Solving a Two-Sided Assembly Line Balancing Problem using Memetic Algorithms*

Solución de un problema de balanceo de línea de ensamble serial de dos lados utilizando algoritmos meméticos**

Solução de um problema de balanço de linha de montagem serial de dois lados utilizando algoritmos meméticos***

*Óscar Rubiano-Ovalle*****
*Alonso Arroyo-Almanza******

* Submitted on March 29, 2009. Accepted on July 16, 2009. This article is derived from the Master's Graduation Work, by the second author, carried out as part of the studies by the Research Group on Logistics and Production at the School of Industrial Engineering and Statistics in Universidad del Valle.

** Fecha de recepción: 29 de marzo de 2009. Fecha de aceptación para publicación: 16 de julio de 2009. Este artículo se deriva del trabajo final de maestría del segundo autor, realizado en el marco del trabajo del grupo de investigación en Logística y Producción, Escuela de Ingeniería Industrial y Estadística, Universidad del Valle.

*** Data de recepção: 29 de março de 2009. Data de aceitação para publicação: 16 de julho de 2009. Este artigo deriva do trabalho final de mestrado do segundo autor, realizado no marco do trabalho do grupo de pesquisa em Logística e Produção, Escola de Engenharia Industrial e Estatística, Universidade do Valle.

**** Ingeniero industrial, Universidad del Valle, Cali, Colombia. Doctorado en Ingeniería Industrial, Universidad de Sevilla, España. Profesor titular de la Escuela de Ingeniería Industrial y Estadística, Universidad del Valle.
Correo electrónico: oscaruba@pino.univalle.edu.co.

***** Físico, Universidad del Valle, Cali, Colombia. Magíster en Ingeniería con énfasis en Ingeniería Industrial, Universidad del Valle. Profesor del Departamento de Ciencias Naturales y Matemáticas, Pontificia Universidad Javeriana, Cali, Colombia. Correo electrónico: alrroyo@puj.edu.co.

Abstract

This research explores a characterization of the Two-Sided Assembly Line Problem (TALBP). There is a growing interest among researchers and assembly line practitioners in the solution to this problem, because it is more related to real-life situations than the Simple Assembly Line Balancing Problem (SALBP). Since the complexity of the TALBP is superior to that of the SALBP, this research emphasizes both the construction and the use of metaheuristics as memetic algorithms for finding a very good solution. Memetic algorithms are supported by genetic algorithms. The solution proposed was implemented in Matlab for a motorcycle assembly line at a local firm. Compared to other recognized heuristics and optimization methods, a most suitable solution was obtained in a shorter time through the use of the constructed algorithm. With this, all the restrictions and complexities inherent to the problem were overcome.

Key words

MATLAB (computer program), assembly-line methods, assembly-line balancing, algorithms.

Resumen

En este artículo se presenta una caracterización del *problema de balanceo de línea de ensamble serial de dos lados* (TALBP, por su sigla en inglés). Existe un creciente interés en este tipo de problemas, pues representan situaciones que se acercan mucho más a la vida real, que los problemas de líneas de ensamble serial simple (SALBP, por su sigla en inglés). Debido a que la complejidad del problema TALBP es superior a la del problema SALBP, en esta investigación se destaca la construcción y uso de metaheurísticas como algoritmos meméticos, para encontrar una muy buena solución. Los algoritmos meméticos se apoyan en los algoritmos genéticos. La solución propuesta fue implementada en el programa Matlab, para una línea de ensamble de motos en una empresa local. Mediante el uso del algoritmo construido se obtuvo una solución óptima en un tiempo más corto, frente a otras reconocidas heurísticas y métodos de optimización, que superaron todas las restricciones y complejidades inherentes al problema.

Palabras clave

MATLAB (programa para computador), métodos de líneas de ensamblado, equilibrio de línea de montaje, algoritmos.

Resumo

Neste artigo apresenta-se uma caracterização do *problema de balanço de linha de montagem serial de dois lados* (TALBP, pela sua sigla em inglês). Existe um interesse crescente neste tipo de problemas, pois representam situações que se aproximam muito mais à vida real, que os problemas de linhas de montagem serial simples (SALBP, pela sua sigla em inglês). Devido que a complexidade do problema TALBP é superior à do problema SALBP, nesta pesquisa destaca-se a construção e uso de meta-heurísticas como algoritmos meméticos, para encontrar uma solução muito boa. Os algoritmos meméticos apóiam-se nos algoritmos genéticos. A solução proposta foi implementada no programa Matlab, para uma linha de montagem de motos em uma empresa local. Mediante o uso do algoritmo construído obteve-se uma ótima solução em um tempo menor, comparado com outras reconhecidas heurísticas e métodos de otimização, que superaram todas as restrições e complexidades inerentes ao problema.

Palavras chave

MATLAB (programa para computador), métodos de linhas de montagem, equilíbrio de linha de montagem, algoritmos.

Introduction

One of the most studied line balancing problems in the last decade is the balancing of the two-sided assembly line known as TALBP in the related literature. Researchers such as (Lee *et al.*, 2001; Simaria y Vilarinho, 2007; Kim *et al.*, 2009; Xiaofeng *et al.*, 2008; Ozcan y Toklu, 2009) have proposed many solutions both exact and metahueristic implemented in programming languages such as C, Pascal, Visual Basic, Access 97, etc. In these languages, the programming codes and data bases require that designers have great expertise in the software thus discouraging engineers from applying it. This is why this paper explores the construction of a solution using memetic algorithms implemented in MATLAB software. The reason for using memethics algorithms was the existing possibility for integrating them to searching technicals impelemented in other metaheuristics, such as local searching and directed searching, which allow obtain solutions much more versatile in comparison to those obtained by a "pure" genetic algorithm (Moscato y Cotta, 2003).

The reasons for using MATLAB are: (a) Ease of adapting the algorithm whenever there is an additional restriction to the problem, (b) convenience in handling the information in matrixes and vectors, and (c) the programming language's great versatility and the facility to show results. As a pilot case, a two-sided assembly line was chosen in a local motorcycle company in Cali, Colombia.

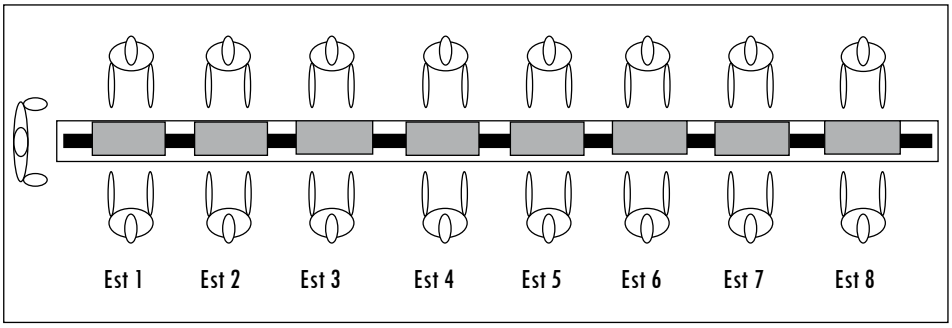
1. Methodology

Initially the two-sided assembly line balance problem was characterized. Then, the general mathematical modeling for TALBP was done to obtain a configuration with the optimum number of stations for a given time cycle. After this, the chosen motorcycle assembly line case was described. Finally, the memetic algorithm applied to the case was developed.

1.1 Characterization of a Two-Sided Assembly Line

In a two-sided assembly line, product flows along a production line with workstations on both sides of the line (Figure 1).

Figure 1. Two-Sided Assembly Line



Source: Jurado y Taborda, 2006.

1.2 TALBP-I Problem Mathematical Model

The model characteristics are the following:

- Objective: Minimize the number of stations for a given cycle time.
- Side restrictions: There are tasks that due to their nature must be assigned to a specific side.
- Precedence restrictions: There are tasks that must be done after others.
- Simultaneity restrictions: These occur when two tasks start at the same time.
- Start time restrictions: When a task is preceded by various tasks; it can not be started if all the other precedent tasks have not been completed. This restriction keeps in mind the termination times of precedent tasks for allowing starting the new task; this makes this type of restrictions different to precedence restrictions.
- Cycle time restrictions: The sum of time for all the tasks assigned to one station must not be greater than the cycle time (equal to Takt time or maximum time allowed to produce a product in order to meet demand).
- Occurrence restrictions: A task can only be assigned to one side and one station must have at least one task assigned.

The following is the assigned nomenclature:

Indexes:
i: one task

j: one station

k: line side; $k = \begin{cases} 1 \ right \ \ side \\ 2 \ \ left \ \ side \\ 3 \ \ any \ \ side \end{cases}$

Parameters:

J: Set of stations; $J = \{1,2,3,\ldots, m\}$

I: Set of tasks; $I = \{1,2,3,\ldots, n\}$

t_i: Processing time of task *i*

L_i: Task side indicator *i*

P(u,v): 1 if *u* precedes *v*; 0, in any other case

S(i,h): 1 if task *i* must be executed simultaneously with task *h*; 0, in any other case

max: Maximum number of workstations

T_{max}: Cycle time (maximum processing time in one station)

n: Number of tasks in the assembly line

Decision variables:

$$y_{jk} = \begin{cases} 1, \ if \ \ work \ \ station \ \ j \ \ \ has \\ a \ \ task \ \ \ on \ \ side \ \ k \\ 0, \ in \ \ \ any \ \ other \ \ case \end{cases}$$

$$x_{ijk} = \begin{cases} 1, \ if \ \ task \ \ i \ \ is \ \ assigned \ \ to \\ station \ \ j \ \ on \ \ side \ \ k \\ 0, \ in \ \ any \ \ other \ \ case \end{cases}$$

tf_i: Finish time of task *i*

m: Number of stations in the line

The model:

Goal: minimize $\sum_{j=1}^{\max} \sum_{k=1}^{3} y_{jk}$

Subject to:

A task can be assigned only to one work station

$$\sum_{j=1}^{m} \sum_{k=1}^{3} x_{ijk} = 1 \qquad \forall i \in I$$

Restriction of task precedence (the task *u* precede the task *v*)

$$\sum_{j=1}^{m}\sum_{k=1}^{3} j * x_{ujk} \leq \sum_{j=1}^{m}\sum_{k=1}^{3} j * x_{vjk} \qquad \forall P(u,v) = 1$$

Dependence restriction among the termination time and the starting time of tasks

$$(tf_i - tf_h)\, y_{jk} \geq t_i,\ tf_i \geq t_i \ \ \forall i \in I, \forall P(i,h) = 1, \forall i,h \in I, \forall j \in J,\ k = 1,2,3$$

Processing time at each station must not exceed the cycle time:

$$\sum_{i=1}^{n}\sum_{k=1}^{3} t_i * x_{ijk} \leq T_{\max} * y_{jk} \qquad \forall j \in J$$

Side restrictions:

$$\sum_{j=1}^{m}\sum_{k=1}^{3} k * x_{ijk} = L_i \qquad \forall i \in I$$

Simultaneity restrictions between tasks:

$$x_{ijk} - x_{hjk} = 0 \quad \forall S(i,h) = 1, \forall i,h \in I, \forall i,h \in I, \quad k = 1,2,3$$

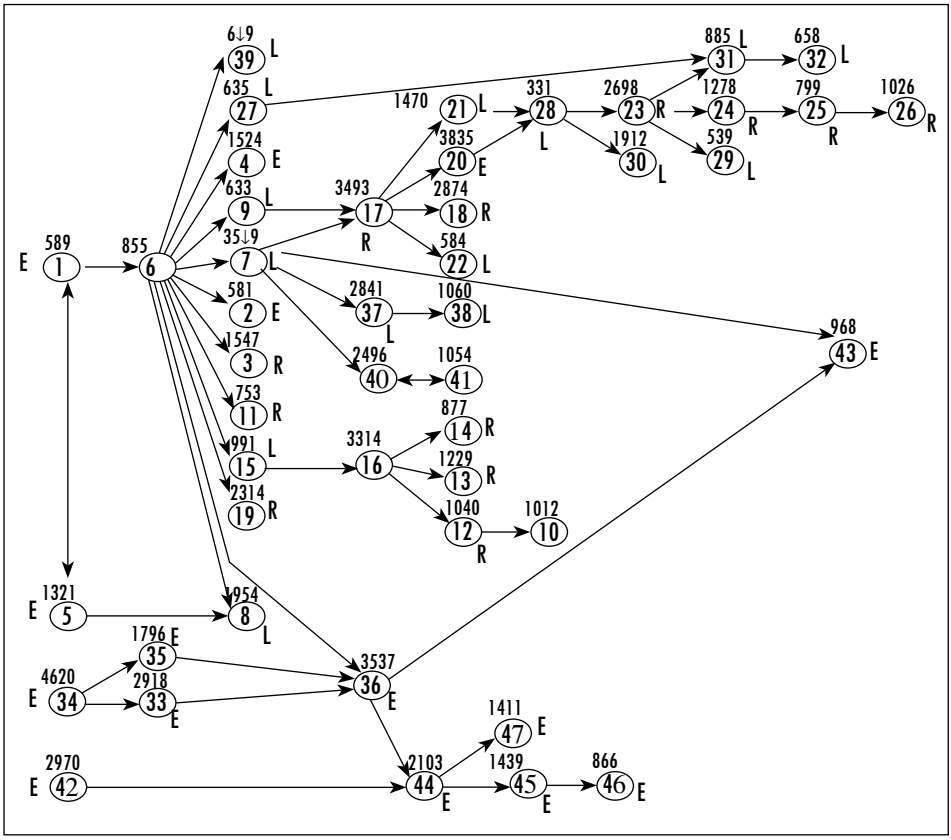
Assures there are no empty stations:

$$y_{j+1,k} \leq y_{jk},\ j = 1,\ldots, \max;\ k = 1,2,3$$

1.3 Precedence Diagram, Legend, and Initial Model Data

The addressed assembly process goes from the placement of the chassis on the conveyor system to the assembly of parts (Figure 2). The numbers in each circle correspond to the tasks, and the processing times are the values written outside the circles. Precedence between tasks is indicated by arrows with one direction. Simultaneity among tasks is shown by arrows with two directions. The side on which a task is to be executed in a specific station is designated by the letter outside the circle (L: left side, R: right side, E: either side).

Figure 2. Configuration and Precedence for the 47 Tasks in the Pilot Case



Source: Authors' elaboration.

In the pilot case's current state the sum of the processing time for all the tasks is 77,875 hundredths of seconds. The daily demand is 300 units and the set cycle time is 10,200 hundredths of seconds.

1.4 Memetic Algorithm

The memetic algorithm's global structure is constructed by a methodological solution to obtain the optimum number of stations in the assembly line by assigning tasks to stations (see Figure 3). This algorithm was implemented in MATLAB software version 7.1.

Figure 3. Memetic Algorithm Implemented to Balance a Two-Sided Assembly Line (Main Algorithm)

```
FUNCION A_M
/ Initial population's generation/
pop ← initial_ population ()
/Task assignment /
asig ← asigna_tareas(pop)
Repeat
/Generate new population/
n_pop ← new_ population (pop)
/Task assignment /
asig ←assignment_t(n_pop)
/ Evaluation of the assignment/
f_n ← aptitude(asig);
If f_n < 10,000;
/Save solution/
Otherwise
If f_n > 20,000
/discard solution/
exit
End_if
End_if
If optimum_found()
/End cycle/
exit
End_if
Increase iteration
Until number of iterations is less than 20,000
/Show solution/
```

Source: Authors' elaboration.

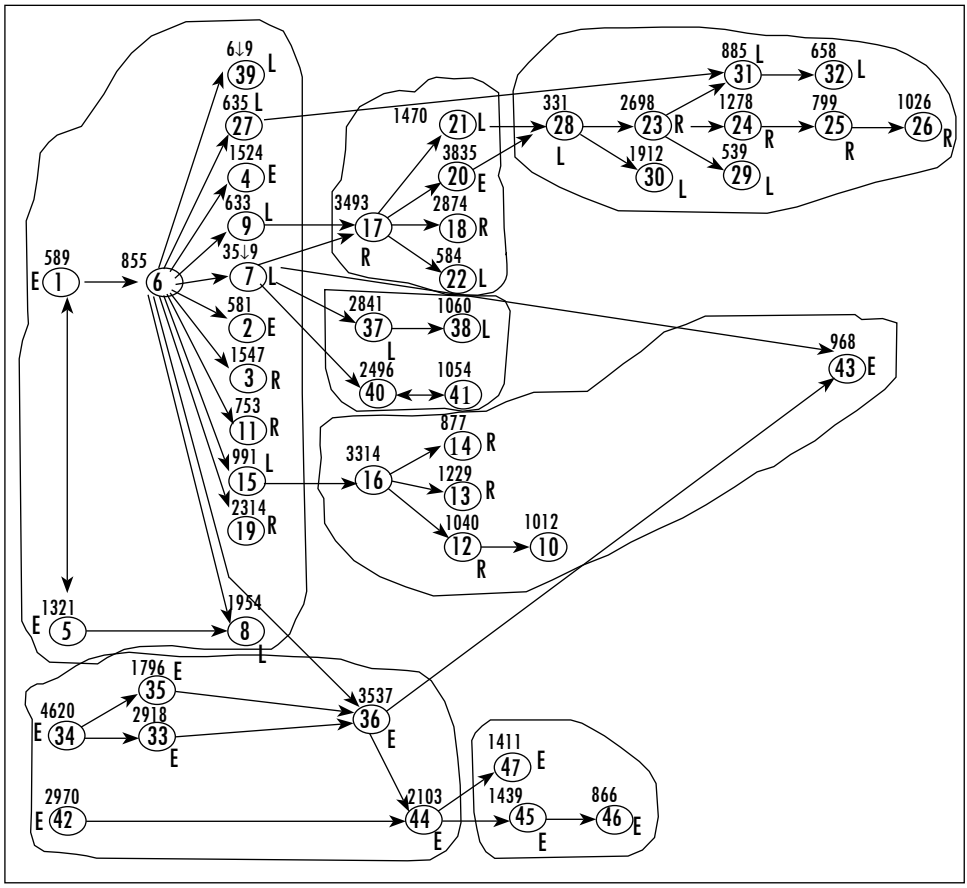
In the following steps the construction of the algorithm is explained.

1.4.1 Initial Population Assignment (Task Assignment)

To carry out the initial population assignment an intentional task assignment procedure was used (Figure 4). This procedure has the advantage of accelerat-

ing the process of reaching the final solution since in the following procedure of exchanging tasks, (generation of new assignments) during the random search process in the proposed algorithm, the majority of previously obtained assignments not involved in each exchange are kept.

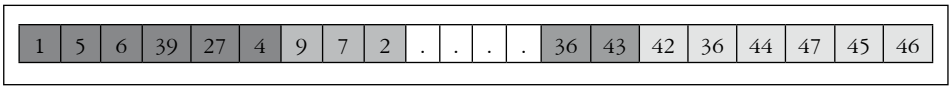
Figure 4. Intentional Task Grouping



Source: Authors' elaboration.

Task grouping is represented in the XT_i arrangement as indicated in Figure 5. This type of plot indicates a specific task grouping for the station. This initial task arrangement (called father assignment), is followed by a station assignment process in line with all the restrictions imposed to reach an initial feasible solution.

Figure 5. XT_i Arrangement



Source: Authors' elaboration.

1.4.2 Evaluation of initial assignment

The initial assignment and any of the generated solutions are evaluated through the proposed aptitude function by (Sabuncuoglu and Tanyer, 2001) shown in the following equation:

$$Aptitude\ \ function = 2\sqrt{\frac{\sum_{k=1}^{n}(T_{\max} - S_k)^2}{n}} + \frac{\sum_{k=1}^{n}(T_{\max} - S_k)}{n}$$

Where n is the number of stations, T_{max} is the cycle time, and S_k is the processing time for the nth k station.

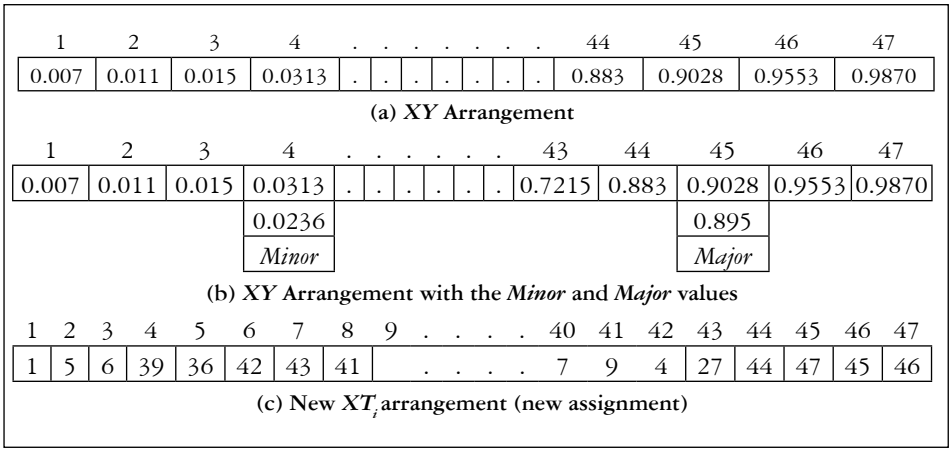
1.4.3 Generation of New Assignments (Sons)

It is done with the following random procedure:

- When n is the total number of tasks to be programmed, n random numbers are generated with uniform distribution between 0 and 1. These random numbers are ordered from least to greatest value thus forming the *XY* arrangement depicted in Figure 6a.

- Additionally, two random numbers between 0 and 1 are generated. The lower value number is called *Minor* and the greater value one *Major*, and they are placed in the lower part of the *XY* arrangement position whose value is closest (Figure 6b). Then, the tasks located between the two selected positions (4 and 45) in the XT_i arrangement are exchanged to generate a new assignment as shown in Figure 6c. This new assignment must be submitted to a new precedence check procedure between the tasks. If the precedence is complied with, this assignment replaces the previous one thus becoming the new assignment.

Figure 6. Generation of New Assignments



Source: Authors' elaboration.

1.4.4 Simultaneity Check

The new assignment must be submitted to a checking process for simultaneity between tasks, i.e. it must be assured that two tasks that are executed at the same time must be in consecutive positions in the XT_i arrangement, for example tasks 36 and 43 (see Figure 5). To check for simultaneity, a check and exchange process is carried out between the XT_i arrangement and an arrangement that has all the tasks to be executed simultaneously.

1.4.5 Assignment of Tasks in the New Assignment

The new assignment of tasks to the workstations must be done. In line with the precedence and simultaneity already established in XT_i, the assignment tasks are assigned sequentially to the workstations in such a manner that the cycle time, task execution side and interval execution times are met.

1.4.6 Evaluation of New Assignment

The new assignment is submitted to an evaluation in accordance to the aptitude function defined in step 1.4.2. If the value of the aptitude function for this new assignment is less than that of the current father assignment, this new assignment replaces the father assignment.

1.4.7 Stop Condition

The new assignment generation process (sons) is repeated as many times as necessary until one of the following conditions is met: (i) the optimum number of stations is obtained, (ii) the number of iterations surpasses an established maximum, or (iii) a satisfactory value determined by the aptitude function is achieved. If any of the two last conditions is attained, an approximate solution has been found but not the optimum one.

2. Results and Discussion

The results are given in Table 1. An optimum number of workstations equal to 4 was obtained with a workstation time of 78,697 hundredth of seconds (13.116 minutes) and a idle time of 2,903 hundredth of seconds (0.483 minutes).

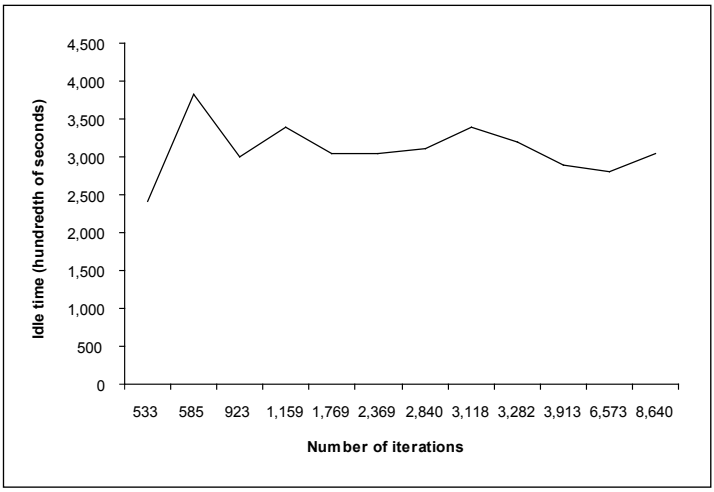
Table 1. Balancing of Two-Sided Assembly Line with 47 Tasks Using A Memetic Algorithm

	Number of Tasks										
	Execution Time (Hundredth of Seconds)										
Station 1	Right Side	1	5	34	35	4	11	33		Workstation Time	Idle time
		589	1321	4620	2918	1524	753	2918		9612	588
	Left Side		6	42	2	39	15	8	9		
			855	2970	581	649	991	1954	633	9954	246
Station 2	Right Side	36	44	17	4	47					
		3537	2103	3493	1524	1411				9977	223
	Left Side	7	16	40	41	27					
		3549	3314	2496	1054	635				9994	206
Station 3	Right Side	3	13	14	18	19	12				
		1547	1229	877	2871	2314	1090			9928	272
	Left Side	22	43	21	37	20					
		584	968	1470	2841	3835				9698	502
Station 4	Right Side	45				23	24	25	26		
		1439				2698				10114	86
	Left Side	38	10	28	30	29	31	32	46		
		1060	1012	331	1912	539	885	658	866	9420	780
								Total	:	78697	2903

Source: Authors' elaboration.

The average time to attain the solution was approximately one minute. Related to the number of executed iterations, in 12 trials that were done, a solution was always found, i.e. there were no non convergence problems in the algorithm. Figure 7 shows the algorithm's downtime versus the number of iterations. A idle time variation between 2,408 and 3,824 hundredth of seconds can be observed (0.401 and 0.637 minutes) with an average value of 3,283.25 hundredth of seconds (0.5472 minutes) and a variation coefficient of 12%. This is an acceptable variation.

Figure 7. Idle Time versus Number of Iterations



Source: Authors' elaboration.

3. Conclusions

In this research a memetic algorithm was developed to solve a deterministic TALBP-I problem. In the literature, many researchers present metaheuristic solutions, but none of these solutions bring together all the specificities implemented in the proposed solution. These specificities are: two-sided assembly line with dependence restrictions between tasks on different sides in one station, simultaneity between tasks in the same or different stations, task execution side, cycle time restrictions and empty station control. Additionally, it is executed with a feasible initial solution (assignment). The memetic algorithm developed in MATLAB was applied to a motorcycle assembly line composed of 47 tasks with the objective of minimizing the number of stations to satisfy the Takt time. In this case, the attained solution was optimum and it was reached in a shorter

time than with other heuristics and optimization methods, thus surpassing all the restrictions and complexities inherent to the problem.

References

JURADO, B. y TABORDA, S. *Análisis y mejora de la capacidad a partir del estudio del trabajo y el uso de Promodel como herramienta de simulación discreta en la línea de ensamble Honda motocicletas de la fábrica nacional de autopartes FANALCA S. A. Cali, Colombia, 2006*. [Trabajo de grado]. Cali: Universidad del Valle, 2006.

KIM, Y. K.; SONG, W. S. and KIIM, J. H. A mathematical model and a genetic algorithm for two-sided assembly line balancing. *Computers & Operations Research*, 2009, vol. 36, num. 3, pp. 853-865.

LEE, T.; KIM, Y. and KIM, YK. Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers & Industrial Engineering*, 2001, vol. 40, num. 3, pp. 273-292.

MOSCATO, P. y COTTA, C. Una introducción a los algoritmos meméticos. *Inteligencia Artificial*, 2003, num. 19, pp. 131-148.

OZCAN, U. and TOKLU, B. Multiple-criteria decision-making in two-sided assembly line balancing: a goal programming and a fuzzy goal programming models. *Computers & Operations Research*, 2009, vol. 36, num. 6, pp. 1955-1965.

SABUNCUOGLU, E. and TANYER, M. Assembly line balancing using genetic algorithms. *Journal of Intelligent Manufacturing*. 2000, vol. 11, núm. 3, pp. 295-310.

SIMARIA, A. and VILARINHO, P. 2-Antbal: An ant colony optimization algorithm for balancing two-sided assembly lines. *Computers & Industrial Engineering*, 2007, vol. 56, num. 2, pp. 489-506.

XIAOFENG, H.; ERFEI, W. and YE, J. A station-oriented enumerative algorithm for two-sided assembly line balancing. *European Journal of Operational Research*, 2008, vol. 186, num. 1, pp. 435-440.