*, Vol. 57, pp. 219-249, DOI: 10.1007/0-306-48056-5_8

Scholl, A., Boysen, N. & Fliedner, M. (2008). The sequence-dependent assembly line balancing problem. *Operations Research Spectrum*, Vol. 30, pp. 579-609.

Scholl, A., Boysen, N. & Fliedner, M. (2009). Optimally solving the alternative subgraphs assembly line balancing problem. *Annals of Operations Research*, vol. 172, No.1, pp. 243-258.

Talbot, F.B, Patterson, J.H. & Gehrlein, W.V. (1986). A comparative evaluation of heuristic line balancing techniques. *Management Science,* Vol. 32, pp. 431-453.



Assembly Line - Theory and Practice
Edited by Prof. Waldemar Grzechca

ISBN 978-953-307-995-0
Hard cover, 250 pages
Publisher InTech
Published online 17, August, 2011
Published in print edition August, 2011

An assembly line is a manufacturing process in which parts are added to a product in a sequential manner using optimally planned logistics to create a finished product in the fastest possible way. It is a flow-oriented production system where the productive units performing the operations, referred to as stations, are aligned in a serial manner. The present edited book is a collection of 12 chapters written by experts and well-known professionals of the field. The volume is organized in three parts according to the last research works in assembly line subject. The first part of the book is devoted to the assembly line balancing problem. It includes chapters dealing with different problems of ALBP. In the second part of the book some optimization problems in assembly line structure are considered. In many situations there are several contradictory goals that have to be satisfied simultaneously. The third part of the book deals with testing problems in assembly line. This section gives an overview on new trends, techniques and methodologies for testing the quality of a product at the end of the assembling line.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen