LINE BALANCING WITH GENETIC ALGORITHMS

Thiago Pereira Berto
Department of Automation and Systems
Universidade Federal de Santa Catarina (UFSC)
Trindade, Florianópolis, SC, Brazil
CEP 88040-900
thiber@das.ufsc.br

João Carlos Espíndola Ferreira
Department of Mechanical Engineering
Universidade Federal de Santa Catarina (UFSC)
Trindade, Florianópolis, SC, Brazil
CEP 88040-900
jcf@grucon.ufsc.br

Abstract. *This work analyzes a version of the line balancing problem of the type GALB (General Assembly Line Balancing), which uses genetic algorithms for its solution and it compares its performance with other heuristics. Hypotheses of the SALB (Simple Assembly Line Balancing) problems were relaxed. In the present problem, there is a reduced number of pieces of equipment allocated between the workstations and each task can be executed only in a specific type of equipment, so that the stations are not capable of executing any task. The workstations don't possess necessarily the same production capacity, so that the processing time of a task can depend on where it is allocated. The applied criterion as objective function is the same of the SALB-II, that is, the maximization of the production rate for a given number of stations.*

Keywords: *Line Balancing, Genetic Algorithms, Heuristic Methods, Optimization, Artificial Intelligence*

1. Introduction

The line balancing problem was defined for the first time by Helgeson (1954) and modeled mathematically by Salveson (1955). According to Baybars (1986), the problem can be of two types, which stand out for the objective function. In the SALB-I (Simple Assembly Line Balancing I) the objective is to minimize the number of workstations for a certain cycle time or a certain production rate. In the SALB-II (Simple Assembly Line Balancing II) the objective is, for a given number of workstations, to maximize the production rate or to minimize the cycle time. The SALB problems still assume other 9 hypotheses:

H1: all of the parameters are deterministic and known;
H2: a task cannot be divided between two or more stations;
H3: the allocation of tasks to the stations should respect the precedence relationships;
H4: all of the tasks should be allocated;
H5: any station has technological capacity to process any task;
H6: the processing time of any task does not depend on the station that will execute it and of the sequence in that will be processed in the station;
H7: any task can be processed in any station;
H8: the line is considered serial; and
H9: the line will only assemble a single model of a single product.

There are still the class GALB (General Assembly Line Balancing) problems. In these, at least a hypothesis of SALB is relaxed. The version being studied is based on the production line of a company in Brazil and it relaxes hypotheses 5, 6 and 7. Each task can only be executed in a specific type of equipment and there is a reduced number of equipment distributed among the stations, so that the workstations are not necessarily capable of executing any task. The processing time of each task depends on the workstation where it is allocated; therefore different stations can have different indexes productivity.

A wide range of products is manufactured in the studied line. At each cycle, however, only the same product model is produced in each station. Thus, the balancing is accomplished separately for each model and the hypothesis 9 doesn't need to be relaxed. However, the variety of the products, together with the reduced readiness of the equipments, has important implications. The products of the company have short life cycle and every 4 months a new collection of products is launched. A pre-balancing can then be carried out for each model, distributing later the equipment among the stations. From those results, the most appropriate distribution it is inferred. Finally, the balancing is carried out again for each model.

It was adopted as objective function the maximization of the production rate for a given fixed number of stations. The problem can be described in mathematical programming as follows:

Maximize u

$$\text{Subject to} \quad \sum_{i=1}^{n} a(i,j) = 1 \qquad j = 1...m \tag{1}$$

$$u \le \frac{e(i)}{t} \sum_{j=1}^{m} a(i,j)p(j) \qquad i = 1...n \tag{2}$$

$$\sum_{i=1}^{n} ia(i,j) \le \sum_{i=1}^{n} ia(i,k) \qquad j,k = 1...m \text{ and } b(j,k) = 1 \tag{3}$$

$$a(i,j) = 0 \qquad i = 1...n, j = 1...m, k = 1...l, c(i,k) = 0 \text{ and } d(k,j) = 1 \tag{4}$$

$$\mathbf{A} \in \{0,1\}^{n \times m}$$

$$u \in \Re$$

where: l is the number of different pieces of equipment in the production line;
m is the number of tasks in which the product manufacture is subdivided;
n is the number of workstations in the production line;
u is such that the floor of u is the number of units produced in a cycle in the production line;
t is the cycle time of the production line;
p is such that $p(i)$ is the processing time of the i-th task;
e is such that $e(i)$ is the efficiency of the i-th station;
A is such that $a(i,j)$=1 if the j-th task is allocated to the i-th station and $a(i,j)$=0 otherwise;
B is such that $b(i,j)$=1 if the i-th task is immediately precedent to the j-th task and $b(i,j)$=0 otherwise;
C is such that $c(i,j)$=1 if the j-th type of equipment is available at the i-th station and $c(i,j)$=0 otherwise; and
D is such that $d(i,j)$=1 if the i-th type of equipment is utilized for processing the j-th task and $d(i,j)$=0 otherwise.

Equation (1) guarantees that all tasks will be allocated in one and only one station. Equation (2) restricts the production of the line to the amount of units produced by the bottleneck. Equation (3) guarantees that the precedence constraints will be satisfied. Equation (4) prevents the tasks from being allocated to stations technologically unable to process them.

2. Algorithms

It is known that the line balancing problem, even in the SALB versions, is a np-hard problem (Bock and Rosenberg, 1997). The search for the optimal solution of instances of realistic size presents an unacceptable computational cost. A solution is to use heuristic methods, which do not guarantee optimality, but provide good results with execution time asymptotically polynomial. Empiric studies (Bautista, Suárez and Mateo, 1999, Ponnambalam, Aravindan and Naidu, 1996) have shown that genetic algorithms overcome the performance of heuristics developed specifically for solving the line balancing problem. For that reason, we decided to implement such method. Also, four classic heuristics were implemented, in order to verify whether they would present a reasonable improvement in relation to the non-algorithmic balancing accomplished currently at the company considered. Those four heuristics are Largest Candidate, Kilbridge-Wester, Ranked Positional Weight and Reversed Ranked Positional Weight.

2.1. Genetic Algorithms

Genetic algorithms constitute a branch of Artificial Intelligence based on evolutionary mechanisms found in nature. These mechanisms were discovered and formalized in 1859 by Charles Darwin in his book "The Origin of Species". The use of those evolutionary mechanisms to solve computational optimization problems began in the 1960's with John Holland's works. Differently of the traditional methods, with the genetic algorithms there is no requirement of previous knowledge of a means of finding the solution for a certain problem.

The genetic algorithms have a lot of variations but all share the following characteristics (Bittencourt, 2001):

- existence of a population of candidate solutions;
- existence of an evaluation function, responsible for attributing a grade to each individual of the population in agreement with his/her aptitude; and
- existence of operators based on the phenomena that govern the evolution: recombination, mutation and selection.

The implemented algorithm has the following structure:

Initialization of the population
For *g* generations do
    Reproduction of all individuals
    Mutation of the genome of a random number of individuals
    Evaluation of all individuals
    Random selection of the most capable individuals and death of the remaining
Assignment of the most capable individual to the solution of the problem

Each one of those stages will be detailed in the following sections.

2.1.1. Initialization

In this first stage, a population of candidate solutions is generated with μ individuals. These individuals can be randomly generated or obtained from conventional techniques. Initially we decided to take advantage of the solutions obtained with the classic algorithms to generate the initial population. Then, to avoid the occurrence of superindividuals, we started to use the random establishment of the population. Each individual is composed by a vector with length j equal to the number of necessary tasks for producing the product. Each element of the vector contains a number, which corresponds to the workstation in which the respective work element is allocated.

The genesis of solutions occurs a single time, while the following stages of the algorithm are repeated for *g* generations.

2.1.2. Reproduction

In reproduction, the individuals generate copies of themselves or form pairs that exchange portions of their solutions. It is common that reproduction is made after the grade assignment to the individuals, representing in this case the possibility of their reproduction. At this point, an algorithm was adopted similar to the evolutionary strategy ($\mu+\lambda$), because a population of descendants is generated with λ individuals, which later will compete for survival with the μ individuals of the ancestral population. The competition between ancestral and descendants is a form of elitism, because, when preserving the ancestors, it is avoided that the best solutions are eliminated from the population (Bittencourt, 2001). Here the reproduction occurs before the grade assignment and it was decided that all the individuals can reproduce once, so that $\mu = \lambda$.

The pairing of individuals is accomplished randomly as well as the rupture point for β recombination. For each pair two descendants are generated. Calling α_1 and α_2 the two ancestors, one of the descendants is composed by the tasks allocated according to α_1 until the β-th element and according to α_2 starting from the next element of the vector that constitutes the individual's genome. The complementary segments of α_1 and α_2 compose the other descendant.

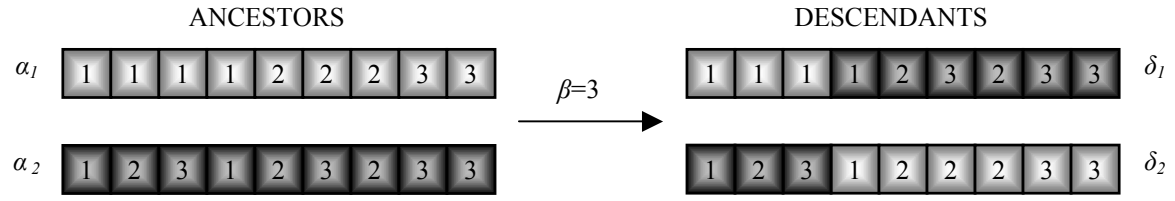

Figure 1. Illustration of the process of recombination of two genomes

2.1.3. Mutation

Mutation consists of the alteration of the value of one of the alleles, which corresponds to one of the tasks in this case. Usually the alleles assume binary values, so that the mutation simply consists of the inversion of the 0 bit to 1 or of 1 to 0. In this case, the tasks can be allocated to any one of the workstations. Therefore, mutation consists of the alteration of the station to other station *i* randomly chosen. Task *j* whose station will be altered is also determined randomly. Each individual in the population of ancestors and descendants have a probability ρ suffering a mutation. That probability is called mutation rate. Although reduced ρ values facilitate the convergence of the population, we applied greater values than normal, as suggested by Anderson and Ferris (1993).

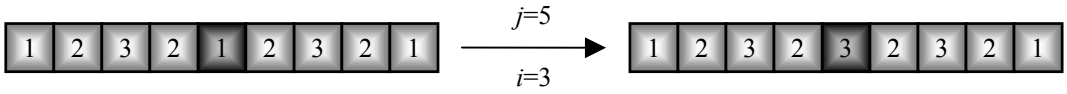

Figure 2. Illustration of the process of mutation of a gene

2.1.4. Evaluation

In the evaluation, each individual receives a grade in agreement with the quality of its solution. In the present case, the $\mu+\lambda$ individuals are evaluated based on the value of u and in the number of times that Eq. (3) and Eq. (4) are violated. Since mutation and recombination are random processes, the execution of prerequisites is completely unknown in a new individual's creation, whose genome can contain an infeasible candidate solution. Equations (1) and (2) are satisfied automatically, due to the structure of the algorithm.

There is no standard formula for evaluation. It should be created based on the variables that one wants to measure and adjusted empirically. The following evaluation function is used:

$$\varphi = \frac{u}{1 + k_P v_P + k_R v_R} \tag{5}$$

where: φ is the quality of the candidate solution;
v_P is the number of times that Eq. (3) is violated by the candidate solution;
v_R is the number of times that Eq. (4) is violated by the candidate solution;
k_P is an consideration empirically adjusted to measure the impact that the violations of Eq. (3) will have about the quality of the candidate solution; and
k_R is an consideration empirically adjusted to measure the impact that the violations of Eq. (4) will have about the quality of the candidate solution.

Thus, if an individual's solution doesn't contain any violation, the evaluation function will be equivalent to the objective function of the problem. Solutions that contain some type of violation are useless as final solution to the problem, but they should not be penalized excessively because they can generate good solutions. It is common that problems treated with genetic algorithms have multimodal behavior, with several local maxima. In these cases bad solutions can sometimes have a genome similar to the global maximum. Its precocious elimination would lead the algorithm to converge to a local maximum (Bittencourt, 2001).

2.1.5. Selection

In selection, individuals are determined that will compose the population of ancestors of the next generation. In other versions of genetic algorithms the selection determines which individuals of the ancestral population will recombine to originate the individuals of the next generation (Bricker, 1997). It is not the case here. As it was already said, the population of ancestors competes with the one of descendants for survival. The j-th individual's probability of passing to the next generation in the k-th selection is given by:

$$\pi(j,k) = \frac{\varphi(j)}{\sum_{i=1}^{\mu+\lambda-k+1} \varphi(i)} \tag{6}$$

where: $\varphi(i)$ is the result of the evaluation function for the i-th individual;
$\varphi(j)$ is the result of the evaluation function for the j-th individual; and
$\mu+\lambda-k+1$ is the number of eligible individuals for the k-th selection.

At each selection an individual is drawn, with proportional probability to its evaluation function, and if it becomes ineligible for the following selections, so that it can only have a copy of itself in the next generation. Is it important to point out that it doesn't prevent a solution from being copied at once to the next generation, since each candidate solution can be present in even $\mu+\lambda$ individuals.

The fundamental theorem of genetic algorithms affirms that the population of solutions that have evaluation above the average tends to grow exponentially at each generation, while the population of those that have evaluation below the average tends to decrease exponentially (Bittencourt, 2001). Thus, after repeating the procedures of reproduction, mutation, evaluation and selection for g generations, a quite homogeneous population should be obtained, with the prevalence, or else hegemony, of the best solution.

2.2. Heuristic Methods

Several specific heuristics were developed for the problem of line balancing. We implemented four of them: Largest Candidate, Kilbridge-Wester, Ranked Positional Weight and Reversed Ranked Positional Weight. All possess the same basic algorithmic structure, adapted here to the case in study:

Calculation of the priority of each task
Sequencing of tasks according to the priority of each of them
Calculation of an upper limit for the number of manufactured units by production cycle
While there is some no allocated task, for each station do
For each station, in ascending order, do
While there is some task not tested, for each task, in ascending order, do
Verification of the previous allocation of the tasks immediately precedents to the *j*-th task
Verification of the readiness of time in the *i*-th station for allocation of the *j*-th task
Verification of the technological adaptation of the *i*-th station for allocation of the *j*-th task
If all of the conditions are satisfied
Allocation of the *j*-th task in the *i*-th station
Return to the beginning of the orderly list of tasks
Return to the first station
Decrease the upper limit for the number of manufactured units allele production cycle

The calculation of an upper limit for *u* is given by:

$$\overline{u} = \left\lfloor c \frac{\sum_{i=1}^{n} e(i)}{\sum_{j=1}^{m} p(j)} \right\rfloor \tag{7}$$

The calculation of the priority of each task is that differentiates a heuristics from another. In the Largest Candidate method, the priority of each task j is proportional to its processing time *p(j)*. The idea behind this algorithm is that, the more slowly it is the work element, less common it will be the occurrence of situations in which there will be readiness of time in the station for its allocation. Thus, long work elements should be allocated as soon is possible.

In the Kilbridge-Wester method, it is assigned column I to the tasks that don't possess any precedence. It is assigned to those follow them column II, and so on, so that the tasks are always in the subsequent column to the one of its predecessor of the greater column. The priority of the task is then inversely proportional to the column where it is located. Kilbridge and Wester developed that algorithm to supply the deficiencies of the largest candidate method. Slow work elements or that are followed by many other elements sometimes took long to be allocated because they are preceded by a very fast element, that it was always ignored for another in the allocation. With the Kilbridge-Wester method that no longer occurs.

In the Ranked Positional Weight method, the priority of each task is proportional to the sum of the processing times of all of its subsequent operations. Differently to the largest candidate method, the time considered here is not just it of the processing of the work element itself, but also of all of the elements that have it as predecessor, immediate or not. Thus, fast methods of which many elements or very slow elements depend have high priority. The logic behind this algorithm is the same of the largest candidate method, but with a global vision of the product instead of the local vision of the task.

The Reversed Ranked Positional Weight method is similar to the previous method. Here the priority of each task is proportional to the sum of the processing times of all its precedent operations. What differentiates it from the previous heuristics is that the allocation of the work elements is accomplished starting from the last workstation. In this way, for an element to be allocated it is necessary that all the elements that follow it have already been allocated. In order to do that, the arrows in the precedence graph of the product to be manufactured are inverted.

3. Results

To evaluate the performance of the algorithms it was simulated through software the line balancing of a sample of products in the company being studied. The results generated by the program were compared with themselves and with the manual balancing that is accomplished in the shop floor. This last comparison, however, is an illustration, because there is a fundamental difference between the implemented model and the company production lines. In these, there is more than a worker for workstation. In the first 5 simulated cases, each production line had 23 workers distributed along 8 workstations and the cycle time was 30 minutes. In these, the production line had 24 workers allocated to 8 stations and the cycle time was 33 minutes. To minimize that problem, the 8 virtual workers' in the simulated production line

was altered so that the working capacity of the 8 virtual workstations was equal to the one with 8 real stations. The two situations would be equivalent if there were not technological constraints in the stations. As there is a limited number of pieces of equipment, Eq. (4) of our model would have to be modified to consider the possibilities in that the pieces of equipment had to be used simultaneously by more than one worker. After all, when linking a task to a station, we are using, implicitly, the worker and equipment resources.

There is other conceptual difference between the model and the real system. In the factories, the tasks were distributed first among the 8 workstations. Soon afterwards, the number of workers by station was designated based on the workload assigned to each station. To lessen the unbalance among the stations, resulting from ineffective balancing, some workers were designated to accomplish tasks in different stations. In this way, the effective workforce of the *i*-th station can be calculated by:

$$w(i) = \sum_{j=1}^{\omega} e(j)f(i,j) \tag{8}$$

where: ω is the number or workers in the production line; and
$f(i,j)$ is the portion of time in which the *j*-th worker remains at the *i*-th station.

In Tab. 1 it can be seen the size of the simulated instances. Tables 2 and 3 provide the main data obtained with the simulation. The first table contains the production per cycle obtained by the line balancing of the 5 implemented algorithms. The second table contains the percentage in relation to the balancing accomplished in the shop floor. They also show the values that would be obtained in the case of an ideal balancing. This ideal balancing occurs when $u=\bar{u}$. But there is not always a feasible solution that satisfies that condition.

Table 1. Number of tasks by product

Instance	1	2	3	4	5	6	Average
Size	*38*	*39*	*54*	*59*	*52*	*44*	*47.7*

Table 2. Quantity of units manufactured by production cycle

Method\Instance	1	2	3	4	5	6	Average
Manual	*34*	*42*	*22*	*26*	*26*	*33*	*30.5*
Largest Candidate	*38*	*43*	*25*	*32*	*32*	*42*	*35.3*
Kilbridge-Wester	*38*	*43*	*24*	*31*	*31*	*42*	*34.8*
Ranked Positional Weight	*38*	*43*	*24*	*32*	*31*	*42*	*35.0*
Reversed Ranked Positional Weight	*38*	*43*	*25*	*32*	*32*	*41*	*35.2*
Genetic Algorithm	*38*	*43*	*25*	*32*	*32*	*42*	*35.3*
Ideal	*40*	*46*	*26*	*33*	*32*	*43*	*36.7*

Table 3. Potential gain compared with the current production

Methdod\Instance	1	2	3	4	5	6	Average
Largest Candidate	*11%*	*2%*	*14%*	*23%*	*23%*	*27%*	*15.7%*
Kilbridge-Wester	*11%*	*2%*	*9%*	*19%*	*19%*	*27%*	*14.1%*
Ranked Positional Weight	*11%*	*2%*	*9%*	*23%*	*19%*	*27%*	*14.8%*
Reversed Ranked Positional Weight	*11%*	*2%*	*14%*	*23%*	*23%*	*24%*	*15.4%*
Genetic Algorithm	*11%*	*2%*	*14%*	*23%*	*23%*	*27%*	*15.7%*
Ideal	*18%*	*10%*	*18%*	*27%*	*23%*	*30%*	*20.3%*

The Largest Candidate method, the one of simpler implementation, obtained, surprisingly, the same performance of the Genetic Algorithm. The sample is too small to have statistical value and it is probable that with a larger sample the result could be different.

The Genetic Algorithm generated a performance significantly worse than the one corresponding to the ideal balancing. This is another indication that some parameters of the Genetic Algorithm can be better fitted and that the number of generations and the size of the population should be increased.

The performance of the 5 algorithms was quite similar. All of them showed significant gain in relation to the manual balancing currently done. However, it is good to recall that those improvements are a preliminary result, which serves as an upper limit, since the applied model contains a relaxation of one of the constraints of the real system.

4. References

Anderson, E. J. and Ferris, M. C., 1993, "Genetic Algorithms for Combinatorial Optimization: The Assembly Line Balancing Problem", University of Cambridge, Cambridge, England.
Bautista, J., Suárez, R. and Mateo, M., 1999, "Local Search Heuristics for the Assembly Line Balancing Problem with Incompatibilities Between Tasks", Institut d'Organització i Control de Sistemes Industrials, Barcelona, Spain.
Baybars, I., 1986, "A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem", Management Science, Vol. 32, No. 8, pp. 909-932.
Bittencourt, G., 2001, "Inteligência Artificial – Ferramentas e Teorias", Editora da UFSC, Florianópolis, Brazil, 362 p.
Bock, S. and Rosenberg, O., 1997, "A new distributed fault-tolerant algorithm for the Simple Assembly Line Balancing Problem 1, University of Paderborn, Paderborn, Germany.
Bricker, D., 1997, "Genetic Algorithm for Line Balancing", University of Iowa, Iowa City, USA.
Helgeson, W. B., 1954, "How to Balance an Assembly Line", Management Report, No. 7, Carr Press.
Ponnambalam, S. G., Aravindan, P. and Naidu1, G. M., 2000, "A Multi-Objective Genetic Algorithm for Solving Assembly Line Balancing Problem", The International Journal of Advanced Manufacturing Technology, Vol. 16, pp. 341-352.
Salveson, M. E., 1955, "The Assembly Line Balancing Problem", Journal of Industrial Engineering, Vol. 6, No. 3, pp. 18-25.

5. Responsibility notice

The authors are the only responsible for the printed material included in this paper.