

A BI-OBJECTIVE GREY WOLF OPTIMIZER WITH A DYNAMIC NUMBER OF LEADERS APPLIED TO SPARE PARTS STOCK LEVEL OPTIMIZATION PROBLEMS

Leonardo Ramos Rodrigues

Instituto de Aeronáutica e Espaço - IAE

Praça Marechal Eduardo Gomes, 50, São José dos Campos, SP, Brasil, 12228-904

leonardolrr2@fab.mil.br

João Paulo Pordeus Gomes

Universidade Federal do Ceará

Rua Campus do Pici, s/n, Fortaleza, CE, Brasil - 60440-554

jpaulo@dc.ufc.br

ABSTRACT

The Grey Wolf Optimizer (GWO) is a metaheuristic optimization algorithm in the field of swarm intelligence that has been successfully used for solving a wide range of optimization problems. The GWO algorithm is based on the leadership hierarchy observed in grey wolf packs. In the original GWO, a fixed number of wolves with the best fitness values for the optimization problem under consideration is used to guide the search process. In a bi-objective optimization problem, we are interested in finding a Pareto front approximation with high cardinality, i.e. a high number of non-dominated solutions. Also, the points should be well distributed along the Pareto front. In this paper, we propose a bi-objective variant of GWO, denoted by BGWODNL. The proposed model uses a data-driven approach to define a dynamic number of leaders to guide the search process in each iteration of the algorithm. Numerical experiments are conducted considering a spare parts list optimization problem. For comparison purposes, experiments with a standard multi-objective variant of GWO are also carried out. Other benchmark multi-objective algorithms such as the Pareto Ant Colony Optimization (PACO) and the Non-dominated Sorting Genetic Algorithm (NSGA-II) are also considered. The results show that the proposed algorithm outperforms the benchmark algorithms in terms of cardinality, simulation time, and distribution of points along the Pareto front.

KEYWORDS. Grey Wolf Optimizer. Multi-objective Optimization. Combinatorial Optimization. Inventory Management. Stock Level Optimization.

Paper topic: MH - Metaheuristics.

1. Introduction

In industrial applications, repairable items are often among the most expensive and critical components [Kim et al., 1996]. When a failure event occurs, failed components must be rapidly replaced to reduce downtime. For this reason, a spare parts inventory is kept. Hence, designing the spare parts stock level for repairable items is an optimization problem that has an impact on modern industrial operations.

In its classical formulation, the spare parts inventory problem is a bi-objective optimization problem that consists of finding a spare parts list that minimizes the inventory cost and maximizes the expected system availability [Sherbrooke, 2004]. Extensions of the classical spare parts inventory problem were addressed in other works such as Rodrigues and Gomes [2018] and Xie et al. [2014]. In both papers, the authors considered the existence of redundant components, i.e. the system may still work even if some of its components fail. The use of redundancies in system architectures is a common practice in safety-critical industry sectors such as space, aeronautical and nuclear power [Rodrigues and Gomes, 2018]. As a consequence of considering redundancies in system architecture, the resulting optimization problem becomes non-convex and thus metaheuristic algorithms arise as valid alternatives to address such a problem. For that, Pareto Ant Colony Optimization (PACO) [Doerner et al., 2004], and the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al., 2002] are among the most popular options.

Recently, Mirjalili et al. [2014] proposed a novel metaheuristic optimization algorithm inspired by the social behavior of grey wolves. The so-called Grey Wolf Optimizer (GWO) uses several mechanisms based on the social hierarchy and the hunting mechanism of grey wolf packs. GWO has achieved competitive results and can be seen as an alternative to popular metaheuristic methods such as Genetic Algorithms (GA) Holland [1992], and Particle Swarm Optimization (PSO) [Kennedy and Eberhart, 1995].

Several variants of GWO have been proposed to improve its performance. In Wang and Li [2019], the authors used differential evolution and applied the “survival of the fittest” principle to eliminate wolves with poor solutions and improve the balance between the exploration and exploitation capabilities of the algorithm. In Gao and Zhao [2019], the authors proposed a variant in which the weights assigned to leaders vary during the execution of the algorithm. Hybrid versions of GWO have also been proposed, combining GWO with the Genetic Algorithm [Tawhid and Ali, 2017], and Particle Swarm Optimization [Teng et al., 2019], among others.

The literature also contains variants of GWO to handle multi-objective optimization problems. A Pareto-based multi-objective version of GWO was proposed by Mirjalili et al. [2016] and achieved remarkable results. A different approach based on decomposition was used in Zapotecas-Martínez et al. [2019] to propose another multi-objective variant of GWO.

In this paper, motivated by the success of the multi-objective versions of GWO, we propose a new variant of GWO named Bi-objective Grey Wolf Optimizer with a Dynamic Number of Leaders (BGWODNL). In BGWODNL, we introduce two main modifications to improve the exploration-exploitation compromise along the iterations. First, we consider that the number of leaders can vary along the iterations. Additionally, we do not impose limits on the number of solutions in the Pareto front. The proposed method is used to solve the spare parts inventory optimization problem with redundant components. The results show that such modifications can produce a large number of non-dominated solutions with reduced simulation time.

The rest of this paper is organized as follows. Section 2 presents the spare parts inventory problem. Section 3 describes the original GWO algorithm. The proposed BGWODNL algorithm is introduced in section 4. The results observed in numerical experiments are presented and discussed

in section 5. Final remarks and opportunities for future research are given in section 6.

2. Problem Definition

In this paper, we consider a spare parts inventory system for multiple repairable items. The inventory system supports the operation of m identical machines. Each machine is composed of n distinct components. In order to support the operation of the machines, the inventory system contains a set of spare parts, denoted by $S = \{s_1, \dots, s_n\}$, where each element s_i is a non-negative integer number that represents the number of spare parts for component c_i . Figure 1 illustrates a spare parts inventory system.

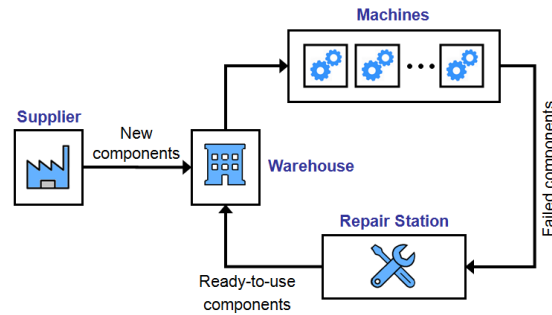


Figure 1: A spare parts inventory system for multiple repairable items.

A supplier sends a set of spare parts S to the system, which is held at the warehouse. When a component fails, it is replaced with a ready-to-use spare part from the warehouse. A system failure occurs whenever a non-redundant component fails. Failed components are sent to a repair station to be repaired. A perfect repair station is considered, i.e. the repair station has an infinite repair capacity, all failed components can be repaired, and repaired components are brought to the “as good as new” condition. At the end of the repair process, the repaired component is sent to the warehouse to be used when a new failure event occurs. We assume that components do not degrade in the warehouse. If a non-redundant component fails and there are no spares at the warehouse, the failed machine remains out of service until a spare part becomes available.

The architecture of each system is represented by a fault tree that describes the interactions among its components. The Time to Failure (TTF) of each component c_i is a random variable that follows an exponential distribution with rate λ_i . Also, the Time to Repair (TTR) each component c_i is a random variable that follows an exponential distribution with rate μ_i .

Each candidate spare part list S has an associated cost, which is the sum of the unit price of each component in S . Also, each candidate spare part list S has an associated expected availability, which is the average percentage of functioning machines. The problem consists in finding a set of non-dominated spare parts lists to build a Pareto front for the bi-objective optimization problem considering spare parts cost and expected system availability as objective functions.

3. Grey Wolf Optimizer

Grey Wolf Optimizer (GWO) is a population-based optimization algorithm inspired by the hierarchical nature and the hunting behavior of grey wolves [Mirjalili et al., 2014]. In the original GWO algorithm the wolves are divided into four hierarchical levels, described as follows:

- **Alpha wolves** (α) are at the top level of the hierarchy. This group takes decisions such as hunting time, prey selection, and sleeping place. Such decisions are followed by all the pack.

- **Beta wolves** (β) are in the second level in the pack hierarchy. They reinforce the commands of alpha wolves all over the pack and advise them in decision making.
- **Delta wolves** (δ) are in the third level of the hierarchy. They are responsible for different tasks such as watching the territory, and taking care of weak wolves.
- **Omega wolves** (ω) are at the lowest hierarchy level. They must submit to all the other dominant wolves.

Also, the algorithm considers the group hunting behavior in its formulation. The grey wolf hunting procedure can be divided into the following three main phases: (1) tracking, chasing, and approaching the prey; (2) pursuing, encircling, and harassing the prey; and (3) attacking the prey.

After defining the hunting behavior and the social hierarchy, the GWO algorithm starts by randomly generating a set of candidate solutions. All solutions are evaluated and the best, the second-best, and the third-best solutions are classified as α , β , and δ . Then, all other solutions (named ω) are updated as a function of their distances to the α , β , and δ wolves.

3.1. Mathematical Model

Let M be the number of wolves in the pack and $X_i = [x_1, x_2, \dots, x_k]$ be the vector that defines the position of the i -th wolf, where k is the dimension of problem under consideration. At each iteration, the position of each wolf is updated according to Equations (1) and (2).

$$D = |C \cdot X_p(t) - X(t)| \quad (1)$$

$$X(t+1) = X_p(t) - A \cdot D \quad (2)$$

where $X(t)$ is the position of the wolf at iteration t , and $X_p(t)$ is the position of the prey. A and C are vectors computed according to Equations (3) and (4), respectively.

$$A = 2a \cdot r_1 - a \quad (3)$$

$$C = 2 \cdot r_2 \quad (4)$$

where $r_1 \in [0,1]$ and $r_2 \in [0,1]$ are random vectors and a is a control parameter that linearly decreases from 2 to 0 during the algorithm execution.

According to the mathematical formulation, a wolf updates its position based on vectors A and C . Approaching a prey is modeled by decreasing the value of a along the iterations. The coefficient vector C can be considered as the effect of obstacles in the hunting path, which prevents the wolves to quickly and conveniently approach the prey [Sahoo and Chandra, 2017].

Even though Equations (1) and (2) indicate that the positions of the wolves are updated based on the position of the prey, such position (i.e. the optimal solution of the problem under consideration) is unknown. As mentioned earlier, the hunting procedure is guided by the α , β , and δ wolves. To model this characteristic, GWO assumes that the α , β , and δ wolves (which are the candidate solutions with the best fitness values) have better estimates of the prey's position. Thus, the positions of the pack are updated according to the position of these wolves, as shown in Equations (5) to (11). A pseudo-code of the GWO algorithm is presented in Algorithm 1.

$$D_{\alpha} = |C_1 \cdot X_{\alpha} - X(t)| \quad (5)$$

$$D_{\beta} = |C_2 \cdot X_{\beta} - X(t)| \quad (6)$$

$$D_{\delta} = |C_3 \cdot X_{\delta} - X(t)| \quad (7)$$

$$X_1 = X_{\alpha} - A_1 \cdot D_{\alpha} \quad (8)$$

$$X_2 = X_{\beta} - A_2 \cdot D_{\beta} \quad (9)$$

$$X_3 = X_{\delta} - A_3 \cdot D_{\delta} \quad (10)$$

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \quad (11)$$

Algorithm 1 GWO algorithm.

- 1: Initialize the population X_i , for $i = 1, \dots, M$
 - 2: Initialize $a = 2$
 - 3: Initialize A and C
 - 4: Compute the fitness value of each wolf
 - 5: X_{α} = the wolf with the best fitness value
 - 6: X_{β} = the wolf with the second best fitness value
 - 7: X_{δ} = the wolf with the third best fitness value
 - 8: **While** $t < \text{max. number of iterations}$ **do**
 - 9: **For** each wolf i **do**
 - 10: Update its position according to Equation (11)
 - 11: **End For**
 - 12: Update a , A , and C
 - 13: Compute the new fitness value of each wolf
 - 14: Update X_{α} , X_{β} , and X_{δ}
 - 15: increment the iteration number ($t = t + 1$)
 - 16: **End While**
 - 17: Return X_{α}
-

4. Proposed Bi-Objective Grey Wolf Optimizer

In this section, we present the proposed Bi-objective Grey Wolf Optimizer with a Dynamic Number of Leaders (BGWODNV). In order to adapt GWO to solve multi-objective problems, we use an archive that stores the current set of all non-dominated solutions during the execution of the algorithm. This archive is the output of the algorithm. Solutions are added to and eliminated from the archive according to the flowchart presented in Figure 2.

We propose a new leadership selection mechanism to select the wolves that will guide the search process in each iteration of the algorithm. In the original GWO, as well as in other GWO variants, the number of leaders is always three (the α , β , and δ wolves). In our proposed model, however, we relax this assumption and adopt a data-driven mechanism to define the number of leaders in each iteration of the algorithm. In order to define the leaders, the solutions in the archive are sorted and the distances between consecutive points are computed. Then, for each objective function, the two consecutive points with the highest distance are selected as leaders. Depending

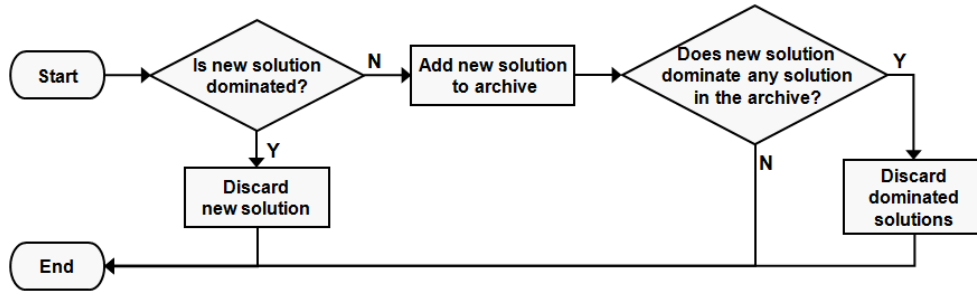


Figure 2: Flowchart for archive update.

on the distribution of points in the archive, the number of leaders in a bi-objective optimization problem may vary from 2 to 4. Figure 3 presents an illustrative example that shows how a different number of leaders may be selected.

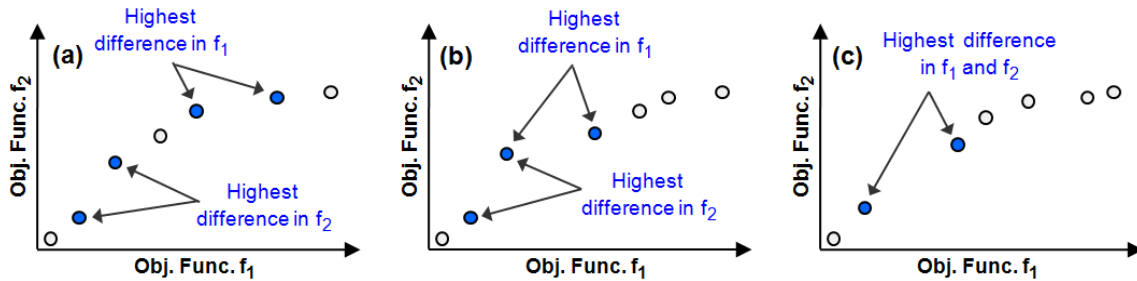


Figure 3: An illustrative example of leaders selection in BGWODNL. (a) Distinct pairs with the highest difference in f_1 and f_2 resulting in four leaders. (b) One common solution in the pairs with the highest difference in f_1 and f_2 resulting in three leaders. (c) The same pair of solutions with the highest difference in both f_1 and f_2 resulting in two leaders.

Once the leaders for iteration t are selected, the position of the other wolves are not updated according to Equation (11). Instead, they are updated according to Equation (12), which is the general form of Equation (11) for a variable number of leaders.

$$X(t+1) = \frac{\sum_{v=1}^{n(t)} X_v}{n(t)} \quad (12)$$

where $n(t)$ is the number of leaders for iteration t , and X_v is computed according to:

$$X_v = X_j(t) - A_j \cdot D_j \quad (13)$$

where $X_j(t)$ is the position of the j -leader in iteration t , A_j is a random vector, and D_j is computed according to:

$$D_j = |C_j \cdot X_j(t) - X| \quad (14)$$

where C_j is a random vector, and X is the current position of the wolf.

4.1. Population Initialization

The opposition-based learning principle is used to initialize the population of the proposed BGWODNL. The opposition-based learning concept was introduced in Tizhoosh [2005] to improve the performance of metaheuristic algorithms. The main idea behind opposition-based learning is that a pair composed of a candidate solution and its opposite has a higher probability to be closer to the optimal solution than a pair composed of two random solutions, considering that the original candidate solution not close to the optimal solution [Tizhoosh, 2005], [Mahdavi et al., 2018].

Consider a d -dimensional problem. Let $\mathbf{z} = (z_1, z_2, \dots, z_d)$ be a candidate solution, where each element $z_i \in [L_i, U_i]$. Also, let $\bar{\mathbf{z}} = (\bar{z}_1, \bar{z}_2, \dots, \bar{z}_d)$ be the opposite solution. Each element $\bar{z}_i$ is computed as follows [Aladeemy et al., 2020]:

$$\bar{z}_i = L_i + U_i - z_i \quad (15)$$

In order to generate the initial population for BGWODNL, the solution of half of the wolves are randomly generated, while the solutions of the remaining wolves are the opposite solutions computed according to Equation (15).

5. Numerical Experiments

In this section, we present the results observed in numerical experiments carried out to evaluate the performance of the proposed algorithm. The performance of BGWODNL is compared with the performance of the multi-objective version of GWO proposed in Mirjalili et al. [2016], denoted by MGWO. We also compare the performance of BGWODNL with two other algorithms: the Pareto Ant Colony Optimization (PACO) [Doerner et al., 2004], and the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al., 2002]. The algorithms are compared in terms of the number of non-dominated solutions, simulation time, and the Δ index [Audet et al., 2018].

The Δ index is a metric that measures the distribution of the non-dominated points in a Pareto front. The Δ index is computed according to Equation (16).

$$\Delta(S) = \sum_{i=1}^{|S|-1} \frac{|d_i - \bar{d}|}{|S| - 1} \quad (16)$$

where S is the set of points belonging to a Pareto front, d_i is the Euclidean distance between two consecutive points of the Pareto front, and $\bar{d}$ is the mean value of d_i .

In the ideal case, all points in S will be uniformly distributed along the Pareto front. In this situation, all distances d_i will be equal to $\bar{d}$, and the Δ index will be zero. In any other situation, the indicator will assume a positive value. Thus, lower indicator values are associated with Pareto fronts with more uniformly distributed points.

5.1. Simulation Data

As mentioned earlier, a bi-objective spare parts stock level optimization problem is considered in the experiments. We conducted experiments with three different multiple-component systems. Figure 4 shows a fault tree representation of the architecture of each system. Table 1 shows the unit price, the failure rate λ_i , and the repair rate μ_i of each component type c_i .

A set of 30 machines was used in all the simulations. For each algorithm, a Monte Carlo approach with 100 repetitions was adopted. Based on experimental tests, a population of 50 wolves

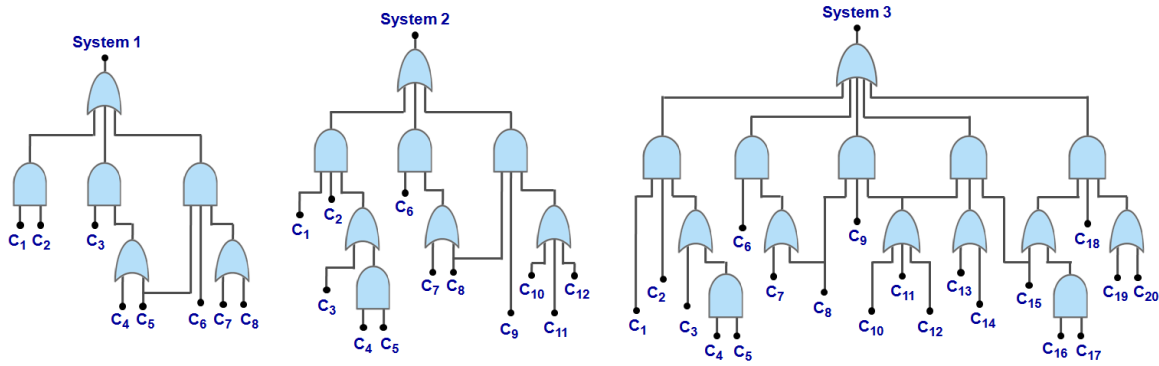


Figure 4: Fault tree representation of system architectures.

Table 1: Component Simulation Data

Component	Price	λ	μ	Component	Price	λ	μ	Component	Price	λ	μ
c_1	270	346	20	c_8	30	186	31	c_{15}	40	166	32
c_2	160	245	38	c_9	150	544	31	c_{16}	120	94	18
c_3	140	149	25	c_{10}	50	206	34	c_{17}	80	90	15
c_4	120	128	16	c_{11}	140	86	17	c_{18}	70	114	25
c_5	90	176	28	c_{12}	30	348	30	c_{19}	40	265	28
c_6	70	280	23	c_{13}	230	169	20	c_{20}	30	225	16
c_7	40	174	31	c_{14}	50	490	30				

and a maximum number of 200 iterations were used for both MGWO and BGWODNL. For the PACO algorithm, a colony with 50 ants, a pheromone evaporation rate of 1%, and a maximum number of 200 iterations were used. For NSGA-II, a population size of 50, a crossover probability of 80%, a mutation probability of 10%, a mutation rate of 5%, and a maximum number of 200 iterations were used. The experiments were conducted using an Intel® Core™ i7, 3.4 GHz processor, and 8GB RAM. The algorithms were coded in Matlab®.

5.2. Simulation Results

The performances obtained with the different algorithms are compared in terms of the number of non-dominated solutions, simulation time, and the Δ index. Figure 5 shows the number of non-dominated solutions obtained with each algorithm for each system.

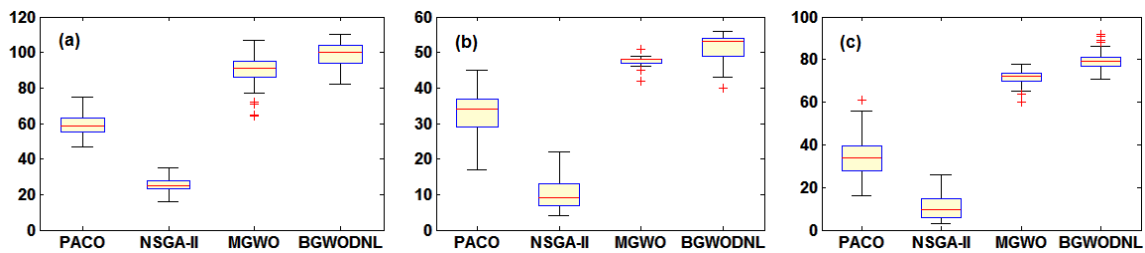


Figure 5: Number of non-dominated solutions. (a) System 1. (b) System 2. (c) System 3.

It can be noticed that, for all the systems, a similar relative result is observed. The two versions of GWO (MGWO and the proposed BGWODNL) found the highest number of non-dominated

solutions, followed by PACO and NSGA-II. It is worth pointing out that the highest number of non-dominated solutions were obtained for system 1, which is the system with the lowest number of components. This result shows that the complexity level to find the Pareto front for a system depends on both the number of components and the interactions among components defined by the system architecture.

The simulation times required to run the experiments are presented in Figure 6. Again, the two GWO versions presented the best results. Also, the two GWO versions presented a lower variability in the simulation time when compared with PACO and NSGA-II.

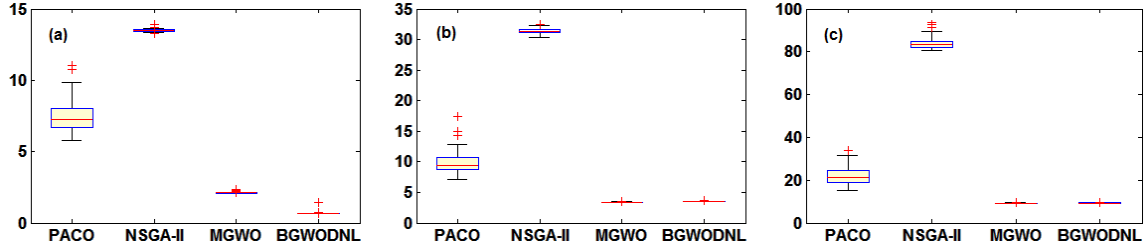


Figure 6: Simulation time required by each algorithm. (a) System 1. (b) System 2. (c) System 3.

Figure 7 shows the Δ index obtained with each algorithm for each system. As mentioned earlier, the Δ index measures the distribution of points along the Pareto front. Lower Δ index values are associated with more uniformly distributed fronts. It can be noticed from Figure 7 that the proposed BGWODNL presented the best performance. In the next section, a statistical significance test is conducted to evaluate the differences in the performances obtained with the different algorithms.

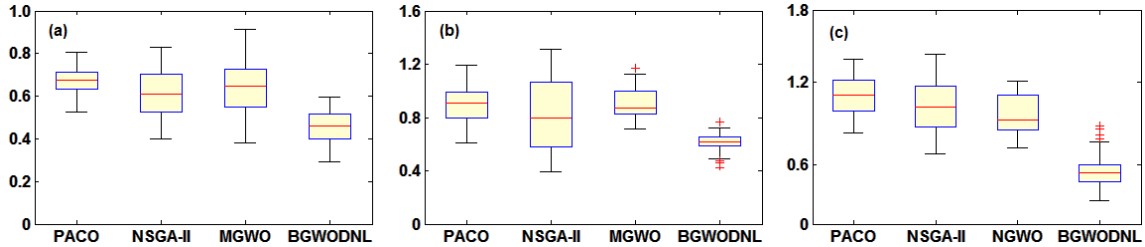


Figure 7: Δ index computed for each metaheuristic algorithm. (a) System 1. (b) System 2. (c) System 3.

5.3. Statistical Significance Test

This section presents the results of a statistical test conducted to check whether the performance obtained with the proposed BGWODNL is statistically different from the performance obtained with the other algorithms.

Let X and Y be two normally distributed random variables. The mean and the variance of X are denoted by μ_X and σ_X^2 , respectively. Similarly, the mean and the variance of Y are denoted by μ_Y and σ_Y^2 , respectively. Also, assume that $\sigma_X^2 \geq \sigma_Y^2$. In order to test whether X and Y have different means, the first step is to conduct an F-Test to test whether X and Y can be assumed to have the same variance. The null hypothesis H_0 for the F-Test is that $\sigma_X^2 = \sigma_Y^2$. The statistics F_{calc} is computed according to Equation (17). The statistics F_{calc} is then compared with a critical value F_{crit} , which is computed according to Equation (18).

$$F_{calc} = \frac{\sigma_X^2}{\sigma_Y^2} \quad (17)$$

$$F(F_{crit}; d_X, d_Y) = 1 - \alpha \quad (18)$$

where $F(\cdot)$ is the cumulative Fisher-Snedecor distribution, α is the confidence level for the test, d_X is the number of degrees of freedom associated with X , and d_Y is the number of degrees of freedom associated with Y . The null hypothesis H_0 can be rejected if $F_{calc} > F_{crit}$.

A T-Test is then conducted to test whether X and Y can be assumed to have different means. The null hypothesis H_0 for the T-Test is that $\mu_X = \mu_Y$. The result of the F-Test is used to define how to compute the statistics for the T-Test. If the null hypothesis of the F-Test is rejected (i.e. X and Y have different variances), then the statistics T_{calc} for the T-Test is computed according to Equation (19). Otherwise, T_{calc} is computed according to Equation (20).

$$T_{calc} = \frac{\mu_X - \mu_Y}{\sqrt{\frac{\sigma_X^2}{n_X} + \frac{\sigma_Y^2}{n_Y}}} \quad (19)$$

$$T_{calc} = \frac{\mu_X - \mu_Y}{\sqrt{\sigma^2 \left(\frac{1}{n_X} + \frac{1}{n_Y} \right)}} \quad (20)$$

where n_X and n_Y are the number of samples used to compute the means and variances of X and Y , respectively, and σ^2 is computed according to Equation (21).

$$\sigma^2 = \frac{(n_X - 1)\sigma_X^2 + (n_Y - 1)\sigma_Y^2}{n_X + n_Y - 2} \quad (21)$$

Again, the statistics T_{calc} must be compared with a critical value T_{crit} , which is computed according to Equation (22).

$$T(T_{crit}; n) = 1 - \alpha/2 \quad (22)$$

where $T(\cdot)$ is the cumulative Student's t distribution, and n is the number of degrees of freedom computed according to Equation (23).

$$n = \frac{\left[\left(\frac{\sigma_X^2}{n_X} + \frac{\sigma_Y^2}{n_Y} \right) \right]^2}{\frac{\left(\frac{\sigma_X^2}{n_X} \right)^2}{n_X - 1} + \frac{\left(\frac{\sigma_Y^2}{n_Y} \right)^2}{n_Y - 1}} \quad (23)$$

The null hypothesis H_0 can be rejected if $T_{calc} > T_{crit}$. A pairwise comparison was conducted between the proposed BGWODNL and each other algorithm considered in this paper for each of the three systems and each performance indicator. The null hypothesis of the T-Test could be rejected in all the tests. It means that the proposed BGWODNL algorithm was able to find a higher number and more distributed non-dominated solutions in a lower computational time for all the systems when compared with the other algorithms considered in the experiments.

6. Conclusions

This paper presented a modified version of the Grey Wolf Optimizer (GWO) algorithm for bi-objective optimization problems, denoted by BGWODNL. An archive is used to store the non-dominated solutions that form the Pareto front. The proposed algorithm uses a data-driven mechanism to define a dynamic number of leaders to guide the search process during each iteration of the algorithm.

The leaders are selected to guide the search process to regions of the Pareto front with a lower concentration of non-dominated solutions. The dynamic number of leaders avoids the selection of wolves that do not contribute to guiding the search process to regions of interest of the Pareto front only to comply with the predefined number of leaders assumption.

Numerical experiments considering the application of the proposed BGWODNL to solve a spare parts list optimization problem were presented. Three different systems were considered. The performance of BGWODNL was compared with the performance of a standard multi-objective variant of GWO and two benchmark multi-objective algorithms: the Pareto Ant Colony Optimization (PACO) and the Non-dominated Sorting Genetic Algorithm (NSGA-II). The performances obtained with the algorithms were compared in terms of cardinality (i.e. the number of points in the Pareto front), simulation time, and Δ index. The proposed BGWODNL presented a very competitive performance.

Future research may extend the scope of this paper by adapting the proposed BGWODNL to higher dimensions optimization problems. Another opportunity to extend the scope of this paper is to investigate the performance of BGWODNL combined with other GWO variants proposed in the literature.

Acknowledgment

The authors would like to thank the Brazilian National Council for Scientific and Technological Development - CNPq (grants 423023/2018-7 and 302289/2019-4).

References

- Aladeemy, M., Adwan, L., Booth, A., Khasawneh, M. T., and Poranki, S. (2020). New feature selection methods based on opposition-based learning and self-adaptive cohort intelligence for predicting patient no-shows. *Applied Soft Computing*, 86:105866.
- Audet, C., Digabel, S. L., Cartier, D., Bignon, J., and Salomon, L. (2018). Performance indicators in multiobjective optimization. In *Technical report, Les Cahiers du GERAD G-2018-90*, p. 1–39.
- Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.
- Doerner, K., Gutjahr, W. J., Hartl, R. F., Strauss, C., and Stummer, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, 131(1):79–99.
- Gao, Z.-M. and Zhao, J. (2019). An improved grey wolf optimization algorithm with variable weights. *Computational Intelligence and Neuroscience*, 2019:1–13.
- Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA. ISBN 0262082136.

- Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, p. 1942–1948.
- Kim, J.-S., Shin, K.-C., and Yu, H.-K. (1996). Optimal algorithm to determine the spare inventory level for a repairable-item inventory system. *Computers & Operations Research*, 23(3):289–297.
- Mahdavi, S., Rahnamayan, S., and Deb, K. (2018). Opposition based learning: A literature review. *Swarm and Evolutionary Computation*, 39:1–23.
- Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61.
- Mirjalili, S., Saremi, S., Mirjalili, S. M., and dos S. Coelho, L. (2016). Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, 47:106–119.
- Rodrigues, L. R. and Gomes, J. P. P. (2018). Spare parts list recommendations for multiple-component redundant systems using a modified pareto ant colony optimization approach. *IEEE Transactions on Industrial Informatics*, 14(3):1107–1114.
- Sahoo, A. and Chandra, S. (2017). Multi-objective grey wolf optimizer for improved cervix lesion classification. *Applied Soft Computing*, 52:64–80.
- Sherbrooke, C. C. (2004). *Optimal Inventory Modeling of Systems: Multi-Echelon Techniques (INTL SERIES IN OPERATIONS RESEARCH & MANAGEMENT SCIENCE)*. Kluwer Academic Publishers, USA. ISBN 1402078498.
- Tawhid, M. A. and Ali, A. F. (2017). A hybrid grey wolf optimizer and genetic algorithm for minimizing potential energy function. *Memetic Computing*, 9:347–359.
- Teng, Z., Lv, J., and Guo, L. (2019). An improved hybrid grey wolf optimization algorithm. *Soft Computing*, 23(15):6617–6631.
- Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 1, p. 695–701, Vienna.
- Wang, J.-S. and Li, S.-X. (2019). An improved grey wolf optimizer based on differential evolution and elimination mechanism. *Scientific Reports*, 9(1):7181.
- Xie, W., Liao, H., and Jin, T. (2014). Maximizing system availability through joint decision on component redundancy and spares inventory. *European Journal of Operational Research*, 237(1):164 – 176. ISSN 0377-2217.
- Zapotecas-Martínez, S., García-Nájera, A., and López-Jaimes, A. (2019). Multi-objective grey wolf optimizer based on decomposition. *Expert Systems with Applications*, 120:357–371.