A Firefighting Resource Dispatch Problem Optimization using Metaheuristics*

Marina A. Matos[1][0000−0001−5387−8771], Rui Gonçalves[1][0009−0002−6561−0021], Ana Maria A. C. Rocha[1][0000−0001−8679−2886], Lino A. Costa[1][0000−0003−4772−4404], and Filipe Alvelos[1][0000−0002−1851−4339]

ALGORITMI Research Centre / LASI, University of Minho, Campus de Gualtar, 4710-057 Braga, Portugal
mmatos@algoritmi.uminho.pt,rgoncalves0320@gmail.com, {arocha,lac,falvelos}@dps.uminho.pt

Abstract. Forest fires are a growing concern in our planet, as they have been increasing in frequency and severity over the last few decades. It is therefore essential to manage firefighting in order to prevent and reduce the damage caused to life, the economy and ecosystems. This work deals with the forest firefighting resource dispatch problem using metaheuristics. It is essential to know when and which suppression resources should be sent to extinguish the ignitions. Note that the faster the resources act, the smaller the burned area. An adapted genetic algorithm and differential evolution algorithm are used to solve a firefighting resource dispatch problem aiming to find the optimal allocation of resources to ignitions, and at what instant of time each resource is assigned to each ignition, in order to minimize the total burned area. The two metaheuristics are applied to a set of 60 randomly generated instances, comprising 5, 7 and 10 resources to extinguish 20 or 30 ignitions. A statistical analysis of the results is used to compare the efficiency of the metaheuristics, where the GA shows more competitive results.

Keywords: Fire Suppression · Dispatch Problem · Genetic Algorithm · Differential Evolution Algorithm

1 Introduction

The increase in forest fires around the world has caused several negative aspects, such as deaths, pollution and damage to infrastructure, making it a very worrying factor for society and planet Earth. Therefore, it is crucial to implement forest fire management strategies, which involve preventing fires, anticipating their occurrence, identifying fire risk locations and applying necessary and effective measures to reduce the damage caused by forest fires. In Europe, in 2021, more

* This work has been supported by FCT Fundação para a Ciência e Tecnologia within the R&D Units Project Scope UIDB/00319/2020 and PCIF/GRF/0141/2019: "O3F - An Optimization Framework to Reduce Forest Fire" and the PhD grant reference UI/BD/150936/2021.

than 1500 km^2 of land burned, of which 1000 km^2 belong to protected areas in the European Union [15]. In 2017, there was a record number of forest fires in Portugal, caused by high temperatures and dry thunderstorms.

Firefighting is a very important area that has been studied in the literature. The number, skills and level of preparedness of firefighting teams are very important factors in fighting fires [1,9]. The better and faster the firefighting team performs, the less damage it causes. In the field of engineering, various optimization techniques have been used to tackle challenges related to fire suppression. The main objectives involve identifying the optimal trajectory and positioning for firefighting vehicles, determining the most efficient route and deciding on the amount and type of resources to be used [5]. A mathematical optimization model was developed in Zeferino et al. [17], aiming to find the best locations for different aircraft on the ground in order to maximize coverage of risk areas. This model was applied to a case study in Portugal. Mendes and Alvelos [12] focused on solving a fire suppression problem represented by a grid network and a fire ignition location. Their objective was to determine the optimal allocation of firefighting resources to minimize both the burned area and the use of resources as a secondary objective. They used a novel iterative local search metaheuristic and a mixed integer programming model, demonstrating the effectiveness of the metaheuristic in identifying optimal solutions.

The firefighting resource dispatch problem optimization involves selecting and allocating resources to a fire, as well as determining the appropriate time to dispatch them [2]. Genetic algorithms have been widely used to solve this type of problem, with the main objective of minimizing the burned area [7]. Matos et al. [11] studied a firefighting resource dispatch problem, involving the allocation of 7 resources (tank vehicles) to extinguish 20 ignitions, aiming to minimize the total burned area. However, in this modeling of the problem, it was not possible to refill the vehicles' tanks whenever they ran out of water. To solve the optimization task, they used a genetic algorithm adapted to this specific problem. In addition, a statistical analysis was carried out comparing different operators of the genetic algorithm, including crossover, mutation and selection, to determine which combination of operators produced the most favorable results. They found that the operators best suited to this type of problem were exponential crossover, polynomial mutation and tournament selection. The same problem was discussed in [10] where an adapted version of the differential evolution algorithm was used. In addition, a statistical analysis was carried out to evaluate its performance with different selection and crossover operators. The results suggested that the current-to-best selection and exponential crossover operators were the most effective for solving this problem.

In this work, a genetic algorithm (GA) and a differential evolution algorithm (DE) adapted and implemented in the *Python* language are applied to solve the firefighting resource dispatch problem. It should be noted that in this implementation, it is possible to refill the vehicles' water tanks (resources) whenever they run out of water, requiring a different representation of solutions by permutation, adapted for the GA and DE algorithms, than that used in the previous works

[10,11]. The main goal is to assign different resources of combat to various ignitions in order to minimize the total burned area. To compare the performance of the two algorithms, a set of 60 instances is randomly generated, comprising 5, 7 and 10 resources to extinguish 20 or 30 ignitions. In addition, a statistical analysis is performed to compare the results obtained by the two metaheuristics.

This paper is organized as follows. The problem is described in Sect. 2. Section 3 explains the permutation representation, and the steps of genetic algorithm and differential evolution algorithm. In Sect. 4 the details and results of the computational experiments, using GA and DE, are presented. Finally, the discussion and conclusions of this study and future work are stated in Sect. 5.

2 Problem description

In this work, a firefighting resource dispatch problem is optimized, aiming the minimization of the total burned area.

In this problem, there may be a number of ignitions, which can be understood as several fronts of the same forest fire. In addition, there are a number of firefighting resources, which are based on forest firefighting land vehicles with water tanks, which can have different capacities.

At the beginning, all the firefighting resources (vehicles) are at the base (fire station) and their water tanks are at full capacity. Concerning the ignitions, they are all active (burning) and located in different places. The main idea is to allocate the vehicles to the ignitions, over various instants of time, in order to completely extinguish them.

Taking into consideration the location of the ignitions, travel times are calculated based on the Euclidean distance between each ignition and the base and between ignitions, satisfying the triangular inequality. The values of the burned area by each ignition at each instant of time are known. To do this, a different exponential growth model of the fire spread over time was applied to each ignition, resulting in different types of growth of the burned area. The water required to extinguish a given ignition over time is calculated using the values of the burned area at each instant of time, based on the perimeter of a circle.

It should be noted that each ignition can be extinguished only once (even taking more than one instant of time) and it may require more than one resource to extinguish it completely. On the other hand, each resource can only be allocated to one ignition at each instant of time.

In each instant of time, each resource is available to be assigned to one ignition or is traveling to the location of an ignition or is being refilled. When a resource is allocated to a particular ignition, it is extinguished instantly, if the resource has enough water in its tank. Whenever the water capacity of a resource is not sufficient to extinguish a given ignition, the ignition remains active an the resource refills in this ignition taking one instant of time.

3 Metaheuristics

In this paper, a genetic algorithm and a differential evolution algorithm are used. These metaheuristics are evolutionary and stochastic algorithms that are applied to optimization problems. Although they share some similarities, they have distinct features in their mechanisms and approaches.

3.1 Solution Representation by Permutation

The permutation representation is relevant for combinatorial problems. In this work, this type of representation allows to search for potential solutions to the problem of dispatching firefighting resources. Both metaheuristics work with a population of individuals (chromosomes for GA and vectors for DE) that is a permutation of the decision variables. This permutation is a unique sequence of integer values, ensuring that each firefighting resource can deal with several ignitions at different instants of time, i.e. the permutation indicates a specific order of priority for assigning resources to ignitions. The size of individual is determined by multiplying the number of resources by the number of ignitions and a parameter (m). This parameter defines the maximum number of times a resource can return to a given ignition, i.e., when a resource runs out of water, it refills at that ignition, and can go to that ignition up to m times, while the ignition is still active. For example, for a problem with 2 resources to extinguish 3 ignitions and with $m = 3$, an individual of size 18 ($2 \times 3 \times 3$) is required to represent a potential solution of the problem, as shown in Table 1. Thus,

Table 1: Permutation representation (example)

Resource	**R1**									**R2**								
Decision Variables	13	4	10	6	14	16	8	15	9	1	12	17	5	3	18	2	7	11
Ignitions Order	I_2	I_1	I_2	I_1	I_3	I_3	I_1	I_3	I_2	I_1	I_3	I_3	I_2	I_1	I_3	I_1	I_2	I_2

according to this individual, resource 1 goes first to ignition 2, then to ignition 1, then again to ignition 2, and so on, ending at ignition 2. Resource 2 starts its journey at ignition 1 and ends at ignition 2. As mentioned earlier, in this example, each resource can be sent to the same ignition up to 3 times ($m = 3$).

3.2 Genetic Algorithm

Genetic algorithm is a metaheuristic inspired by the principles of biological evolution [6]. It was developed as a machine learning and optimization algorithm and applied to solve optimization problems with the aim of finding the best possible solution in complex search spaces. GA working comprises the following essential components, a population of individuals (chromosomes), a fitness function, a selection mechanism and genetic operators such as crossover and mutation. The population is evolved until a given termination criterion [8].

In GA, potential solutions of the problem are represented as chromosomes made up of genes (a vector of elements). Genes encode various aspects of the solution, and genetic operators, such as crossover and mutation, act on these genes to produce offspring chromosomes. Algorithm 1 starts with a given population of NP individuals generated at random. Each individual is evaluated by a fitness function and the fittest individuals have higher probability of being selected to generate offspring. The selected individuals generate offspring by the operators of crossover and mutation. The offspring individuals are then evaluated. This cycle continues until some stopping criterion is reached.

Algorithm 1 Genetic Algorithm

```
1: Initialize population
2: Evaluate population
3: while stopping criterion is not met do
4:     Selection
5:     Crossover
6:     Mutation
7:     Evaluate population
8: end while
```

In this work, a tournament selection was adopted. The GA operators used in this work are the exponential crossover and the polynomial mutation. These choices are based on a previous work in which several different operators were tested for the problem of dispatching forest firefighting vehicles, and these were the ones that gave the best results [11].

In the tournament selection two or more parent chromosomes are randomly selected from population, and the one with the highest fitness function value is selected to generate offspring. In this work, a binary tournament selection was used. Thus, tournaments between two selected chromosomes are successively performed. Each chromosome of the population can only participate in a single tournament. Exponential crossover is applied to two parent individuals to generate two new offspring. An initial starting position is generated at random that guaranties that at least the offspring will differ from the parents in one position. Then, for the remaining positions of the parent individuals, a random number between 0 and 1 is generated; if it is less than the the control parameter (CR, crossover rate), the parent genes are copied to the offspring; otherwise, the parent genes are swapped and then copied to the offspring individual [13,19]. The crossover rate determines the probability of crossover occurring [14]. Thus, the offspring vector u_i is componentwise computed either from the elements of first parent vector $x_{i,\cdot} = x_{i,j} = x_{i,1}, \ldots, x_{i,n}$ or the elements of the second parent vector $v_{i,\cdot} = v_{i,j} = v_{i,1}, \ldots, v_{i,n}$ ($i = \{1, \ldots, NP\}$, $j = \{1, \ldots, n\}$ and n is the dimension of the individual):

$$u_{i,j} = \begin{cases} v_{i,j}, \text{ if } \text{rand}_{i,j} \geq \text{CR} \text{ or } j = j_{\text{rand}} \\ x_{i,j}, \text{ otherwise} \end{cases}$$

where $\text{rand}_{i,j} \sim U(0,1)$ is a uniformly distributed random number which is generated for each j and $j_{\text{rand}} \in \{1,\ldots,n\}$ is a random integer used to ensure that $u_{i,\cdot} \neq x_{i,\cdot}$ in all cases.

In the polynomial operator, a polynomial probability distribution is used to generate a solution in the vicinity of a parent. The polynomial mutation employs a distribution parameter, η_m, which is typically set in the range between 20 and 100. This parameter plays a crucial role in controlling the spread effect of the mutation. The probability distribution, both to the left and right of a variable value, is adjusted so that values outside the specified interval $\left[x_j^L, x_j^U\right]$ can not be created by the mutation operator. A mutated individual $y_{i,\cdot}$ is created from the parent individual $x_{i,\cdot}$ by mutating each position $j \in \{1,\ldots,n\}$ of the parent individual, according to a polynomial distribution where $u \sim U(0,1)$ [4].

$$y_{i,j} = \begin{cases} x_{i,j} + \delta_L(x_{i,j} - x_j^L), & \delta_L = (2u)^{\frac{1}{1+\eta_m}} - 1 \quad \text{for } u \leq 0.5 \\ x_{i,j} + \delta_R(x_j^U - x_{i,j}), & \delta_R = 1 - (2(1-u))^{\frac{1}{1+\eta_m}} \quad \text{for } u > 0.5 \end{cases} \tag{1}$$

The termination criteria determines the end of the search. Possible termination conditions are related with the stagnation of the population where there is no further improvement in the value of the objective function, reaching the specified maximum number of iterations, or reaching a predetermined value of the objective function evaluations. If the stopping criterion is not met, the algorithm continues and repeats all steps again to generate a new generation.

3.3 Differential Evolution

Differential evolution is an evolutionary algorithm and was first introduced by Storn and Price in 1997 [16]. Since then, it has been widely adopted due to its simplicity and effectiveness in solving complex optimization problems [14]. The DE steps are presented in Algorithm 2, where NP is the population size, F is the mutation factor and CR is the crossover rate. DE has some similarities with GA, differing in the order of operators. It starts with a given population of individuals (vectors). The population evolves through mutation, crossover and finally selection. This evolution takes place while the stopping criterion has not been reached.

Algorithm 2 Differential Evolution Algorithm

```
Initialize population
Evaluate population
while stopping criterion is not met do
    Mutation
    Crossover
    Evaluate population
    Selection
end while
```

In DE, potential solutions of the problem are represented as vectors in the search space. DE uses mutation, crossover and selection operators. The operators define different variants of DE, which nomenclature is as follows: DE/Selection/1/Crossover. In this work, the variant chosen was DE/Current-to-best/1/Exponential, since it was the best variant for the dispatch problem, found in [10].

During the mutation phase, a new candidate is formed by adding a scaled differential term to an existing base vector (individual). A key factor influencing this mutation process and subsequently the effectiveness of the algorithm is the mutation factor (F_1 and F_2), typically limited to a range of 0 to 2. This parameter plays a fundamental role in regulating the rate of evolution within the population, dictating the magnitude of the adjustment to the differential term. A higher value of F_1 and F_2 corresponds to a greater probability of breaking free from local optima, thus promoting the exploration of the solution space. The mutation in the variant DE/current-to-best/1 is defined by a mutant vector v_i computed by [18]

$$v_i = x_i + F_1 \left(x_{\text{best}} - x_i\right) + F_2 \left(x_{r_1} - x_{r_2}\right)$$

where x_i is the selected individual (i.e. $r_1 \neq r_2 \neq r_3 \neq i$) and F_1 and F_2 are real constants, $F_1, F_2 \in [0, 2]$.

In this work, an exponential crossover is used, as in GA. In DE, crossover involves combining mutant vectors with the candidate solutions present in current population. In other words, the algorithm combines two populations: the current one and the mutated one, and a new experimental population is created.

In the selection phase, a mechanism is used to select the individuals that will form the next generation. Based on their objective values, a comparison is made between the experimental and current individuals. The individuals with the lowest function value (when considering a minimization problem) are selected to form the next generation. Thus, the new target vector is obtained and its value is computed as follows:

$$x_i = \begin{cases} u_i, \text{ if } f(u_i) < f(x_i) \\ x_i, \text{otherwise} \end{cases}$$

where $f(u_i)$ is the objective function value of the trial vector and $f(x_i)$ is the objective value of the parent vector.

As with GA, the DE algorithm terminates when a predefined stopping criterion is reached. If the stopping criterion is not reached, the algorithm continues and repeats all the steps to generate a new generation.

4 Computational Experiments

In this section, the details used in the algorithms (GA and DE), the results obtained by them and finally a discussion of the results through statistical analysis will be presented.

4.1 Details

In this work, the GA and DE are adapted from the framework *pymoo*: Single-objective Optimization in Python [3]. The experiments were carried out on a PC 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz, 2803 Mhz, 4 Nucleus(s), 8 Processor(s) Logic(s), 16 Gb RAM. The code was implemented in *python* (version 3.9.35) using *VScode* (Version 1.87).

In order to test our approach, these metaheuristics are used for a set of 60 instances of the firefighting resource dispatch problem. These instances were randomly generated for 20 and 30 ignitions and 5, 7 and 10 resources, available at `https://github.com/MarinaAMatos/Inst_Firefighting`. For each pair of ignitions and resources, 10 instances are randomly generated. There are different types of resources (with predefined water tank capacities - 500 l, 1000 l, 1500 l, 2000 l and 3000 l - randomly assigned). The values of the burned area of each ignition at the initial instant of time are also randomly defined for each instance and range from 1 to 500 m^2. The water required to extinguish a given ignition is calculated using the perimeter of a circle, whose radius is obtained by multiplying the burned area by 10, which is the amount of water needed to extinguish 1 m of the ignition area.

The initial population size used was set at 200 and the maximum number of generations and the maximum number of function evaluations were 1000 and 100000, respectively, for the two metaheuristics. The m parameter chosen was 5, i.e. each resource is allowed to return up to 5 times to a given ignition when its water tank needs refilling. This value was chosen because it was considered sufficient to obtain feasible solutions, since the higher it is, the longer the algorithms will run. The operators exponential crossover, polynomial mutation and tournament selection were used in GA. For DE, the variant applied is "DE/current-to-best/1/exp", i.e. the exponential crossover, polynomial mutation and current-to-best selection operators were applied. The $CR = 0.3$, $F_1 = F_2 = 0.5$ and all the other parameters used were the default ones.

4.2 Results

The results obtained from GA are described below. It should be noted that in all instances the ignitions were all extinguished. For each number of ignitions, NI, and number of resources, NR, Table 2 shows the results obtained by GA and DE, taking into account the average values of the best values (GA_{best} and DE_{best}) and the average of the values (GA_{avg} and DE_{avg}) over 30 runs. Columns 4 to 9 show the average of the values obtained in the 10 instances for each pair of NI and NR. TBA_{avg} represents the average value of the total burned area TBA, UW_{avg} is the average value of used water, $Time_{avg}$ is the average value of the instants of time required to extinguish all ignitions, PT_{avg} the average value of the processing time in seconds, nfe_{avg} the average value of the number of function evaluations, and $ngen_{avg}$ the average value of the number of generations. The values obtained for TBA_{avg}, UW_{avg} and $Time_{avg}$ in GA are lower than DE.

Table 2: Solution values for GA and DE

NI	NR		TBA_{avg}	UW_{avg}	$Time_{avg}$	PT_{avg}	nfe_{avg}	$ngen_{avg}$
20	**5**	GA_{best}	**5210.5**	10802.3	114.0	270.8	20500.0	103.5
		DE_{best}	5318.1	10929.0	130.0	254.6	18400.0	93.0
		GA_{avg}	**5273.2**	10876.3	124.4	219.3	17289.3	87.4
		DE_{avg}	5388.2	11008.6	146.6	201.5	15051.3	76.3
	7	GA_{best}	**5420.1**	11101.6	101.0	271.4	20060.0	101.3
		DE_{best}	5481.6	11170.9	108.0	257.1	19440.0	98.2
		GA_{avg}	**5455.8**	11145.3	104.5	224.4	16208.7	82.0
		DE_{avg}	5520.4	11216.4	116.3	184.1	13890.0	70.5
	10	GA_{best}	**5433.1**	11059.9	80.0	294.4	21780.0	109.9
		DE_{best}	5458.7	11096.8	84.0	250.2	18260.0	92.3
		GA_{avg}	**5450.9**	11082.9	80.2	196.9	13580.0	68.9
		DE_{avg}	5477.4	11114.6	86.4	167.5	11922.7	60.6
30	**5**	GA_{best}	**8042.3**	16654.6	185.0	535.9	39800.0	200.0
		DE_{best}	8606.1	17301.5	242.0	316.8	21300.0	107.5
		GA_{avg}	**8309.7**	16970.5	212.7	392.3	26726.0	134.6
		DE_{avg}	8909.0	17584.6	264.4	291.2	17812.7	90.1
	7	GA_{best}	**7812.3**	16319.4	139.0	662.3	38080.0	191.4
		DE_{best}	8110.2	16633.8	175.0	347.9	21420.0	108.1
		GA_{avg}	**7971.2**	16514.8	162.4	413.4	24499.3	123.5
		DE_{avg}	8264.7	16835.1	193.6	280.5	16858.7	85.3
	10	GA_{best}	**7326.9**	15843.1	107.0	583.7	33460.0	168.3
		DE_{best}	7469.4	16004.1	130.0	361.8	23120.0	116.6
		GA_{avg}	**7390.9**	15921.7	115.6	419.7	23416.7	118.1
		DE_{avg}	7531.5	16089.7	131.2	245.2	14426.0	73.1

In relation to the values of PT_{avg}, nfe_{avg} and $ngen_{avg}$, it was found that the solutions obtained by DE were smaller than those obtained by GA.

Table 3 shows the statistical analysis of the TBA values between GA and DE for all instances. For the statistical analysis, a one-tailed paired sample t-student test was used, and in all tests, a significance level of 5% was considered. The table shows the solutions for all pairs of number of ignitions and number of resources, where the p-values and the difference in the TBA value between the two metaheuristics are shown. It was observed significant differences between all the results, since the p-value is less than 0.05.

Table 3: Statistical analysis between GA and DE

NI	**20**			**30**		
NR	**5**	**7**	**10**	**5**	**7**	**10**
GA-DE (best)	-107.7	-61.5	-25.6	-563.8	-298.0	-142.5
GA-DE (avg)	-115.0	-64.6	-26.5	-599.3	-293.5	-140.6
p-value	<0.001					

5 Conclusions

Currently, fires have become an area of interest in the academic community, due to the numerous damages caused. Managing the use of forest firefighting resources is a crucial issue, and they need to be used more efficiently so that the fire can be controlled in a shorter period of time.

This paper presents a firefighting resource dispatch problem that aims to know which and when each resource goes to each ignition, in order to minimize the total burned area. To this end, the GA and DE metaheuristics were adapted to this problem and applied to 60 randomly generated instances with 5, 7 and 10 resources to extinguish 20 ignitions, and 5, 7 and 10 resources to fight 30 ignitions. It was concluded that GA is better in terms of minimizing the value of TBA when compared to DE. However, DE takes less time to find solutions, i.e. it has a shorter runtime than GA.

In future work, a forest firefighting resource dispatch problem using real data will be solved.

References

1. Attri, V., Dhiman, R., Sarvade, S.: A review on status, implications and recent trends of forest fire management. Archives of Agriculture and Environmental Science **5**(4), 592–602 (2020)
2. Bélanger, V., Lanzarone, E., Nicoletta, V., Ruiz, A., Soriano, P.: A recursive simulation-optimization framework for the ambulance location and dispatching problem. European Journal of Operational Research **286**(2), 713–725 (2020)

3. Blank, J., Deb, K.: pymoo: Multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)
4. Deb, K., Deb, D.: Analysing mutation schemes for real-parameter genetic algorithms. International Journal of Artificial Intelligence and Soft Computing **4**(1), 1–28 (2014)
5. Granberg, T.A.: Optimized dispatch of fire and rescue resources. In: Computational Logistics: 13th International Conference, ICCL 2022, Barcelona, Spain, September 21–23, 2022, Proceedings. pp. 132–146. Springer (2022)
6. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA, USA (1975)
7. HomChaudhuri, B., Kumar, M., Cohen, K.: Genetic algorithm based simulation–optimization for fighting wildfires. International Journal of Computational Methods **10**(06), 1350035 (2013)
8. Lambora, A., Gupta, K., Chopra, K.: Genetic algorithm- a literature review. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). pp. 380–384 (2019)
9. Martell, D.L.: A review of recent forest and wildland fire management decision support systems research. Current Forestry Reports **1**(2), 128 – 137 (2015). https://doi.org/10.1007/s40725-015-0011-y
10. Matos, M.A., Gonçalves, R., Rocha, A.M.A., Costa, L.A., Alvelos, F.: Resource dispatch optimization for firefighting using a differential evolution algorithm. In: International Conference on Optimization, Learning Algorithms and Applications. pp. 63–77. Springer (2024)
11. Matos, M.A., Rocha, A.M.A.C., Costa, L.A., Alvelos, F.: Resource dispatch optimization for firefighting based on genetic algorithm. In: Gervasi, O., Murgante, B., Rocha, A.M.A.C., Garau, C., Scorza, F., Karaca, Y., Torre, C.M. (eds.) Computational Science and Its Applications – ICCSA 2023 Workshops. pp. 437–453. Springer Nature Switzerland, Cham (2023)
12. Mendes, A.B., e Alvelos, F.P.: Iterated local search for the placement of wildland fire suppression resources. European Journal of Operational Research **304**(3), 887–900 (2023)
13. Meng, Z., Chen, Y.: Differential evolution with exponential crossover can be also competitive on numerical optimization. Applied Soft Computing **146**, 110750 (2023). https://doi.org/https://doi.org/10.1016/j.asoc.2023.110750
14. Price, K.V.: Differential Evolution, pp. 187–214. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-30504-7_8
15. San-Miguel-Ayanz, J., Durrant, T., Boca, R., Maianti, P., Liberta, G., Artes, T.V., Oom, D.J.F., Branco, A., De Rigo, D., Ferrari, D., Pfeiffer, H., Grecchi, R., Onida, M., Loffler, P.: Forest Fires in Europe, Middle East and North Africa 2021. No. EUR 31269 EN, Publications Office of the European Union, Luxembourg (2022). https://doi.org/10.2760/34094
16. Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization **11**, 341–359 (1997)
17. Zeferino, J.A.: Optimizing the location of aerial resources to combat wildfires: a case study of portugal. Natural Hazards **100**(3), 1195–1213 (2020)
18. Zeng, Z., Zhang, M., Chen, T., Hong, Z.: A new selection operator for differential evolution algorithm. Knowledge-Based Systems **226**, 107150 (2021)
19. Zhao, S.Z., Suganthan, P.N.: Empirical investigations into the exponential crossover of differential evolutions. Swarm and Evolutionary Computation **9**, 27–36 (2013)