A hybrid modified-NSGA-II VNS algorithm for the Multi-Objective Critical Disruption Path Problem

Donatella Granata*[2,3] and Antonino Sgalambro†[1,2]

[1] *University of Sheffield, Management School, Conduit Road S10 1FL, Sheffield, United Kingdom*

[2] *Istituto per le Applicazioni del Calcolo "Mauro Picone", National Research Council, via dei Taurini 19, 00185 Rome, Italy*

[3] *Dipartimento di Scienze Matematiche, Fisiche e Informatiche, Plesso di Matematica, Università degli Studi di Parma, Parco Area delle Scienze 53/A, 43124 Parma, Italy*

Abstract

This paper considers a Multiple Objective variant of the Critical Disruption Path problem to extend its suitability in a range of security operations relying on path-based network interdiction, including flight pattern optimisation for surveillance. Given a pair of nodes s and t from the network to be monitored, the problem seeks for loopless $s-t$ paths such that, within the induced subgraph obtained via deletion of the path, the size of the largest connected component is minimised, the number of connected components is maximised, while concurrently reducing as much as possible the cost of such disruption path. These three objectives are possibly in conflict with each other, and the scope of this work is to allow for an efficient and insightful approximation of the Pareto front, looking for a trade-off between costs and effectiveness to secure the most convenient paths for security and surveillance operations. We first introduce and formulate the Multi-Objective Critical Disruption Path Problem (Multi-Objs-CDP) as a mixed integer programming formulation (MO-CDP), then we propose an original evolutionary metaheuristic algorithm hybridising modified-NSGA-II and VNS for finding an approximation of the Pareto front, as well as a procedure securing the efficient generation of a high quality pool of initial solutions. The experimental performance of the proposed algorithm, as compared with a variety of competing approaches, proves to be fully satisfactory in terms of time efficiency and quality of the solutions obtained on a set of medium to large benchmark instances.

Keywords: Networks; Critical Disruption Path; Mixed Integer Programming; Multiple Objective Optimisation; Metaheuristics.

*email: donatella.granata@cnr.it

†Corresponding author: a.sgalambro@sheffield.ac.uk, antonino.sgalambro@cnr.it

1 Introduction: the role of critical disruption paths in network surveillance and security

Many real-world problems arising in safety and security can be accurately represented through networks, enabling the adoption of optimisation methods to support better decision making, also in presence of multiple and often conflicting criteria. Network modeling is typically utilised to predict and evaluate the behaviour of a system, to identify the most vital (also called vulnerable or critical) components of a network, via measuring the rules governing individual nodes, arcs, or associated substructures, whose failure will prevent the functionality of the network as a whole. In the last decades a growing scientific attention has been devoted towards a specific class of problem, referred to as Network Interdiction, where one is concerned with studying those disruption interventions which are expected to induce a residual network as damaged as possible. Work in the field of network interdiction dates back to [23] and, over the years, network interdiction models have been increasingly applied in many different areas [6, 32, 42, 53, 54]. These problems are relevant from both a protection and an interdiction perspective. In the latter case, the interdiction is an attack to the network that leaves the network fragmented or disconnected and the interdictor chooses his optimal strategy to attack a given network, or a part of it. In the former, the defender identifies which network components are the most critical ones for maintaining the integrity of the network, and therefore should be protected or reinforced. Network interdiction problems, besides focusing on scenarios that cause the network to become non-operational after the failure of some nodes or arcs, are also aimed at measuring the network's communication capability or level of degradation [1, 24, 35]. The definition of network failure varies, but it typically involves either nodes or arcs or specific topological substructure that may fail under certain conditions or specific attacks. It is often advisable to evaluate the number of necessary disruptive events to experience given levels of disruption in the target network, in order to assess its vulnerability. This led to studies on how to lower the overall pairwise connectivity as a measure of the network performance [11]. Furthermore, examples of arc removal interdiction models have been introduced in [27, 38], whereas interdiction models based on node removal have been broadly studied by [3, 4, 8, 29, 40, 43, 48–50, 52, 58], and the shortest path interdiction problem has been considered in [27]. In other variants of interdiction problems, the aim of nodes and/or arcs removal is to get minimum weight on specific structural and topological properties in

the remaining graph [42], including cliques [17, 33] and maximum matching [56]. When planning complex security operations, an insightful analysis and assessment of network's vulnerability and control is sometimes best achieved by seeking those loopless paths in the network whose removal maximally impedes network operability or maximises its disruption. These are referred to as Critical Disruption Paths (CDPs) and were firstly introduced in [21]. CDP's applications arise for instance when defining optimal patterns for surveillance purposes, thus including: flight patterns for military surveillance and reconnaissance missions, realised through remotely piloted or traditional aircraft, optimal pedestrian or cycling paths for surveillance and control workforce.
With the aim of enhancing the outcome of the surveillance activity for network control purposes, different measures have been adopted and investigated as an objective for the CDP problem, including:

- minimising the size of the largest connected component in the induced subgraph obtained via deletion of the CDP: this goal will lead to containing the residual risk of adverse activities taking place in the residual network after the surveillance intervention [21];

- maximising the number of connected components in the induced subgraph obtained via deletion of the CDP: this goal is aimed at decreasing the connectivity of isolated components after the disruptive action [20].

A preliminary bi-objective study presented in [20] showed how a combined use of both objective functions can yield a range of non-dominated solutions to choose from, while evaluating the most appropriate intervention to be implemented for practical purposes. Those experiments also revealed how the cost allowed for the critical path impacts on the trade-off between such different and often conflicting goals, thus revealing the presence of a prominent research gap: how to design an efficient algorithm to approximate the Pareto front for multiple-objective variants of the CDP with conflicting relevant objectives. Furthermore, the cost of the CDPs clearly reveals a feature to be considered as a major decision making driver, as it influences both: the capability to implement surveillance operations within given limitations in times, budget and vehicle endurance, and the chance of finding proper trade-off between conflicting network disruption measures. In this work we contribute to bridge these gaps, as follows.
First, we introduce the Multiple Objective Critical Disruption Path problem where, given a pair of nodes s and t and the network to be monitored

with weights associated to arcs, one seeks for the loopless $s-t$ paths such that, within the induced subgraph obtained via deletion of the path, the size of the largest connected component is minimised, the number of connected components is maximised, and the cost of such a disruption path is concurrently minimised. The three considered objectives are in conflict with each other, and the scope of this work is to foster an efficient and insightful approximation of the Pareto front of this optimisation problem, looking for the trade-off between costs and effectiveness in the solutions, thus supporting the decision maker at identifying the most suitable paths for security and surveillance operations.

Secondly, we propose an original evolutionary metaheuristic algorithm which hybridises modified-NSGA-II and VNS for approximating the Pareto front of the considered Multi-objective Critical Disruption Path problem. To the best of our knowledge, this is overall the first solution approach proposed for a multiple-objective CDP, and its performance is compared to a variety of rigorous competing approaches, again proposed and implemented in this work.

As a further contribution, we propose an original polynomial time procedure aimed at generating a pool of tailored feasible solutions by identifying in polynomial time nodes which cannot belong to any CDP, thus at complementing and boosting the performance of the metaheuristic scheme. The experimental performance of the proposed algorithm proves to be fully satisfactory in terms of time efficiency and quality of the solutions obtained on a set of medium-to-large benchmark instances. The considered testbed is comprehensive and large enough to check and secure the scalability of the proposed method on any realistic size application. The remainder of our paper is organised as follows. In Section 2 we introduce formally the Multi-Objective Critical Disruption Path problem (Multi-Objs-CDP) providing a mixed integer programming formulation (MO-CDP). In Section 3 we discuss algorithmic strategies to restrict the search for Pareto efficient solutions as the generation of an initial pool of solutions. In section 4 we present our Hybrid Multi-Objective Modified-NSGA-II Variable Neighborhood Search (MO-NSGA-VNS) approach to solve efficiently the Multi-Objs-CDP problem. The experimental performance of the proposed MO-NSGA-VNS algorithm is presented in Section 5 on a set of medium to large benchmark instances, as compared to a variety of competing approaches, including two further NSGA-based algorithms and a scalarisation technique implemented to calculate non-dominated solutions by applying an off-the-shelf solver to the MO-CDP MIP model resolution. Some final remarks and further research avenues conclude the paper.

2 Problem statement

In this section we first introduce the Multi-Objective Critical Disruption Path Problem (Multi-Objs-CDP) and its Mixed-Integer formulation (MO-CDP). The Multi-Objective Critical Disruption Path Problem (Multi-Objs-CDP) proposed in this paper is stated as follows.
We are given a directed graph $\mathcal{G} = (V, E, w, s, t)$ with node and arc sets V and E of size n and m respectively and two special nodes: a *source* node $s \in V$ and a *destination* node $t \in V$. We assume w.l.o.g. that if $(i, j) \in E$ then also $(j, i) \in E$, and w_{ij} is the weight assigned to arc $(i, j) \in E$. Given any path ρ, we refer to $\mathcal{G}^\rho := (V^\rho, E^\rho)$ as the induced subgraph obtained via deletion of ρ from $\mathcal{G}$, $V^\rho := V \setminus V(\rho)$, $E^\rho := E \cap (V^\rho \times V^\rho)$, being $V(\rho)$ the subset of nodes of V which are included in path ρ.

Definition 2.1. The Multi-Objective Critical Disruption Path Problem (Multi-Objs-CDP) is defined as the problem of finding a simple loopless path ρ from s to t such that the following objectives are pursued:

- minimise the size of the largest connected component in $\mathcal{G}^\rho$;
- maximise the number of connected component in $\mathcal{G}^\rho$;
- minimise the cost of the path ρ.

The Multi-Objective Critical Disruption Path can be now formulated through the following multiple objective Mixed Integer Programming model.

We term this model as MO-CDP.

$$
\begin{aligned}
\text{MO-CDP}: f_1 &:= \min \psi \\
f_2 &:= \max \sum_{i \in V \setminus \{s,t\}} d_i \\
f_3 &:= \min \sum_{i,j:(i,j)\in E} w_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in E} x_{ij} = \sum_{j:(j,i)\in E} x_{ji} \qquad \forall i \in V \setminus \{s,t\} && (1)\\
& \sum_{i:(s,i)\in E} x_{si} = 1 && (2)\\
& \sum_{i:(i,t)\in E} x_{it} = 1 && (3)\\
& x_{ij} + x_{ji} \leq 1 \qquad \forall (i,j) \in E,\ j > i && (4)\\
& \sum_{i,j \in S:\ (i,j)\in E} x_{ij} \leq |S| - 1 \qquad \forall S \subset V, \quad |S| \geq 2 && (5)\\
& y_{ii} = 1 - \sum_{l:(l,i)\in E} x_{li} \qquad \forall i \in V \setminus \{s,t\} && (6)\\
& y_{ij} \geq y_{ih} - \sum_{l:(l,j)\in E} x_{lj} \\
& \qquad\qquad \forall h,i,j \in V \setminus \{s,t\} :\ j \neq i,\ (h,j) \in E && (7)\\
& \psi \geq \sum_{j \in V\setminus\{s,t\},\ j \geq i} y_{ij} \qquad \forall i \in V \setminus \{s,t\} && (8)\\
& n \cdot d_i \leq n \cdot y_{ii} - \sum_{j \in V\setminus\{s,t\}: j<i} y_{ji} \qquad \forall i \in V \setminus \{s,t\} && (9)\\
& \psi \geq 0 && (10)\\
& d_i \in \{0,1\} \qquad \forall i \in V \setminus \{s,t\} && (11)\\
& x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E && (12)\\
& y_{ij} \in \{0,1\} \qquad \forall i,j \in V \setminus \{s,t\} && (13)
\end{aligned}
$$

Among all the $s-t$ paths, here we are concurrently looking for the shortest ones (as required by the third objective function f_3) and the most disruptive ones, such that the arising connected components in the residual graph after the path removal present the largest amount of connected components with reduced size: this is obtained by including f_2 and f_1 as objective functions, respectively. The proposed model is an arc-based formulation requiring the selection of one simple $s-t$ path in the network, in the following indicated as Multi-Objs-CDP. The binary variables x_{ij} encode the decision on the choice

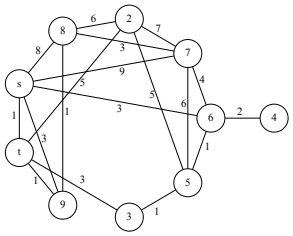
of the CDP, and the binary decision variables y_{ij} take value 1 if and only if both nodes $i \in V$ and $j \in V$ belong to the same connected component after path extraction and 0 otherwise; y_{ii} has value 1 if node $i \in V$ is not belonging to Multi-Objs-CDP path and 0 otherwise. A nonnegative variable ψ is used to indicate the cardinality of the largest connected component arising in the residual graph. Binary variables d_i are used to count the number of connected components: as it is necessary to have a member of each component in the residual graph to represent that component, we use the node with the highest index as such one representative, hence d_i equals 1 if and only if node i is the rightful component representative, 0 otherwise. Constraints (1)-(5) ensure that variables x_{ij} identify a simple $s-t$ path, where each path node has one associated arc in and out as verified by the balance constraints (1). The complexity of sub-tour elimination constraints (5) is reduced by a separation mechanism and by constraints (4). Constraints (6) and (7) identify the connected components in the network: where two nodes $j \in V$ and $i \in V$ are forced to be in the same connected component, if j does not belong to the selected path and there exists a further node $h \in V$ connected via an arc to node j into the same connected component. Constraints (8) assign the largest connected component size value to variable ψ, whereas constraints (9) count the number of connected components. It is worth recalling how the CDP detection problem was proven in [21] to be NP-complete by reduction from the Hamiltonian Path problem, thus characterising also the class of complexity of the Multi-Objs-CDP problem proposed in this paper.

2.1 Relevance of the MO-CDP for application purposes

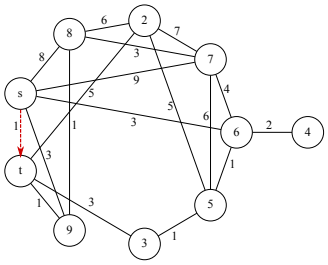
The enhanced potential for application purposes unleashed by the multiple objective model above introduced, as compared to the single objective variant of the CDP, can be better understood by observing the set of non-dominated solutions obtained on the small example network presented in Figure 1. In this toy example we are given a graph $G = (V, E, w, s, t)$ with nodes $V = \{s, t, 2, 3, 4, 5, 6, 7, 8, 9\}$ and 27 arcs, and two special nodes: a *source* node $s \in V$ and a *destination* node $t \in V$. The weight w_{ij} assigned to each arc $(i, j) \in E$ is shown along the arc in the figure. Multi-Objs-CDP efficient solutions for the graph example a) are presented in the following subfigures {b),d)}, {c),e)} ,{f),j)}, {g),k)},{h),l)}, {i),m)}, where each subfigure couple shows on the left the s-t CDP path with the corresponding values of the three objective functions (f_1, f_2, f_3) and on the right a representation of the remaining connected components into residual graph $\mathcal{G}_i^{\rho}$

after the removal of CDP ρ_i as depicted into subfigure i). We recall that the objective function f_1 minimises the size of the largest connected component in $\mathcal{G}^\rho$, f_2 maximises the number of connected component in $\mathcal{G}^\rho$ and f_3 minimises the cost of the path ρ. Let us assume that the CDP is utilised here to define the best flight pattern for an unmanned aerial vehicle (UAV) aimed at automated aerial surveillance. The single objective CDP would simply suggest *h)* as a flight pattern, as a way to minimise the size of the largest component in the residual network *l)*. Such a solution might require a surveillance pattern exceeding (or not) the endurance of the adopted UAV, depending on the amount of budget invested in purchasing the fleet for security operations. By using the MO-CDP model it is possible to explore the progressive growth in the surveillance quality while allowing increasing cost for surveillance paths, thus showcasing all the trade-offs between costs and solutions quality. Computing and comparing solutions which approximate the Pareto front of the MO-CDP will allow to appreciate, for each given level of intervention cost, the associated expected impact on security and surveillance operations, thus informing accurately the decision making process to identify the most convenient level of investment.

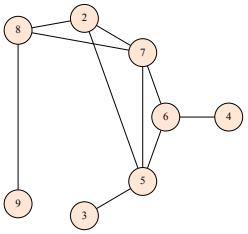
Figure 1: Pareto front for a toy example.



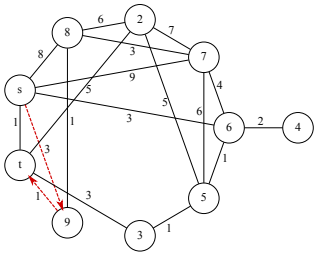
a)



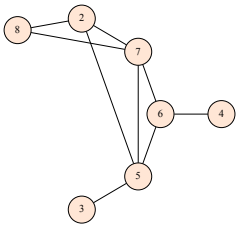
b) $\rho_b = \{s, t\}$,
$f_1 = 8, f_2 = 1, f_3 = 1$



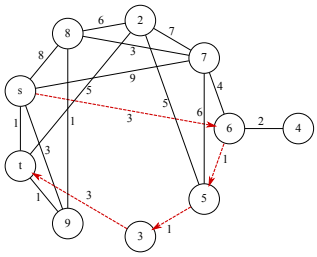
d)

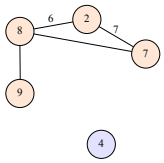


c) $\rho_c = \{s, 9, t\}$,
$f_1 = 7, f_2 = 1, f_3 = 4$



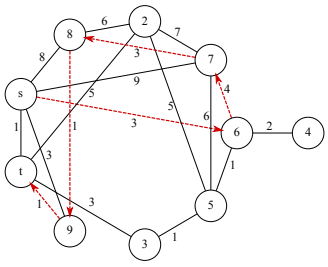
e)

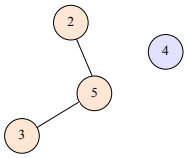




j)

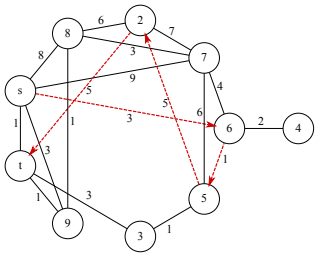
f) $\rho_d = \{s, 6, 5, 3, t\}$,
$f_1 = 4, f_2 = 2, f_3 = 8$

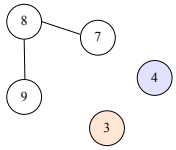




k)

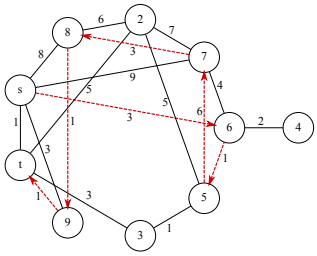
g) $\rho_h = \{s, 6, 7, 8, 9, t\}$,
$f_1 = 3, f_2 = 2, f_3 = 12$

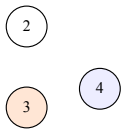




l)

h) $\rho_i = \{s, 6, 5, 2, t\}$,
$f_1 = 3, f_2 = 3, f_3 = 14$





m)

i) $\rho_j = \{s, 6, 5, 7, 8, 9, t\}$,
$f_1 = 1, f_2 = 3, f_3 = 15$

3 Generating an initial set of feasible solutions

With the twofold goal of generating a pool of good initial solutions and increase the efficiency in computing high quality solutions, we are interested in identifying quickly those nodes which result in suitable candidates for inclusion in a CDP, and in excluding all the others from our search process. Our approach is underpinned by the preliminary observation that a node which cannot be included in any simple $s-t$ path, can be excluded while seeking for feasible solutions. In the following we refer to these as unreachable nodes. In order to find such nodes, we design a polynomial-time pair node disjoint algorithm, denoted as *Pool-Init-Gen*, which draws upon a variant of the procedure presented in [45] and is applied on a modified network. In this section, we first briefly recall some major results from the literature on disjoint paths, then we describe the approach adopted in this paper for generating an initial pool of feasible solutions.

Finding disjoint paths: variants and complexity. The term k shortest disjoint paths is plainly interpreted as follows: given an undirected graph $G = (V, E)$ and k distinct pairs of nodes $(s_1, t_1), \ldots, (s_k, t_k)$, the objective is finding whether there exist k pairwise disjoint paths $P_1, \ldots, P_k$ such that P_i is a path from s_i to t_i, for every $1 \leq i \leq k$. One may consider several variants: directed or undirected, node or arc disjoint. The node disjoint path problem was shown to be NP-hard by Li et al. [30]. Fortune et al. [15] proved that the directed version is NP-hard even if $k = 2$. Shiloach [41] presented a a linear $O(n \cdot m)$ algorithm that, given an undirected graph $G = (V, E)$ and nodes s_1, s_2, t_1, t_2, determines whether or not G admits two node disjoint paths, one connecting s_1 to t_1 and the other one s_2 to t_2. Eilam-Tzoreff [12] proved that directed or undirected and node or arc disjoint path problems are also NP-complete for arbitrary values of k even for planar graphs with unit arc-costs. But the author actually provided a polynomial algorithm for the case of $k = 2$ with positive arc-costs. Furthermore, in [47] the problem of finding a pair of length-bounded disjoint paths between nodes s and t of an undirected graph was proven NP-complete. The problem of finding two disjoint paths from s to t such that the length of the longer path is minimised was proven to be NP-complete on directed and undirected graphs[31]. Whenever the min-sum disjoint path variant is considered, namely where k disjoint paths with the total cost to be minimised are to be found, the problem is known to be polynomially solvable [44, 45]. Suurballe and Tarjan [45] proved that given a directed graph $G = (V, E)$ with m arcs and with non negative weight assigned to each arc, finding a pair

of shortest node/arc disjoint paths from s to a single sink t can be obtained in $O(m \cdot log_{(1+m/n)}n)$ time. More results on the complexity of finding disjoint paths can be found in [16, 51]. A recent heuristic for the computation of node disjoint path pair for any set of at least two intermediate nodes has been presented in [34], which can be used on undirected and directed symmetric graphs. The paper also succinctly describes a procedure to obtain a min weight path visiting a specific node in undirected networks. In the next section, we propose and detail a novel algorithm which can be adopted on any graph. This routine is based on Suurballe's approach and considers one single intermediate node. Its worst-case complexity is bounded by $O(m + n \cdot log(n))$ which stems from shortest path tree calculation.

3.1 *Pool-Init-Gen* algorithm construction

To generate a pool of good initial solutions and to timely identify those unreachable nodes which cannot belong to any CDP, we design a polynomial-time algorithm, namely *Pool-Init-Gen*, able to find a min-hop $s-t$ path passing through a specific node i, by solving via an auxiliary modified network. This algorithm draws inspiration from [45], where finding a shortest $s-t$ pair of arc/node disjoint paths is used as a minimum-cost flow problem via Dijkstra algorithm. Its adaptation to this particular case is similar to the approach detailed in the introductory section of [34].

The first step of *Pool-Init-Gen* algorithm consists in a building phase, to produce an auxiliary graph $G' = (V', E', s, t)$ from the original input graph $G = (V, E, s, t)$, which contains $|V'| = 2 \cdot |V - 2| + 2$ nodes and $|E'| = |V - 2| + |E|$. This building phase of G' incorporates the following finite steps:

- Each node $v \in V \setminus \{s, t\}$ is split in a pair of nodes denoted as v and v'.
- An arc $e = (v, v')$ is created connecting each couple of split nodes (v, v').
- All the outcoming arcs of v are moved to be outcoming arcs of v'.
- A weight $w(u, v) = 1$ is assigned to each arc $(u, v) \in G'$.

The unitary weight value assigned in the last step to each arc in the auxiliary graph is instrumental at enabling the search for a min-hop path in original path. The effort of our algorithm is based on the idea that finding path p from s to t passing through a node i in the original graph G means

looking for a shortest pair of node-disjoint paths, one from s to i and one from i' to t in the auxiliary graph G'. The *Pool-Init-Gen* involves these polynomial operations:

1. Find the shortest path tree $SPT(s)$ rooted at node s by running the Dijkstra's algorithm on the graph G', and let call p_1 the found shortest path $(s \rightsquigarrow i)$ from source s to i.

2. Modify the weight of each arc (u, v) in the graph by replacing its weight $w(u, v)$ by $w'(u, v) = w(u, v) - d(s, v) + d(s, u)$, being $d(i, j)$ the distance between any nodes i and j.

3. Create a residual graph G'^{p_1} formed from G' by reversing the direction of the zero weight arcs along path p_1 and by removing the arcs that are directed into s and, for all arcs $(u, v) \in |E| : v \in V(p_1) \setminus \{s, i\}$, set the weight $w(u, v) = |E|$.

4. Find the shortest path $p_2 = (i' \rightsquigarrow t)$ from the node i' to the sink t in the residual graph G'^{p_1} by running Dijkstra's algorithm. For each arc $(u, v) \in p_2$ remove arc (v, u), if it exists, from both paths p_1 and p_2.

5. Construct the $s - t$ path p concatenating p_1 and p_2.

6. Modify p by shrinking any node splits previously operated during the building phase, thus preserving only those nodes and arcs which were originally part of the original graph G.

An example of auxiliary graph construction is presented in Figure 2.b) applied to the dummy graph example represented in Figure 2.a). So, Figures c)–f) depict the algorithm'steps in finding the path between the node s and t that passes through the node 1; both steps 1-2 of *Pool-Init-Gen* algorithm are depicted in Figure 2.c), step 3 in Figure 2.d), step 4 in Figure 2.e) and steps 5-6 in Figure 2.f).

Claim 1. If the cost of path p_2, defined as $W(p_2) = \sum_{(u,v)\in E(p_2)} w(u, v)$, is greater than or equal to $|E|$, then there exist no $s-t$ path p passing through a given node i.

Proof. Seeking for the path p_2, during the fourth step of the *Pool-Init-Gen* algorithm, means seeking for a simple path from i' to t in the residual graph G'^{p1}, produced by step 3. In G'^{p1}, all the arcs in $SPT(s)$ have weights equal to 0, and there are some arcs whose weight equals $|E|$, namely, those arcs entering any node v belonging to $V(p_1)\setminus\{s, i\}$ (by algorithmic construction).

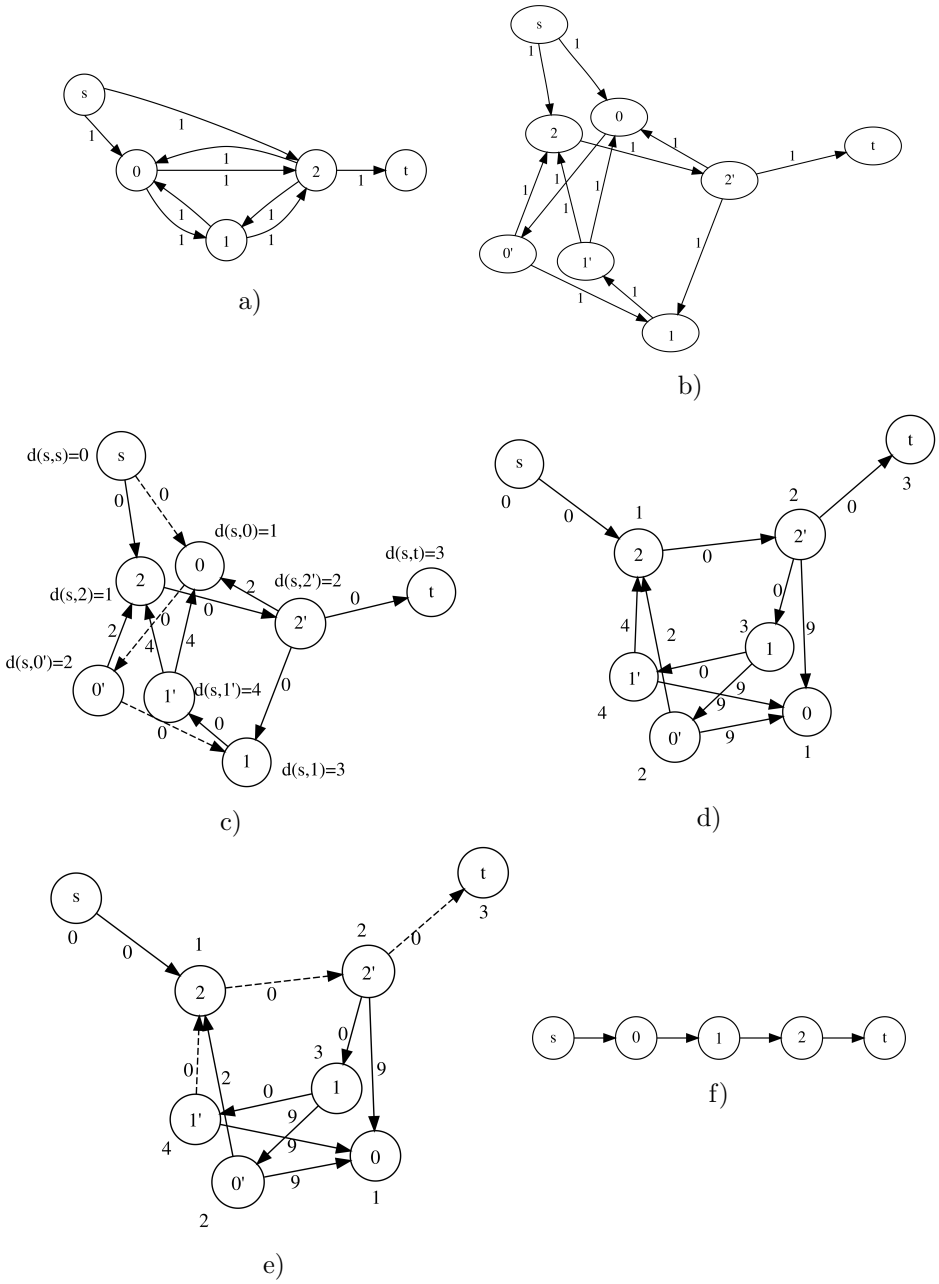
If an arc (u, v) with $w(u, v) > 0$ is included in p_2, this arc does not belong to the shortest path tree $SPT(s)$, because all the arcs in $SPT(s)$ have weights equal to 0. Furthermore, we can assert that the distance $d(i', v)$ from i' to v is bounded from above by $d(i', u) + w(u, v)$, namely, it holds $d(i', v) \leq d(i', u) + w(u, v)$, otherwise there would exist a shorter path to v via u, such that $d(i', u) - d(i', v) + w(u, v)$ is non negative and less than the longest path, which in turn can be at most equal to $n - 1$. Suppose for absurd, that $w(u, v) = |E|$: this would imply that during the construction of p_2 via Dijkstra's algorithm, one arc incident on $V(p_1) \setminus \{s, i\}$ has been included into the path p_2, but this means that the final path p is not a feasible $s - t$ path solution for the graph G, and it is not passing through the node i, so a contradiction follows. □

Furthermore, it is easy to show that G' contains a pair of arc-disjoint paths p_1 and p_2, the former from s to i and the latter from i' to t, if and only if the original graph contains a path from s to t through node i.

3.2 Generating an initial population

Instead of simply using any preliminary random population generator as it often happens for genetic algorithms, we adopt iteratively the procedure presented above, with the scope of generating tailored initial solutions, taking into account the main features of the Multi-Objs-CDP, thus also securing increased chances to include non-dominated solutions in the starting population P_0. Here we explain how the *Pool-Init-Gen* algorithm utilised the Algorithm 3.1) to generate the initial population P_0. The algorithm $Init\_phase$ is fed with a weighted directed graph $\mathcal{G} = (V, E, w, s, t)$ with source s and destination t and returns as an output two sets: $UN \subset V$ and ND, filled with unreachable nodes and with a first pool of non-dominated solutions, respectively, with respect to the set of feasible solutions currently considered. $Init\_phase$ works by initially exploiting an implementation of the Dijkstra's algorithm for generating as its first solution a path which becomes in turn the first solution to be included in the set of non-dominated solutions ND (lines 1- 2). The procedure *Pool-Init-Gen* is repeatedly executed upon each node i other than s and t (as detailed in Algorithm 1, lines 3-13). If a feasible solution is identified, the path p is included in the set ND, whereas if a feasible solution cannot be identified, the node i is labelled as unreachable and added to UN, as it is not possible to find any simple path from s to t through i.

Figure 2: Example of construction of shortest pair of node-disjoint paths s to 1 and from $1'$ to t.



Algorithm 1 Init_phase$(\mathbf{G(V, E, w, s, t)}, \mathbf{ND}, \mathbf{UN})$

1: $p \leftarrow Dijkstra(G(V, E, w, s, t))$
2: $ND \leftarrow p$ {set of non-dominated solutions }
3: **for all** $i \in V \setminus \{s, t\}$ **do**
4: **if** $degree[i] = 2$ and arcs $(i, k), (k, i)$ exist **then**
5: $ADD(UN, i)$
6: **end if**
7: Use ***Pool-Init-Gen*** algorithm for finding a path p passing through node i
8: **if** p is not a feasible solution **then**
9: $ADD(UN, i)$
10: **else**
11: $ADD(ND, p)$
12: **end if**
13: **end for**
14: **return** ND, UN

4 Hybrid Multi-Objective Modified-NSGA-II Variable Neighborhood Search (MO-NSGA-VNS)

In this section we present the original algorithmic procedure we designed *ad-hoc* to approximate the Pareto front for large size instances of the Multi-Objs-CDP problem. The adoption of bio-inspired search paradigms has proved to be effective in solving many multi-objective optimisation problems [14, 60], as they are able to find multiple solutions simultaneously in a single execution. Methods such as Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [10], Non-dominated Sorting Genetic Algorithm-III (NSGA-III) [9], Strength Pareto Evolutionary Algorithm 2 (SPEA2) [59], S-Metric Selection Evolutionary Multiobjective Optimisation Algorithm (SMS-EMOA) [13], and Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) [57] have become extremely popular when it comes to solving multi-objective optimisation problems. NSGA-II [10] can be taken as a representative of Pareto-based approaches, the advantage to use this kind of method is the necessity to have few configuration parameters and the possibility to work well with a lot of objective functions meanwhile it is difficult to guarantee and measure the convergence of the solutions. SMS-EMOA [13] is a representative of the hypervolume indicator-based approaches: for these methods, it is possible to assess convergence at a high computational

cost, since a set's hypervolume is measured in relation to a reference point and it equals the total size of the space dominated by the solutions in the set. MOEA/D [57] is a good representative of the decomposition-based approaches, for these methods is possible to incorporate various scalarisation methods but this requires some prior knowledge of the position of the Pareto front. Though, at a certain point it has become evident that a single metaheuristic is not sufficient to deal with the real world and large scale problems, so hybridised metaheuristics started to be presented in literature as reported by [18]. The interaction among metaheuristics can take place at different levels, at low-level using specific functions from each metaheuristics or at high-level using a portfolio of metaheuristics for automated hybridisation [46]. Indeed, it has become popular the hybridisation between genetic algorithms and local search, that is also referred to as genetic local search, or *memetic algorithms* [25, 26]. Hybrid metaheuristics provide a more efficient behavior and a higher flexibility. For instance, a two-phased approach based on the combination of a multi-objective evolutionary algorithms and single-objective techniques to solve Vehicle Routing Problems has been proposed by [28] and two methods hybridised with the path relinking procedure, a Pareto ant colony optimisation algorithm and a variable neighborhood search method by [39]. A first hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible workshop scheduling problem has been introduced in [5]. Extended surveys have been provided in [7, 14, 46, 55, 60]. Our procedure is based on a hybridisation of NSGA-II and Variable Neighborhood search (VNS), and we refer to this as the Hybrid Multi-Objective Modified-NSGA-II Variable Neighborhood Search (MO-NSGA-VNS). Similarly to other population-based algorithms, the Pareto fronts (PFs) are formed and re-elaborated throughout the whole search process. At each step, solutions in the fronts are either kept or discarded according to a set of criteria and a new offspring population is generated. While many variants of NSGA algorithms generate preliminary random populations, we utilise the *Pool-Init-Gen* algorithm (see Algorithm 3.1) to create the initial population P_0. Furthermore, instead of using basic operators such as mutation and crossover in order to generate new populations, in our algorithmic approach we adopt a modified Variable Neighborhood Search scheme (see Section 4.1), which combines a variety of neighboring structures and intensification procedures. A previous example of hybridisation between NSGA and VNS had been presented in the literature [5]: in this work, the algorithm was applied to a bi-objective problem, the initial input solution to VNS was a random offspring obtained by a mutation operator, the neighbours were applied to random jobs into layers and no intensification features were exploited. Be-

fore presenting the complete pseudocode description of MO-NSGA-VNS as applied to our problem in Algorithm 2, we describe some of the used support structures, parameters and useful recalling functions, to allow for a thorough understanding of the procedure scheme, as follows:

UN: set of unreachable nodes.

ND: set where all found non-dominated solutions are stored.

α: parameter related to neighborhood $N_\alpha(p)$ (as presented in section 4.1), increased at each iteration i.

β: parameter related to neighborhood $N_\beta^{max}(p)$ (as presented in section 4.1), increased at each iteration i.

$\hat{Q}$: population size limit.

$\hat{T}$: time limit in seconds.

$\hat{L}$: intensification iterations limit.

$ADD(ND, p)$: handler function defined to update the set ND. Given any solution p this function adds p to the current set of the non-dominated ND if and only if p is non-dominated by any other solution belonging to the set ND.

An initial population P_0 of size at most $\hat{Q}$ is generated using Algorithm 1, then a generational loop is repeated until time limit $\hat{T}$ is reached. At any i_{th} population generation, an offspring population Q_i of size $\hat{Q}$ is created using the $MOVNS$ Algorithm (see Algorithm 3).

The new Pareto front population is obtained as $PQ_i = P_i \bigcup Q_i$ of size $2\hat{Q}$, this is divided into different non-dominated classes, or fronts, using the procedure $Fast\_non\_dominated\_sort$ presented by [10]. The domination counter is used to count how many solutions dominate the solution p, and it is adopted to determine whether a solution p belongs to a different non-dominated class, or front. The domination counter starts from zero for the first non-dominated front and reaches $|V| - 1$ for the last non-dominated front. Only the first $\hat{Q}$ individuals are kept to form the next generation, even if some of them can be members of the same last front. Note that no sub-procedures have been included to select and rank the last front. A graphical description of the whole evolutionary metaheuristic procedure is provided in Figure 3.

Algorithm 2 NSGAII-MOVNS$(\mathbf{G}(\mathbf{V}, \mathbf{E}, \mathbf{w}, \mathbf{s}, \mathbf{t}), \hat{\mathbf{Q}}, \hat{\mathbf{T}}, \hat{\mathbf{L}})$

$UN \leftarrow \emptyset$ {Set of unreachable nodes of any path from s to t}
$ND \leftarrow \emptyset$ {Set of the non-dominated Pareto solutions}
$t = 0$ {Time counter}
$i = 0$ {Iteration counter}
$P_i \leftarrow$ **Init_phase**$(\mathbf{G}(\mathbf{V}, \mathbf{E}, \mathbf{w}, \mathbf{s}, \mathbf{t}), \mathbf{ND}, \mathbf{UN})$
$F \leftarrow$ **Fast_non_dominated_sort**$(\mathbf{P_i})$
$Q_i \leftarrow$ **MOVNS**$(\mathbf{G}, \mathbf{P_i}, \hat{\mathbf{Q}}, \alpha, \beta, \mathbf{t}, \hat{\mathbf{T}}, \mathbf{ND}, \hat{\mathbf{L}})$
while t has not reached the time limit $\hat{T}$ **do**
  $PQ_i \leftarrow P_i \bigcup Q_i$
  $F \leftarrow$ **Fast_non_dominated_sort**$(\mathbf{PQ_i})[10]$
  ADD(ND, F) {update the set of non-dominated solutions with those ones coming from frontiers stored in F}
  $P_{i+1} \leftarrow F[1 : \hat{Q}]$ {Take the first frontiers until to fill the next population P_{i+1} with at most $\hat{Q}$ solutions }
  $Q_{i+1} \leftarrow$ **MOVNS**$(\mathbf{G}(\mathbf{V}, \mathbf{E}, \mathbf{w}, \mathbf{s}, \mathbf{t}), \mathbf{P_{i+1}}, \hat{\mathbf{Q}}, \alpha, \beta, \mathbf{t}, \hat{\mathbf{T}}, \mathbf{ND}, \hat{\mathbf{L}})$
  $i \leftarrow i + 1$
  if $\alpha > (|V|/2)$ **then**
    $\alpha = 1$ {re-initialise α to 1 }
  else
    α++ {Increase α until to reach half size of node number }
  end if
  if $\beta > (|V|/2)$ **then**
    $\beta = 1$ {re-initialise β to 1 }
  else
    β++ {Increase β until to reach half size of node number }
  end if
end while

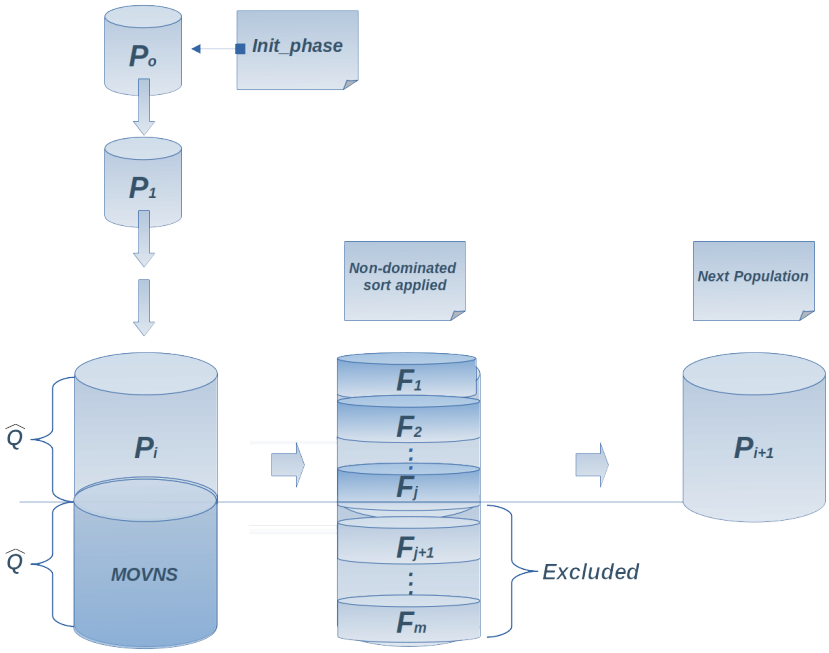


Figure 3: Graphical sketch of the MO-NSGA-VNS procedure.

4.1 Using a modified Multi-Objective Variable neighborhood search to generate the offsprings

Variable neighborhood search, introduced by [36], is a metaheuristic method whose search process draws upon systematic changes of neighborhood. The effectiveness of this procedure for solving single-objective optimisation problems has been broadly proved [22, 36, 37]. The first multi-objective VNS (MOVNS) was applied to a machine scheduling problem and proposed in [19], differing from single-objective VNSs for having introduced two main arbitrary choices: the base unvisited non-dominated solution (starting point of the next neighborhood search) and the used neighborhood, both chosen at random from those available. This procedure has been further developed by [2], where some non-dominated solutions have been constructed from partial results found by the inner procedures.
In our algorithm, we propose and adopt three parametric neighborhoods $N_\alpha(x')$, $N_{\beta^{max}}(x')$, N_γ^{CROSS} and, at each iteration of the algorithm, all of the defined neighborhoods are applied to a non-dominated solution p as randomly selected from the set P_i. Such neighborhoods are:

$\mathbf{N}_\alpha(\mathbf{p})$: obtained by removing up to α nodes from p and adding up to α nodes, completely at random.

$\mathbf{N}_\beta^{\mathbf{max}}(\mathbf{p})$: obtained by removing up to β nodes from p and adding up to β nodes, where such nodes are chosen from those belonging to the maximum connected component. This choice is aimed at reaching better solutions by creating paths passing through nodes belonging to larger connected components.

$\mathbf{N}_\gamma^{\mathbf{CROSS}}(\mathbf{p})$: is obtained from p and a random non negative index γ, with $1 \leq \gamma \leq |p| - 1$ by concatenation of the subpaths p_γ and p_i, where $p_\gamma = \{v_s, \ldots, v_\gamma\}$, (that is the subpath of p truncate at position γ) and $p_i = \{v_b, \ldots, v_t\} \subset \rho \in ND$ with $v_b = v_\gamma$ (that is the subpath of a solution $\rho \in ND$ having v_γ as inner node).

A pseudo-code describing the complete modified Multi-Objective Variable Search (MOVNS) including the intensification phase is provided in Algorithm 3, which is also depicted in Figure 4.

The aim of the intensification procedures is to add some new nodes to the current solution in order to shake the search procedure and escape from any local minima.This is realized by a recursive function, that is calling itself until no improvement can be performed on the best current solution (base case) or until one of the following terminating conditions is met: time limit $\hat{T}$ and depth of the recursion tree $\hat{T}$. We present two different procedures, both of them aimed at cutting a path p, which can be depicted by a sequence of nodes, in a specific position i, thus obtaining two subpaths $p_1 = s, \ldots, i$ and $p_2 = j, \ldots, t$, and filling the gap between nodes i and j with a new shortest path passing through a specific node u. The two procedures differ from how the node $u \in V$ is chosen (see Algorithm 4, and refer to lines 6 and 9). Hence a node u is selected at random from two sets, V^- and V^{max}, for $Intensification\ TYPE$ 1 and $Intensification\ TYPE$ 2 respectively. The set V^- includes all reachable nodes except those belonging to the path p, whereas V^{max} is the set of all nodes belonging to the maximum size connected component. At each offspring generation, both intensification procedures are applied on a random solution chosen from the parent population, differently from [2] where intensification is aimed at improving a partial VNS solution.

Algorithm 3 $MOVNS(G(V, E, w, s, t), P_i, \hat{Q}, \alpha, \beta, t, \hat{T}, ND, \hat{L})$

1: $Q_i \leftarrow \emptyset$ {set of offspring solutions}
2: **while** $\mathbf{t} \leq \hat{\mathbf{T}}$ AND $|\mathbf{Q_i}| \leq \hat{\mathbf{Q}}$ {Until time or Offspring size limits are reached} **do**
3: Select randomly a solution p from the population set P_i
4: and mark p as visited
5: $pd_1 \leftarrow$ **Intensification**$(\mathbf{G(V, E, w, s, t)}, \mathbf{p}, \hat{\mathbf{L}}, \mathbf{TYPE} \leftarrow \mathbf{1})$
6: $pd_2 \leftarrow$ **Intensification**$(\mathbf{G(V, E, w, s, t)}, \mathbf{p}, \hat{\mathbf{L}}, \mathbf{TYPE} \leftarrow \mathbf{2})$
7: **ADD**(Q_i, pd_1) AND **ADD**(Q_i, pd_2)
8: Determine randomly a solution p' from $\mathbf{N}_\alpha(\mathbf{p})$
9: **for all** $p'' \in \mathbf{N}_\alpha(\mathbf{p'})$ **do**
10: Evaluate the solution p''
11: **ADD**(Q_i, p'') {p'' is non-dominated by any solution of offspring set Q_i}
12: **end for**
13: Determine randomly a solution p' from $\mathbf{N}_\beta^{\mathbf{max}}$
14: **for all** $p'' \in \mathbf{N}_\beta^{\mathbf{max}}(\mathbf{p'})$ **do**
15: Evaluate the solution p''
16: **ADD**(Q_i, p'')
17: **end for**
18: Determine randomly a non negative index γ, with $1 \leq \gamma \leq |p| - 1$
19: Determine randomly a solution p' from $\mathbf{N}_\gamma^{\mathbf{CROSS}}(\mathbf{p})$
20: **for all** $p'' \in \mathbf{N}_\gamma^{\mathbf{CROSS}}(\mathbf{p'})$ **do**
21: Evaluate the solution p''
22: **ADD**(Q_i, p'')
23: **end for**
24: **end while**
25: **for all** $p \in Q_i$ **do**
26: **ADD**(ND, p)
27: **end for**
28: **return** Q_i

Algorithm 4 $Intensification(G(V,E,w,s,t),p,\hat{L},\hat{T},TYPE)$

1: $p' \leftarrow p$
2: **while** $\hat{L}$ and $\hat{T}$ limits are not satisfied **do**
3: **for** $i = 0$ **to** $|V(p) - 2|$ **do**
4: **for** $j = i + 1$ **to** $|V(p) - 1|$ **do**
5: **if** $TYPE = 1$ **then**
6: pick a random node u from V^-,
7: $V^- \leftarrow V \setminus \{s, t, V(p_s, \ldots, p_i), V(p_j, \ldots, p_t), UN\}$
8: **else**
9: pick a random node u from V^{max},
10: { $V^{max} \leftarrow CC \setminus \{UN\}$ where $|CC|$ is the maximum connected component obtained removing p }
11: **end if**
12:
13: $r' \leftarrow Dijkstra(G(V \setminus V(p), E \setminus E(p), w, s, t)i, u, j)$ { that is a shortest i-j path passing through node u, where i is the source and j is the sink }
14: **if** $r' \neq \emptyset$ **then**
15: $p' = (p_s, p_1, \ldots, \overbrace{p_i = r_i, \ldots, u, \ldots r_j = p_j}^{r'}, \ldots, p_t)$
16: **end if**
17: **if** $p' \prec p$ **then**
18: $\hat{L}++$ // increment to handle the recursion tree depth
19: $Intensification(G(V,E,w,s,t),p',\hat{L},\hat{T},TYPE)$ //recursive case
20: $add(ND, p')$
21: **else**
22: **return** p' //case base
23: **end if**
24: **end for**
25: **end for**
26: **end while**
27: **return** p'

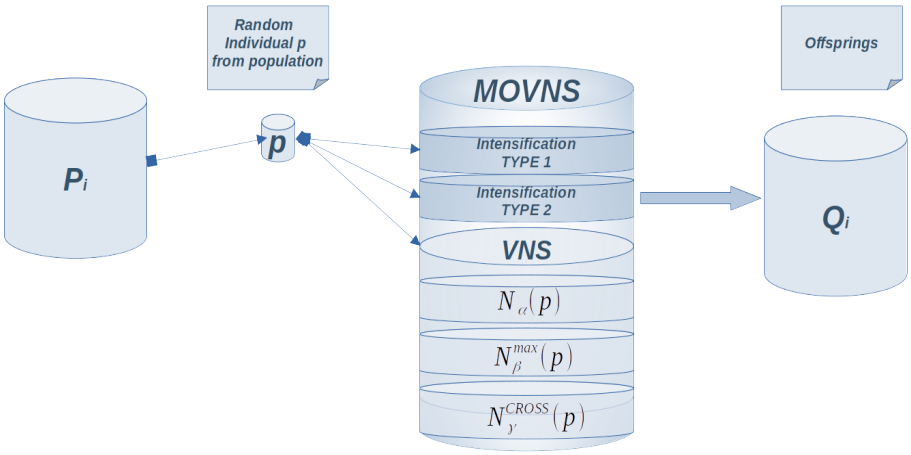


Figure 4: Graphical sketch of the Algorithm 3.

5 Testbed and Computational Experiments

In this section, we present and analyse the results of the articulated computational experience that has been developed in this work. The scope of the experiments is assessing the efficiency of the MO-NSGA-VNS algorithm and its efficacy at approximating the Pareto front of the Multi-Objective Critical Disruption Path Problem. We first provide here a description of the adopted testbed and of the range of different methods we considered in order to assess the performance of our algorithm. Then in Section 5.1 the results of the computational experiments are presented and discussed.

Computational testbed. A large set of increasing size random instances has been generated uniformly distributed as a test bed, overall made up of two different classes of instances:

- six groups of medium size, with a number of nodes $n = 40 + 10 \times \kappa$, $\kappa \in [0, 1, \ldots, 5]$

- ten groups of large size instances, with a number of nodes $n = 100 + 100 \times \sigma$, $\sigma \in [0, 1, \ldots, 9]$.

For both classes, the number of arcs is $m = n \times (n-1) \times \rho$ where $\rho \in \{0.1, 0.5, 0.9\}$ and, for each pair (n, m), two distinct randomly generated instances are created and referred to by using the notation (n, m, o), with $o = \{0, 1\}$, to denote the instance occurrence. The algorithm for instance generation primarily consists of two phases: the connecting phase, with at most $n-1$ steps, where arcs are iteratively generated at pseudo-random connecting one node already inserted in the building graph and the other one not yet connected, until none of the nodes are left out; the building phase, where all other arcs are generated at pseudo-random, by iteratively selecting origin and destination in V with uniform probabilities, until the required target number of arcs m is reached. Furthermore, for each arc e a weight l_e is generated at random with a uniform distribution in the interval $[1, n]$. The MO-NSGA-VNS was coded in ANSI C++-14. All the computations have been performed on an Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz with 16 GB of RAM. The following parameters were adopted:

$\hat{T} = 3600$ *seconds*, as a computational time limit.

$\hat{L} = 100 * |E|$, with a maximum value of 10000, (i.e., the recursion tree depth cannot exceed the value of 10000) as an intensification depth parameter, used in the Intensification procedures (see Algorithm 4).

$\hat{Q} = 20$, as an offspring set size.

Note that the parameters $\hat{L}$ and $\hat{Q}$ have been set by following the results of a preliminary calibration phase based on the execution of the algorithm on a sample of instances.

Benchmarking against scalarisation technique. In order to assess quality and exhaustiveness of the MO-NSGA-VNS at generating the Pareto front, a comparison with the results of a multi-objective scalarisation technique, implemented by using a state-of-art off-the-shelf solver, is also provided in this paper for the class of medium size instances. To this aim, several convex combinations of the three objective functions were considered and experiments conducted at varying weight coefficients λ_i , $i = 1, 2, 3$, in the range [0,1] with such interval divided in 100 steps for each dimension, such that $\sum \lambda_i = 1$ for each triple of weight coefficients considered. As the goal of this specific comparison is concerned with challenging the capability of the MO-NSGA-VNS to produce a comprehensive Pareto front, rather

than its computational efficiency, each experiment was executed by using IBM ILOG CPLEX 12.9 on the same machine without any time limitation in order to get the largest possible number of solutions, although clearly not exhaustive. The proposed model has been implemented using ANSI C++ and Concert Technologies libraries, using Cplex callbacks to cope with subtour elimination contraints (5). Thus, the set of non-dominated solutions $ND(\text{MO-CDP}) = \bigcup_{\lambda_i} ND_{\lambda_i}(\text{MO-CDP})$ found through implementing and solving the (MO-CDP) model by CPLEX consists in the union of all non-dominated solutions found whilst varying the convex weight λ_i combinations through the above described scalarisation technique.

Benchmarking against evolutionary algorithms. In order to test also time-efficiency and effectiveness of the MO-NSGA-VNS algorithm, two further comparisons against standard evolutionary algorithms were developed, through the implementation of the standard Multi-objective Evolutionary algorithms: *basic-NSGA* and *subpaths-NSGA*. The *basic-NSGA* is based on the classic implementation of NSGA-II where the mutation operator is obtained by replacing a node with a new one, as follows: given a solution path $p = \{s, \ldots, i-1, i, i+1, \ldots, t\}$, a random node i is substituted by another random node j as it is expected within the *basic-NSGA* algorithmic framework. As this simple mutation operator often does not guarantee the existence of another valid path, we also introduced *subpaths-NSGA* algorithm, whose main idea is to replace a node i by a simple sub-path $\{i\text{-}1, \ldots i\text{+}1\}$, created between the adjacent nodes i-1 and i+1 whose nodes are not belonging to the original path p. We use the $Intensification$ procedure of $TYPE$ 1 (see Algorithm 4) as operator to find the best fitting subpath.

Adopted benchmarking metrics. non-dominated

$$NDsbyR(ND, R) = |nd \in ND| \nexists r \in R : r \prec nd \tag{14}$$

$$NDsRbyR(ND, R) = \frac{NDsbyR(ND, R)}{|ND|} \tag{15}$$

The reference set R is defined as a collection of candidate solutions with respect to which we can compare two algorithms. In other words, these metrics estimate, respectively, the amount of solutions in the set ND which are dominated by any solution belonging to the reference set R, expressed as absolute number and percentage, respectively.

5.1 Discussion of the results

The results of the computational experiments are now presented and discussed, focusing on the performance of MO-NSGA-VNS algorithm as measured against the Pareto front in terms of quality of the obtained solutions by adopting a variety of convergence-diversity criteria. Experiments are presented as follows. Firstly we present the comparison between the results obtained by MO-NSGA-VNS and those provided by the scalarisation technique approach on the (MO-CDP) model. Table 1 is aimed at comparing the overall number of non-dominated solutions provided (NDs) and the required CPU times, against each distinct instance. Recall for each pair (n, m) a couple of instance occurrences are generated, denoted with two triplets (n, m, o), with $o = \{0, 1\}$. Computational times for the MO-CDP scalarisation technique express the sum of all computational times needed to solve each convex combination obtained at varying λ_i into the objective function, where no time limit was applied. As regards MO-NSGA-VNS, $Min(Time)$ and $AVG(Time)$ report respectively the time when the first solution was found and the average time when non-dominated solutions ND were identified, recalling how a time limit of 3600 seconds was set for this case. A value of the $Min(Time)$ column equal to 0 suggests therefore how at least one solution among the final set of non-dominated solutions was found during the initialisation phase by the *Pool-Init-Gen* algorithm (see Section Algorithm 3.1). From the results, it is apparent how the MO-NSGA-VNS provided clear advantages in terms of computing times and number of identified solutions. It is also worth clarifying that the measured CPU time needed for executing the whole $Init\_phase$ iterative algorithm on instances of any size was negligible as regularly less than one second. This evidence matches with the worst-case analysis already provided for the proposed *Pool-Init-Gen* algorithm, and corroborates experimentally its high performance, as it requires very limited computational resources but in some cases even produces non-dominated solutions. In order to explore and compare the quality of the solutions obtained by the two approaches, the results in Table 2 adopt the metrics 14 and 15 above described to appreciate how only a few solutions provided by MO-NSGA-VNS (around 1.25 on the average) are dominated by other solutions in the reference set, which includes the solutions found by the MO-CDP algorithm, whereas the MO-NSGA-VNS outperformed the MO-CDP approach by 25 solutions on the average. Results in Table 3 report the range of values for the three objective functions: if the Pareto front solutions span on a broader interval, then a better approximation of the reference set is reflected. These results are therefore devoted

to evaluate diversity and inclusiveness of the set of obtained non-dominated solutions for each algorithm, and assess how broadly these are spread over the Pareto front. The two approaches show a quite different behaviour in this respect, as functions f_1 and f_2 span a quite wider range on most of the solved instances, with exception for the larger ones, while the range for f_3 is consistently wider for the metaheuristic approach.

Also, it is worth noticing how the lowest values in terms of cost of the CDP are regularly identified by the evolutionary metaheuristic, thus hitting one of the underlying goals of this multi-objective extension of the CDP approach: containing as much as possible the transportation costs for surveillance operations while securing the highest possible effectiveness in terms of quality of the surveillance.

The results of the experiments aimed at benchmarking MO-NSGA-VNS against standard evolutionary algorithms are provided in Table 4 by reporting, for each instance, the number NDs of solutions provided as an output by each algorithm, and the percentage $NDsRbyR$ of such solutions which result to be dominated by using the metric 15 presented above. Note such a metric is applied with a *one-vs-all* method, where each non-dominated solution set ND produced by a given algorithm is compared to a reference solution set R, being the latter equal to the union of all the solutions produced by the remaining algorithms. Clearly, such a reference set R differs from the one adopted in Table 2. It can be observed how the MO-NSGA-VNS outperforms both the competing approaches, providing somewhat regularly the highest number of solutions, and presenting at the same time the lowest values for the dominance metric. Indeed, although at a first glance on some instances the number of solutions returned by the *subpaths-NSGA* may appear higher than those provided by the MO-NSGA-VNS, $NDsRbyR$ scores are effective at showing how most of such produced solutions are dominated and thus do not represent suitable candidates to approximate the Pareto front. Figures 5-6 are instrumental at visualising the prevailing profile of the MO-NSGA-VNS algorithm solutions as compared to the competing approaches.

Conclusions

In this paper we proposed a multiple objective approach to extend the Critical Disruption Path problem, a path-based network interdiction problem aimed at optimising surveillance operations. While coupling the size of the maximal connected component and the number of such components in the

Table 1: Performance comparisons between MO-CDP and MO-NSGA-VNS. Column 1 describes the instance triplet (n,m,o). Columns 2 and 4 present the number of non-dominated solutions NDs for both algorithms: MO-CDP and MO-NSGA-VNS. Running times are reported into columns 3, 5 and 6.

	MO-CDP		**MO-NSGA-VNS**		
(n,m,o)	NDs	Time (s)	NDs	$Min(Time(s))$	$AVG(Time(s))$
(40,156,0)	16	25816.76	33	1133	3024.15
(40,156,1)	11	19098.29	21	0	2947.00
(40,780,0)	9	153529.76	29	0	2461.55
(40,780,1)	13	189728.45	26	0	2093.81
(40,1404,0)	17	246153.33	29	2	2333.97
(40,1404,1)	14	260671.31	27	126	1578.22
(50,245,0)	20	39137.43	39	0	2705.69
(50,245,1)	17	27888.02	16	7	2826.88
(50,1225,0)	9	123010.44	27	1	2792.30
(50,1225,1)	6	139596.33	26	1822	3147.50
(50,2205,0)	-	-	22	1365	3126.77
(50,2205,1)	-	-	28	315	2052.75
(60,354,0)	23	81649.94	48	196	1470.44
(60,354,1)	10	63908.25	54	182	2120.20
(60,1770,0)	-	-	40	449	2462.05
(60,1770,1)	-	-	48	336	1868.25
(60,3186,0)	-	-	31	409	2767.23
(60,3186,1)	-	-	25	0	2744.80
(70,483,0)	16	184657.06	53	49	1765.02
(70,483,1)	28	153343.14	68	119	1602.93
(70,2415,0)	-	-	32	5	2288.56
(70,2415,1)	-	-	44	693	2602.36
(70,4347,0)	-	-	22	0	3018.64
(70,4347,1)	-	-	27	748	2596.74
(80,632,0)	5	48229.90	21	0	2412.76
(80,632,1)	12	169140.29	18	20	1533.89
(80,3160,0)	-	-	46	82	1673.41
(80,3160,1)	-	-	50	0	2306.84
(80,5688,0)	-	-	14	1457	2325.43
(80,5688,1)	-	-	22	3	2603.00
(90,801,0)	1	63118.42	26	19	2628.27
(90,801,1)	1	52530.52	19	0	3055.53
(90,4005,0)	-	-	26	961	2793.58
(90,4005,1)	-	-	26	0	2903.46
(90,7209,0)	-	-	21	1745	2753.76
(90,7209,1)	-	-	30	0	2899.10

Table 2: Results of the capacity performance measure for MO-NSGA-VNS algorithm, where the reference set R contains the solutions found by the MO-CDP algorithm. Column 1 describes the instance triplet (n,m,o). Columns 2-3 report the NDs for algorithms MO-CDP and MO-NSGA-VNS. Columns 4-5, report $NDsbyR$ and $NDsRbyR$ metric values (Metrics 14, 15). Last column 6 reports the solutions in common with the reference set R.

	MO-CDP	**MO-NSGA-VNS**			
(n,m,o)	NDs	NDs	NDsbyR	NDsRbyR	$\cap$
(40,156,0)	16	33	5	0.16	6
(40,156,1)	11	21	3	0.15	6
(40,780,0)	9	29	2	0.07	4
(40,780,1)	13	26	6	0.24	1
(40,1404,0)	17	29	9	0.32	2
(40,1404,1)	14	27	8	0.30	1
(50,245,0)	20	39	5	0.13	1
(50,245,1)	17	16	2	0.13	2
(50,1225,0)	9	27	1	0.04	2
(50,1225,1)	6	26	1	0.04	1
(50,2205,0)	-	22	-	-	-
(50,2205,1)	-	28	-	-	-
(60,354,0)	23	48	9	0.19	4
(60,354,1)	10	54	3	0.06	4
(60,1770,0)	-	40	-	-	-
(60,1770,1)	-	48	-	-	-
(60,3186,0)	-	31	-	-	-
(60,3186,1)	-	25	-	-	-
(70,483,0)	16	53	4	0.08	2
(70,483,1)	28	68	8	0.12	4
(70,2415,0)	-	32	-	-	-
(70,2415,1)	-	44	-	-	-
(70,4347,0)	-	22	-	-	-
(70,4347,1)	-	27	-	-	-
(80,632,0)	5	21	1	0.05	2
(80,632,1)	12	18	3	0.17	1
(80,3160,0)	-	46	-	-	-
(80,3160,1)	-	50	-	-	-
(80,5688,0)	-	14	-	-	-
(80,5688,1)	-	22	-	-	-
(90,801,0)	1	26	0	0.00	1
(90,801,1)	1	19	0	0.00	1
(90,4005,0)	-	26	-	-	-
(90,4005,1)	-	26	-	-	-
(90,7209,0)	-	21	-	-	-
(90,7209,1)	-	30	-	-	-

Table 3: Performance comparisons between MO-CDP and MO-NSGA-VNS algorithms, column 1 describes the instance triplet (n,m,o). Columns 2-5 report the range of the three objective functions f_1,f_2 f_3 and the cost as a number of hops for the MO-CDP problem. Columns 6-9 report these ranges for the MO-NSGA-VNS problem.

	MO-CDP				**MO-NSGA-VNS**			
(n,m,o)	f_1	f_2	f_3	$Hops$	f_1	f_2	f_3	$Hops$
(40,156,0)	[1-36]	[2-19]	[24-711]	[2-22]	[4-36]	[2-17]	[24-417]	[2-15]
(40,156,1)	[1-30]	[3-15]	[203-827]	[7-25]	[4-30]	[3-15]	[203-587]	[7-19]
(40,780,0)	[1-32]	[2-7]	[122-1241]	[6-32]	[10-32]	[2-3]	[122-2313]	[6-27]
(40,780,1)	[1-37]	[2-7]	[134-1327]	[1-33]	[12-37]	[2-3]	[134-2825]	[1-26]
(40,1404,0)	[0-35]	[0-3]	[84-2440]	[4-39]	[10-35]	[1-2]	[84-9237]	[4-29]
(40,1404,1)	[0-36]	[0-3]	[130-2302]	[3-39]	[11-36]	[1-2]	[130-8301]	[3-27]
(50,245,0)	[1-36]	[11-32]	[93-970]	[3-20]	[5-36]	[11-26]	[93-944]	[3-17]
(50,245,1)	[1-30]	[14-32]	[73-986]	[6-21]	[10-30]	[14-22]	[73-486]	[6-17]
(50,1225,0)	[1-45]	[1-7]	[83-3269]	[4-45]	[21-45]	[1-2]	[83-3822]	[4-27]
(50,1225,1)	[1-45]	[1-7]	[166-3103]	[4-45]	[18-45]	[1-2]	[166-9588]	[4-31]
(50,2205,0)	[-]	[-]	[-]	[-]	[19-46]	[1-1]	[117-10107]	[3-30]
(50,2205,1)	[-]	[-]	[-]	[-]	[18-46]	[1-2]	[129-15299]	[3-30]
(60,354,0)	[1-52]	[5-30]	[88-1969]	[3-31]	[8-52]	[5-26]	[88-1897]	[3-21]
(60,354,1)	[1-52]	[4-27]	[135-2080]	[4-32]	[9-52]	[4-25]	[135-2638]	[4-25]
(60,1770,0)	[-]	[-]	[-]	[-]	[24-53]	[1-9]	[247-10659]	[6-30]
(60,1770,1)	[-]	[-]	[-]	[-]	[27-57]	[1-4]	[135-8213]	[2-30]
(60,3186,0)	[-]	[-]	[-]	[-]	[28-56]	[1-3]	[458-17497]	[3-29]
(60,3186,1)	[-]	[-]	[-]	[-]	[25-57]	[1-3]	[188-17405]	[2-32]
(70,483,0)	[1-57]	[9-34]	[239-3346]	[4-36]	[6-57]	[9-30]	[239-2533]	[4-21]
(70,483,1)	[1-59]	[8-33]	[219-3039]	[3-37]	[10-59]	[8-29]	[219-2436]	[3-21]
(70,2415,0)	[-]	[-]	[-]	[-]	[35-64]	[2-6]	[119-12206]	[4-28]
(70,2415,1)	[-]	[-]	[-]	[-]	[33-68]	[1-7]	[122-12359]	[1-30]
(70,4347,0)	[-]	[-]	[-]	[-]	[36-63]	[1-2]	[123-21697]	[6-32]
(70,4347,1)	[-]	[-]	[-]	[-]	[38-63]	[1-2]	[249-27361]	[5-30]
(80,632,0)	[3-67]	[6-35]	[402-2371]	[5-36]	[37-67]	[6-21]	[402-1537]	[5-20]
(80,632,1)	[1-73]	[4-47]	[64-3133]	[3-36]	[39-73]	[4-23]	[64-359]	[3-14]
(80,3160,0)	[-]	[-]	[-]	[-]	[48-76]	[1-11]	[312-18037]	[3-22]
(80,3160,1)	[-]	[-]	[-]	[-]	[40-75]	[1-11]	[165-15957]	[4-32]
(80,5688,0)	[-]	[-]	[-]	[-]	[51-71]	[1-1]	[280-12378]	[8-28]
(80,5688,1)	[-]	[-]	[-]	[-]	[48-70]	[1-2]	[210-18573]	[9-30]
(90,801,0)	[86-86]	[1-1]	[170-170]	[3-3]	[54-86]	[1-1]	[170-4218]	[3-35]
(90,801,1)	[88-88]	[1-1]	[60-60]	[1-1]	[61-88]	[1-1]	[60-2008]	[1-28]
(90,4005,0)	[-]	[-]	[-]	[-]	[63-84]	[1-2]	[451-21072]	[5-25]
(90,4005,1)	[-]	[-]	[-]	[-]	[57-83]	[3-7]	[159-18480]	[4-27]
(90,7209,0)	[-]	[-]	[-]	[-]	[61-83]	[1-2]	[261-44696]	[5-28]
(90,7209,1)	[-]	[-]	[-]	[-]	[62-87]	[1-2]	[288-45496]	[2-26]

Table 4: Results of the capacity performance measure for all the created algorithms, where the reference set R is the union of all the ND sets found by the other algorithms. Column 1 describes the instance triplet (n,m,o). Columns 2-3, 4-5 and 6-7 report the NDs value and the $NDsRbyR$ metric value (see Metric 15) for each genetic algorithm.

	MO-NSGA-VNS		**Subpaths-NSGA**		**Basic-NSGA**	
(n,m,o)	NDs	NDsRbyR	NDs	NDsRbyR	NDs	NDsRbyR
(40,156,0)	33	0.15	35	0.29	3	0.33
(40,156,1)	21	0.14	20	0.35	4	0.50
(40,780,0)	29	0.07	30	0.70	1	1.00
(40,780,1)	26	0.23	23	0.83	4	0.25
(40,1404,0)	29	0.31	26	0.77	2	0.50
(40,1404,1)	27	0.30	35	0.66	2	1.00
(50,245,0)	39	0.21	37	0.35	7	0.71
(50,245,1)	16	0.19	38	0.29	5	0.40
(50,1225,0)	27	0.04	44	0.64	3	0.33
(50,1225,1)	26	0.04	32	0.66	3	0.67
(50,2205,0)	22	0.00	30	0.60	2	0.50
(50,2205,1)	28	0.21	27	0.48	2	1.00
(60,354,0)	48	0.31	50	0.32	5	0.80
(60,354,1)	54	0.17	44	0.41	4	1.00
(60,1770,0)	40	0.02	58	0.36	4	0.50
(60,1770,1)	48	0.04	50	0.48	6	0.50
(60,3186,0)	31	0.00	39	0.56	4	0.50
(60,3186,1)	25	0.00	36	0.47	4	0.75
(70,483,0)	53	0.15	53	0.45	6	0.50
(70,483,1)	68	0.25	48	0.38	6	0.67
(70,2415,0)	32	0.03	50	0.40	6	0.33
(70,2415,1)	44	0.00	43	0.56	5	0.20
(70,4347,0)	22	0.00	23	0.61	3	0.67
(70,4347,1)	27	0.00	23	0.74	2	1.00
(80,632,0)	21	0.19	37	0.19	7	0.86
(80,632,1)	18	0.17	51	0.24	4	1.00
(80,3160,0)	46	0.04	36	0.22	6	0.67
(80,3160,1)	50	0.06	47	0.19	6	0.50
(80,5688,0)	14	0.00	35	0.31	3	0.67
(80,5688,1)	22	0.00	31	0.32	3	0.67
(90,801,0)	26	0.00	25	0.80	3	1.00
(90,801,1)	19	0.00	26	0.58	4	0.25
(90,4005,0)	26	0.00	34	0.59	4	0.75
(90,4005,1)	26	0.00	24	0.25	6	0.50
(90,7209,0)	21	0.00	23	0.43	2	0.50
(90,7209,1)	30	0.03	27	0.56	3	0.67

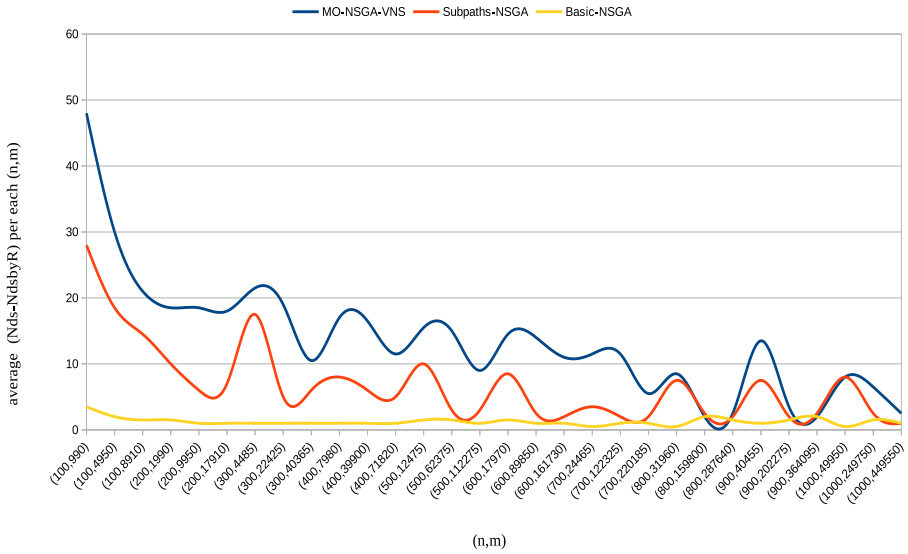


Figure 5: Comparisons among the different evolutionary algorithms reporting $NDs\text{-}NDsbyR$ values for each pair (n, m).

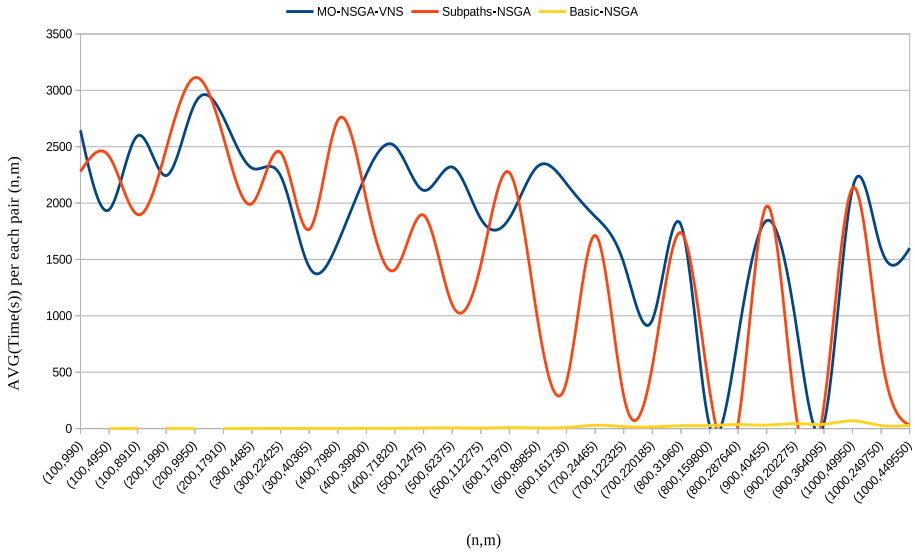


Figure 6: Comparisons among the different evolutionary algorithms reporting average time values for each pair (n, m).

effort to maximise the surveillance effectiveness, our variant of the CDP problem concurrently seeks ~~for~~ the minimisation of the critical path cost, thus increasing the impact of adopting a CDP approach for practical purposes, supporting surveillance planning for a variety of application fields and different transportation means. Motivated by the provably high level of complexity of the considered problem, we developed an original evolutionary metaheuristic algorithmic approach, which hybridises modified-NSGA-II and VNS for finding efficiently an approximation of the Pareto front on increasing size networks. We also complemented this algorithm with a tailored preliminary procedure, based on a minisum variant of the shortest disjoint path pair problem, and aimed at allowing a preliminary efficient calculation of high quality initial solutions to feed the evolutionary algorithm, concurrently decreasing its computational effort requirements by reducing the set of candidates.

References

[1] R. Albert, H. Jeong, and A. L. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406:378–382, aug 2000.

[2] J. E. C. Arroyo, R. dos Santos Ottoni, and A. de Paiva Oliveira. Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. *Electronic Notes in Theoretical Computer Science*, 281:5– 19, 2011. Proceedings of the 2011 Latin American Conference in Informatics (CLEI).

[3] A. Arulselvan, C. W. Commander, P. M. Pardalos, and O. Shylo. Managing network risk via critical node identification. In B. Rustem and N. Gulpinar, editors, *Risk Management in Telecommunication Networks*. Springer, 2007.

[4] A. Arulselvan, C. W. Commander, L. Elefteriadou, and P. M. Pardalos. Detecting critical nodes in sparse graphs. *Computers & Operations Research*, 36(7):2193–2200, 2009.

[5] H. Asefi, F. Jolai, M. Rabiee, and M. E. Tayebi Araghi. A hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 75(5-8):1017–1033, 2014.

[6] M. O. Ball, B. L. Golden, and R. V. Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8(2):73–76, 1989.

[7] M. Basseur, A. Talbi, E.-G.and Nebro, and E. Alba. Avancées des métaheuristiques pour l'optimisation combinatoire multi-objectif. Technical report, INRIA research report, 2006.

[8] R. Cohen, S. Havlin, and D. Ben-Avraham. Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, 91(24):247–901, Dec 2003.

[9] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[11] T. N. Dinh, Y. Xuan, M. T. Thai, E. K. Park, and T. Znati. On approximation of new optimization methods for assessing network vulnerability. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 2678–2686, Piscataway, NJ, USA, 2010. IEEE Press.

[12] T. Eilam-Tzoreff. The disjoint shortest paths problem. *Discrete Applied Mathematics*, 85(2):113 – 138, 1998.

[13] M. Emmerich, N. Hochstrate, and B. Naujoks. An EMO algorithm using the Hypervolume Measure as Selection Criterion. In *In international Conference on Evolutionary Multi-Criterion Optimization*, pages 62–76. Springer, 2005.

[14] M. T. M. Emmerich and A. H. Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609, September 2018.

[15] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111 – 121, 1980.

[16] A. Frank. *Packing paths, circuits and cuts: a survey.* Report. Sonderforschungsbereich 303, 1988.

[17] F. Furini, I. Ljubić, S. Martin, and P. San Segundo. The maximum clique interdiction problem. *European Journal of Operational Research*, 277(1):112 – 127, 2019.

[18] X. Gandibleux and M. Ehrgott. 1984-2004–20 years of multiobjective metaheuristics. but what about the solution of combinatorial problems with multiple objectives? In *Evolutionary Multi-Criterion Optimization*, pages 33–46. Springer, 2005.

[19] M. J. Geiger. Randomised variable neighbourhood search for multi objective optimisation. *CoRR*, abs/0809.0271, 2008.

[20] D. Granata and A. Sgalambro. Network interdiction through length-bounded critical disruption paths: a bi-objective approach. *Electronic Notes in Discrete Mathematics*, 2016.

[21] D. Granata, G. Steeger, and S. Rebennack. Network interdiction via a critical disruption path: Branch-and-price algorithms. *Computers & Operations Research*, 40(11):2689–2702, 2013.

[22] Pierre Hansen and Nenad Mladenović. Variable neighborhood search for the p-median. *Location Science*, 5(4):207–226, 1997.

[23] T. E. Harris and F. S. Ross. Fundamentals of a method for evaluating rail net capacities. Research Memorandum RM-1573, The Rand Corporation, Santa Monica, CA, 1955.

[24] P. Holme, B. J. Kim, C. N. Yoon, and S. K. Han. Attack vulnerability of complex networks. *Physical Review E*, 65:056109, May 2002.

[25] H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, pages 119–124, May 1996.

[26] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 28(3):392–403, Aug 1998.

[27] E. Israeli and R. K. Wood. Shortest-path network interdiction. *Networks*, 40:97–111, 2002.

[28] N. Jozefowiez, F. Semet, and E.G. Talbi. From single-objective to multi-objective vehicle routing problems: Motivations, case studies, and

methods. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research & Computer Science Interfaces*, pages 445–471. Springer US, 2008.

[29] M Lalou, Tahraoui M. A., and Kheddouci H. The critical node detection problem in networks: A survey. *Computer Science Review*, 28:92 – 117, 2018.

[30] C.L. Li, S.T. McCormick, and D. Simchi-Levi. On the computational complexity of combinatorial problems. *Networks*, 5:45–68, 1975.

[31] C.L. Li, S.T. McCormick, and D. Simchi-Levi. The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics*, 26(1):105 –115, 1990.

[32] C. Lim and J. C. Smith. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions*, 39 (1):15–26, 2007.

[33] P. F. Mahdavi, V. Boginski, and E. L. Pasiliao. Minimum vertex blocker clique problem. *Networks*, 64(1):48–64, 2014.

[34] L. Martins, T. Gomes, and D. Tipper. Efficient heuristics for determining node-disjoint path pairs visiting specified nodes. *Networks*, 70(4): 292–307, 2017.

[35] I. Mishkovski, M. Biey, and L. Kocarev. Vulnerability of complex networks. *Communications in Nonlinear Science and Numerical Simulation*, 16(1):341 – 349, 2011.

[36] N. Mladenović and P. Hansen. Variable neighborhood search. *Computer and Operations Research*, 24(11):1097–1100, November 1997.

[37] M. Polacek, R. F. Hartl, K. Doerner, and M. Reimann. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics*, 10(6):613–627, 2004.

[38] J. O. Royset and R. K. Wood. Solving the bi-objective maximum-flow network-interdiction problem. *INFORMS Journal on Computing*, 19 (2):175–184, 2007.

[39] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3): 179–201, 2009.

[40] S. Shen, J. C. Smith, and R. Goli. Exact interdiction models and algorithms for disconnecting networks via node deletions. *Discrete Optimization*, 9(3):172–188, 2012.

[41] Y. Shiloach. A polynomial solution to the undirected two paths problem. *J. ACM*, 27(3):445–456, July 1980.

[42] J. C. Smith and Y. Song. A survey of network interdiction models and algorithms. *European Journal of Operational Research*, 283(3):797 – 811, 2020.

[43] M.D. Summa, A. Grosso, and M. Locatelli. Complexity of the critical node problem over trees. *Computers & Operations Research*, 38(12): 1766–1774, 2011.

[44] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4(2):125–145, 1974.

[45] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14(2):325–336, 1984.

[46] E-G. Talbi, M. Basseur, A. J Nebro, and E. Alba. Multi-objective optimization using metaheuristics: non-standard algorithms. *International Transactions in Operational Research*, 19(1-2):283–305, 2012.

[47] S. Tragoudas and Y. Varol. Computing disjoint paths with length constraints. In F. d'Amore, P. Franciosa, and A. Marchetti-Spaccamela, editors, *Graph-Theoretic Concepts in Computer Science*, volume 1197 of *Lecture Notes in Computer Science*, pages 375–389. Springer Berlin Heidelberg, 1997.

[48] M. Ventresca. Global search algorithms using a combinatorial unranking-based problem representation for the critical node detection problem. *Computers & Operations Research*, 39(11):2763–2775, 2012.

[49] M. Ventresca and D. Aleman. A derandomized approximation algorithm for the critical node detection problem. *Computers & Operations Research*, 43:261– 270, 2014.

[50] M. Ventresca and D. Aleman. A fast greedy algorithm for the critical node detection problem. In Z. Zhang, L. Wu, W. Xu, and D.-Z. Du, editors, *Combinatorial Optimization and Applications*, volume 8881 of *Lecture Notes in Computer Science*, pages 603–612. Springer International Publishing, 2014.

[51] J. Vygen. *Disjoint paths*. Citeseer, 1994.

[52] J. L. Walteros, A. Veremyev, P. M. Pardalos, and E. L. Pasiliao. Detecting critical node structures on graphs: A mathematical programming approach. *Networks*, 73(1):48–88, 2019.

[53] R. Wollmer. Removing arcs from a network. *Operations Research*, 12 (6):934–940, 1964.

[54] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

[55] G. R. Zavala, A. J. Nebro, F. Luna, and C.A. Coello Coello. A survey of multi-objective metaheuristics applied to structural optimization. *Structural and Multidisciplinary Optimization*, 49(4):537–558, 2014.

[56] R. Zenklusen. Matching interdiction. *Discrete Applied Mathematics*, 158(15):1676 – 1690, 2010.

[57] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731, 2007.

[58] T. Zhou, Z. Q. Fu Fu, and B. H. Wang. Epidemic dynamics on complex networks. *Progress in Natural Science*, 16:452, 2006.

[59] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report, 2001.

[60] E. Zitzler, M. Laumanns, and S. Bleuler. A tutorial on evolutionary multiobjective optimization. In *Metaheuristics for multiobjective optimisation*, pages 3–37. Springer, 2004.