An elitist non-dominated heuristic resolution for the dynamic asset protection problem

Quentin Peña, Aziz Moukrim, Mehdi Serairi

An elitist non-dominated heuristic resolution for the dynamic asset protection problem

Quentin Peña[1], Aziz Moukrim[1], and Mehdi Serairi[1]

Université de Technologie de Compiègne, CNRS,
Heudiasyc (Heuristics and Diagnosis of Complex Systems),
CS 60 319 - 60 203 Compiègne Cedex
{quentin.pena, aziz.moukrim, mehdi.serairi}@hds.utc.fr

Abstract. During escaped wildfires, community assets are at risk of damage or destruction. Preventive operations requiring dispatching resources and cooperation can be taken to protect these assets. The planning of such operations is sensitive to unforeseen disruptions that may occur. To account for the effects of the disruption, it may be necessary to alter the initial routes of the vehicles. The problem rising from the rescheduling of the vehicles is a bi-objective optimization problem known as the Dynamic Asset Protection Problem (D-APP). We propose a genetic algorithm based on the Non-dominated Sorting Genetic Algorithm (NSGA-II) to solve the D-APP. We define new mutation and crossover operators adapted to our problem, and we propose procedures to repair and evaluate a solution based on Mixed Integer Programming (MIP).

Keywords: bi-objective optimization · vehicle routing · team orienteering · synchronization · NSGA-II.

1 Introduction

In the recent years, wildfires break out more frequently throughout the world. When wildfires are not controlled, they quickly expand and can burn thousands of hectares of vegetation. In urban areas, the fire can also harm people and damage infrastructure. Emergency response teams and resources must be deployed to respond to these escaped wildfires. Multiple operations are jointly carried out, from fire containment to evacuation, sheltering operations and including asset protection. In this paper, we will focus on the preventive actions for the protection of community assets.

Depending on the community asset, different actions can be taken to mitigate or nullify the damages caused when the wildfire reach them. Such actions include removing fuel materials, wetting down buildings, or reducing fire. Preventive protection actions must be taken in a timely manner: it has to be performed before the fire reaches the asset, but not too early to be efficient. Some interventions may require several trucks with specific capacities, thus requiring different teams to collaborate to perform the task in a synchronous way. In particular, we will focus on rerouting the vehicles after a disruption occurs that invalidates their

initial routes. A wide range of disruptions can impact our initial plans in different ways. For example, we may not have all the resources available, due to a faulty equipment or a vehicle breakdown. The time windows on some assets may be updated after unforeseen wind or weather changes, altering the propagation of the fire. Travel times between assets may also change if traffic jams are caused by people evacuating, or a road might be blocked by a fallen tree.

The problem of routing vehicles for preventive protection operations can be viewed as a variant of the Team Orienteering Problem (TOP) with time windows and synchronization constraints. This problem was first introduced by Merwe et al. [9] as the Asset Protection Problem (APP). The authors proposed a Mixed Integer Linear Programming (MILP) formulation of the problem. Later, Merwe et al. [10] introduced the dynamic APP (D-APP), which is a bi-objective problem for rerouting the vehicles after a disruption. The authors updated the MILP formulation from the mono-objective version of the problem to account for the deviation. They generated theset of solutions offering optimal trade-off between protection of the assets and deviation from the initial routes using an ϵ-constraint scheme. Peña et al. [13] proposed a new mathematical formulation and valid inequalities based on the properties of the D-APP.

In this paper, we will present a heuristic solution method for the D-APP based on the Non-dominating Sorting Genetic Algorithm (NSGA-II) [3]. We will introduce different crossover and mutation operators specific to our problem, including a destruction/construction operator as well as different MILP to repair and evaluate a solution.

2 Dynamic Asset Protection Problem

During a wildfire, community assets such as schools, hospitals, bridges are at risk of being damaged. A fleet of heterogeneous vehicles must be dispatched to the different assets to perform preventive protection operations. These operations must be accomplished within a specific time window, and often require the cooperation of multiple vehicles.

An asset is protected if, within its time window, enough vehicles are present at the asset to accomplish the protection operation. The protection of an asset requires some resources (e.g., crew size, number of fire hoses, ...), that need to be met by the vehicles assigned to the asset.

In the dynamic APP, we already have routes assigned to the vehicles. However, an unforeseen disruption occurred and these initial routes may no longer be feasible nor optimal. We want to recompute the routes to take into account the consequences of the disruption. We then have two competing objectives:

- maximizing the total value of the protected assets
- minimizing the deviation from initial routes

We define the deviation as the number of vehicle/asset reassignments, i.e., if an asset is added to or removed from the initial route of a vehicle, then it implies a deviation of one.

2.1 Problem presentation

An instance of our problem is represented by a graph $G = (V, A)$, with V representing the locations and A the arcs. There are n total locations. The first m locations represent the depots from which the vehicles depart, and the n-th location is a fictitious sink node. The remaining $n-m-1$ locations represent the assets to protect. We define subsets of V: the depots V^d and the assets V^a. Each asset i has a value v_i, a requirement vector r_i, a service duration a_i and a time window $[o_i; c_i]$. The set $\mathcal{P}$ represents the available vehicles. Each vehicle p has a capability vector cap_p. In order to be protected, the vehicles assigned to an asset must collectively meet the resource requirement of the asset. For example, an asset with resource requirement $r_i = (1, 2, 1)$ can be protected by vehicles p and q with respective capability $cap_p = (1, 1, 0)$ and $cap_q = (1, 1, 1)$. All the vehicles assigned to the asset must be present when the service starts, and throughout the entirety of the service. Parameters t_{ijp} are the travel time between locations i and j for vehicle p. Travel times satisfy the triangle inequality. We note Φ the solution representing the routes of the vehicles before disruption.

Before proceeding further, we introduce some definitions. An arc between two assets i and j is called *a valid arc* for vehicle p if $o_i + a_i + t_{ijp} \leq c_j$. In other words, vehicle p can visit asset i before asset j within the respective time windows of the assets. Additionally, we say that the insertion of an asset k between two assets i and j in the route of vehicle p is at a valid position if arcs (i, k) and (k, j) are valid arcs for vehicle p.

2.2 Bi-objective optimization

The D-APP is a bi-objective optimization problem. We recall some terminology related to Multi-objective Optimization Problems (MOP).

In MOP, a solution is evaluated according to an objective function vector $f = (f_1, ..., f_d)$ with d objectives. Without loss of generality, we suppose that all the objectives are to be maximized. These d objectives are competing against each other: improving one of the objective will often degrade one or multiple other objectives. Hence, we want to find the set of efficient solutions based on a dominance relation between solutions [7].

Definition 1. *Let u and v be vectors of $\mathbb{R}^d$, we say that u dominates v if and only if $u_i \geq v_i$ for each $i \in \{1, ..., d\}$ and there exists $j \in \{1, ..., d\}$ such that $u_j > v_j$. We denote this dominance relation by $u \succ v$.*

Definition 2. *A solution s is efficient if there is no other solution s' such that $f(s') \succ f(s)$, with $f(s)$ the objective function vector associated with solution s.*

For ease of use, we say that a solution s dominates a solution s' if and only if its objective function vector $f(s)$ dominates $f(s')$. The set of all efficient solutions is known as the efficient set. The set of objective vectors with respect to the efficient set is call the non-dominated set, or Pareto front [12].

In many vehicle routing problems, multiple competing objectives have been considered [6]. A popular approach is to solve the multi-objective problem using a decomposition approach. The multi-objective problem is decomposed in multiple single objective problems, using aggregation functions. For instance, the bi-objective traveling salesman problem has been solved using ant colony optimization based on decomposition [2]. A metaheuristic method that combines a Pareto ant colony optimization algorithm and a variable neighborhood search method has been proposed for the bi-objective TOP (BTOP) [14]. Finally, a two-phase decomposition method based on Local Search has been proposed to solve Selective Pickup and Delivery Problems with Time Windows (SPDPTW) [1].

Several approaches extend the fast and elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II) [3]. It has been efficiently applied to various multi-objective problems including but not limited to the BTOP [11], the Green Vehicle Routing Problem [4] and the Vehicle Routing Problem with Route Balancing [5].

3 NSGA-II

In this section, we will discuss the implementation of the NSGA-II algorithm to solve the D-APP. We will first present in Section 3.1 an overview of the NSGA-II algorithm. We will then present in Section 3.2 the encoding we encounter in the literature for a genetic algorithm on a problem similar to the problem at hand. We will introduce mutation and crossover operators based on the properties of our problem in Section 3.3. Finally, we will define two different procedures for repairing and evaluating a solution in Section 3.4.

3.1 Overview

NSGA-II is an iterative algorithm. For each generation t, we consider a population R_t of size $2N$, that is the combination of two subpopulations of size N: P_t, the parents, and Q_t, the offspring. There are three main steps in the NSGA-II algorithm, described below. A solution i has two fitness criteria relative to the current population: a rank r_i and a crowding distance d_i. The rank represents the quality of the solution with regards to the dominance relation presented in Section 2.2. The crowding distance represents the quality of the solution in terms of diversification. For more information on how these criteria are computed, we refer the reader to [3].

At generation t, the three steps are:

Step 1 - Initialization. Create the population R_t by combining the parent and offspring populations. Compute the rank of the solutions in R_t and identify all the non-dominated fronts $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, ...)$. Compute the crowding distance of the solutions within each non-dominated front.

Step 2 - Parent population selection. Create the parent population for next generation P_{t+1} by selecting the N solutions from population R_t. Between two solutions with different ranks, we prefer the solution with the lowest rank. If

both solutions belong to the same front, we prefer the solution with the lowest crowding distance.

Step 3 - Offspring creation. Create offspring population Q_{t+1} from P_{t+1}. Details are given in Algorithm 1. The tournament operator is binary tournament, as described in [3]. Two solutions are selected at random, the solution with lowest rank is selected, or with lowest crowding distance if there is a tie. The crossover and mutation operators are discussed in Section 3.3. The repair and evaluation procedure is discussed in Section 3.4.

Algorithm 1 Offspring creation

Data: Parent population P, mutation rate μ
Result: Offspring population Q
$Q \leftarrow \emptyset$;
while $|Q| \leq N$ **do**
  $p_1 \leftarrow tournament(P)$;
  $p_2 \leftarrow tournament(P)$;
  $s \leftarrow crossover(p_1, p_2)$; (See Section 3.3)
  if rand() $< \mu$ **then**
    $s \leftarrow mutate(s)$; (See Section 3.3)
  end if
  $s \leftarrow repair\_and\_evaluate(s)$; (See Section 3.4)
  $Q \leftarrow Q \cup \{s\}$;
end while
return Q

3.2 Encoding

We based the implementation of the NSGA-II algorithm for our problem on a genetic algorithm proposed for the mono-objective version of the APP with a homogeneous fleet of vehicles [8].

A solution s is represented by an array of integers, representing the order in which assets are visited for each vehicle. The route of a vehicle always starts at a depot and ends at the sink node. For instance, there are three vehicles in solution $[1, 2, 6, 4, 11, 1, 5, 7, 3, 11, 1, 7, 3, 11]$, the route of the first vehicle is $(1 \rightarrow 2 \rightarrow 6 \rightarrow 4 \rightarrow 11)$, the second $(1 \rightarrow 5 \rightarrow 7 \rightarrow 3 \rightarrow 11)$ and the last $(1 \rightarrow 7 \rightarrow 3 \rightarrow 11)$.

We note $\mathcal{P}_i^s$ the set of vehicles assigned to asset i in solution s, and $\overline{\mathcal{P}_i^s}$ the set of available vehicles not assigned to asset i in solution s.

3.3 Operators

Valid crossover operator (CXVAL). This crossover operator between two solutions s_1 and s_2 selects a vehicle at random. The route for this vehicle in

s_1 is cut after a random asset i_k. The route for this vehicle in s_2 is also cut, after asset j_l. The offspring route for this vehicle is constructed by the taking the part of the route up to, and including, asset i_k in s_1 first, and then the part of the route after asset j_l in s_2. For example, suppose we have two routes $(i_1 \rightarrow i_2 \rightarrow \mathbf{i_3} \rightarrow i_4 \rightarrow i_5)$ and $(j_1 \rightarrow j_2 \rightarrow j_3 \rightarrow \mathbf{j_4} \rightarrow j_5 \rightarrow j_6)$, and assume the cuts happen after assets i_3 and j_4 respectively, indicated in bold. The resulting route would be $(i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow j_5 \rightarrow j_6)$. The route of the second vehicle is cut in a way such that arc (i_k, j_{l+1}) is a valid arc. This crossover may result in duplicate assets in the route of a vehicle; we only keep the first occurrence of an asset to fix this issue.

Time crossover operator (CXTIM). This crossover operator between two solutions s_1 and s_2 selects a time at random within the time horizon. The routes for the vehicles in s_1 are cut when the start time of service of the asset exceeds the chosen time, and represent the first part of the offspring routes. We then cut the routes of the vehicles in s_2 such that there is a valid arc between the last asset of the first part of the route and the first asset of the second part of the route.

Single-change operators. We define two different mutation operators that perform a single change on the solution, with same probability of being used: an insertion operator and a removal operator.

Insertion operator. The insertion operator adds one randomly selected asset to the route of one or multiple vehicles. An asset is selected at random. The asset is added at a random valid position in the route of vehicles, taken in a random order, until the resource requirement of the asset is met.

Removal operator. The removal operator removes one randomly selected asset from the route of one or multiple vehicles. An asset is selected at random. The asset is removed from the route of all the vehicles.

Multi-change operator. We define a mutation operator that performs multiple changes on the solution, first removing multiple assets from the solution in the destruction phase, then inserting multiple protected assets in the construction phase.

During the destruction phase, the operator randomly selects d assets to be removed from the current solution. The number of assets removed is randomly selected between 1 and d_{max}. The destruction parameter d_{max} is initially set to 3. If there is no improvement on the optimal Pareto front $\mathcal{F}_1$, its value is increased, and resets to 3 when an improvement is found. In the random selection process, we can assign weights to the assets in order to favor removing assets that induce most deviation. We note w_i^- the weight associated to asset i. The probability of selecting asset i to be removed is thus $p^-(i) = w_i^- / \sum_i w_i^-$. If $w_i^- = 1$ for all

assets, we have fully random behavior. Alternatively, we can use a weight based on the deviation induced by the removal of asset i from solution s, with γ a parameter to be determined:

$$w_i^- = \left(1 + max(0, |\mathcal{P}_i^s| - |\overline{\mathcal{P}_i^s}|)\right)^\gamma \quad (1)$$

During the construction phase, the operator uses a Best Insertion Heuristic (BIH) to insert a subset of assets to the current solution. The number of assets to add is chosen randomly between d and $d + c_{max}$. The construction parameter c_{max} is initially set to 3. The assets to be inserted are randomly selected. We can assign weights to the assets in the selection process. We note w_i^+ the weight associated to asset i. We can use a weight based on the profit v_i associated with the protection of asset i and a lower bound on the deviation necessary for the protection of the asset nb_i^+, with α and β parameters to be defined:

$$w_i^+ = v_i^\alpha / (1 + nb_i^+)^\beta \quad (2)$$

We want to add each asset to the route of enough vehicles for the resource protection to be met. We also want to minimize the number of vehicles we use to protect the asset. As we do not know how many vehicles will be required to meet the resource requirement, we will generate multiple insertion patterns and select the one minimizing our criterion. We detail the process in Algorithm 2. In order to account for the deviation from the pre-disruption routes, we first select the vehicles for which the asset is in the pre-disruption route. If these vehicles are not sufficient to meet the resource requirement, we continue the process with the remaining vehicles. We select the vehicles in a random order, until the protection requirement is met.

Adaptive parameters. The multi-change operator relies on parameters α, β and γ to control the relative importance of the different factors when associating weights to assets. They are first initialized with $\alpha = 1$, $\beta = 1$ and $\gamma = 0.5$, and then adaptively tuned during the offspring creation phase. We generate M offspring solutions with slightly different values of α, β and γ. The values leading to the best offspring subpopulation are recorded to be used in the next iteration. All the offspring solutions generated are considered in the offspring population Q of the current step.

3.4 Repair and evaluation procedure

A solution is represented by the route of each vehicle. It is sufficient to know the routes of the vehicles to compute the deviation from the pre-disruption routes. However, we cannot determine which assets are effectively protected: we must check if it is possible to synchronize the visits of all assigned vehicles within the time window of the asset, and if the resource requirement is met by these vehicles.

Some solutions are not feasible. For instance, two vehicles may visit two assets in a different order, thus causing the synchronization to be impossible.

Algorithm 2 Construction: Add an asset

Data: Solution S, asset k, number of insertion patterns nb_p
Result: A solution that protects asset k, if possible

if the available vehicles cannot meet the resource requirement **then**
  return S
end if
for $cpt = 1...nb\_p$ **do**
  $V_{cpt} \leftarrow \emptyset$; {Set of selected vehicles at iteration cpt}
  $cost_{cpt} = 0$;
  Determine a random order on the vehicles that prioritizes vehicles in $\mathcal{P}_k^{\Phi}$
  for each vehicle p following the previously defined order **do**
    if there is a valid position in the route of vehicle p **then**
      $V_{cpt} \leftarrow V_{cpt} \bigcup \{p\}$
      $cost_{cpt} \leftarrow cost_{cpt} + 1$
      if vehicles in V_{cpt} meet the resource requirement of asset k **then**
        Begin new insertion pattern (next cpt)
      end if
    end if
  end for
end for
Select set of vehicles V_* with lowest cost
Insert asset k in the routes of vehicles in V_* in solution S
return S

Our repair procedure aims at finding the best subroutes of the solution, to make it feasible and maximize total protected value. We do not modify the order in which assets are visited by a vehicle, nor do we add new assets to the routes. The repair procedure determines which assets can actually be protected, thus contributing to the total protected value. It also gives data to correct the deviation, if unprotected assets have been added to the route of a vehicle for instance. At the end of the repair procedure, we know the value of the two objective functions for the solution we have just repaired. Hence, we can use the repair procedure as the evaluation procedure for our solutions. By doing so, we also ensure that all the solutions we consider are feasible.

We propose two different MIPs used for repairing and evaluating solutions for our problem. We note $\mathcal{P}_i$ the set of vehicles that have asset i in their route. We note X_p the set of arcs (i_k, i_l) between assets in the route of vehicle p, with $k < l$.

Asset penalization. The first MIP tries to find a feasible solution from the given routes. Assets can be visited outside of their time windows, but these assets cannot be protected. Infeasibilities are lifted by removing assets entirely from the solution.

We define three sets of decision variables:

- Binary variables Y_i, set to 1 if asset i is protected. Asset i is protected when service starts within its time window and its resource requirement is met.

– Binary variables θ_i, set to 1 if asset i is removed from the solution.
– Continuous variables S_i, that represent the start time of service of asset i.

$$Maximize \sum_i v_i Y_i \tag{3}$$

$$(1-\theta_i)\sum_p cap_p \geq r_i Y_i \ \ \forall i \in V^a \tag{4}$$

$$S_i + t_{ijp} + a_i \leq S_j + M_1(\theta_i + \theta_j) \ \ \forall p \in \mathcal{P}, (i,j) \in X_p \tag{5}$$

$$o_i - M_2(1-Y_i) \leq S_i \leq c_i + M_2(1-Y_i) \ \ \forall i \in V^a \tag{6}$$

$$Y_i \in \{0,1\}, \ \theta_i \in \{0,1\}, \ S_i \in \mathbf{R} \ \ \forall i \in V^a \tag{7}$$

Objective function (3) maximizes the total protected value.

Constraints (4) ensure that the protection requirement is met for protected assets. Assets that have been removed from the solution (with $\theta_i = 1$) cannot be protected.

Constraints (5) set correct start time of service for assets i and j when asset i is visited by the vehicle before asset j. The order in which the assets are visited is fixed within the solution. However, as assets can be removed, we need to consider every pair of assets (i,j) visited by the vehicle such that asset i is visited before asset j.

Constraints (6) ensure that a protected asset is visited within its time window. Constraints (7) define the domain of the decision variables.

Assignment penalization. The second MIP tries to find a feasible solution from the given routes. Infeasibilities are lifted by removing assets from the routes of individual vehicles.

We use binary variables Y_i and continuous variables S_i. We replace variables θ_i by variables θ_{pi}, set to 1 if asset i is removed from the route of vehicle p.

$$Maximize \sum_i v_i Y_i \tag{8}$$

$$\sum_{p \in \mathcal{P}_i} (1-\theta_{pi}) cap_p \geq r_i Y_i \ \ \forall i \in V^a \tag{9}$$

$$S_i + t_{ijp} + a_i \leq S_j + M_1(\theta_{pi} + \theta_{pj}) \ \ \forall p \in \mathcal{P}, \ (i,j) \in X_p \tag{10}$$

$$Y_i + \theta_{pi} \geq 1 \ \ \forall p, \ \forall i \in V^a \tag{11}$$

$$o_i - M_2(1-Y_i) \leq S_i \leq c_i + M_2(1-Y_i) \ \ \forall i \in V^a \tag{12}$$

$$Y_i \in \{0,1\}, \ S_i \in \mathbb{R} \ \ \forall i \in V^a \tag{13}$$

$$\theta_{pi} \in \{0,1\} \ \ \forall i \in V^a, \ p \in \mathcal{P}_i \tag{14}$$

Objective function (8) maximizes the total protected value.

Constraints (9) ensure that the protection requirement is met for protected assets. If asset i is removed from the route of the vehicle (with $\theta_{pi} = 1$), the vehicle does not contribute to the protection.

Constraints (10) set correct start time of service for assets i and j when asset i is visited by the vehicle before asset j, similarly to constraints (5).

Constraints (11) ensure that unprotected assets are removed from the routes of all vehicles.

Constraints (12) ensure that a protected asset is visited within its time window. Constraints (13) and (14) define the domain of the decision variables.

Local search. After repairing a solution, we explore its neighborhood to find a dominating solution. We base our local search on the MIP used in the ϵ-constraint method for the D-APP introduced in [13]. We use the MIP that maximizes total protected value with deviation limited to the value of the deviation of the solution we are considering. This solution is used as a warm-start for the MIP. We set a high relative gap tolerance in our solver, meaning that the resolution will stop before optimality is proven. For example, with a tolerance of 0.05, a solution is returned when it is proved to be within 5% of optimal.

4 Computational results

We carried out computational testing on a computer with an Intel Core i7-8550U processor and 8 GB of RAM. We implemented the method in Julia.

We generated 10 benchmark instances[1], following the guidelines provided by Merwe et al.[9]. Each instance has 100 assets randomly distributed within a 80 km by 80 km grid. Instances of less than 100 assets are created using a subset of the 100-asset instances.

In order to evaluate our algorithm performance, we will use a quality indicator to compare approximate PFs: the hypervolume (HV) [16]. The hypervolume (or S-volume) is widely used in multi-objective optimization as we can compute it without knowing the optimal PF. We suppose, without loss of generality, that we want to maximize objective function f_1 and minimize objective function f_2. The hypervolume requires two reference points in order to be computed: it is important to use the same reference points when we compare two approximate fronts. For a set of approximate fronts, the references points called nadir and ideal are defined as $nadir = (f_1^{min}, f_2^{max})$ and $ideal = (f_1^{max}, f_2^{min})$, where f_i^{min} and f_i^{max}, $i = 1, 2$ refer to the minimum and maximum values of objective functions f_1 and f_2 encountered in the set of approximate fronts. Let $\Lambda(a_i)$ be the size of the rectangular area a_i constructed with a solution s_i from an approximation set A and the nadir as corners. For approximate set A, we compute the HV as follows:

[1] See https://www.hds.utc.fr/p̃enaquen/dokuwiki/doku.php for the detailed instances and pre-disruption routes.

$$HV(A) = \frac{\Lambda\left(\bigcup_{a_i \in A} a_i\right)}{(f_1^{max} - f_1^{min})(f_2^{max} - f_2^{min})} \tag{15}$$

4.1 Mutation rate tuning

In this section, we want to test the influence of the mutation rate μ on the output of our algorithm. We launched the algorithm with a time limit of 60 seconds on all our benchmark instances with 30, 40, 50 and 60 assets, with two different vehicle breakdowns as the disruption.

Based on preliminary tuning work, we used fixed values for some of our parameters. The population size is set to $N = 100$. We use the time crossover operator as crossover operator and multi-change operator as mutation operator. For the choice criteria w_i^- and w_i^+, the parameters are set to $\alpha = 1.0$, $\beta = 0.5$ and $\gamma = 1.0$. Destruction and construction parameters c_{max} and d_{max} are initially set to 3. The initial population is generated by applying the multi-change operator with high c_{max} and d_{max} values on the solution representing the initial routes.

We report in Figure 1 the average gap between the hypervolume of the non-dominated front $\mathcal{F}_1$ obtained with each mutation rate and the hypervolume of the best known Pareto front.

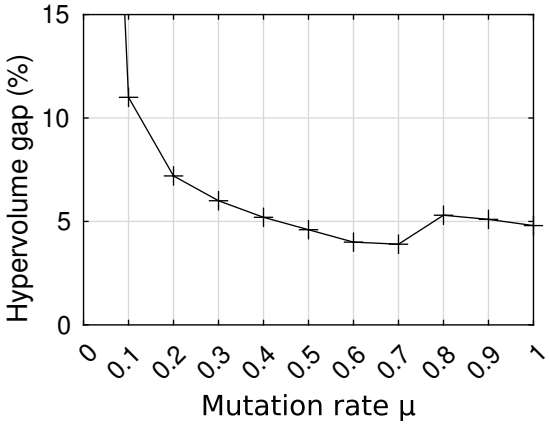

Fig. 1. Average gap between the front obtained by NSGA-II and best known front, based on mutation rate μ.

We can see that our mutation operator impacted the quality of the front we generated. We obtained the worst results when the mutation operator was disabled ($\mu = 0$), with a gap superior to 58%. The gap decreased when μ increased up to 0.6, from 11 to 4% on average. The gap stabilized for $\mu = 0.7$ and deteriorated around 5% for higher mutation rates.

4.2 Performance analysis

In this section, we test the influence of our evaluation models and operators, and the impact of our additional components. Following preliminary work, we chose not to consider the valid crossover operator (CXVAL). Hence, we will only present results using the time crossover operator (CXTIM). We will first compare the results obtained using our two different evaluation models, with our single-change operators and our multi-change operator. Then, we will evaluate the impact of the adaptive scheme and the local search procedure we presented.

We launched our NSGA-II algorithm with the different sets of operators for each of our ten benchmark instances with 30, 40, 50 and 60 assets, and two different random vehicle breakdowns as the disruption. We used the parameters presented in Section 4.1, and set the mutation rate $\mu = 0.6$.

Table 1 shows the results of NSGA-II within a time limit of 300 seconds. For each evaluation model (shown in row "Eval.") and operator (shown in row "Operator") combination, we give the average hypervolume of the non-dominated front $\mathcal{F}_1$ we obtained. In the last column "$\epsilon - 300$", we indicate the hypervolume of the front obtained using the ϵ-constraint method with the model introduced in [13], with a 300-second time limit. Due to the time limit, this method does not always yield the full optimal Pareto front.

Table 1. Average hypervolume of fronts obtained by NSGA-II in 300 seconds

Eval.	**Asset penalization**		**Assignment penalization**		$\epsilon - 300$
Operator	**Single-change**	**Multi-change**	**Single-change**	**Multi-change**	
n = 30	73.2 %	78.0 %	74.1 %	77.7 %	82.3 %
40	70.2 %	73.0 %	70.5 %	73.4 %	78.5 %
50	67.4 %	70.1 %	69.1 %	71.9 %	68.7 %
60	66.2 %	65.9 %	68.0 %	67.4 %	54.4 %

We can see that the second evaluation model on average gave fronts with higher hypervolume on average than the first model for the same operators. For instances with 30, 40 and 50 assets, the multi-change operator performed better than the single-change operators. The multi-change operator offers more stable results than the single-change operators, and find solutions with higher profit. For larger instances, we obtain better fronts on average than the $\epsilon - constraint$ method within the same time limit.

Based on Table 1, we will consider the second evaluation model with multi-change operator to evaluate our additional components. We performed a parameter analysis similar to Section 4.1 to determine good values for our adaptive method and local search parameters. We set the initial population to $N = 50$, and offspring population to $M = 50/4$. For the local search, we apply it to 5% of the solutions, with a relative gap tolerance of 0.05. Each model is run five times on each instance, to ensure the robustness of our results.

Table 2 summarizes the results of our algorithm with no additional component, with the adaptive parameters enabled and our local search procedure. It shows the average value of the hypervolume found in the best run (HV_{max}) and the average value in all the runs (HV_{avg}).

Table 2. Comparison of the components of our NSGA-II implementation

Method	**No component**		**Adaptive**		**Local Search**		$\epsilon - 300$
	HV_{max}	HV_{avg}	HV_{max}	HV_{avg}	HV_{max}	HV_{avg}	HV
n = 30	79.5%	77.6%	79.8%	77.8%	82.3%	81.9%	82.3%
40	75.8%	73.5%	75.8%	73.6%	79.9%	79.2%	79.1%
50	74.5%	72.2%	75.1%	73.1%	80.7%	79.5%	68.9%
60	70.5%	67.8%	73.0%	69.7%	80.0%	78.3%	56.4%

The adaptive component yielded similar results for instances with 30 and 40 assets and slightly better results for 50 and 60 assets when enabled. We obtained significant improvements for all instances when enabling our local search procedure, up 10% for instances with 60 assets on average. The local search procedure also improved the stability of our algorithm, reducing the gap between the best solution and the average solution for all size of instances.

5 Conclusion

NSGA-II is a popular algorithm for multi-objective heuristic resolution that has proven efficient for multiple vehicle routing problems. We proposed an implementation of NSGA-II for the D-APP. Due to the numerous constraints that are part of the D-APP, we introduced mutation and crossover operators based on properties of the problem and MIPs to repair and evaluate solutions. It is the first heuristic solution method dedicated to the D-APP. The approach can be improved by defining operators that better take the deviation into account, and finding faster procedures to repair and evaluate a solution.

Acknowledgment

This study was carried out within the framework of GEOSAFE (Geospatial Based Environment For Optimization Systems Addressing Fire Emergencies). This work was partially supported by the framework of the Labex MS2T, funded by the French Government, via the program Investments for the future managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02).

References

1. A. Ben-Said, A. Moukrim, R.N. Guibadj and J. Verny, “Using decomposition-based multi-objective algorithm to solve Selective Pickup and Delivery Problems

with Time Windows," *Computers & Operations Research*, Volume 145, 2022, doi: /10.1016/j.cor.2022.105867.
2. J. Cheng, G. Zhang, Z. Li and Y. Li, "Multi-objective ant colony optimization based on decomposition for bi-objective traveling salesman problems," *Soft Comput*, 16, 597614, 2012, doi: 10.1007/s00500-011-0759-3
3. K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 2, pp. 182-197, April 2002, doi: 10.1109/4235.996017.
4. J. Jemai, M. Zekri and K. Mellouli, "An NSGA-II Algorithm for the Green Vehicle Routing Problem," In: Hao, JK., Middendorf, M. (eds) Evolutionary Computation in Combinatorial Optimization. EvoCOP 2012. Lecture Notes in Computer Science, vol 7245. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-642-29124-1_4
5. N. Jozefowiez, F. Semet and EG. Talbi, "Enhancements of NSGA II and Its Application to the Vehicle Routing Problem with Route Balancing," In: Talbi, EG., Liardet, P., Collet, P., Lutton, E., Schoenauer, M. (eds) Artificial Evolution. EA 2005. Lecture Notes in Computer Science, vol 3871. Springer, Berlin, Heidelberg. doi: 10.1007/11740698_12
6. N. Jozefowiez, F. Semet, EG. Talbi, "Multi-objective vehicle routing problems," *European Journal of Operational Research*, vol. 189, Issue 2, pp. 293-309, 2008, ISSN 0377-2217, doi:10.1016/j.ejor.2007.05.055.
7. A. Mas-Colell, M.D. Whinston, J.R. Green and others. "Microeconomic theory;" *New York: Oxford University Press.*, vol. 1, 1995.
8. M. Merwe, "An optimisation approach for assigning resources to defensive tasks during wildfires," [Ph.D.thesis], 2015, RMIT University. URL researchbank.rmit.edu.au/view/rmit:161622
9. M. Merwe, J. Minas, M. Ozlen and J. Hearne, "A mixed integer programming approach for asset protection during escaped wildfires," *Canadian Journal of Forest Research*, 45, 04 2015, doi: 10.1139/cjfr-2014-0239.
10. M. Merwe, M. Ozlen, J. Hearne, and J. Minas, "Dynamic rerouting of vehicles during cooperative wildfire response operations," *Annals of Operations Research*, 254, 07 2017, doi: 10.1007/s10479-017-2473-8.
11. M. H. Mirzaei, K. Ziarati and M.-T. Naghibi, "Bi-objective version of team orienteering problem (BTOP)," *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)*, 2017, pp. 1-7, doi: 10.1109/ICCKE.2017.8167930.
12. V. Pareto, "Manual of Political Economy," *New York: Augustus M. Kelley Publishers*, 1971.
13. Q. Peña, M. Serairi, A. Moukrim, "Reformulation and valid inequalities for the dynamic asset protection problem during an escaped wildfire," *submitted for publication*, 2022.
14. M. Schilde, K. Doerner, R. Hartl and G. Kiechle, "Metaheuristics for the bi-objective orienteering problem," *Swarm Intelligence*, 3, 179-201, 2009. doi: 10.1007/s11721-009-0029-5.
15. E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms : a comparative case study," In: Eiben, A.E., Bck, T., Schoenauer, M., Schwefel, HP. (eds) Parallel Problem Solving from Nature PPSN V. PPSN 1998. Lecture Notes in Computer Science, vol 1498. Springer, Berlin, Heidelberg. doi: 10.1007/BFb0056872
16. E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, Nov. 1999, doi: 10.1109/4235.797969.