Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

https://doi.org/10.23913/ride.v14i27.1577

Artículos científicos

Aplicación de algoritmos genéticos con reglas de decisión en el balanceo de líneas en forma de U estocástico

Application of genetic algorithms with decision rules in stochastic u-shaped line balancing

Aplicação de algoritmos genéticos com regras de decisão no balanceamento estocástico de linhas em U

Demetrio Fermán Alvarez
Tecnológico Nacional de México, México
m20112714@cdjuarez.tecnm.mx
https://orcid.org/0000-0002-0655-7761

Ulises Martínez Contreras
Tecnológico Nacional de México, México
ulises.mc@cdjuarez.tecnm.mx
https://orcid.org/0000-0002-1631-4448

Mirella Parada González
Tecnológico Nacional de México, México
mirella.pg@cdjuarez.tecnm.mx
https://orcid.org/0000-0002-8257-685X

Arturo Woocay Prieto
Tecnológico Nacional de México, México
arturo.wp@cdjuarez.tecnm.mx
https://orcid.org/0000-0001-9235-0494

Adán Valles Chávez
Tecnológico Nacional de México, México
avalles@itcj.edu.mx
https://orcid.org/0000-0002-6559-0123

Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

Resumen

Actualmente, la mayoría de investigaciones acerca del problema de balaceo de líneas de ensamble consideran que los tiempos de las tareas son determinados. Sin embargo, en los procesos de fabricación siempre existe la posibilidad de obtener en los procesos variaciones que impactan en los tiempos de las tareas. Por eso, en el presente trabajo, con base en un enfoque estocástico, se presenta un método que utiliza técnicas metaheurísticas mediante un algoritmo genético, el cual tiene como objetivo brindar una solución al problema de balanceo tipo 1 de líneas en forma de U con tiempos de tarea estocásticos. Para ello, se han tomado como referencia problemas existentes en la literatura para luego ofrecer una comparación entre las soluciones existentes. En el proceso de validación se utilizaron siete categorías de problemas resueltos por otro método. La solución brindada por el algoritmo se sometió a un análisis experimental de los datos para comprobar si era capaz de dar una o más soluciones mejores a las existentes; de ese modo, se buscó balancear la línea con la menor cantidad de recursos humanos posible. Los datos muestran mejores soluciones para los problemas de alta varianza únicamente en el resultado *WS mayor*, donde se observa una diferencia del 4 %; en los demás hallazgos los porcentajes son mejores. Además, se encontraron seis soluciones mejores a las existentes.

Palabras clave: técnicas metaheurísticas, solución al problema de balanceo, líneas en forma de U, estocásticos, validación.

Abstract

Currently, most of the research on the assembly line balancing problem considers that the task times are determined. However, in manufacturing processes there is always the possibility of obtaining variations in the processes, these variations lead to variations in the task times, which leads to address this type of problem from a stochastic approach. This paper presents a method that uses metaheuristic techniques, through a genetic algorithm which aims to solve the problem of balancing type 1 of U-shaped lines with stochastic task times using existing problems in the literature and then make a comparison between the existing solutions.

Seven categories of problems solved by another method were used for the validation process. The solution provided by the algorithm was subjected to an experimental analysis of the data to check if it is capable of providing one or more solutions that are better than the existing ones, seeking to balance the line with the least amount of human resources possible. The results show better solutions for the high variance problems, only for the WS Major result a difference of 4% is observed, but in the remaining results the percentages are better. It can be observed that 6 better solutions were found than the existing ones.

Keywords: metaheuristic techniques, solution to the balancing problem, U-shaped lines, stochastics, validation.

Resumo

Atualmente, a maioria das pesquisas sobre o problema de balanceamento de linha de montagem considera que os tempos das tarefas são determinados. Porém, em processos de fabricação sempre existe a possibilidade de se obter variações nos processos que impactam os tempos das tarefas. Por esse motivo, no presente trabalho, baseado em uma abordagem estocástica, é apresentado um método que utiliza técnicas metaheurísticas por meio de um algoritmo genético, que visa fornecer uma solução para o problema de balanceamento tipo 1 de linhas em forma de U com tempos de tarefas estocásticas . Para isso, foram tomados como referência problemas existentes na literatura para posteriormente oferecer uma comparação entre as soluções existentes. No processo de validação, foram utilizadas sete categorias de problemas resolvidos por outro método. A solução fornecida pelo algoritmo foi submetida a uma análise experimental dos dados para verificar se era capaz de dar uma ou mais soluções melhores que as existentes; Desta forma, buscou-se equilibrar a linha com a menor quantidade de recursos humanos possível. Os dados mostram melhores soluções para problemas de alta variância apenas no maior resultado de WS, onde se observa uma diferença de 4%; nos demais achados as porcentagens são melhores. Além disso, foram encontradas seis soluções melhores que as existentes.

Palavras-chave: técnicas metaheurísticas, solução do problema de balanceamento, linhas em forma de U, estocástica, validação.

Fecha Recepción: Noviembre 2022 **Fecha Aceptación:** Julio 2023

Introduction

In industrial production processes there are countless operations carried out directly by the human being, each of which must be balanced according to the different needs of the production process, hence it is important to have an adequate balancing of lines to meet the estimated demands of the product. In this regard, Orejuela and Flórez (2019) highlight that the first designs of assembly lines were developed to obtain efficiency and eliminate production costs in operations that commonly work against inventories. For this reason, research has been carried out to create optimal methods of assigning tasks in the stations of an assembly line, which are called the assembly line balancing problem (ALBP).

Assembly lines can be linear and U-type; the latter offer improved productivity and quality, which is why they are considered one of the best for implementing just-in-time (JIT) systems. Although there is a growing interest in the literature to arrange straight or linear assembly lines as U-shaped lines to improve performance, literature works are still limited. The U-type Assembly Line Balancing Problem (UALBP) is an extension of the Straight Line Balancing Problem (SALBP), in which tasks can be assigned from both sides of the precedence diagram (Baykasoğlu & Özbakır, 2006).).

Line balancing problems are divided into two types: type 1 and type 2. In the first, the cycle time is already known, so tasks are assigned to work stations to minimize the number of stations. In problem type 2, the aim is to reduce the cycle time when the number of stations is fixed.

Heuristic and metaheuristic techniques have allowed the development of solution methodologies for assembly line balancing problems that cannot be addressed with conventional methods. For example, Gallego et al. (2015) mention that metaheuristic techniques are very useful to solve optimization problems, which cannot be solved by other types of techniques.

Metaheuristics operate by means of algorithms that are not common order, but special because, basically, they are not governed by a predictive, causal, or organized pattern, but random. This algorithm acquires its optimal form through roaming or trials that approximate the solution. "The best known algorithms in metaheuristics are genetic algorithms, tabu search, ant colony algorithm (ACO), simulated annealing, particle swarm optimization (PSO)" (Maldonado, 2016, p. 173).

Genetic algorithms were originally developed by J. Holland. They have the ability to learn, which is the most determining feature in the evolution of any living system or that exhibits life. This search technique uses a population of solutions that are independently manipulated (Maldonado, 2016).

Currently, in most ALBP studies, determined parameters are considered. However, in actual manufacturing processes there is always uncertainty, as there may be variations in manual and machine operating times. Therefore, to minimize the negative effects of all these unexpected problems, stochastic theory has been applied in the SALBP and UALBP (Zhang et al., 2018).

Now, although in recent years different authors have proposed methodologies to solve the ALBP, the present investigation is developed within the stochastic approach in the balancing of U-shaped lines type 1. Genetic algorithms, being more efficient methods, allow us to provide more options for possible solutions to the problem of stochastic balancing of U-shaped lines type 1. In this sense, Martínez (2015) developed and published a new algorithm that uses metaheuristic techniques through genetic algorithms with heuristic rules, which can help to solve ALBP and UALBP, since they provide one or more good solutions, and in some cases optimal, to apply to any process.

To solve the stochastic UALBP type 1, the algorithm is adapted by incorporating equations to calculate the probabilities that the times in the workstations exceed the cycle times. The performance of the algorithm is evaluated and compared with existing solutions in the literature of Adil Baykasoğlu y Lale Özbakır (2006) "Stochastic U-line balancing using genetic algorithms" (p. 139).

Methods and materials

Genetic algorithm

To Cortez (2004) A computational process, also called an algorithmic process or algorithm, is fundamental to computer science, since a computer cannot execute a problem that does not have an algorithmic solution. Evaluating the efficiency of algorithms, therefore, has a lot to do with assessing their complexity. In this sense, the theory of computational complexity is the part of the theory of computation that studies the resources required during computation to solve a problem. The resources commonly studied are time (number of execution steps of an algorithm to solve a problem) and space (amount of memory used to solve a problem). An algorithm that solves a problem, but takes a long time to do so, will hardly be of any use.

Genetic algorithms are part of the so-called evolutionary techniques, originally proposed in the 1950s, which have a common basic structure: they reproduce, carry out random variations, promote competition, and execute the selection of individuals from a given population. Whenever these four processes are present, whether in nature or in a computer simulation, evolution is the byproduct.

In computer simulations —according to Gallego et al. (2015)— genetic algorithms, like other evolutionary techniques, simulate a process of natural selection to obtain the solution of optimization problems. In this case, the problem to be solved plays the role of the environment and each individual in the population is associated with a candidate solution. In this way, an individual will be more adapted to the environment whenever it corresponds to a more effective solution to the problem.

Evolutionary computing has the advantage of being able to solve problems through simple mathematical descriptions. "In this way, evolutionary computation must be understood as a set of generic and adaptable techniques and procedures, to be applied in solving complex problems, for which other known techniques are ineffective or not applicable" (Gallego et al., 2015, p.6).

Evolutionary algorithms are techniques based on a population of individuals, which are in constant communication and sharing information through reproduction and mutation operators. The population is made up of several individuals, which are generally represented by a binary string called a chromosome, where each bit of this string is known as a gene. (Esparza, 2009).

Modified direct coding algorithm to solve the type 1 stochastic UALBP

In direct coding, the genetic algorithm is fed with the specific data of each problem. Balancing problems have a number of tasks, task times, restrictions and precedence of these, as well as a given cycle time. In the direct coding algorithm, each gene represents a task, that is, the number of genes is equivalent to the number of tasks. The previously mentioned data is introduced to the algorithm and it will generate an initial population; then the search for an ideal chromosome (one that generates an optimal number of workstations) begins. If no such chromosome is found, new populations are generated using breeding, crossing, and mutation genetic operations. (Martínez, 2015).

Coding

As Martínez (2015) comments, the first step to build a genetic algorithm is to define a genetic representation called coding. Thus, each task is numbered sequentially in the order in which it will be assigned to the workstations, and each chromosome gene contains the task number it represents (Martínez, 2015).

The chromosome is symbolized by a line graph or isomorphic diagram, so called in graph theory. The isomorphic diagram contains the same precedence configuration as the original diagram, ie the isomorphic diagram is equivalent to the precedence diagram. This is used to build a chromosome.

The method used to construct a valid random sequence of genes on the chromosome (isomorphic diagram) is as follows:

Step 1: Generate an empty chromosome with a number of genes equal to the number of tasks.

Step 2: Select a task set that does not have precedence.

Step 3: Select an available task at random and add it to the chromosome.

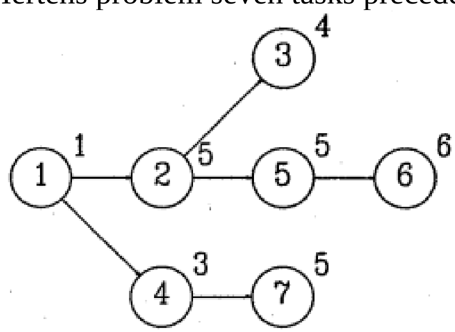
Step 4: Remove the selected task from the task set without precedence.

Step 5: Add all immediate successor tasks to the aggregated task, as long as all of its predecessors are already on the chromosome.

Step 6: If there are still unassigned tasks, go back to step 3; otherwise, terminate the chromosome.

Figures 1 and 2 show the precedence diagram and the isomorphic representation, respectively, for the Mertens problem.

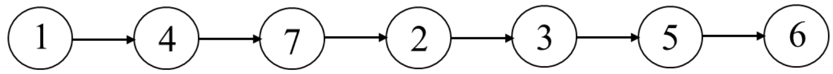
Figure 1. Mertens problem seven tasks precedence diagram



Source: Scholl (1993)

Ride Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

Figure 2. Mertens problem isomorphic representation



Source: Own elaboration

Initial population

The initial population of chromosomes is generated randomly, and the number of chromosomes to use is defined by the user. Many of the possible gene combinations are irrelevant because they violate precedence constraints. To generate the initial population, the isomorphic diagram construction method is used. Thus, it is guaranteed that the generated chromosomes maintain a valid sequence of genes. Table 1 shows a chromosome for the Mertens problem.

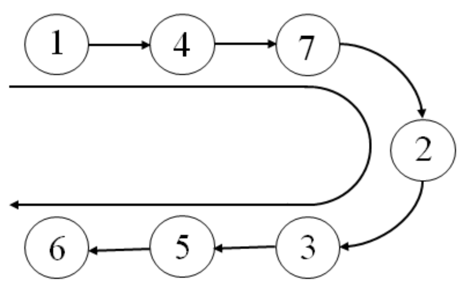
Table 1. Mertens problem chromosome

Chromosome	Genes						
	1	4	7	2	3	5	6

Source: Own elaboration

Since each chromosome is represented by an isomorphic diagram, this can be used to graphically show how the U-shaped line would be represented once the problem has been solved. Figure 3 shows the U-shaped graphical representation for the Mertens problem.

Figure 3. Mertens problema U-shaped graphical representation



Source: Own elaboration

Decoding

Chromosomes are generated in such a way that the sequence does not violate precedence constraints, which allows tasks to be assigned multiple workstations instead of one (Martínez, 2015). The decoding process refers to the procedure by which the chromosome genes (tasks) are assigned to the workstations and the way in which they are generated.

When this process ends, a solution is obtained, which shows a fitness index (number of workstations), a smoothness index and a computational time.

The following notations used by Baykasoğlu and Özbakır (2006) are used for the development of the algorithm.

N	number of tasks
T	Cycle time
μi(Tj)	Average processing time of task i
σi	Standard deviation of the processing time of task i
Pk	Probability that the station time exceeds the cycle time
Zk	Random variable with mean of 0 and standard deviation of 1
F(Zk)	Accumulated value of the Zk function
α	Upper limit of the probability that the station time exceeds the cycle time
Kα	α quantile of the standard normal distribution
σi^2	Variance of the processing time of task i

The method used to decode the chromosome is described below:

1. Create an empty workstation.
2. 2. Select the start and end tasks, and assign one of them to the first workstation.
3. 3. Calculate the probability that the station time exceeds the cycle time using equations 1 and 2 (Baykasoğlu y Özbakır, 2006).

$$Pk = 1 - F(Zk) \quad (1)$$

$$Zk = \frac{(T - \Sigma\mu i)}{\sqrt{\Sigma\sigma i^2}} \quad (2)$$

4. If the probability that the station time exceeds the cycle time is less than the value of α, the assignment of tasks to the station continues.
5. If the probability that the station time exceeds the cycle time is greater than the value of α, the next station is opened and the assignment of tasks continues.

6. The leftmost tasks are added if their ancestors are already on the chromosome, and the rightmost tasks are added if their successors have already been assigned.
7. Go back to step 3, and repeat the process until you finish the assignment of tasks; then finish the process.

Variance Generation

The literature on the stochastic U-shaped line balancing problem is very limited. Although the methodologies that have been proposed show the development of the method to arrive at the solution, they do not teach specific values for the mean and the variance of the tasks. In this sense, Armin Scholl (1993) proposed a set of problems, which have been used by different authors in solutions to the line balancing problem; however, it is difficult to find problems in the literature that show the specific values for the variance of the tasks; consequently, it was necessary to develop a method and combine it with the Carraway approach used by Urban and Chiang (2006) for the generation of such variances.

The variance is randomly generated using part of the Carraway approach. In this, random variance values are generated in two intervals $[0, (Ti/4)^2]$ for low variance and $[0, (Ti/2)^2]$ for high variance and using the minimum cycle times to generate a range of random values. The Mertens problem is used to show the procedure. Table 2 shows the mean tasks time for this problem.

Table 2. Mertens (1967) problem mean tasks time

Task	Mean tasks time
1	1
2	5
3	4
4	3
5	5
6	6
7	5
Cycle time 8	

Source: Own elaboration

1. The maximum values of Zk were determined as 1.28, 1.645, and 1.96 (Urban and Chiang 2006). Using equation 2, the following equations can be developed and a maximum value of the variance determined for each task.

$$\sigma i = \frac{(C-\mu i)}{Zk} \quad (3)$$

$$\sigma i^2 = [\frac{(C-\mu i)}{Zk}]^2 \quad (4)$$

2. Calculation of variance task 1.

$$\sigma i = \frac{(8-1)}{1.96} = 3.571, \qquad \sigma i = \frac{(8-1)}{1.645} = 4.255, \qquad \sigma i = \frac{(8-1)}{1.28} = 5.469$$

$$\sigma i^2 = 12.755 \qquad \sigma i^2 = 18.105 \qquad \sigma i^2 = 29.909$$

It is observed that the smallest variance calculated is for the Zk value of 1.96. This is the value that is selected as the maximum variance for this task. Also, it is used as the maximum value for the range of random values for the variance of this task. This is selected because any larger value of variance would not generate any solution, that is, there is no way to assign the task (in this case, 1 to some workstation9, since a larger variance would exceed the probability that the The station time exceeds the cycle time.The same procedure is carried out for the missing tasks. Then, in the results table 3, the Carraway approach for the selection of the interval of the variance is included.

Table 3. Calculated variance results

Task	Carraway variance range	Standar deviation (σ) for Zk values			Variance (σ²)	
	$[0,(Tj/4)^2]$	Z=1.96	Z=1.645	Z=1.28	σ^2 Calculated	σ^2 Random
1	0.0625	3.571	4.255	5.469	12.755	0.017
2	1.5625	1.531	1.824	2.344	2.343	0.165
3	1	2.041	2.432	3.125	4.165	0.257
4	0.5625	2.551	3.040	3.906	6.508	0.541
5	1.5625	1.531	1.824	2.344	2.343	0.987
6	2.25	1.020	1.216	1.563	1.041	0.976
7	1.5625	1.531	1.824	2.344	2.343	1.556

Source: Own elaboration

3. The values of the columns $[0,(Tj/4)^2]$(Carraway variance range) and calculated σ^2 are compared, and the smaller values are selected. In this example, the values in column $[0,(Tj/4)^2]$ are selected, since they are the smallest for tasks 1, 2, 3, 4, 5 and

Ride Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

7; for task 6 the value of the calculated σ² column is selected. Table 4 shows the intervals of the variance and the random results for it.

Table 4. Randomized results for variance

Task	Maximum variance values	Interval variance	σ² Random
1	0.0625	0 - 0.625	0.017
2	1.5625	0 - 1.5625	0.165
3	1	0 - 1	0.257
4	0.5625	0 - 0.5625	0.541
5	1.5625	0 - 1.5625	0.987
6	1.041	0 - 1.041	0.976
7	1.5625	0 - 1.5625	1.556

Source: Own elaboration

4. With the randomly generated variance data, Table 5 is created with the mean and variance values for the algorithm.

Table 5. Algorithm data

Task	Mean task time	Variance
1	1	0.017
2	5	0.165
3	4	0.257
4	3	0.541
5	5	0.987
6	6	0.976
7	5	1.556

Source: Own elaboration

Algorithm Development

The steps for the solution of the generated chromosome are described below:

1. Place the possible assignable tasks (1,6) to the lock station 1.
2. Select one of the tasks at random.
3. Determine the probability that the station time will exceed the cycle time.
4. If the probability that the station time exceeds the cycle time is less than the value of α, the assignment of tasks to station 1 continues.
5. If the probability that the station time exceeds the cycle time is greater than the value of α, station 2 is opened and the assignment of tasks continues.

To exemplify the solution process of the algorithm, the chromosome previously shown in table 1 and the data from table 5 are used. Table 6 shows the steps for the solution of the mentioned chromosome. The proposed cycle time is 10 and the proposed probability is 95% (α = 0.05).

Table 6. Chromosome solution

Probability 95 %, α = 0.05, CT = 10								
Tasks to be assigned	Selected task	μ	Σμ	σ^2	Σσ^2	√Σσ^2	P*k*	Work station
1,6	1	1	1	0.017	0.017	0.130	1-F((10-1)/0.130) = 0	1
4,6	6	6	7	0.976	0.993	0.996	1-F((10-7)/0.996) = 0.001	1
4,5	4	3	10	0.541	1.534	1.239	1-F((10-10)/1.239) = 0.5	2
7,5	5	5	8	0.987	1.528	1.236	1-F((10-8)/1.236) = 0.052	3
7,3	3	4	9	0.257	1.244	1.115	1-F((10-9)/1.115) = 0.184	4
7,2	7	5	9	1.556	1.813	1.346	1-F((10-9)/1.346) = 0.228	5
2	2	5	10	0.165	1.721	1.312	1-F((10-10)/1.312) = 0.5	6

Source: Own elaboration

- Operation 1

Calculation of the probability that the time at station 1 exceeds the cycle time using equations 1 and 2 with task 1 assigned:

1-F((10-1)/0.130)

$$Zk = \frac{(10-1)}{0.130}$$

Z*k* = 69.23

P value from Z table:

F(Z*k*) = P(x<10) = 1

P*k* = P(x>10) = 1 - P(x<10) = 0

- Operation 2

1-F((10-7)/0.996)

$$Zk = \frac{(10-7)}{0.996}$$

Z*k* = 3.012

P value from Z table:

F(Zk) = P(x<10) = 0.9987

Pk = P(x>10) = 1 - P(x<10) = 0.001

- Operation 3

  1-F((10-7)/0.996)

$$Zk = \frac{(10-10)}{1.239}$$

Zk = 0

P value from Z table:

F(Zk) = P(x<10) = 0.5

Pk = P(x>10) = 1 - P(x<10) = 0.5

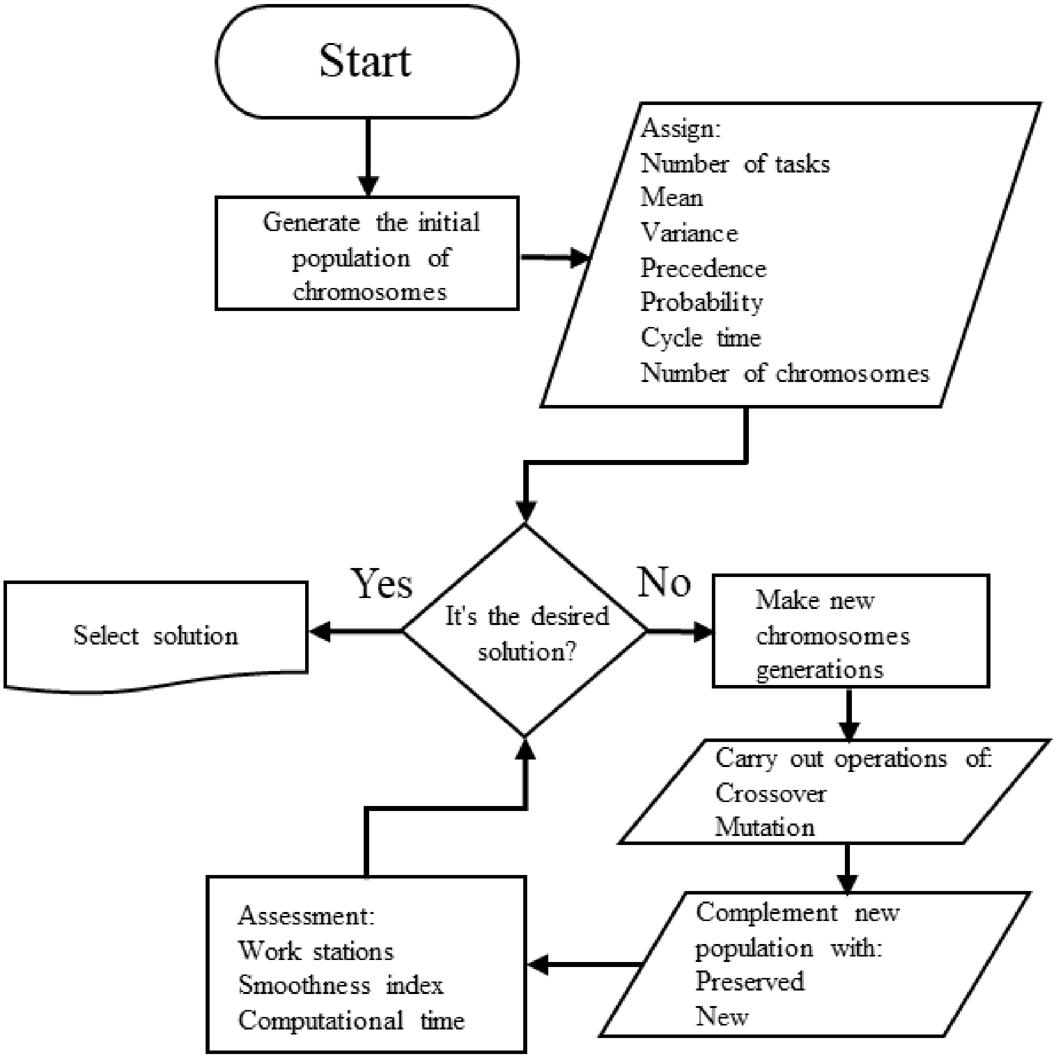
At the end of operation 3, it is observed that the probability that the time of station 1 exceeds the cycle time is greater than the value of α; therefore, Season 2 opens.

The operations for each of the tasks selected to be assigned to the following stations are performed in the same way. The solution for this chromosome results in six workstations.

Computational solution

The computational algorithm develops solutions by searching for chromosomes that generate feasible solutions through genetic operations. The process involves the following: the user defines the initial populations, the most suitable chromosomes are selected to carry out the crossover operation, a random selection of chromosomes is made for the mutation operation, and the new population is complemented with more suitable chromosomes for be conserved and with new chromosomes. This process continues until the established number of generations is reached. The block diagram of figure 4 represents the process of the algorithm.

Figure 4. Algorithm process



Source: Own elaboration

Results

To evaluate the algorithm for the stochastic type 1 U-shaped line balancing problem, the set of line balancing problems presented by Armin Scholl (1993) was used, which has been used by several authors to test different solution methodologies. to the line balancing problem. This set of problems proposes task times, which were considered as the mean task time (μi).

Seven problem categories are used for algorithm evaluation: Mertens (7 tasks), Bowman (8 tasks), Jaeschke (9 tasks), Jackson (11 tasks), Mitchell (21 tasks), Heskiaoff (28 tasks), and Killbridge. (45 tasks). The problems are evaluated in two ranges of variance (high and low variance), which allows to visualize the impact on the solutions with different ranges in the variance of the tasks. Task completion probabilities were set to 0.90, 0.95, and 0.97 (Kα = 1.28, 1.645, and 1.96, respectively). The combination of these categories with their respective cycle times, variance ranges, and different probabilities generate a total of 165 problems. These were solved with a 2.3 GHz personal computer. Tables 7 and 8 show the results highlighted in bold of the computational development of the algorithm and the existing solutions in the literature of Baykasoğlu y Özbakır (2006).

Table 7. Algorithm computational development results for low variance

Low variance																	
Problems	Tasks numbers	Cycle time	K(1-α) = 1.96 Probability 97.5 %					K(1-α) = 1.645 Probability 95 %					K(1-α) = 1.28 Probability 90 %				
			SI	Solution WS		CPT		SI	Solution WS		CPT		SI	Solution WS		CPT	
					Existing		Existing			Existing		Existing			Existing		Existing
Mertens	7	8	1.354	6	5	0.118	0.203	1.354	6	5	0.053	0.281	1.354	6	5	0.049	0.078
		10	1.414	5	4	0.102	0.17	1.414	5	4	0.053	0.201	2.179	4	4	0.045	0.18
		15	0.577	3	3	0.11	0.079	0.577	3	3	0.056	0.203	0.577	3	3	0.049	0.22
		18	0.707	2	2	0.112	0.281	0.057	2	2	0.707	0.155	0.707	2	2	0.043	0.187
Bowman	8	20	5.082	6	6	0.141	0.172	5.082	6	6	0.045	0.203	2.75	5	5	0.037	0.094
Jaeschke	9	6	N/S/F		8	N/S	0.172	N/S/F		8	N/S/F	0.203	N/S/F		8	N/S	0.156
		7	1.541	8	7	0.161	0.157	1.541	8	7	0.113	0.172	1.62	8	7	0.055	0.23
		8	1.62	8	7	0.058	0.172	1.62	8	7	0.032	0.09	1.927	7	7	0.032	0.171
		10	2.12	6	5	0.181	0.141	2	5	5	0.152	0.141	2	5	5	0.109	0.13
		18	0.816	3	3	0.187	0.14	0.816	3	3	0.127	0.11	2.08	3	3	0.057	0.203
Jackson	11	9	1.414	8	7	0.097	0.17	1.414	8	7	0.055	0.204	1.5	8	7	0.048	0.2
		10	1.69	7	7	0.151	0.063	1.69	7	7	0.071	0.183	1.69	7	7	0.068	0.172
		13	1.095	5	5	0.137	0.14	1.414	5	5	0.066	0.204	1.264	5	5	0.066	0.13
		14	1.264	5	4	0.132	0.188	1.264	5	4	0.073	1.1	0.707	4	4	0.057	0.24
		21	0.816	3	3	0.126	0.187	0.816	3	3	0.069	0.14	0.816	3	3	0.063	0.157
Mitchell	21	15	2.774	10	N/S/F	0.153	N/S/F	1.632	9	N/S/F	0.101	N/S/F	1.563	9	9	0.101	0.297
		21	1.647	7	6	0.173	0.5	0.707	6	6	0.105	0.843	0.707	6	6	0.103	0.26
		26	1.183	5	5	0.154	0.34	0	5	5	0.128	0.344	0.183	5	5	0.126	0.234
		35	1.5	4	4	0.254	0.28	5.408	4	4	0.227	0.21	2.692	4	4	0.167	0.281
		39	8.139	4	4	0.301	0.21	1.29	3	4	0.225	0.235	1.29	3	3	0.218	0.156
Heskiaoff	28	205	12.69	6	6	0.138	11.031	14.85	6	6	0.166	3.297	10.23	6	6	0.131	0.343
		216	12.11	6	6	0.134	1.97	17.34	6	6	0.186	0.1	23.54	6	6	0.171	0.25

		256	12.88	5	5	0.311	0.405	18.15	5	5	0.183	0.36	23.75	5	5	0.202	0.328
		324	18.61	4	4	0.282	0.453	20.84	4	4	0.205	0.25	28.52	4	4	0.216	0.454
		342	29.77	4	4	0.247	0.328	36.78	4	4	0.197	0.32	65.6	4	4	0.239	0.406
Killbridge	45	79	6.073	9	9	0.346	0.39	6.904	9	9	0.275	0.39	8.062	9	8	0.289	5.203
		92	11.34	8	8	0.349	0.594	5.707	7	8	0.353	0.391	5.644	7	7	0.294	1.37
		110	4.163	6	6	0.395	0.4	5.887	6	6	0.334	0.4	5.228	6	6	0.336	0.594
		138	9.777	5	5	0.412	0.578	12.17	5	5	0.407	0.2	21.24	5	5	0.42	0.39
		184	33.79	4	4	0.48	0.391	42.05	4	4	0.437	0.112	47.15	4	4	0.399	0.45
N/S/F No feasible solution found																	

Source: Own elaboration

Table 8. Algorithm computational development results for high variance

High variance																	
Problems	Tasks numbers	Cycle time	K(1-α) = 1.96 Probability 97.5 %					K(1-α) = 1.645 Probability 95 %					K(1-α) = 1.28 Probability 90 %				
			SI	Solution WS		CPT		SI	Solution WS		CPT		SI	Solution WS		CPT	
					Existing		Existing			Existing		Existing			Existing		Existing
Mertens	7	8	1.354	6	N/S/F	0.134	N/S/F	1.354	6	N/S/F	0.055	N/S/F	1.354	6	5	0.056	6.172
		10	1.354	6	5	0.078	0.1	1.354	6	5	0.032	0.18	2.489	5	5	0.031	0.14
		15	0.577	3	3	0.141	0.13	0.577	3	3	0.071	0.125	0.577	3	3	0.03	0.11
		18	0.577	3	3	0.149	0.09	0.577	3	3	0.11	0.078	0.707	2	2	0.049	0.075
Bowman	8	20	6.928	7	6	0.07	7.12	5.016	6	6	0.032	0.171	5.016	6	6	0.108	0.2
Jaeschke	9	8	1.62	8	7	0.132	0.922	1.62	8	7	0.036	0.531	1.62	8	7	0.033	1.47
		10	1.927	7	7	0.033	0.125	1.927	7	7	0.071	0.219	2.121	6	7	0.059	0.187
		18	0.816	3	3	0.152	0.234	0.816	3	3	0.121	0.175	0.816	3	3	0.1	0.3
Jackson	11	10	1.414	8	N/S/F	0.068	N/S/F	1.581	8	N/S/F	0.024	N/S/F	1.5	8	7	0.027	0.203
		13	1.732	6	5	0.098	2.04	1.732	6	5	0.074	0.985	1.095	5	5	0.036	1.402
		14	1.095	5	5	0.068	0.891	1.095	5	5	0.084	0.25	2.236	5	5	0.1	0.772
		21	0.816	3	3	0.116	0.766	0.816	3	3	0.044	0.187	0.816	3	3	0.035	0.31
Mitchell	21	21	1.274	8	8	0.08	0.344	1.362	7	7	0.083	0.231	1.647	7	7	0.093	0.516
		26	2.121	6	6	0.196	0.782	1.957	6	6	0.092	0.344	2.366	5	5	0.092	1.89
		35	0.866	4	4	0.257	5.468	0.866	4	4	0.173	0.562	0.866	4	4	0.16	0.281
		39	2.692	4	4	0.269	0.174	6.224	4	4	0.219	0.235	6.576	4	4	0.185	0.344
Heskiaoff	28	205	17.41	8	8	0.298	0.547	20.37	8	7	0.143	1.641	14.75	7	7	0.101	0.437
		216	25.95	8	7	0.491	1.976	27.15	7	7	0.135	0.563	23.76	7	6	0.128	5.593
		256	23.12	6	6	0.2	0.48	18.75	6	6	0.157	0.53	24.06	6	5	0.149	1.453
		324	19.57	5	5	0.15	0.691	41.19	5	4	0.184	0.328	13.1	4	4	0.114	0.531
		342	48.67	5	4	0.249	0.531	5.787	4	4	0.192	0.657	11.25	4	4	0.21	0.18
Killbridge	45	92	10.65	9	8	0.198	0.61	4.769	8	8	0.201	5.547	7.632	8	8	0.172	0.594
		110	8.115	7	7	0.288	0.609	9.433	7	7	0.214	0.984	21.42	7	6	0.226	4.14
		138	22.61	6	6	0.313	0.782	4.289	5	6	0.302	0.39	8.148	5	5	0.276	0.797
		184	11.85	4	4	0.345	0.593	14.35	4	4	0.331	0.781	31.6	4	4	0.302	0.593
N/S/F: No feasible solution found																	

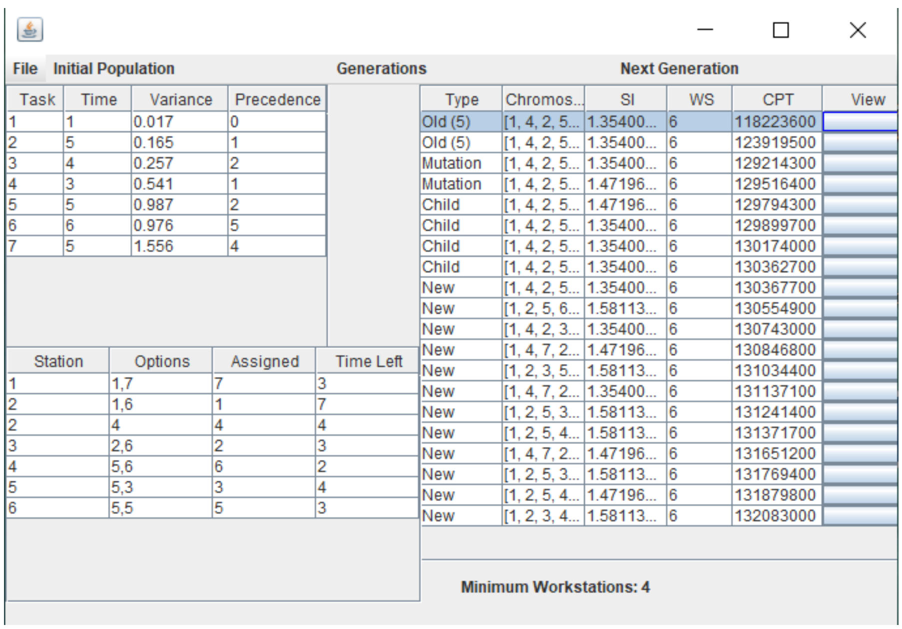
Source: Own elaboration

At the end of the evaluation process, the algorithm shows the following results:

- Smoothness index (SI)
- Number of workstations (WS)
- Computational time (CPT)

The smoothness index shows how close the generated chromosome (solution) is to achieving equilibrium on the production line. A number closer to zero is better, because the smaller this value is, the closer you are to achieving perfect balance. The number of work stations indicates the number of work stations that are generated by each chromosome. The computational time indicates the time taken by the algorithm to generate the chromosomes (time units are shown in nanoseconds). Figure 5 shows an example of the computational solution of the algorithm.

Figure 5. Algorithm computational solution



Source: Own elaboration

Discussion

From the results obtained, a comparative table is made with the existing solutions of Baykasoğlu and Özbakır (2006) for high and low variance, where the quantity and percentage for the following results are shown:

- Ws major. Problems in which one more WS (workstation) was generated.

- Similar WS, lower CPT. Problems in which the amount of WS is similar with a lower CPT (computational time).
- Similar WS, higher CPT. Problems in which the amount of WS is similar with a higher CPT.
- WS minor. Issues for which a lower WS number was generated.
- No feasible solution was found. Problems for which no solution was found.
- Total problems. Number of problems performed.

Table 9. Comparison results for low and high variance

Low variance						
	WS larger	WS Similar smaller CPT	WS Similar larger CPT	WS smaller	No feasible solution found	Overall problems
	18	52	16	1	3	90
%	20.0	57.8	17.8	1.1	3.3	
High variance						
	WS larger	WS Similar smaller CPT	WS Similar larger CPT	WS smaller	No feasible solution found	Overall problems
	18	46	5	6	0	75
%	24.0	61.3	6.7	8.0	0.0	

Source: Own elaboration

From the analysis of the results, we can affirm that the evaluated algorithm provides better solutions for high variance problems, since only for the highest WS result is a difference of 4 % observed. On the other hand, in the remaining results the percentages are better. In addition, it can be seen that six better solutions were found than the existing ones. In this sense, the solutions for some problems show variation for a maximum additional task, although in most the number of workstations are similar to the existing ones. Regarding the computation times of the solutions, no great difference is observed, since most of the times are below 1 second. Table 10 shows the time averages for both variances.

Table 10. Computational times averages

Computational times averages		
Variance	Algorithm solutions	Existing solutions
Low	0.177	0.522
High	0.146	0.991

Source: Own elaboration

In short, it can be observed that the average times of the algorithm's solutions are better than the existing ones, which shows the algorithm's ability to find solutions in a shorter computational time.

Conclusions

Carrying out this work reaffirmed the effectiveness of computational genetic algorithms for solving complex problems, since results similar to those found in the literature were found through validation.

On the other hand, it should be noted that in many cases the balancing of lines is carried out based on the experience of the personnel in charge of this task, that is, methodologies based on metaheuristics or other tools are not used. An empirical balancing, therefore, is not always the most appropriate, since it may imply increases in production costs. However, with this new tool based on genetic algorithms, a more adequate balancing can be performed on U-shaped lines with stochastic task times. One of its most outstanding characteristics is its versatility, since it allows different parameters to be varied to obtain a considerable number of solutions. This serves to experiment and observe the different aspects that can improve or optimize the operation, with which a balance can be achieved with the least amount of human resources possible.

Likewise, the validation of the algorithm was a very extensive process, since the problems carried out were developed with different probability values and variance ranges. This was very important because it allowed the algorithm to be subjected to different scenarios in order to achieve as even a comparison as possible.

Finally, it is important to note that the variances of both solutions were randomly generated, so it is difficult to conclude that an algorithm provides the best solution to the type 1 stochastic U-shaped line balancing problem. realistically, it would be necessary to carry out the computational study with equal variances.

Future lines of research

With the results obtained, we have a clearer idea of the solutions that the algorithm can show. In fact, as a solution evaluation measure, the algorithm shows the SI (smoothness index), although there are also three measures that help to evaluate the solution. Future work, therefore, could improve the algorithm and incorporate these three measures of solution evaluation:

1. Balance delay.
2. Line efficiency.
3. Balance sheet efficiency.

References

Baykasoğlu, A. and Özbakır, L. (2006). Stochastic U-line balancing using genetic algorithms. *The International Journal of Advanced Manufacturing Technology*, *32*, 139-147. https://doi.org/10.1007/s00170-005-0322-4.

Cortez, A. (2004). Teoría de la complejidad computacional y teoría de la computabilidad. *Revista de Investigación se Sistemas e Informática*, *1*(1), 102–105. http://inventio.uaem.mx/index.php/inventio/article/view/324

Esparza, D. (2009). *EDA para la resolución de problemas de optimización con restricciones* (tesis de maestría). Centro de Investigación en Matemáticas. https://cimat.repositorioinstitucional.mx/jspui/bitstream/1008/192/2/TE%20311.pdf

Gallego, R. A., Escobar, A. y Toro, E. M. (2015). *Técnicas heurísticas y metaheurísticas de optimización*. Universidad Tecnológica de Pereira.

Maldonado, C. E. (2016). Metaheurísticas y resolución de problemas complejos. *Revista Colombiana de Filosofía de la Ciencia*, *16*(33), 169-185. https://doi.org/10.18270/rcfc.v16i33.1938

Martínez, U. (2015). *Metaheuristics Approach to Solving U-Shaped Assembly Line Balancing Problems using a Rule-Base Coded Genetic Algorithm* (doctoral dissertation). Department of Mechanical Engineering, Colorado State University.

Orejuela, J. P. y Flórez, A. (2019). Balanceo de líneas de producción en la industria farmacéutica mediante programación por metas. *INGE CUC*, *15*(1), 109-122. http://dx.doi.org/10.17981/ingecuc.15.1.2019.10

Scholl, A. (1993). *Data of assembly line balancing problems*. TH Darmstadt. https://assembly-line-balancing.de/wp-content/uploads/2017/01/Scholl-1993-ALBData.pdf

Urban, T. L. and Chiang, W. C. (2006). An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. *Eur J Oper Res, 168*(3), 771–782. https://doi.org/10.1016/j.ejor.2004.07.027

Zhang, H., Zhang, C., Peng Y., Wang, D., Tian, G., XU Liu, X. and Peng, Y. (2018). Balancing Problem of Stochastic Large-Scale U-Type Assembly Lines Using a Modified Evolutionary Algorithm. https://doi.org/10.1109/ACCESS.2018.2885030

Contribution role	Autor (es)
Conceptualization	Demetrio Fermán Alvarez
Methodology	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same)
Software	Ulises Martínez Contreras
Validation	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same) Mirella Parada González (support) Arturo Woocay Prieto (support) Adán Valles Chávez (support)
Formal analisis	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same) Mirella Parada González (same) Arturo Woocay Prieto (same) Adán Valles Chávez (same)
Investigation	Demetrio Fermán Alvarez (main) Ulises Martínez Contreras (support)
Resources	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same) Mirella Parada González (support) Arturo Woocay Prieto (support) Adán Valles Chávez (support)
Date curation	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same) Mirella Parada González (support) Arturo Woocay Prieto (support) Adán Valles Chávez (support)
Original draft writing and preparation	Demetrio Fermán Alvarez (main) Ulises Martínez Contreras (support) Mirella Parada González (support) Arturo Woocay Prieto (support) Adán Valles Chávez (support)
Writing, proofreading and editing	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same) Mirella Parada González (same) Arturo Woocay Prieto (same) Adán Valles Chávez (same)
Visualization	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same) Mirella Parada González (same)

	Arturo Woocay Prieto (same) Adán Valles Chávez (same)
Supervision	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same)
Project management	Demetrio Fermán Alvarez (same) Ulises Martínez Contreras (same)
Fund acquisition	Demetrio Fermán Alvarez