






New Hybrid Algorithm for Supply Chain Optimization

Carlos Cevallos¹, Mario Peña² , and Lorena Siguenza-Guzman³  

¹ Faculty of Engineering, Universidad de Cuenca, Cuenca, Ecuador
carlos.cevallos@ucuenca.edu.ec

² Research Department, Universidad de Cuenca, Cuenca, Ecuador
mario.pena@ucuenca.edu.ec

³ Department of Computer Sciences, Faculty of Engineering,
Universidad de Cuenca, Cuenca, Ecuador
lorena.siguenza@ucuenca.edu.ec

Abstract. Optimization is the process of obtaining the best solutions to specific problems. In the literature, those problems have been optimized through a plethora of algorithms. However, these algorithms have many advantages but also disadvantages. In this article, a New Hybrid Algorithm for Supply Chain Optimization, NHA-SCO has been proposed in order to improve the benefits of objective function convergence. For the analysis of the results, three assembly companies have been utilized as case studies. These companies present their supply chains, i.e., networks where products flow from their raw material to the final products delivered to clients. These supply chains must satisfy different objectives, such as maximize benefits and service level and minimize scrap. For the evaluation of results, NHA-SCO has been compared to other well-known optimization algorithms. In the presented case studies, the NHA-SCO algorithm performs faster, or it converges in fewer iterations, obtaining similar or even better results than the other algorithms tested.

Keywords: Optimization · Supply chain · Metaheuristics · Hybrid algorithm

1 Introduction

The technological wave has grown in the last years. These changes have greatly improved decision-making; however, they have also brought new problems that involve more variables and information, complicating and streamlining these processes. In this scenario, it is essential to find fast and optimal solutions. Optimization is the process of obtaining the best solutions (optimal state) to specific problems. In this sense, Global Optimization is a technique to solve any optimization problem to find an optimal state from a large set of elements or states [1]. In the literature, a plethora of algorithms has been developed to optimize different problems. However, these algorithms have many advantages but also disadvantages. Thus, new algorithms have been developed which combine different characteristics of other algorithms to optimize faster while getting better results [2].

In the industry, companies try to improve their services by optimizing their Supply Chain (SC). [3] defined SC as an entity network where the raw material or product flows through it. This network includes distribution centers, markets, clients, transporters, industries, and providers. The study of SC is significant within organizations since it offers information to company stakeholders to make decisions and changes. These decisions and changes could enhance the behavior of an organization to improve its benefits and the quality provided to final users [4]. In this sense, global optimization allows optimizing a SC, but an optimizer must make the process; in this case, the chosen optimizer is an algorithm.

In [5], a hybrid algorithm was developed to optimize a SC. It consisted of combining different characteristics of several optimization algorithms to obtain a new algorithm that optimizes faster than others. The optimization algorithms used are known as metaheuristics. Metaheuristics are algorithms usually inspired by nature [6] that help to solve general problems utilizing objective functions [7–9]. This paper aims to describe an improvement of the previous algorithm proposed. Specifically, the changes made to enhance the algorithm are discussed, and also the reasons why these changes were made. To analyze the benefits of the new algorithm, three case studies were utilized. The first case study is the same as shown in [10], but variables were modified. The second case is similar to the previous; however, the SC is handled individually per product in each period. Finally, the last case study is the optimization of raw materials cutting through patterns. The rest of the paper is organized as follows. In Sect. 2, the case studies are presented. Section 3 briefly explains the optimization algorithms utilized. Section 4 describes the New Hybrid Algorithm for Supply Chain Optimization, NHA-SCO. Section 5 reports the results of the proposed algorithm. In Sect. 6, these results are discussed. Finally, the conclusions are included in the last part.

2 Case Studies

Three case studies were utilized in this work. The first case study is related to a company where televisions are assembled. The second and third case study belong to the same company in charge of assembling furniture. The names of the companies are not mentioned due to a confidentiality agreement signed with the companies.

2.1 Case Study I

The first case study corresponds to maximizing the benefit and service level in an assembly company. **Benefit:** Maximize the profit of the company, i.e., the resulting value of the sum of all distributed products multiplied by its sell price, subtracted by the amount of the costs of assembly, distribution, transportation, and storage (1). **Service Level:** Maximize the service level offered to clients (2).

$$\begin{aligned}
 f1 = & \sum_z \sum_j \sum_k \sum_b (PD_{zjkb} * PV_{kb}) - \sum_z \sum_j \sum_b P_{zjb} * (CM_{zjb} + CE_{zjb}) \\
 & - \sum_z \sum_j \sum_k \sum_b (PD_{zjkb} * CT_{zjkb}) - \sum_z \sum_j \sum_b (I_{zjb} * CI_{zjb})
 \end{aligned}
 \tag{1}$$

$$f2 = \frac{\sum_z \sum_j \sum_k \sum_b PD_{zjkb}}{\sum_z \sum_k \sum_b D_{zkb}} \tag{2}$$

where: b is the product, j is the plant, k is the market, z is the period. **Variables:** P_{zjb} is the total amount of product b assembled in plant j at period z , PD_{zjkb} is the quantity of product b sent from plant j to market k at period z , I_{zjb} is the final inventory of product b in plant j at period z . **Constants:** CM_{zjb} is the unit cost of raw material supplied to assemble b to plant j at period z , CE_{zjb} is the cost of assembly of product b in the plant j at period z , CI_{zjb} is the unit holding cost of product b in plant j at period z , PV_{kb} is the sale price of product b in market k , CT_{zjkb} is the cost of transporting one unit of product b from plant j to market k at period z , D_{zkb} is the demand for product b of customer k at period z , I_{jb0} is the initial inventory level of product b in the plant j , pt_{jb} is the time required to produce product b in plant j , tt_{zj} is the time available to produce in plant j at period z , SS_{zjb} is the safety stock of product b in plant j at period z .

Subject to the following **constraints**. Constraint (3) ensures that the quantity of distributed products in a market at each instant is less than the total amount of produced products in every plant. Constraint (4) guarantees that the time available at each plant in each moment is higher than the time required to produce products. Constraint (5) ensures that the products distributed to the markets from the plants should be less than the demand at each instant. Equation (6) defines the inventory balance for the products in each plant at each moment. Constraint (7) ensures that the inventory of products at the end of each instant is higher than the safety stock in each plant. The quantity of products produced, stored, and distributed in a market in each plant at every moment is always positive and are guaranteed through Constraint (8).

$$\sum_j P_{zjb} \geq PD_{zjkb}; \quad \forall z, k, b \tag{3}$$

$$\sum_b pt_{jb} * P_{zjb} \leq tt_{zj}; \quad \forall z, j \tag{4}$$

$$\sum_j PD_{zjkb} \leq D_{zkb}; \quad \forall z, k, b \tag{5}$$

$$I_{zjb} = I_{(z-1)jb} + P_{zjb} - \sum_k PD_{zjkb}; \quad \forall z, j, b \tag{6}$$

$$I_{zjb} \geq SS_{zjb}; \quad \forall z, j, b \tag{7}$$

$$P_{zjb}, PD_{zjkb}, I_{zjb} \geq 0; \quad \forall z, j, k, b \tag{8}$$

2.2 Case Study II

The second case study is the maximization of the SC benefit. **Benefit:** Maximize the value of the benefit obtained subtracted by the sum of distribution, inventory, and transportation costs (9).

$$f3 = \sum_{k,b} PVP_{kb} * PV_{kb} - \sum_b PR_b * CP_b - I * \sum_b CP_b * Inv_b * T_b - \sum_{k,b} CT_b * PD_{kb} \quad (9)$$

where: b is the product, k is the market. **Variables:** PV_{kb} is the quantity of product b sold in market k , PR_b is the production of product b , PD_{kb} is the quantity of product b distributed to market k . **Constants:** PVP_{kb} is the selling price of product b in market k , CP_b is the unit production cost of product b , Inv_b is the inventory of product b , T_b is the average inventory time of product b , I is the defined interest rate, CT_b is the cost of transport of product b , D_{kb} is the demand in the market k of the product b .

Subject to the following **constraints**. Constraint (10) ensures that the products distributed to the markets should be higher than the demand. Constraint (11) guarantees that the production plus the inventory of the products meet at least the demand. The quantity of products produced, distributed, and sold in a market is always positive and are guaranteed through Constraint (12).

$$\sum_{k,b} PD_{kb} \geq \sum_{k,b} D_{kb}; \quad \forall k, b \quad (10)$$

$$PR_b + Inv_b \geq PD_{kb}; \quad \forall k, b \quad (11)$$

$$PR_b, PD_{kb}, PV_{kb}, \geq 0; \quad \forall k, b \quad (12)$$

2.3 Case Study III

The last case study corresponds to the minimization of scrap produced by the cut of raw material. **Scrap:** Minimize scrap generated by plants (13).

$$f4 = \min\{cx : x \in Z_+^n\} \quad (13)$$

where: c is the cut pattern, x is the cut frequency, i is the raw material, j is the pattern. **Variables:** a_{ij} are the produced cuts, x_j are the frequency cuts applied. **Constants:** e_j is the inventory available, d_i is the demand for cut pieces.

Subject to the following constraints. Constraint (14) guarantees that the demanded quantity of cuts pieces is satisfied by the sum of the produced cuts multiplied by the frequency of the cuts. Constraint (15) verifies that the frequency cuts applied to each pattern do not exceed the available inventory. Constraint (16) satisfies that the quantity of cut produced and the frequency cuts implemented are always positive.

$$\sum_{j=1}^n a_{ij}x_j \geq d_i; \quad \forall i, j \quad (14)$$

$$\sum_{j=1}^n x_j \leq e_j; \quad \forall j \quad (15)$$

$$a_{ij}, x_j, \geq 0; \quad \forall i, j \quad (16)$$

3 Optimization Algorithms Utilized

The algorithms utilized were the following, Multi-objective Pareto archives evolutionary strategy (M-PAES), Multi-objective particle swarm optimization (MOPSO), Non-dominated Sort Genetic Algorithm 2 (NSGA-II) and NSGA-II K-means.

M-PAES is a memetic algorithm designed for multi-objective problem resolution. It uses the technique of local searching, specifically with the PAES (Pareto archives evolutionary strategy) procedure. The individuals used in each iteration combine and mate themselves to generate new individuals [11]. This algorithm takes time to be executed since iterations consists of many processes with several operations. MOPSO is a modification of the Particle Swarm Optimization (PSO) algorithm. It uses the Pareto frontier to determine the movement of the particles through the search space [12]. This change was made to compete with algorithms that work with multi-objective problems. NSGA-II is a genetic algorithm modified to work in multi-objective problems. It changes the sorting of individuals through a new method called non-dominated sorting [13]. uNSGA-II is a modification of NSGA-II that works with a few numbers of individuals. Therefore, the runtime is also reduced [14]. NSGA-II K-means is a modification of the NSGA-II algorithm. In this case, this algorithm tries to split the population into different clusters. Each cluster will pass for a NSGA-II process that takes less time for each group due to the number of members [15].

4 New Hybrid Algorithm Developed

4.1 Previous Hybrid Algorithm

[10] presented a hybrid algorithm which consisted of a combination of three algorithms: micro-algorithm (i.e., an optimization algorithm with a few number of individuals), NSGA-II, and MOPSO. NSGA-II was the basis algorithm. The parents' selection was made with MOPSO procedures. After each iteration, the population was reduced multiplying by a factor to decrease runtime and computer memory. As genetic algorithms, the crossover and mutation are based on probability, which relies on a randomly generated number. Mutation usually takes a random value for the entire space of the problem, but in this algorithm, a sub-space was created to find solutions faster. The previous hybrid algorithm was compared with the other two algorithms in only one case study. The case study was the minimization of costs of a SC and the maximization of the service level offered to clients. The algorithm was compared with NSGA-II and MOPSO algorithms. The results obtained showed that the hybrid algorithm was faster in runtime in comparison to the other algorithms [10].

4.2 Creation of Sub-spaces

The creation of sub-spaces was developed to improve the algorithm mutation. This consists of reducing the search space of individuals in each iteration. At the start of the problem, this sub-space would be like the same space, which is a width space. However, in the next generations, this should be reduced to have fewer options knowing that there

are the best individuals. This reduction relies on a variable xz , which will be cut in some iterations. This is something similar to PSO, where the particle is limited to the max speed.

4.3 New Hybrid Algorithm for Supply Chain Optimization

This algorithm for SC optimization, which was named NHA-SCO, has substantially changed since many features have been replaced, and others eliminated. In the selection operation, the MOPSO process is not always perfect because on some occasions could not converge. Therefore, it was necessary to change this selection method to NSGA-II non-dominated sorting. In each generation, the crossover operation is executed. The crossover and the mutation are similar to the previous algorithm. The mutations of the individuals have more diverse values from the new sub-space generated by xz . After generating the new child, the population is re-evaluated through NSGA-II non-dominated sorting. Before finishing the generation, it is necessary to calculate the generation mod of a number. This number depends on the max number of iterations. If the result is zero; then, xz reduces its value, and the population is increased. Micro-algorithm made a twist, instead of reducing the population, it has increased. This change was made due to, at the start of the process, individuals have more variable values. Thus, it would be easier to get parents in a short set of individuals. However, at a large number of iterations, the individuals will have similar values. For that reason, it is necessary to have more options to choose from. It is essential to know that the algorithm starts with a few individuals, e.g., 10 or 20. These changes produce that NHA-SCO takes more iterations to converge, but the speed has been reduced incredibly. Figure 1 presents the resulting flowchart of the new hybrid algorithm NHA-SCO.

5 Results

5.1 Case Study I

Many runs were made to find a mean of runtime and the number of iterations of each algorithm. Some algorithms converged to better values than others. These values are presented as the SC benefit and service level. In Table 1, the results of running the algorithms are presented. The results obtained were the runtime and the number of iterations required to converge; and values of the benefit and service level offered.

Table 1. Comparison of optimization algorithms for the Case Study I

Algorithm	M-PAES	MOPSO	Previous hybrid	NHA-SCO
Runtime	1020 s	51.32 s	24.79 s	8.08 s
Number of iterations	8141	2848	3110	3050
SC benefit (\$)	4314,239	4263,268	4359,815	4360,030
Service level	99%	98.6%	99.9%	99.9%

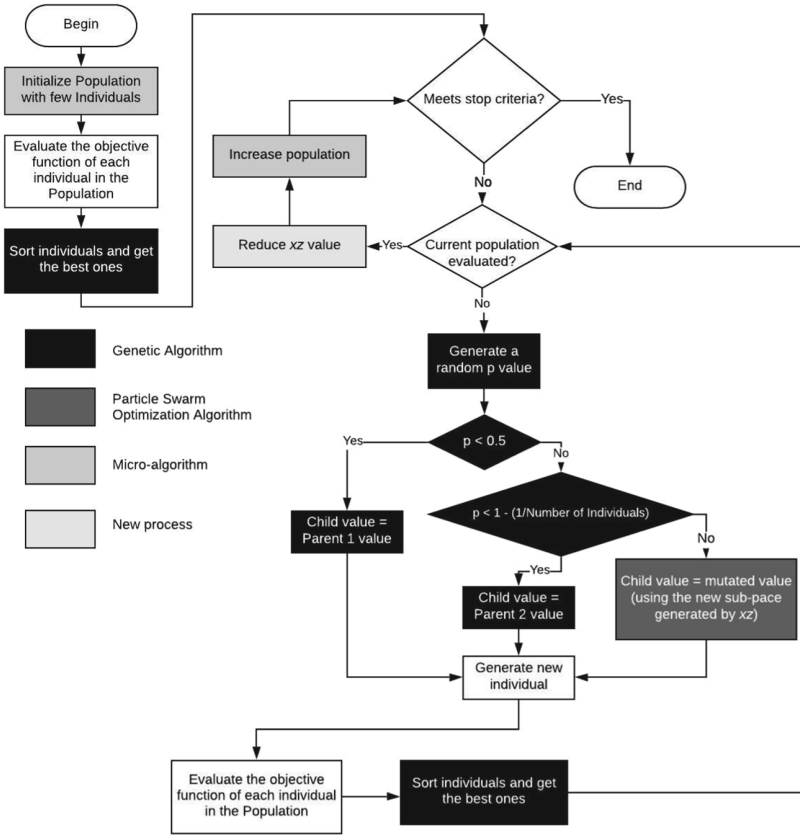


Fig. 1. Flowchart of the new hybrid algorithm NHA-SCO.

5.2 Case Study II

It is necessary to maximize the SC benefit. However, for this case, the SC is partially optimized. The running of the algorithm is for each product per period. There are 12 months; however, it is presented only the first two months of the year for illustration purposes. Table 2 and Table 3 show the expected benefit taken from [15], and the output values after running the four optimization algorithms.

Table 2. Results after running case study II with the optimization algorithms in January

Product	Expected value	uNSGA-II output value	NSGA-II K-means output value	Previous hybrid algorithm output value	NHA-SCO algorithm output value
PTC002B	1,258.25	1,084.21	1,258.25	1,258.25	1,258.25
PTC007B	1,027.91	801.91	1,027.91	1,027.91	1,027.91
PTC009D	156.73	148.65	148.65	156.73	156.73
PTC012B	801.87	676.73	801.02	801.87	801.87
PTC018B	775.21	533.49	775.21	775.21	757.46
PTC021B	593.47	509.77	593.47	593.47	593.47

Table 3. Results after running case study II with the optimization algorithms in February

Product	Expected value	uNSGA-II output value	NSGA-II K-means output value	Previous Hybrid algorithm output value	NHA-SCO algorithm output value
PTC001B	70.4	70.39	70.39	70.39	70.39
PTC002B	428.58	418.59	418.59	428.57	428.57
PTC007B	12.58	0	0	12.57	12.57
PTC008B	1,681.5	1,681.49	1,681.49	1,681.49	1,681.49
PTC009D	519.19	519.19	519.19	519.19	519.19
PTC010B	2.66	0	0	2.65	2.65
PTC012B	269.56	266.81	266.81	269.56	266.81
PTC018B	913.43	913.43	913.43	913.43	913.43
PTC021B	609.7	609.69	609.69	609.69	609.69

5.3 Case Study III

In this case, it is necessary to minimize the scrap generated by the plants after cutting the raw material. It would be running each product per month. As in Case Study II, Table 4, and Table 5 present the expected values and output values of each optimization algorithm.

Table 4. Results after running Case Study III with the optimization algorithms in January

Product	Expected value	uNSGA-II output value	NSGA-II K-means output value	Previous hybrid algorithm output value	NHA-SCO algorithm output value
PTC002B	1,258.25	1,084.21	1,258.25	1,258.25	1,258.25
PTC007B	1,027.91	801.91	1,027.91	1,027.91	1,027.91
PTC009D	156.73	148.65	148.65	156.73	156.73
PTC012B	801.87	676.73	801.02	801.87	801.87
PTC018B	775.21	533.49	775.21	775.21	757.46
PTC021B	593.47	509.77	593.47	593.47	593.47

Table 5. Results after running Case Study III with the optimization algorithms in February

Product	Expected value	uNSGA-II output value	NSGA-II K-means output value	Previous hybrid algorithm output value	NHA-SCO algorithm output value
PTC001B	70.4	70.39	70.39	70.39	70.39
PTC002B	428.58	418.59	418.59	428.58	428.58
PTC007B	12.58	0	0	12.577062	12.577062
PTC008B	1,681.5	1,681.5	1,681.5	1,681.5	1,681.5
PTC009D	519.19	519.19	519.19	519.19	519.19
PTC010B	2.66	0	0	2.655796	2.655796
PTC012B	269.56	266.81	266.81	269.56	266.81
PTC018B	913.43	913.43	913.43	913.43	913.43
PTC021B	609.7	609.69	609.69	609.69	609.69

6 Discussion

Several algorithms have been tested to compare the results. Each case study has its algorithms to execute. Findings represent important algorithms features, such as speed and convergence values, and the number of iterations that the algorithm takes to obtain the objective value. In the next subsections, the results obtained in each case study will be discussed.

6.1 Case Study I

Figure 2a, and b show the convergence of the benefit obtained and service level offered to clients, respectively. These figures present the next features:

- M-PAES takes more runtime, but it gets better solutions to the problem.
- MOPSO does not converge to good solutions, but it takes less time than M-PAES.
- Hybrid algorithms have faster runtime, but they use more iterations to converge than MOPSO.

NHA-SCO and the previous algorithm are better in convergence to the solution. However, the previous algorithm converges faster in the first iterations. Although this could seem a disadvantage, it is not, since the number of iterations is compensated with the speed of the algorithm. This is presented in Table 1, where the number of iterations to converge of the hybrid algorithms are similar, but the runtime of the new algorithm and the old are different. New hybrid algorithm runtime is 33,33% of the previous hybrid algorithm runtime. The number of iterations of both hybrid algorithms is more significant than MOPSO, but the service level convergence is better than MOPSO, as shown in Figure 2a, and b.

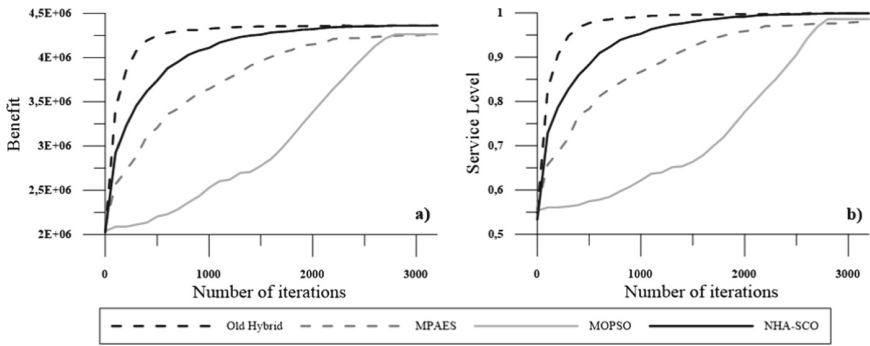


Fig. 2. a) Convergence of the benefit function. b) Convergence of the service level function.

6.2 Case Study II

Case Study II results, presented in Table 6, show that NHA-SCO takes more time in completing the running than the old algorithm. However, the new hybrid algorithm converges to the solution in fewer iterations than other algorithms. uNSGA-II takes the minimum runtime, but it has the worst number of iterations to converge. NSGA-II K-means takes more time but gets better results than uNSGA-II.

Table 6. Comparison of optimization algorithms for Case Study II.

	uNSGA-II	NSGA-II K-means	Previous hybrid	NHA-SCO
Runtime	0.33 s	59.54 s	2.31 s	7.32 s
Iterations	1013088	390410	148059	131695

6.3 Case Study III

In the last case study shown in Table 7, there is a change. NHA-SCO is faster than the previous algorithm; however, the number of iterations is the same as Case Study II. As the results presented during the last case study, uNSGA-II takes the minimum runtime and the worst number of iterations to converge. NSGA-II K-means takes more time but gets better results.

Table 7. Comparison of optimization algorithms for Case Study III.

	uNSGA-II	NSGA-II K-means	Previous hybrid	NHA-SCO
Runtime	0.66 s	104.4 s	51.74 s	18.7 s
Iterations	71180	40178	28018	26570

NHA-SCO has shown to be faster than the old algorithm. This is produced because the new algorithm at the beginning of the running has few individuals with different values, so it is easier to get new parents with the best values. In the next generations, when individuals have similar values, the number of individuals is more significant to have more options to choose from. Whereas the previous hybrid algorithm, at the start of the running, has many individuals; and, in the next iterations, the number of individuals is reduced. This is a problem because it would be harder for the algorithm to get the new best solutions due to the few individuals who have similar values between them.

Jamshidi, Fatemi, and Karimi developed an algorithm based on the Taguchi method [16]. This method is in charge of selecting parents for the next iterations. As NHA-SCO, the algorithm presented by the authors uses PSO methods for the selection. Therefore, the algorithm is as fast as the new hybrid algorithm. Although this last algorithm uses Micro-algorithms, thus, the population will decrease; therefore, the algorithm will be faster and will use less machine memory. Kuo and Han present three variations of algorithms [17]. These hybrid algorithms use the processes of the PSO and Genetic Algorithm. PSO is very slow in runtime; hence, the algorithms take more time to converge to the results. On the contrary, NHA-SCO uses the genetic process to increase speed to converge to the solution.

7 Conclusion

There are always failures in the systems; thus, there will always be space for improvements. A newly developed algorithm could still be improved. The new hybrid algorithm has found flaws in the theory of the previous algorithm. NHA-SCO has taken the old algorithm disadvantages to strengthen it and get better results and solutions to the problems related to SC. M-PAES algorithm has shown excellent results at the expense of having a high value of runtime. MOPSO has not converged to best solutions but optimizes faster than M-PAES. The proposed algorithm has demonstrated to be better in speed and convergence in comparison to MOPSO and M-PAES.

It is important to mention that, in this paper, three different case studies have been utilized to evaluate the algorithm. However, this not ensure that the algorithm would be better in other cases. The algorithm uses a parameter xz to change the size of th sub-space. Hence, this parameter must change according to the problem. In the future, this algorithm developed could be enhanced with new techniques of optimization.

Acknowledgments. This study is part of the research project “Modelo de optimización de costos en la Cadena de suministro en Empresas de ensamblaje,” supported by the Research Department of the University of Cuenca (DIUC). The authors gratefully acknowledge the contributions and feedback provided by the IMAGINE Project team.

References

1. Yang, X.-S.: Computational Optimization, Methods and Algorithms (2011)
2. Xiong, F., Gong, P., Jin, P., Fan, J.F.: Supply chain scheduling optimization based on genetic particle swarm optimization algorithm. *Clust. Comput.* **22**(6), 14767–14775 (2018). <https://doi.org/10.1007/s10586-018-2400-z>
3. Lummus, R.R., Vokurka, R.J.: Defining supply chain management: a historical perspective and practical guidelines. *Ind. Manag. Data Syst.* **99**, 11–17 (1999). <https://doi.org/10.1108/02635579910243851>
4. Garcia, D.J., You, F.: Supply chain design and optimization: challenges and opportunities. *Comput. Chem. Eng.* **81**, 153–170 (2015). <https://doi.org/10.1016/j.compchemeng.2015.03.015>
5. Cevallos, C., Siguenza-Guzman, L., Peña, M.: A hybrid algorithm for supply chain optimization of assembly companies. In: 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Guayaquil, Ecuador, pp. 1–6 (2019)
6. Slawomir, K., Xin-She, Y.: Computational Optimization, Methods and Algorithms. Springer, Heidelberg (2011). <https://doi.org/10.1007/978-3-642-20859-1>
7. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Comput. Surv. (CSUR)*. **35**, 268–308 (2003)
8. Gendreau, M., Potvin, J.-Y. (eds.): Handbook of Metaheuristics. ISORMS, vol. 272. Springer, Cham (2019). <https://doi.org/10.1007/978-3-319-91086-4>
9. Gonzalez, T.F.: Handbook of Approximation Algorithms and Metaheuristics: Contemporary and Emerging Applications. CRC Press Taylor & Francis Group, New York (2018)
10. Cevallos, C., Siguenza-Guzman, L., Peña, M.: A hybrid algorithm for supply chain optimization of assembly companies. Presented at the IEEE LA-CCI, Ecuador (2019)
11. Knowles, J., Corne, D.: M-PAES: a memetic algorithm for multiobjective optimization. In: Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, 1 (2000). <https://doi.org/10.1109/CEC.2000.870313>
12. Coello Coello, C.A., Lechuga, M.S.: MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002 (Cat. No. 02TH8600), vol. 2, pp. 1051–1056 (2002)
13. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**, 182–197 (2002). <https://doi.org/10.1109/4235.996017>
14. Coello Coello Coello, C.A., Toscano Pulido, G.: A micro-genetic algorithm for multiobjective optimization. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 126–140. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44719-9_9

15. Berrezueta, N.: Optimización de la cadena de suministro mediante uso de un algoritmo genético basado en clusterización (Working Paper). Presented at the (2020)
16. Jamshidi, R., Fatemi Ghomi, S.M.T., Karimi, B.: Multi-objective green supply chain optimization with a new hybrid memetic algorithm using the Taguchi method. *Scientia Iranica* **19**, 1876–1886 (2012). <https://doi.org/10.1016/j.scient.2012.07.002>
17. Kuo, R.J., Han, Y.S.: A hybrid of genetic algorithm and particle swarm optimization for solving bi-level linear programming problem - a case study on supply chain model. *Appl. Math. Model.* **35**, 3905–3917 (2011). <https://doi.org/10.1016/j.apm.2011.02.008>