A Genetic Algorithm for Reassigning Work on the Assembly Line: A Real Scenario

Sergio Ramirez-Campos[1] and Luis Torres-Trevino[2] and Gaston Cedillo-Campos[2] and Jorge Villegas-Leza[3] and Pedro Perez Villanueva[2]

[1] Instituto Tecnologico de Saltillo, Blvd. V. Carranza 2400, Saltillo Coahuila, Mexico
sramirez@its.mx
[2] Corporacion Mexicana de Investigacion en Materiales, S. A. de C. V. Blvd. Oceania 190, Saltillo Coahuila, Mexico
ltorres@comimsa.com.mx, mgaston@comimsa.com.mx, pperez@comimsa.com.mx
[3] Daimler Chrysler Planta Derramadero Saltillo Coahuila, Mexico
jv@dcx.mx

Abstract. The problem at hand corresponds to a real situation with characteristics that make it a flow shop NP-hard type of combinatorial optimization problem. This is a unidirectional line flow with serial workstations in a sequential order for all products. Additionally, restrictions of precedence, different workstations, number of operators and jobs greater than one per workstation, and zoning restrictions for certain equipment are considered. This scenario makes it highly complex, under an increasing number of work elements, to reassign within the workstations. A simple genetic algorithm to readjust the assembly line subject to perturbations that force the reduction of cycle time was designed. To meet this demand, work elements are reassigned in order to find the minimum number of workstations without exceeding cycle time. Finally, experimental results in a real-life automobile assembly plant indicate the effectiveness and applicability of the proposed approach in practice.

1 Introduction

The term production line is applied to a flow oriented production system, which is still typical in the automotive industry. According to Carnahan *et al.* [1], a production line of this type, is defined as a series of manual or automated assembly workstations through which one or multiple product(s) are sequentially assembled. The workers, equipment, tools and other devices are located in each workstation according to the tasks that are to be carried out in each one of them.

All the components flow toward each workstation at the last possible moment and in the required quantity to avoid unnecessary inventories and consequently, an unnecessary use of space.

Ideally, the demand (a variable of independent flow) produces work in process ending up with minimum inventory of finished product (a variable of dependent

Advances in Computer Science and Engineering.
Research in Computing Science 23, 2006, pp. 17-29

Received 10/07/06
Accepted 12/10/06
Final version 16/10/06

state). However, these dynamic systems are subject to interferences like changes in the mixture or volume of products, delays, and production speed. Consequently, plants must adjust their production due to unexpected behavior in a market with growing uncertainty. Companies must analyze frequently and have to react to competitive pressure. In other words, appropriate analytical tools are required in order to offer effective solutions with quick implementation. If this is a common practice, the company will be ahead of its competitors having more possibilities to satisfy demand.

2 Background

Several authors have carried out studies in relation to the treatment of disturbances in manufacturing. Matson *et al.* [4], [5] define some important concepts in relation to the nature of the disturbances, analyze how to evaluate their impact on production goals, and determine a procedure for identifying the most important disturbances, although they recognizes that such a procedure does not provide enough information for a detailed study of the response mechanisms. Other investigations have been limited to the process of decision-making, like Meskill *et al.* [6], who explore the way the person who makes the decisions recognizes and identify the interferences. This can be useful for purposes of selection and training.

On the other hand, Kritchanchai and MacCarthy [3] outline the necessity of more investigation, specifically in regard to the process of satisfying production orders. Said authors clarify the definitions and descriptions of the related components through the analysis of the corresponding state of the art, and the study of several industrial sectors, in order to find similarities and differences.

The work that is presented in this article was inspired by the study carried out by Hui Chi and Ng San [2] who outline a mathematical model used to evaluate the impact of variations in the time standards on the balance of an assembly line.

3 The Problem under Study

In order to adequately identify the problem presented in this paper, it is necessary to differentiate between the conventional flow shop problem and an assembly line. A flow shop is characterized by a flow that is almost continuous using multiple serial machines. The flow is unidirectional, and all jobs follow the same route designed by the machine. Although this description of a flow shop is similar to the operation of an assembly line, there are various differences.

First: a flow shop is equipped to handle a variety of jobs, whereas an assembly line handles a standard product. Second: the work in a flow shop does not have to be processed on all of the machines. Some jobs omit operations according to technological requirements. However, in an assembly line, all work must follow all stations without skipping any. Third: in a flow shop, every machine is independent from the others and can carry itself as such. In an assembly line,

every workstation depends on the preceding one. Finally, in a flow shop every job has its own process time on each machine, yet on an assembly line, all the operations in the production of a product have the same standard time at each station.

It is also important to mention that there are a great number of articles that study the balance of a flow shop, such as the single assembly line balancing problem (SALBP). A study that analyzes a variant of SALBP is given by Nicosia *et al.* [14] who formulate a solution based on a dynamic programming algorithm related to the problem of operation assignment in an ordered sequence of different workstations, observing the restrictions of precedence and cycle time, and minimizing the cost. In this scenario the unidirectional flow is not contemplated, since the work does not have to follow the same route of the machines. Another approach is described by Ponnabalam *et al.* [15] who analyze the problem of a mixed model assembly line (MMAL) in which different workstations, standard times and methods, different equipment, and components and raw material are all considered. Ponnabalam separates the problem in two aspects: (a) the assignment of jobs to workstations and (b) the sequence of models in a line. Said author only addresses the second case.

A scenario that is closer to the problem we are discussing here is shown by Dimitradis [16] who proposes a different design in the assembly line. Dimiatradis proposes that instead of workstations with a single operator, more operators without surpassing the "maximum concentration of workers per product" should be considered established according to the structure and size of the product so that the operators are not in each other's way. The use of the right number of tools reduces the wait times along with an adequate distribution of workstations. One omission is that the time for each operator to go from one job to another is not considered.

For the description of a certain type of programming problem, the nomenclature used by Pinedo [17] $(\alpha|\beta|\gamma)$ can be used, in which α represents the area where the machines are consistent with a single entry, β gives details of characteristics and restrictions and it is possible for it to contain either no entry or multiple entries, and finally γ describes the objective to be minimized and usually contains a single entry.

In our study, the problem is in the reassignment of tasks to the workstations and it can be defined as $Fm|block, prec|Ct$ since it is a discrete system of unidirectional in line flow with: m workstations in series with zero capacity buffers, blocks in between stations, and restrictions of precedence. The goal is to minimize the cycle time. Other important considerations are:

- The number of operators in a single workstation is not limited to one.
- The number of operations assigned to a workstation is not limited to one.
- If equipment or tools are not to be relocated in other stations, the corresponding restriction is respected.
- Interruptions are not allowed in operations once initiated.
- The order, in which the jobs are processed, is the same for all.

We have not found an analysis of the defined problem in any literature. Furthermore, Pinedo [17] determines that the problem $Fm||Cj$ is NP-hard for $m \geq 2$ and our problem stated is generalized by incorporating the restrictions of precedence, and therefore also is NP-hard. The solution proposed represents a contribution to this area.

4 Modeling the System

The Critical Path Method (CPM) provides a simple form of capturing the most representative facts of the process when there is a great number of tasks and resources along with the interaction among the activities. Mathur and Solow [7] explain this technique. The necessary minimum information consists of the description of each activity, its standard time, and its predecessors and successors. With this information available, the corresponding network of the production line is designed. In this way, Ramirez-Campos and Salais-Fierro [8] illustrate an example of the use of this technique in the case of a process of on-line flow.

It is also possible to incorporate another class of restrictions referring to position, like the fact that some work elements must be carried out particularly in a certain station or, on the contrary, should not be carried out in one or several specific stations.

Network is a group of precedent restrictions that allows the identification of feasible solutions, i.e., the solutions that satisfy this group of restrictions and the restrictions referring to position.

5 The Simple Genetic Algorithm

The simple genetic algorithm (SGA) belongs to an evolutionary computation paradigm which emulates the processes proposed in the theory of natural selection of Darwin [9]. Genetic algorithms have been used to solve problems of optimization through the generation of solutions called "population by analogy to natural process". The process of evaluation, selection and reproduction is applied to the population of solutions in a cyclical way.

According to Goldberg [10], the simple genetic algorithm uses non-complex mechanisms to carry out selection and reproduction, however, as in any evolutionary algorithm, a key point is the representation of the possible solutions to the problem and the corresponding evaluation.

The first step is to design an initial network that is revised and modified until the representation of the process that should be carried out in the production line is reached. After obtaining the definitive network containing m work elements, the following step is to represent each feasible solution using a chain or chromosome. This chain represents a group of stations and each station represents a gene of the chromosome. Each station contains a certain number of assigned work elements.

To generate a feasible solution (or chromosome), the desired minimum speed is chosen, for example 47 vehicles per hour (47 vph), and the current speed of the

production line that is shown here. This speed implies a maximum cycle (t_{cm}) of 1.277 minutes per vehicle (1.277 mpv), i.e., 60/47. The following algorithm is applied in order to obtain a feasible initial solution:

1. $k = 0$ (station number)
2. $k = k + 1$
3. Generate a random integer r_i so that $1 \leq r_i \leq m$ (i^{th} random integer)
4. If the precedent restrictions and positions are satisfied, we continue on step 5. if not, we continue on step 3.
5. If t_{cm} is not exceeded, the r_i^{th} work element is assigned to station k and we continue on step 6. If not, we continue on step 2.
6. If the all work-elements were already assigned, the algorithm is stopped. If not, we continue on step 3.

After the quantity of desired solutions is generated, an initial population is available with p individuals or p feasible solutions. Since the aptitude value depends on the cycle time of the slowest station, the population is sorted according to the cycle times to identify the best solution reached to the moment. The following step is reproduction. A rate is established, t_R recommended between 0.5 and 0.8 according to Jahangirian and Conroy [11]. If this rate is established at 0.7 it implies that 70 percent of the population will change one aptitude value to another.

Starting from the sorted population, there are two selection methods for carrying out reproduction:

1. A selection of mates according to $(1, 2), (3, 4), (5, 6), \ldots$
2. A random selection of mates

After selecting a couple (s_i, s_j), the next step is to proceed to reproduction by generating a random whole number R_k so that $1 \leq R_k \leq m$. R_k represents the work element and is in the position to carry out the exchange between s_i and s_j, in order to obtain two new solutions. It is important to verify the feasibility of the new solutions, h_1 and h_2. If not feasible, one or both are discarded and reproduction continues with another couple until completing the t_R rate. Once this rate has been reached, insertion by choosing a t_r rate that can be up to a 100 percent according to Badibaru and Cheung [12] is the next step. A t_r value of 0.5 divides the current sorted population in two parts with the worst group being the second part which will be replaced by the first half (the best group) of the new sorted generation.

According to Badibaru and Cheung [12], there are two methods for conforming the new population:

1. The proportion used in the substitution is compounded exclusively for new solutions.
2. The proportion used in the substitution is composed of new solutions, the part of the current population's solutions that did not reproduce and the solutions that result from mutation.

For mutation, a rate (t_m) is chosen with a value around 0.2 according to Jahangirian and Conroy [11]. If the value of t_m is 0.2, it implies that 20 percent of the current population will change. A mutation is obtained by the following algorithm:

1. A random integer is generated so that $0 < r_j < p$ selecting the next solution to be submitted to mutation. A new feasible solution is generated in order to partially replace the r_j solution.
2. A random number r_i is generating so that $0 \leq r_i \leq 1$ (i^{th} random number)
3. The probability of mutation is calculated using:

$$P_m = \frac{1}{2}\left(\frac{1}{p} + \frac{1}{l}\right) \tag{1}$$

   where p = initial population, l = length of chromosome or number of stations in the particular solution
4. If $r_i < P_m$, the mutation is applied, otherwise the mutation does not proceed and the algorithm ends.
5. If the mutacion proceeds, another random integer is generated so that $0 < r_k < l$ for defining the position where the mutation starts. This is from the gene in the r_k position.
6. Finally, the mutation is executed and its feasibility is verified. In case of an unacceptable solution, step 5 is repeated until a feasible solution is achieved.

As a result of each iteration, the process described creates new generations and is stopped when it reaches an established goal or satisfies a detention rule. Three measures can be selected for the aptitude function F:

1. Smaller standard deviation
2. Smaller cycle time
3. Greater average efficiency

The smallest standard deviation is calculated using the cycle times of each station, t_{ci}. For instance, a certain solution with 14 stations has 14 cycle-times to calculate their standard deviation. In this way, for a number (n) of stations, the smallest cycle time is obtained using the group of t_{ci} (or genes) of a given solution. The efficiency average (E_a) is the average of the efficiencies of all stations (E_i):

$$E_a = \frac{\sum_{i=1}^{n} E_i}{n} \tag{2}$$

The efficiency of a station i (E_i) is the percentual relationship of the cycle time of the station (t_{ci}) and the allowed maximal cycle time (t_{cm}) that it is determined by the user:

$$E_i = \left(\frac{t_{ci}}{t_{cm}}\right)100 \tag{3}$$

6 The COMSOAL Heuristic Approach

COMSOAL was originally a solution approach for the assembly-line balancing problem. In fact, according to Johnson and Montgomery [13] the COMSOAL approach finds solutions as follows:

1. Select the tasks without precedences and open the first station ($k = 1$).
2. Assign the tasks of the step 1 to the station k without exceeding the cycle time (t_{mc}).
3. In the event it is necessary to continue with step 2, make $k = k+1$; otherwise, go to step 4.
4. Given the tasks already assigned, identify the possible inmediate tasks (m).
5. Given the tasks already assigned, generate the possible sequences (permutations) of m tasks.
6. Select the task(s) with the smallest number of possible different positions and, if there is tie, randomly choose one of them.
7. Assign task s to station k without exceeding the t_{mc} and go to step 9; otherwise go to step 8.
8. Open a new station ($k = k + 1$).
9. If all the tasks of the network were already assigned, end the procedure; otherwise go to step 4.

7 Experimental results

In Section 7.1 steps are shown for applying the response mechanism to readjustments in the assembly due to changes in the configuration of the product. Results are obtained. In Section 7.2 the comparative results of said algorithms are explained.

7.1 The Response Mechanism

For application the response mechanism described was implemented in a section of the process of an assembly plant. This section is the door line that at the moment consists of 14 work stations and runs to a speed of 47 vehicles per hour (vph). The desired objective was a new configuration of this line to reach a minimum speed of 48 vph. The steps followed are shown.

a. The standard times of each work element were upgraded and the corresponding network was designed, and verified by the analysis of the department of Industrial Engineering so that the structure was attached to the process. Once the network was concluded, it was captured on the interface. Table 5 shows partial information corresponding to this network.
b. Since work elements 1 and 6 must be carried out only in station 1, two additional restrictions were considered.

c. In order to conclude correctly, it was decided to make 5 runs at each one of the speeds: 45, 46, 47, 48 and 49 vph. Tables 1, 2, and 3, where t_{cpu} is cpu seconds, show the corresponding results. We took into account the following parameters:

$$m = 71 \qquad p = 15 \qquad t_R = 0.7 \qquad t_m = 0.2 \qquad t_r = 0.5$$

The fitness function (F) is the smallest standard deviation (minutes). Also, the detention rule is activated when 15 consecutive intents are accumulated and in each case the difference between the cycle times of one solution and another is smaller than 0.0001.

Table 1. Results for 45 and 46 vph

vph	F	n	t_{cpu}	E_a	vph	F	n	t_{cpu}	E_a
45.079	0.089	13	112.344	92.898	46.620	0.087	14	114.734	88.181
45.249	0.074	13	89.734	92.898	46.404	0.089	14	103.094	88.181
45.079	0.071	13	128.563	92.898	46.440	0.092	14	128.641	88.181
45.249	0.109	13	93.969	92.898	46.048	0.062	13	90.203	94.064
45.249	0.076	13	107.688	92.898	46.765	0.075	14	133.563	88.181

d. When analyzing the results of the runs, we can observe that the answer mechanism is consistent, i.e., for a lower speed, a lower number of stations (n) and vice versa. Also, the same efficiency average was always obtained

Table 2. Results for 47 and 48 vph

vph	F	n	t_{cpu}	E_a	vph	F	n	t_{cpu}	E_a
48.154	0.068	14	163.313	90.094	48.154	0.070	14	112.063	92.011
47.170	0.082	14	133.219	90.094	48.077	0.085	14	103.625	92.011
47.356	0.076	14	109.281	90.094	48.232	0.096	14	116.188	92.011
47.133	0.081	14	125.438	90.094	48.820	0.080	14	107.609	92.011
48.309	0.088	14	119.781	90.094	48.154	0.071	14	103.672	92.011

(E_a) for the speeds of 47 and 48 vph.
For 46 vph, four runs give 14 stations and one gives 13 stations, but the efficiency average is low in the first case. Considering that the network is a constant, we can see that, seemingly, for this particular network the speed of 49 vph offers the highest efficiency average, with n = 14. In other words, with a higher or lower speed than 49 vph, E_a diminishes or increases or another number of stations is required. In the case of 49 vph, if the first

Table 3. Results for 49 vph

vph	F	n	t_{cpu}	E_a
49.342	0.068	14	163.313	93.928
49.140	0.118	15	125.266	87.666
49.669	0.088	15	108.047	87.666
50.000	0.082	15	128.031	87.666
49.020	0.109	15	115.234	87.666

run had not been better than the other runs, the low efficiency of the other four runs would indicate the possibility of a better solution. It is advisable to make other runs by increasing the number of intents in the stop rule, and the number of iterations, or by making other types of changes.
To illustrate how to make a more thorough search, table 4 shows the results of 5 runs, applying the following changes:

$$p = 60 \qquad tries = 100$$

The results of this search revealed that four of them reached the same efficiency average of 93.928 percent with only 14 stations, showing that these are the best solutions. The cost of this is the time invested, which went from 1.5 to 3 minutes in the first part of the search, changing to an approximate range of between 4 and 28 minutes in an intensive search (table 4).

Table 4. Results for a thorough search

vph	F	n	t_{cpu}	E_a
49.221	0.064	14	250.328	93.928
49.261	0.075	14	255.438	93.928
49.342	0.078	14	243.641	93.928
50.761	0.073	15	1682.328	87.666
49.180	0.066	14	115.234	93.928

e. In conclusion, the best solution for a desirable minimum speed of 48 vph is the run with the smallest value of F shown in line 1 of table 4.
Corresponding configuration is partially shown in table 5, because there are 71 work elements and 14 stations and, therefore, the description of the work element is omitted. In this table, column 1 is the number of work elements, column 2 is the station number i, columns 3 and 4 the nodes (a, b) of work element k, column 5 the element time, column 6 the cycle time of the station i, and column 7 the efficiency of station i. Columns 8 to 14 have the same meaning.

Table 5. Final configuration for 48 vph

		nodes							nodes				
k	i	a	b	t_k	t_{ci}	E_i	k	i	a	b	t_k	t_{ci}	E_i
1	1	1	5	0.158			15	2	1	500	0.254		
2	1	5	15	0.305			16	2	1	185	0.266		
3	1	5	25	0.119			17	2	1	195	0.164		
4	1	5	20	0.115			18	2	1	200	0.152		
5	1	20	25	0			19	2	185	500	0		
6	1	15	30	0.260			20	2	195	500	0		
7	1	25	30	0			21	2	200	500	0	1.214	99.14
8	1	1	180	0.240			.	.	.	.	.	.	.
9	1	180	500	0	1.197	97.75	.	.	.	.	.	.	.
10	2	5	10	0.034			.	.	.	.	.	.	.
11	2	10	30	0			68	13	148	160	0.758		
12	2	5	38	0.063			69	13	160	165	0.229	0.987	80.60
13	2	25	35	0.281			70	14	155	500	0.502		
14	2	35	38	0			71	14	165	500	0.634	1.136	92.77

Table 6. The best results

	Number of stations	Cycle time (minutes)
SGA	14	1.222
COMSOAL	15	1.178

Table 7. Average performance data shown in Figure 1

Iteration	1	2	3	4	5	6	7	8
SGA	18.015	17.887	17.825	18.015	17.985	17.658	17.786	17.477
COMSOAL	19.436	19.350	19.342	19.301	19.223	19.464	19.389	19.375
Iteration	9	10	11	12	13	14	15	16
SGA	17.820	17.761	17.748	17.741	17.706	17.765	17.902	17.770
COMSOAL	19.375	19.194	19.315	19.241	19.398	19.352	19.241	19.191
Iteration	17	18	19	20	21	22	23	24
SGA	17.815	17.759	17.932	17.932	17.773	17.985	17.713	17.825
COMSOAL	19.443	19.445	19.362	19.422	19.289	19.410	19.345	19.304
Iteration	25							
SGA	17.797							
COMSOAL	19.259							

7.2 Performance

The comparison between the SGA and the COMSOAL was based on specifc runs for each technique. In the case of SGA, 100 runs with 25 iterations each were done. In the COMSOAL case, 2500 runs were completed. This comparison process hads two main objectives. The first one was to find the smallest number of work stations. The second one was to get the smallest cycle time not exceeding 1.2245 minutes. This time corresponds to a speed of 49 vph. For the SGA case, 100 values of each respective iteration were averaged and in the COMSOAL case, sets of 25 serial values were taken and averaged.

As a result, this analysis found that the SGA obtained solutions with a smaller number of stations than the COMSOAL approach. COMSOAL achieved a feasible solution of 15 work stations with a cycle time of 1.178 minutes. On the other hand, SGA was able to find a better solution with 14 work stations that perform in a cycle time of 1.222 minutes (see Tables 6 and 7 and Figure 1).

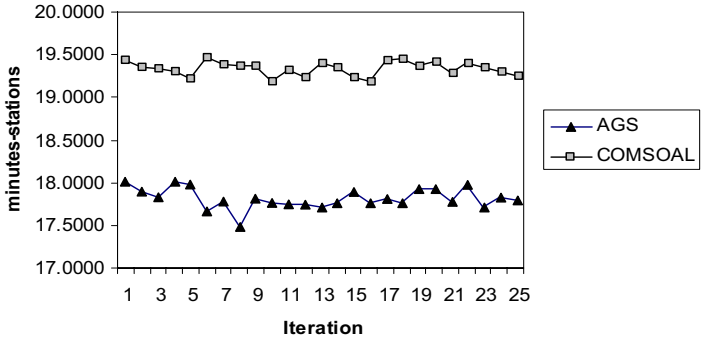

Fig. 1. Comparative performance

8 Conclusions

The door line at the moment is running at 47 vph with 14 work stations, however, a solution has been found with the same 14 work stations reaching a speed of 49 vph. This is a very high increase, considering that in the automotive industry the cost of relocating materials, tools and equipments, is very low in comparison to the benefit of a substantial increase in speed. This mechanism can be applied to any section of the automotive line for similar analysis, subject to an attractive cost-benefit relationship. It is therefore necessary to stress the importance of the estimate of the costs of adjusting the production line according to configuration as a result of applying the response mechanism. These costs are derived from locating work elements in different stations or incorporating new work elements into one or more work stations. It is therefore necessary to relocate tools, diverse devices, material containers and equipment.

If the cost of adjustment is very near the benefit, then a careful additional analysis must be made of the convenience of carrying out the involved changes.

In this light, the mechanism provides valuable information that allows us to avoid making inconvenient decisions about adjustments in a production line. On the other hand, this mechanism can be applied when the production line has not been put into operation yet. In this case, the benefit is greater than in a line in operation, since relocation costs do not exist.

More exhaustive comparison is needed to determine the efficiency of this genetic algorithm, so we needed further heuristics from literature to apply to the stated scenario. Although it is important to note that there is a large amount of literature with heuristics in relation to certain types of flow shop problems, they apply to a multitude of different scenarios, so it is not easy to find a heuristic that adapts to a specific scenario such as the one stated in this paper. However, the genetic algorithm can be improved by taking into account other focuses such as: a multi-objective function of fitness, a small initial population that can grow from one generation to the next, the generation of only feasible solutions, and a more efficient representation of a chromosome to reduce epistasis in the code.

References

1. Carnahan, Brian J., Norman Bryan A., and Redfern Mark S.: Incorporating physical demand criteria into assembly line balancing. IIE Transactions, **33** (2001) 875–887
2. Hui Chi L. P. and Ng Sau F. F.: A study of the effect of time variations for assembly line balancing in the clothing industry. International Journal of Clothing Science and Technology, **11** (1999) 181–188
3. Kritchanchai Duangpun and MacCarthy, B. L.: Responsiveness of the order fulfilment process. International Journal of Operations & Production Management, **19** (1999) 812–833
4. Matson, Jeremy B. and McFarlane, Duncan C.: Tools for Assesing the responsiveness of existing production operations. In Proceedings of IEE Workhshop, Responsiveness in Manufacturing, London, February (1998)
5. Matson, Jeremy B. and McFarlane, Duncan C.: Assesing the responsiveness of existing production operations. International Journal of Operations & Production Management, **19** (1999) 776–784
6. Meskill Margaret, Mouly Souchitra and Dakin, Stephen: Managerial disturbance handling: a case study approach. Managerial Psychology, **14** (1999) 443–454
7. Mathur, Kamlesh and Solow, Daniels: Investigacion de Operaciones. Prentice-Hall Hispanoamericana, S. A. first edition, (1996) 492–541
8. Ramirez-Campos, Sergio and Salais-Fierro, Tomas.: Flexibility and Time Optimization in an Automotive Assembly Line: A Neural Network Approach. In proceedings of Business & Industry Symposium: Advanced Simulation Technology Conference, Arlington, VA, USA, April, (2004)
9. Darwin, Charles: El origen de las especies UNAM, (1997)
10. Goldberg, David E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Pub Co. (1989)
11. Jahangirian, M. and Conroy, G. V.: Intelligent dynamic scheduling system. Integrated Manufacturing Systems, **11** (2000) 247–257

12. Badibaru, A. B. and Cheung, J. Y.: Fuzzy Engineering Expert Systems with Neural Network Applications. John Wiley & Sons, Inc. (2002)
13. Johnson, Lynwood A. and Montgomery Douglas C.: Operations Research in Production Planning, Scheduling, and Inventory Control. John Wiley & Sons, Inc, first edition, (1974) 373–374
14. Nicosia, G. and Pacciarelli, D. and Pacifici, A.: Optimally balancing assembly lines with different workstations. Discrete Applied Mathematics, **118** (2002) 434–456
15. Ponnabalam, S. G. and Aravindan, P. and Rao, M. Subba: Genetic algorithms for sequencing problems in mixed model assembly lines. Discrete Applied Mathematics, **45** (2003) 669–690
16. Dimitriadis, S. G.: Assembly line balancing and group working: A heuristic procedure for workers'groups operating on the same product and workstation. Computers & Operations Research, **33** (2005) 2757–2774
17. Pinedo, M.: Scheduling, Theory, Algorithms and Systems. Prentice Hall, second edition, (2002)