ILP based heuristics for a Virtual Network Function placement and routing problem

Bernardetta Addis[a], Giuliana Carello[b], Meihui Gao[c]

[a] *Université de Lorraine, CNRS, LORIA, F-54000 Nancy, France*
[b] *Dipartimento di Elettronica, Informazione e Bioingegneria Politecnico di Milano, Milano, Italy*
[c] *Zhejiang University of Technology, China*

Abstract

Thanks to the increased availability of computing capabilities in data centers, the recently proposed virtual network function paradigm can be used to keep up with the increasing demand for network services as internet and its applications grow. The problem arises then of managing the virtual network functions, that is, to decide where to instantiate the functions and how to route the demands to reach them. While it arises in an application field, the Virtual Network Function placement and routing problem combines location and routing aspects in an interesting, challenging problem.

In this paper, we propose several ILP-based heuristics and compare them on a dataset that includes instances with different sizes, network topologies and service capacity. The heuristics prove effective in tackling even large size instances, with up to 50 nodes and more than 80 arcs.

Keywords: virtual network functions, routing, location, matheuristics

1. Introduction

Over the past years, the amount of traffic on the network increased due to the diffusion of internet services and their widespread availability through mobile devices, and the arrival of 5G will boost it even further. Among the challenges that this increase will create, network providers will be facing a growing demand for certain network functions

Email addresses: bernardetta.addis@loria.fr (Bernardetta Addis), giuliana.carello@polimi.it (Giuliana Carello), gaomeihui@zjut.edu.cn (Meihui Gao)

June 17, 2021

10.1002/net.22073 - Networks. 2021 78:328–349.

such as firewalls, proxies or WAN optimizers, which brings with them a high demand for computing capacity. Network functions have been historically provided on hardware, i.e., by installing expensive, specifically designed devices on some network nodes to provide the required functions. However, these devices are hardly able to keep up with the increasing demand and cannot be upgraded easily and cost-effectively to increase their capacity or provide new functionalities. On the other hand, a great amount of computing capacity has been recently made available on the network thanks to the diffusion of data centers and cloud computing facilities. It has therefore been proposed to provide network functions on a virtual, rather than hardware, basis. According to the Network Function Virtualization paradigm, part of the computing capacity available on the network nodes can be reserved to provide the required network functions in a cheap, flexible and easy-to-update way.

The problem arises then of determining the position of the Virtual Network Functions (VNFs) in the graph: node capacity must be reserved to provide virtual services, usually by allocating some virtual machines. In addition, the routing of each demand must be selected to guarantee that the demand can use the functions it requires by passing through a node hosting one VNF instance of the required type.

In this paper, we consider a network where each node is connected to a server or a data center that provides computing capacity. A set of demands is given, each requiring the same type of VNF. The problem is to decide where to allocate computing capacity to VNFs and how to route all the demands to satisfy their request. We assume the VNF to be capacitated, as well as the network connections, thus limiting the amount of traffic that can pass on each network link and be served by each VNF. The goal is to minimize the number of allocated VNF instances.

Besides its relevance from the application point of view, the problem is interesting and worth studying from an optimization point of view, even with only one type of service. Indeed, the problem combines features of the facility location and the network routing problems, but the very way in which they are combined is new and produces an interesting, challenging problem.

We will focus on the underlying problem structure, rather than addressing a very spe-

cific version of it. Due to its relevance from the application point of view, the problem has been widely addressed in the telecommunication literature, where the solution proposed have been mainly models and heuristics. However, only in [1], there is a comparison between formulations. Even the most promising formulation proposed in [1] cannot deal with large size instances. The goal of this work is therefore to develop and compare heuristics based on mathematical models to tackle large size instances.

By considering a single service case, we focus on the underlying structure of the problem rather than addressing a real-life case. By modifying the used models, ILP-based heuristics allow us to develop approaches that can be applied (hopefully with the same success) to other versions of the problem.

The paper is organized as follows. In Section 2, we report a short literature review about VNFs location and routing problems. In Section 3, we describe the problem and the formulation used in the ILP-based heuristics. We describe the heuristics in Section 4. In Section 5, we report the comparison of the different heuristics and the assessment of their performances. Conclusions (Section 6) end the paper.

2. Literature

Although VNFs related optimization problems have appeared rather recently (see, e.g., the early works in [2] and [3]), they have received significant attention, especially in the telecommunication community. Due to the many features that can be considered, several variants of the problem have been considered in the literature, the main focus being the application perspective, and, as a consequence, formulations and problem-tailored heuristics.

From the point of view of the problem features and definition, the objective functions and the constraints may differ and yield to several variants of the problem. Objective functions are mainly related to VNF costs: they can be considered on their own (see, e.g., [4] and [5]) or combined with the link costs (see, e.g., [6] and [7]). Links and VNFs or nodes are considered as capacitated in almost all the studies.

The different technological features considered give rise to different variants of the problem as well, and to different constraints, such as maximum allowed end-to-end latency [4], [6], [8], partial or total order in the VNFs chain [6], incompatibility among

VNFs [6]. In some works, even when is not considered explicitly as a constraint, the maximum allowed delay is introduced as a penalization in the objective function [9].

Besides the main application environment, represented by a wired telecommunication network to which end-users are connected (the so-called NFVI-PoP), the same underlying problem structure can be found in many contexts and with different network architectures, such as Mobile core [10], DataCenters [11], or Wireless [12]. This leads to a wide number of variants of the problem, and the subsequent research works can be hard to classify and compare in terms of computational efficiency.

Focusing on the solution approach, ILP models are often proposed together with classical constructive and improving heuristics [13], [2]. ILP formulations are used also within column generation approaches [14], [15], relaxation and rounding [16], or Benders' decomposition approaches [17], [18], and formulation-based heuristics [6].

In some papers (e.g., [19], [20], [9]), besides the offline and static version, an online version is also proposed, where the demands are not present all the time, but may appear and disappear. Such a version is usually dealt with using an incremental version of the approaches proposed for the offline case.

In most of the papers, only a specific version of the problem is handled, and only few case studies based on the same topology are considered, thus making it difficult to compare the proposed approaches or to derive wide spectrum insights. A guide to the broad literature, mainly from the telecommunication community, can be found in [21].

In this paper, we consider the underlying structure of the problem, focusing on the features that are common to almost all the versions of the problem. We consider the VNFs opening problem together with the demand assignment and routing problem. We assume that VNFs and links are capacitated. The VNF cost minimization is a very commonly used objective function in the literature, we apply the same metric. As we consider all the nodes equally expensive, our objective turns out to be the minimization of the number of open VNFs. The considered problem has been studied from the computational complexity perspective in [22], where it was proved to be NP-complete even if only one capacity is considered (either link or VNF). Different formulations are analyzed and compared in [1]. We exploit the results of such analysis, and we develop ILP-based heuristics with the goal of successfully tackling large-size instances, in terms of network

size and number of demands. We tested our methods on a large data-set including instances with different topologies and capacities. This element is not common in the current literature.

3. Problem description & formulation

We consider a problem where a set of demands must be routed on a telecommunication network and must receive a service provided by VNF instances allocated on the computing nodes. Each computing node is connected to a telecommunication node. The demands originate and end in telecommunication nodes. An example of such a system, with six telecommunication nodes, is illustrated by Figure 1. The telecommunication nodes are represented as circles, while the computing nodes are represented as squares. There are two demands: the first one has origin in node 1 and destination in node 6, and the second one has origin in node 1 and destination in node 5. In the figure, we also report a possible solution. The first demand is routed along the path $1-3-4-6$ and is served by the VNF allocated on the computing node connected to (telecommunication) node 3, namely 3-C. The second demand is routed along the path $1-2-5$ and is served by the VNF allocated on the computing node 5-C.

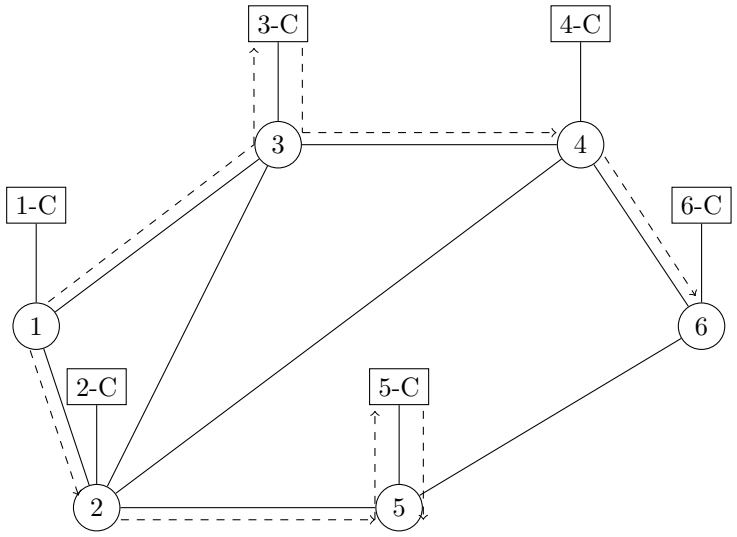


Figure 1: Small example - original graph: computing facilities are represented as square nodes.

As each telecommunication node is connected with one and only one computing node and vice versa, without loss of generality, we collapse each computing node into the connected telecommunication node and represent them with a single node, as illustrated by Figure 2.

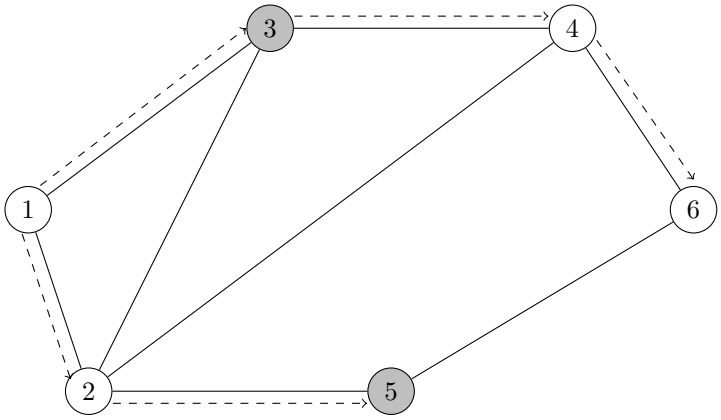


Figure 2: Small example - collapsed graph: computing nodes are merged with the telecommunication nodes. In light-gray we highlight nodes that host a VNF.

The network is represented by a graph $G(N, A)$, where N represents the set of nodes and A represents the set of arcs (or links). An instance of the VNF can be installed on any node in N, as we assume that computing capability is available on each node of the network. If needed, such a representation can be used also if computing capability is available only on a subset of nodes. We assume uniform capacities both for links and VNF instances: let u denote the arc capacity and q the VNF or service (in what follows we will use these two terms interchangeably) capacity. The set D represents the network demands: each demand $k \in D$ is characterized by a source (origin) node o_k, a destination (terminal) node t_k and a demand amount d_k. The demand must be served by (i.e., pass through) an instance of the VNF, but a demand can pass through a node without using the service installed on it. Demands cannot be split and must be routed on simple paths. The problem is to decide on which nodes to open a VNF instance together with the routing and the VNF assignment of each demand.

In this paper, we propose matheuristic approaches that first try to compute a feasible solution and then refine it. The approaches are mainly based on the k-opt neighborhood

Notation	
Sets	
N	set of nodes
A	set of arcs
D	set of demands
Capacities	
u	arc capacity
q	service capacity
Demand parameters	
o_k	origin of demand $k \in D$
t_k	destination of demand $k \in D$
d_k	amount of demand $k \in D$
Assignment/location variables	
y_i	1 if a service is opened on node $i \in N$
z_i^k	1 if demand $k \in D$ uses the service on node $i \in N$
Routing variables	
x_{ij}^{k1}	1 if arc $(i,j) \in A$ is used by demand k in sub-path 1
x_{ij}^{k2}	1 if arc $(i,j) \in A$ is used by demand k in sub-path 2

Table 1: Mathematical notation.

search [23]. We use the most promising formulation analyzed in [1]. In the following, we briefly describe such formulation. The opening of a VNF instance on node $i \in N$ is modelled with a binary variable y_i, which is equal to 1 if a VNF instance is opened on node i and 0 otherwise. The assignment of demand k to the instance of the VNF opened on the node i is modelled with a binary variable z_i^k. To model the demand k passing through a VNF instance, the demand's path is split into two sub-paths: the first one from the origin o_k to the service node to which k is assigned (described by binary variables x_{ij}^{k1}) and the second one from the service node to the destination t_k (described by binary variables x_{ij}^{k2}). Table 1 recaps the notation and the variables.

The resulting model (Split Path - SP) is:

$$\text{(SP)}\ \min \sum_{i \in N} y_i \tag{1}$$

$$\sum_{i \in N} z_i^k = 1 \qquad \forall k \in D \tag{2}$$

$$z_i^k \leq y_i \qquad \forall k \in D, i \in N \tag{3}$$

$$\sum_{k \in D} d_k (x_{ij}^{k1} + x_{ij}^{k2}) \leq u \qquad \forall (i,j) \in A \tag{4}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{k1} - \sum_{j:(j,i)\in A} x_{ji}^{k1} = \begin{cases} 1 - z_i^k & \text{if } i = o_k \\ -z_i^k & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N \tag{5}$$

$$\sum_{j:(i,j)\in A} x_{ij}^{k2} - \sum_{j:(j,i)\in A} x_{ji}^{k2} = \begin{cases} z_i^k - 1 & \text{if } i = t_k \\ z_i^k & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N \tag{6}$$

$$\sum_{j:(j,i)\in A} (x_{ji}^{k1} + x_{ji}^{k2}) \leq 1 \qquad \forall k \in D, i \in N \tag{7}$$

$$\sum_{j:(i,j)\in A} (x_{ij}^{k1} + x_{ij}^{k2}) \leq 1 \qquad \forall k \in D, i \in N \tag{8}$$

$$\sum_{k \in D} d_k\ z_i^k \leq \bar{q_i}\ y_i \qquad \forall i \in N \tag{9}$$

$$\sum_{i \in N} y_i \geq VNF_{LB} \tag{10}$$

$$y_i \in \{0,1\} \qquad \forall i \in N \tag{11}$$

$$z_i^k \in \{0,1\} \qquad \forall i \in N, k \in D \tag{12}$$

$$x_{ij}^{k1}, x_{ij}^{k2} \in \{0,1\} \qquad \forall (i,j) \in A, k \in D \tag{13}$$

where

$$\bar{q_i} = \min \left\{ q, \max \left\{ \sum_{(i,j)\in A} u + \sum_{k \in D: t_k = i} d_k, \sum_{(j,i)\in A} u + \sum_{k \in D: o_k = i} d_k \right\} \right\} \tag{14}$$

Equation (14) strengthens the value of the capacity parameters. Indeed, the amount of demand that can be served by a service opened on the node i is limited not only by the service capacity q, but also by the overall capacity of the links incident in i. However,

the demands originated in i can be served in i without impacting the incoming links' capacity. Similar reasoning can be applied to the outgoing arcs, leading to the value of $\bar{q}_i$.

Constraints (2) force each demand to be assigned to one service. Constraints (3) guarantee that a demand is assigned to a node only if a service instance is opened on the node. Constraints (4) are the link capacity constraints. Constraints (5) and (6) are flow balancing constraints, and they link the choice of a service to the first half (and second half) of the routing path (respectively). Constraints (7) and (8) forbid the two paths from entering both into (or going both out from) the same node, thus forbidding the use of the same link (represented by the two arcs (i, j) and (j, i)) for both sub-paths. These constraints, together with the routing constraints (5)-(6), imply that a simple path is used to route each demand. Isolated cycles with no VNF may occur, which however can be removed without worsening the objective function, as shown in [1]. Constraints (9) limit the amount of demand served by each service instance, and link the opening and assignment variables.

Constraint (10) bounds the number of needed service instances. In [1] the value of the bound VNF_{LB} is set equal to $\left\lceil \frac{\sum_{k \in D} d_k}{q} \right\rceil$. We further improve the formulation by computing the value of VNF_{LB} as the optimal solution of the following Bin-Packing-like Problem (BPP), which uses the same parameters and variables summarized in Table 1:

$$
\begin{aligned}
(BPP)\ \min & \sum_{i \in N} y_i & & \\
& \sum_{i \in N} z_i^k = 1 & & \forall k \in D \\
& \sum_{k \in D} d_k \; z_i^k \leq \bar{q}_i \; y_i & & \forall i \in N
\end{aligned}
$$

However, if solving (BPP) proves to be too time-consuming, we resort to the looser bound and set $VNF_{LB} = \left\lceil \frac{\sum_{k \in D} d_k}{q} \right\rceil$.

4. Heuristic approaches

In this paper we present the most promising approaches proposed in [21]. All of them are two step matheuristic methods:

1. Finding a starting feasible solution;

2. Improving the feasible solution from the first step with a local search procedure.

In both steps, mathematical programming models other than SP are solved using a standard solver. These models have been developed to speed up the solution phase. We derived them by the original SP by fixing some variables, changing the objective function, or reducing the search space.

We propose two different strategies to find a feasible solution and two different local search procedures, obtaining a total of four different complete matheuristics. In the following paragraphs, we will first sketch the key elements used to compose our strategies, and then show how we combined them into a whole matheuristic approach.

4.1. Key ingredients of the matheuristics

In this section, we describe the key ingredients that compose the matheuristics. The first one is a sub-problem that aims at serving the maximum number of demands with a given number of VNFs, rather than assigning all of them and minimizing the number of VNFs. The second ones are the k-opt constraints that are used to define the neighborhoods of the local search.

4.1.1. Maximization of served demands model

The routing sub-problem, namely the problem of finding a feasible path for each demand given the location of the VNF instances, is in itself a challenging problem. In fact, it is difficult to even check feasibility when a VNF instance is open in each node, as it is a special case of the integer multicommodity flow problem. To speed up the feasibility check for a given number of VNF instances VNF_{fix}, we exploit an auxiliary problem, whose goal is to maximize the number of served demands. We refer to it as the Split Path Maximizing the number of served demands problem (SP-MD). As we want to keep the representation of the auxiliary problem as similar as possible to the original, we introduce a slightly different graph. We add one auxiliary node a equipped with a VNF instance with infinite capacity. An uncapacitated arc is added from every demand source node to the auxiliary node and from the auxiliary node to every demand destination node. The resulting graph is denoted as:

$$G^{\mathrm{MD}} = (N \cup \{a\}, A^{\mathrm{MD}}),$$

where N is the set of original nodes and A^{MD} is the increased set of arcs:

$$A^{\mathrm{MD}} = A \cup \{(s_k, a) : k \in D\} \cup \{(a, t_k) : k \in D\}$$

A trivial feasible solution can always be found on such a graph: each demand is served by the service open in a and is routed on the two-arcs path from its source to a and from a to its destination. Then, the auxiliary problem (SP-MD) is defined as follows:

$$\text{(SP-MD)} \ \max \sum_{k \in D, i \in N} z_i^k \tag{15}$$

$$\sum_{i \in N \cup \{a\}} z_i^k = 1 \qquad \forall k \in D \tag{16}$$

$$\sum_{j:(i,j) \in A^{\mathrm{MD}}} x_{ij}^{k1} - \sum_{j:(j,i) \in A^{\mathrm{MD}}} x_{ji}^{k1} = \begin{cases} 1 - z_i^k & \text{if } i = o_k \\ -z_i^k & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N \cup \{a\} \tag{17}$$

$$\sum_{j:(i,j) \in A^{\mathrm{MD}}} x_{ij}^{k2} - \sum_{j:(j,i) \in A^{\mathrm{MD}}} x_{ji}^{k2} = \begin{cases} z_i^k - 1 & \text{if } i = t_k \\ z_i^k & \text{otherwise} \end{cases} \qquad \forall k \in D,\ i \in N \cup \{a\} \tag{18}$$

$(3), (4), (7), (8), (9)$

$$\sum_{i \in N} y_i = VNF_{fix} \tag{19}$$

$$\sum_{i \in N:(i,a) \in A^{\mathrm{MD}}} (x_{ia}^{k1} + x_{ia}^{k2}) \leq z_a^k \quad \forall k \in D \tag{20}$$

$$y_i \in \{0, 1\} \qquad \forall i \in N \tag{21}$$

$$z_i^k \in \{0, 1\} \qquad \forall i \in N \cup \{a\}, k \in D \tag{22}$$

$$x_{ij}^{k1}, x_{ij}^{k2} \in \{0, 1\} \qquad \forall (i, j) \in A^{\mathrm{MD}}, k \in D \tag{23}$$

The objective function (15) maximizes the number of demands served by the given number of open VNFs, imposed by constraint (19). Constraints (20) prevent from using arcs incident in a for routing a demand that is not served by the VNF on a. Indeed, such arcs do not belong to the original set of arcs, and cannot be used for finding a feasible routing on the original graph.

Constraints (3), (4), (7), (8), and (9) are the same as in SP and are defined only on nodes and arcs of the original graph G. Instead, constraints (2), (5) and (6) are replaced by (16), (17) and (18), respectively, as such constraints are extended to the graph G^{MD}.

4.1.2. Local search based on k-opt neighborhood

A k-opt neighborhood [23] is defined as the set of solutions such that only a given number of changes for the variables is allowed w.r.t. the current solution. The neighborhood is explored solving an ILP model, whose goal is to find an improving solution. As an example, let us consider a general binary problem, with variables ξ. The current solution is described by the set of variables equal to 1, $\bar{S}$. The k-opt neighborhood is given by the set of feasible solutions satisfying the additional constraint

$$\sum_{i \in \bar{S}} (1 - \xi_i) + \sum_{i \notin \bar{S}} \xi_i \leq \kappa \tag{24}$$

where κ is the maximum allowed number of changes, defining the size of the neighborhood.

In our problem, we may limit the number of changes in VNF opening or in the assignments. Thus, we consider two different k-opt constraints.

k-opt-L constraint limiting the changes in the opened VNF:

Let us denote with $\bar{S}_f$ the set of nodes selected to host a VNF instance and with κ_f the maximum allowed number of changes. The k-opt constraint limits the number of opening variables that can change their values w.r.t. the current solution:

$$\sum_{i \in \bar{S}_f} (1 - y_i) + \sum_{i \in N \setminus \bar{S}_f} y_i \leq \kappa_f \tag{25}$$

k-opt-A constraint limiting the assignments changes:

Let us denote with $\bar{S}_d \subset D \times N$ the assignment selected in the current solution and with κ_d the maximum number of changed assignments. The k-opt constraint is as follows:

$$\sum_{(k,i) \in \bar{S}_d} (1 - z_i^k) + \sum_{(k,i) \in (D \times N) \setminus \bar{S}_d} z_i^k \leq \kappa_d \tag{26}$$

Such constraints can be used either separately or in combination, leading to different neighborhoods. We call 'L' neighborhood the neighborhood where the number of changes in the VNF's opening is limited (defined by constraint (25)), and 'A' neighborhood the one where the number of changes in the assignments is limited (defined by constraint (26)). The term 'LA' is used for their combination (using both constraints (25) and (26)).

4.1.3. Summary of the formulations used in the matheuristics

Combining the described models with the k-opt constraints, we formulated several models. The four models that we use in the two different heuristic steps[1] (initial feasible solution, local search procedure) are reported in Table 2. The last column summarize the steps where each model is used. AFR and DFR are the names of the two first-step procedures.

Formulation name	Model Obj.	Model Constraints	k-opt constraints	Heuristic steps
SP	(1)	(2)-(9), (10)	-	-
SP-MD	(15)	(3),(4),(7)-(9), (16)- (18), (19)-(20)	-	DFR
SP-MD-A	SP-MD		(26)	AFR
SP-L	SP		(25)	DFR, local search
SP-LA	SP		(25), (26)	local search

Table 2: Recap of models.

4.2. Initial solution

Both first-step procedures fix the number of opened VNF instances and try to find a feasible assignment and routing, but they differ in the number of VNF instances they use as a starting point and in the way a feasible assignment and routing is searched. The first approach, All-open Feasible Routing (AFR), opens a VNF instance in each node and tries to serve all the demands. Instead, in the second procedure, Dichotomic placement Feasible Routing (DFR), the number of VNF instances to be open is set through a dichotomic-like procedure. If the dichotomic-like procedure is unable to find a feasible

[1]Some combination were not effective and where discarded during a preliminary analysis, see [21].

solution, a further step is applied, which opens a VNF instance in each node. While both procedures may end up opening a VNF in each node, the DFR procedure may reduce the number of open VNFs, as opposed to AFR, which never does.

In the following, the procedure schemes are reported and commented.

4.2.1. All-open Feasible Routing (AFR)

The AFR scheme is given in Algorithm 1.

Algorithm 1 Overview of the AFR heuristic

1: $VNF_{fix} = |N|$
2: $\bar{S}_d = \emptyset$ and $\text{res}^\star = 0$
3: **while** total time-limit is not exceeded **do**
4: res = **solve** (SP-MD-A, solver_TL)
5: **if** $res == |D|$ **then**
6: **return** solution with number of VNF equal to $|N|$ and assignment $\bar{S}_d$
7: **end if**
8: **if** $\text{res}^\star < \text{res} < |D|$ **then**
9: $\bar{S}_d$ = current assignment solution
10: $\text{res}^\star = \text{res}$
11: **end if**
12: **end while**
13: **return** "No feasible solution found"

AFR opens (and keeps) a VNF instance in each node (line 1). It tries to find a feasible assignment and routing solution applying a k-opt neighborhood based local search on the assignments to the problem of maximizing the number of served demands (line 4). It starts with the trivial solution where no demand is served by the VNFs on the original nodes and all the demands are assigned to the auxiliary node a. The partial assignment solution $\bar{S}_d$ is empty and the number of assigned demands $\text{res}^\star$ is equal to 0 (line 2). The k-opt neighborhood based local search tries to increase the number of served demands. A time-limit is set for solving SP-MD-A (denoted as solver_TL in the algorithm scheme). The overall procedure ends if one of the following conditions holds: i) all the demands can be served and therefore a feasible solution with $|N|$ VNFs for the original problem

is provided, ii) the overall time-limit is reached, iii) there is no improving solution in the neighborhood. In the two latter cases, no feasible solution is provided.

4.2.2. Dichotomic placement Feasible Routing (DFR)

The DFR scheme is given in Algorithm 2.

Algorithm 2 Overview of the DFR heuristic

1: stop = false, $VNF_{fix} = (|N| + VNF_{LB})/2$
2: **while** total time-limit is not exceeded or stop == false **do**
3: res = **solve** (continuous relaxation of SP-MD, solver_TL)
4: **if** $res == |D|$ **then**
5: **solve** (SP-MD, solver_TL)
6: **if** $res == |D|$ **then**
7: **return** solution with number of VNF equal to VNF_{fix}
8: **end if**
9: **end if**
10: **if** $VNF_{fix} \geq |N|$ **then**
11: $stop = true$
12: **else**
13: $VNF_{fix} = (|N| + VNF_{fix})/2$
14: **end if**
15: **end while**
16: **if** total time-limit is not exceeded **then**
17: **return** output of the **Recovery strategy** (Algorithm 3)
18: **end if**

We start with the number of VNF instances VNF_{fix} equal to the middle value between opening all the nodes and the lower bound VNF_{LB}, i.e., $VNF_{fix} = (|N| + VNF_{LB})/2$ (line 1). Although the VNF_{LB} might be feasible, preliminary tests showed that the BPP bound is loose in most of the cases. Thus, starting from this value would excessively slow down the overall procedure.

We perform the following steps:

- The continuous relaxation of model SP-MD is solved with the given value VNF_{fix} (line 3).
- If the continuous relaxation assigns and routes all the demands, then a feasible solution for the integer problem is searched by solving SP-MD (line 5). Otherwise, the integer problem solution is skipped.
- If the procedure is unable to serve all the demands, the number of opened service VNF_{fix} is increased. Its value is the middle point of the new search interval that is delimited by $|N|$ (opening all the nodes) and the previous VNF_{fix} (line 13).

A time-limit is set for each single run of the solver. Checking the solution of the continuous relaxation allows us to quickly identify unfeasible problems, speeding up the overall procedure.

The dichotomic-like procedure ends if one of the following conditions holds: i) a feasible solution is found (line 7), and the solution with a number of VNFs equal to VNF_{fix} is returned, ii) the time-limit of the overall procedure is exceeded (line 2), iii) a VNF has been opened in each node and yet a feasible routing has not been found (line 11).

Algorithm 3 Recovery strategy

1: **if** total time not exceeded **then**
2: $\quad \bar{S}_f = N$;
3: $\quad$ **solve** (SP-L, solver_TL)
4: $\quad$ **if** problem solved **then**
5: $\quad\quad$ **return** solution with num. of VNF between $|N| - k_f$ and $|N|$
6: $\quad$ **end if**
7: **end if**
8: **return** "No feasible solution found"

If no solution is found and the overall time-limit is not exceeded, a further step is performed. We call this step Recovery strategy, and we report its scheme in Algorithm 3. A VNF instance is opened in each node (line 2) and the model SP-L is solved (line 3): both assignment and routing variables can be modified, while at most κ_f VNF instances

can be closed. The SP-L is solved only once, as the goal of the procedure is to quickly find a feasible solution, and not to obtain the smallest number of VNF instances. Reducing the search space using constraint (25) allows us to speed up the search. If even such a step fails, the procedure returns a no feasible solution output.

4.3. Improving step

Once a feasible solution is found, a k-opt neighborhood search is applied to improve it. The scheme of the local search based on the k-opt neighborhood is described in Algorithm 4.

At each iteration, the procedure explores the neighborhood of the current solution $\bar{S}$, solving an ILP model of the problem with additional constraints of the form (24). If an improving solution is found, then the current solution is updated (line 5). A feasible solution must be provided to initialize the local search.

Algorithm 4 Overview of the local search procedure

Input: a feasible solution $\bar{S}$ (output of AFR or DFR)

1: $res^\star = value(\bar{S})$
2: improve = True
3: **while** improve and total time-limit is not exceeded **do**
4: res = **solve** (SP + k-opt constraint (L or LA), solver_TL)
5: **if** $res \leq res^\star$ **then**
6: $\bar{S}$ = current solution
7: $res^\star = res$, improve = True
8: **else**
9: improve = False
10: **end if**
11: **end while**
12: **return** $\bar{S}$, $res^\star$

Two different neighborhoods are considered, L and LA, as described in Section 4.1.2. Neighborhood L limits the changes in the opening variables using constraint (25), while, neighborhood LA limits the changes in both opening and assignments instead, using

constraints (25) and (26). The two neighborhoods are applied to both initial solution procedures, thus yielding to 4 different heuristics.

In the LA neighborhood, when a local minimum is reached, the size of the assignment based neighborhood is increased by setting the value of κ_d equal to $|D|$ (thus allowing to change up to half of the current demand to VNF assignments). If an improving solution is found, the local search size is restored to its original value. Otherwise, the procedure stops, having explored two neighborhoods of different size without improvement. After some preliminary tests, we decided to fix the k-opt parameters as follows:

- κ_f (on opening variables) to $\lceil |N|/10 \rceil$
- initial κ_d (on assignment variables) to $\lceil |D|/2 \rceil$.

The 4 different matheuristics are summarized in Table 3.

Method	Init. solution	Improving step
AFR-L	AFR	k-opt-L
AFR-LA	AFR	k-opt-LA
DFR-L	DFR	k-opt-L
DFR-LA	DFR	k-opt-LA

Table 3: Summary of matheuristic approaches.

5. Computational results

This section is devoted to the analysis of the computational results. First, we describe the test settings (Section 5.1) and the structure of the tables and the figures (Section 5.2); later, we present two different analyses: we compare the performance of the heuristics among themselves and with the solver (Section 5.3); finally, we evaluate the impact of combining two different heuristics (Section 5.4).

5.1. Test settings

The proposed heuristics have been coded in AMPL and tested on a set of instances based on the SNDLib [24]. The models are solved with IBM ILOG CPLEX 12.7.1.0. The

tests run on an Intel Xeon, CPU E5-1620 v2 (4 cores), 3.7 GHz with 32 GB of RAM. We set a tree memory limit of 3000 MB. The time-limits depend on the instances, and we will detail them below. We use the solution of the model SP(1)-(10) with a standard solver as a term of comparison. We set a 10 seconds time-limit for the Bin-Packing-based bound computation. If no solution is found within the time-limit, we resort to the bound obtained by rounding up the ratio between the overall demand and the service capacity.

The test set is the same presented in [21] and [1]. We report here a short description of how the instances were generated. Network topologies and demands are taken from the SNDLib. We generated link capacity (u) and service capacity (q) as follows.

As for the services, three levels of capacity are considered: low, medium, and high. In the high capacity case, the parameter q is set to guarantee that all the demands can be served by a single VNF instance (service uncapacitated instances), namely $q = \sum_{k \in D} d_k$. In the low VNF capacity case, the parameter q is twice the total amount of the demands divided by the number of nodes $q = \left\lceil 2 \frac{\sum_{k \in D} d_k}{|N|} \right\rceil$, which means, we need to install a VNF instance in at least half of the nodes. Medium capacity is the average between the high one and the low one.

As for the links, we consider just one of the levels proposed in [21] and [1], as it turned out that instances with higher link capacity values can be easily solved by a standard solver (see computational results of [1]). Therefore, the link capacity u is computed as the minimum capacity for a feasible routing to exist, neglecting the services. This value is obtained solving a suitable MILP model.

Preliminary results allowed us to group instances in different classes based on the performances of the solver applied to the SP formulation. We distinguish three classes:

group1: topologies that the solver can solve to optimality within one hour for any capacity setting. For these instances we set an overall time-limit of 10 minutes for the heuristics and a time-limit of 600s for every single call to the solver within the heuristic;

group2: topologies that the solver cannot solve to optimality within one hour (or due to the memory limit) for at least one capacity setting. For these instances we set an overall time-limit of 20 minutes for the heuristics and a time-limit of 600s for every single call to the solver within the heuristic;

group3: topologies that the solver cannot solve to optimality within two hours (or due to the memory limit) for at least one capacity setting. For these instances, we used two overall time-limits for the heuristics: one hour and two hours (the two hours time-limit is used to perform tests combining the heuristics). In both cases, a time-limit of 900s is set for every single call to the solver.

The 48 instances are presented in Table 4, where for each instance size (number of nodes, number of arcs and number of demands) and capacity values are reported.

	Data from SNDLib				Capacity			
					VNF (q)			Link (u)
	Network	$\|N\|$	$\|A\|$	$\|D\|$	high	medium	low	
group1	atlanta	15	22	210	136726	77478	18230	19404
	geant	22	36	462	2999992	1636359	272726	359868
	nobel-eu	28	41	378	1898	1016	135	214
	nobel-us	14	21	91	5420	3097	774	486
	polska	12	18	66	9943	5800	1657	995
	sun	27	51	67	476	255	35	53
group2	france	25	45	300	99830	53908	7986	9413
	janos-us	26	42	650	80000	43076	6153	7624
	newyork	16	49	240	1774	997	221	66
	norway	27	51	702	5348	2872	396	358
group3	cost266	37	57	1332	679598	358166	36735	53562
	germany50	50	88	662	2365	1229	94	123
	giul39	39	86	1471	7366	3871	377	363
	india35	35	80	595	3292	1740	188	121
	janos-us-ca	39	61	1482	2032274	1068246	104219	180471
	pioro40	40	89	780	115953	60875	5797	7609

Table 4: Instances' details.

5.2. Tables and figures description

We report the results for the model and the heuristics in Tables 5, 6 and 7, for group1, group2 and group3, respectively. In each table, the instances are grouped by the VNF capacity value (high, medium or low).

For each instance, we report:

- for the SP model solution:
  - the dual bound (lower bound - LB)
  - the best integer solution found (upper bound - UB)
  - the computational time (Time)
- for the heuristics (Sol Heur), as a summary:
  - the best value found by the heuristics (Best)
  - the worst value found by the heuristics (Worst)
- for each heuristic:
  - the solution (Sol)
  - the computational time (Time)

The symbol '-' represents the cases where no solution can be found. 'ML' stands for memory limit. When the time-limit is reached, we report its value, i.e., one hour ('1h'), for the solver, and 10 minutes ('10m'), 20 minutes ('20m') and one hour ('1h') for the heuristics on group1, group2 and group3, respectively. We highlight in bold the best result obtained by the heuristic(s) and with a light-gray background the heuristic that finds the best value within the smallest computational time.

We report summarized results in Figures 3, 4 and 5. In Figure 3, for group1, we report:

#best: the number of instances for which the heuristic finds the best solution among the solutions found using the heuristics,

#opt: the number of instances for which the heuristic finds the same UB as the model,

#faster: the number of instances for which the heuristic finds a feasible (not necessarily optimal) solution and requires a smaller computational time than the model.

In Figures 4 and 5, for group2 and group3, respectively, we report:

#best: as for group1,

#UB imp: the number of instances in which the heuristic finds the same number of VNFs as the model or a smaller one.

We report **# UB imp** instead of **#opt**, because the model does not always find the optimal solution on these instances. We do not report the value #faster because the model reaches its time-limit in almost all the instances (except for some low capacity ones) and the time-limit imposed on the heuristics (600s for group2, 1 hour for group3) is always smaller than the one imposed on the solver.

5.3. Heuristics comparison

In this section, we compare the proposed heuristics and the model. We show that the heuristics can be successfully applied and can outperform the model.

The model provides good results on the group1 instances. Its behavior worsens for the other groups (especially for the high and the medium capacity cases), where the heuristics improve upon it. The heuristics are, in general, faster than the model.

There is not a clear winner among the heuristics. In general, they provide rather similar results. The best performing heuristic is not the same for all the instances. Nevertheless, the best and the worst values are similar, at least as far as group1 and group2 are concerned. The difference increases for the instances of group3. When they do not provide similar results, the AFR and DFR-based heuristics are somehow complementary. In fact, when the performance of one worsens, the performance of the other improves. A question then arises if combining the two different policies is profitable. This question is addressed in Section 5.4.

We hereby report a detailed analysis of the results, a subsection for each group of instances. We report more detailed results in Appendix A: the comparison of the first step heuristics (AFR and DFR) and the impact of the local search procedure on the overall algorithm.

5.3.1. Group1 instances

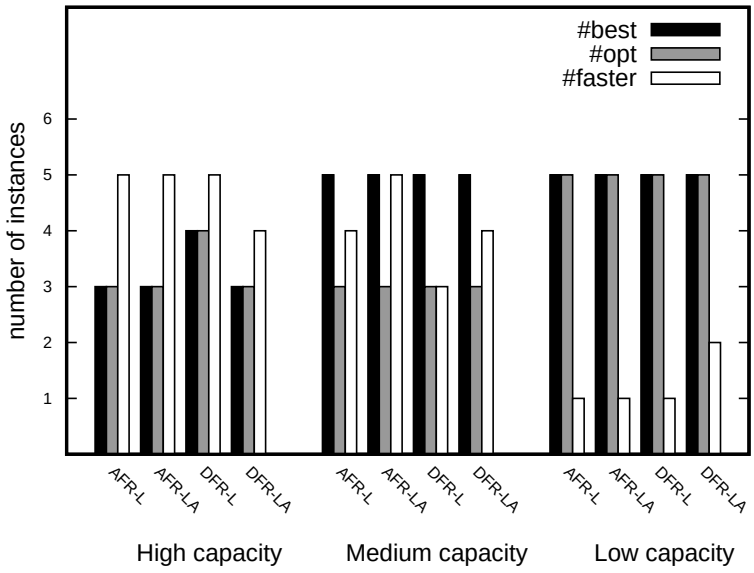


Figure 3: Summary of heuristics' results on instances of group1 for different cases of VNF capacity.

On the group1 instances, the heuristic behavior varies depending on the capacity case. The polska instance represents an exception: it is very challenging for the heuristics, but not for the model. Indeed, polska can be solved using the model for all the capacity cases, while only the AFR-based heuristics can find a feasible solution for the medium capacity case.

Put aside the instance polska, the heuristics show a good behavior in the high capacity case. Heuristics are in general faster than the model (the only exception being DFR-LA on instance nobel-us). They fail to find the optimal solution in 1 or 2 cases. When they fail, their solution has only one VNF instance more than the optimal one. On the overall, DFR-L behaves slightly better than the others, while DFR-LA provides the poorest performance.

All the heuristics provide the same solution in the medium capacity instances, and they all fail in finding the optimum in two out of five instances (leaving aside polska, which is challenging in all the capacity cases, as mentioned). AFR-LA provides the best performance, being the fastest on most of the instances (it is outperformed just twice,

and not by the same heuristic), while DFR-L is faster than the model in only three instances. AFR-L and DFR-L reach the time-limit in nobel-us (and polska).

The low capacity instances are not challenging for the model nor for the heuristics, aside from polska. In fact, all the heuristics can find the optimal solution in all the instances. They are slower than the model in all the instances but atlanta, except DFR-LA, which is faster than the model in both atlanta and nobel-us.

	Instance	SP			Sol Heur		AFR-L		AFR-LA		DFR-L		DFR-LA	
	Network	LB	UB	Time	Best	Worst	Sol	Time	Sol	Time	Sol	Time	Sol	Time
high	atlanta	3	3	1528.7	3	4	4	43.7	4	62.8	**3**	29.8	4	24.1
	geant	1	1	299.0	1	1	**1**	175.7	**1**	164.4	**1**	84.3	**1**	94.7
	nobel-eu	3	3	1406.5	3	3	**3**	410.6	**3**	206.0	**3**	438.4	**3**	481.4
	nobel-us	4	4	265.9	4	5	**4**	65.9	**4**	98.9	**4**	116.1	5	10m
	polska	4	4	534.2	-	-	-	10m	-	10m	-	10m	-	10m
	sun	2	2	44.1	2	3	3	33.9	3	31.6	3	18.8	**2**	9.8
medium	atlanta	3	3	800.3	4	4	**4**	47.4	**4**	38.3	**4**	51.1	**4**	48.3
	geant	2	2	241.8	2	2	**2**	218.7	**2**	113.9	**2**	255.1	**2**	304.9
	nobel-eu	3	3	2595.9	3	3	**3**	10m	**3**	285.0	**3**	10m	**3**	339.9
	nobel-us	4	4	262.4	4	4	**4**	80.6	**4**	94.1	**4**	150.3	**4**	101.9
	polska	4	4	902.4	6	-	**6**	10m	**6**	10m	-	10m	-	10m
	sun	2	2	213.0	3	3	**3**	29.3	**3**	19.2	**3**	12.1	**3**	86.2
low	atlanta	8	8	58.1	8	8	**8**	17.6	**8**	32.4	**8**	13.6	**8**	13.2
	geant	12	12	23.1	12	12	**12**	79.8	**12**	163.6	**12**	51.9	**12**	71.8
	nobel-eu	15	15	24.7	15	15	**15**	62.7	**15**	78.8	**15**	53.7	**15**	61.8
	nobel-us	8	8	10.9	8	8	**8**	24.2	**8**	61.9	**8**	19.9	**8**	5.3
	polska	7	7	286.0	-	-	-	10m	-	10m	-	10m	-	10m
	sun	14	14	1.7	14	14	**14**	8.8	**14**	14.6	**14**	5.4	**14**	22.0

Table 5: Results of heuristics on group1 compared with ILP model solved by CPLEX. Computational time is expressed in seconds. The time-limit for heuristics is set to 10 minutes ($10m = 600s$).

5.3.2. Group2 instances

The instances of group2 are more challenging both for the model and the heuristics. In fact, for the high and medium capacity case, the model finds an upper bound only in one instance out of four, while for the others it cannot compute a feasible solution within the time-limit (one hour). Instead, the low capacity instances are less challenging for

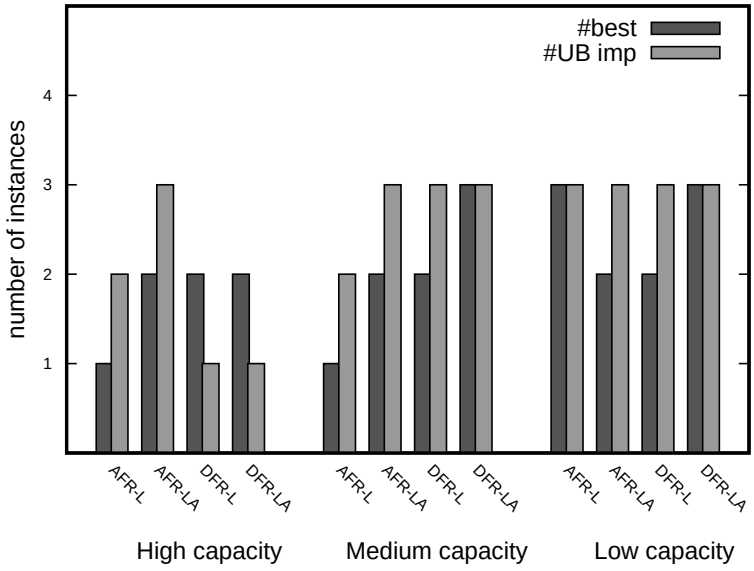


Figure 4: Summary of heuristics' results on instances of group2 for different cases of VNF capacity.

the model, except for the instance france. Although the group2 instances are challenging for the heuristics (they reach the time-limit in almost all the cases), they clearly show the heuristics' benefit. Only in france with high capacity, no heuristic can find a feasible solution. The DRF-based heuristics can find a solution for france with medium and low capacity, while the AFR-based heuristics (and the model) cannot. For the newyork instance, it is the reverse: for both high and low capacities, AFR-based heuristics can find a feasible solution, while the DFR-based heuristics fail. When all heuristics can find a solution, the DRF-based methods perform better. In norway with the low and medium capacity, DRF-based heuristics obtain solutions with a value that is half the one found by the AFR-based ones. The instance newyork with medium capacity represents an exception: AFR-LA can find a solution with a value of 6, while all the other methods find a value of 8 or 9. As mentioned before, the low capacity instances are less challenging for the model. It is also true for the heuristics, which manage to find the optimal solution almost always. Overall, the group2 instances show that the heuristics can outperform the model, by finding feasible solutions where the model cannot. The heuristics' solutions are very close to the lower bound provided by the model (it is worth noting that the

time-limit imposed on the heuristics is one-third of the one imposed on the model).

	Instance	SP			Sol Heur		AFR-L		AFR-LA		DFR-L		DFR-LA	
	Network	LB	UB	Time	Best	Worst	Sol	Time	Sol	Time	Sol	Time	Sol	Time
high	france	4	-	ML	-	-	-	20m	-	20m	-	20m	-	20m
	janos-us	3.91	5	1h	6	6	**6**	20m	**6**	20m	**6**	20m	**6**	20m
	newyork	2.6	-	1h	5	-	9	20m	**5**	20m	-	20m	-	20m
	norway	4.6	-	1h	7	13	11	20m	13	20m	**7**	20m	**7**	20m
medium	france	4	-	ML	14	-	-	20m	-	20m	**14**	20m	**14**	20m
	janos-us	3.9	5	1h	6	6	**6**	20m	**6**	20m	**6**	20m	**6**	20m
	newyork	2.7	-	1h	6	9	8	20m	**6**	20m	8	20m	9	20m
	norway	4.6	-	1h	6	13	13	20m	13	20m	7	20m	**6**	20m
low	france	13	-	1h	17	-	-	20m	-	20m	19	20m	**17**	20m
	janos-us	14	14	84.5	14	14	**14**	116.8	**14**	139.2	**14**	105.1	**14**	122.7
	newyork	9	9	1972.6	9	-	**9**	837.5	14	20m	-	20m	-	20m
	norway	14	14	315.3	14	14	**14**	688.3	**14**	904.3	**14**	833.7	**14**	1021.1

Table 6: Results of heuristics on group2 compared with ILP model solved by CPLEX. Computational time is expressed in seconds. The time-limit for heuristics is set to 20m= 1200s.

5.3.3. Group3 instances

The model can find a feasible solution only in one instance out of 6 of the group3 for the high and medium capacity case. Instead, as for the group2, the low capacity instances are less challenging: the model can solve three of them to optimality (two in a short computational time). Nevertheless, in three instances of the low capacity case, the model cannot find a feasible solution. Furthermore, for giul39 with the high and medium capacity, the model cannot even find a lower bound. On the contrary, all the heuristics can find a feasible solution for all the instances. The AFR-based heuristics perform better than the DFR-based ones on the high and medium capacity cases, with some exceptions. DFR-L obtains a solution that is half the one found by the other heuristics on janos-us-ca with high capacity. DFR-LA finds a solution that is at least one-third smaller than the other heuristics on pioro40 with medium capacity. On the low capacity instances, the DFR-based heuristics perform better than the AFR-based ones. However, except for a few cases (i.e., janos-us-ca with low capacity, pioro40 with medium capacity, and cost266 with low capacity), the difference between the best and the worst

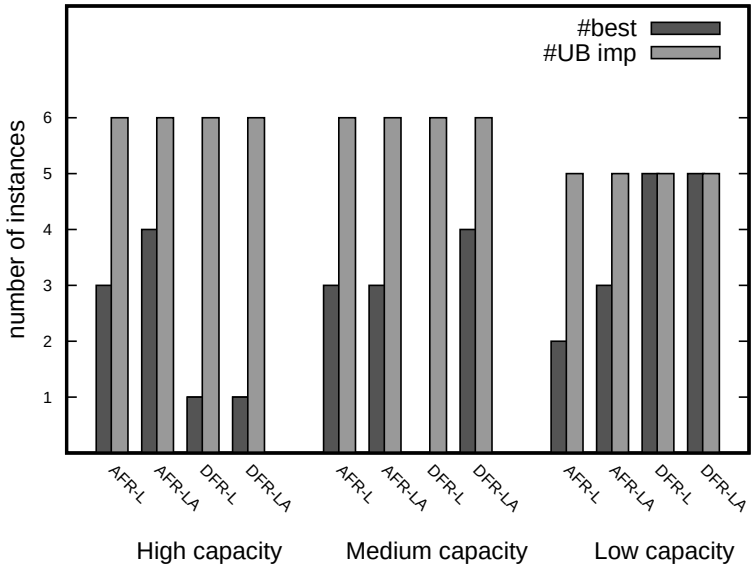


Figure 5: Summary of heuristics' results on instances of group3 for different cases of VNF capacity.

heuristic is not significant. In the low capacity case, the best heuristic provides optimal, or nearly optimal (janos-us-ca and pioro40), solutions.

As for the computational time, low capacity instances are less time-consuming. The local search based on VNF location (L) is, in general, faster. AFR-L reaches the time-limit only in half of the instances with high and medium capacity. DFR-L reaches the time-limit in one instance for the high and low capacity cases and in two for the medium ones. AFR-LA and DFR-LA are more time-consuming: they run out of time in more than half the instances for the high and medium capacity cases and in three for the low capacity ones.

	Instance	SP			Sol Heur		AFR-L		AFR-LA		DFR-L		DFR-LA	
	Network	LB	UB	Time	Best	Worst	Sol	Time	Sol	Time	Sol	Time	Sol	Time
high	cost266	3	6	2h	4	6	**4**	1h	**4**	1h	5	2965.4	6	3169.3
	germany50	3.35	-	2h	5	7	7	2012.4	**5**	2544.6	7	1311.8	6	2583.5
	giul39	-	-	2h	27	30	**27**	1h	**27**	1h	30	1h	30	1h
	india35	3.46	-	2h	8	10	**8**	1h	**8**	1h	10	3053.7	10	1h
	janos-us-ca	2.84	-	2h	6	13	13	1h	12	1h	**6**	3383.1	12	1h
	pioro40	3	-	2h	11	40	37	2143.9	40	2142.6	27	2692.9	**11**	1h
medium	cost266	3	6	2h	4	6	**4**	1h	**4**	1h	5	2892.5	6	3100.6
	germany50	3.35	-	2h	5	7	7	1998.5	**5**	2543.9	7	1278.0	**5**	3341.7
	giul39	-	-	2h	24	30	27	1h	27	1h	30	1h	**24**	3130.2
	india35	3.46	-	2h	8	10	**8**	1h	**8**	1h	10	2936.6	10	1h
	janos-us-ca	2.84	-	2h	6	12	**6**	1h	12	1h	10	1h	**6**	1h
	pioro40	3	-	2h	11	40	37	2142.7	40	2142.8	27	2689.4	**11**	1h
low	cost266	19	-	2h	19	28	28	1939.4	**19**	1934.9	**19**	1663.5	**19**	1893.7
	germany50	26	26	296.3	26	26	**26**	274.3	**26**	509.9	**26**	247.5	**26**	386.6
	giul39	20	20	2754.6	32	36	36	2196.6	33	1h	35	2164.0	**32**	1h
	india35	18	18	287.5	18	18	**18**	1861.3	**18**	1563.9	**18**	1130.1	**18**	1054.4
	janos-us-ca	20	-	2h	23	29	27	2403.3	29	1h	**23**	1h	29	1h
	pioro40	21	-	ML	23	28	28	2877.0	24	1h	**23**	2963.2	**23**	1h

Table 7: Results of heuristics on group3 compared with the ILP model solved by CPLEX. Computational time is expressed in seconds. The time-limit for heuristics is set to 1h= 3600s.

5.4. Extending the time-limit and combining heuristics

In this section, we focus on the group3 instances. The aim is to evaluate if the behavior of the heuristics changes using a longer computational time. More precisely, we use the same time-limit we imposed on the model, i.e., two hours. The results are reported in Table 8. For ease of reading, we report two additional columns from Table 7: the best and worst values found by the heuristics in one hour (columns three and four). We highlight in bold the instances where the result obtained in two hours improves upon the result obtained within one hour.

	Instance	SP	Heur 1h		Heur 2h		AFR-L		AFR-LA		DFR-L		DFR-LA	
		LB	Best	Worst	Best	Worst	Sol	Time	Sol	Time	Sol	Time	Sol	Time
high	cost266	3	4	6	4	6	**4**	4408.2	**4**	4444.4	5	2965.4	6	3169.3
	germany50	3.3	5	7	5	7	7	2012.4	**5**	2544.6	7	1311.7	6	2583.5
	giul39	-	27	30	**15**	30	**15**	6695.6	21	7455.4	18	6885.2	30	5344.9
	india35	3.5	8	10	**6**	10	8	3593.8	**6**	5462.7	10	3053.7	10	3889.8
	janos-us-ca	2.8	6	13	6	**12**	7	5374.8	12	4437.9	**6**	3383.1	12	4157.3
	pioro40	3	11	40	**8**	40	37	2143.9	40	2142.6	27	2692.9	**8**	5644.2
medium	cost266	3	4	6	4	6	**4**	4257.7	**4**	4426.9	5	2892.5	6	3100.6
	germany50	3.3	5	7	5	7	7	1998.5	**5**	2543.9	7	1278.0	**5**	3341.7
	giul39	-	24	30	**15**	**24**	**15**	6596.7	21	7449.9	18	6610.7	24	7385.1
	india35	3.5	8	10	**6**	10	8	3586.3	**6**	5473.3	10	2936.6	7	5595.5
	janos-us-ca	2.8	6	12	6	**7**	**6**	4226.0	**6**	7040.8	7	4530.2	**6**	5218.5
	pioro40	3	11	40	**8**	40	37	2142.7	40	2142.8	27	2689.4	**8**	5571.1
low	cost266	19	19	28	19	28	28	1939.4	**19**	1934.92	**19**	1663.5	**19**	1893.7
	germany50	26	26	26	26	26	**26**	274.3	**26**	509.924	**26**	247.5	**26**	386.6
	giul39	20	32	36	**29**	36	36	2196.6	33	4426.01	35	2164.0	**29**	5843.2
	india35	18	18	18	18	18	**18**	1861.4	**18**	1563.88	**18**	1130.1	**18**	1054.4
	janos-us-ca	20	23	29	**20**	27	**27**	2403.3	**20**	4615.0	**20**	5992.6	21	7052.2
	pioro40	21	23	28	**21**	28	28	2877.0	24	4792.78	23	2963.2	**21**	3656.8

Table 8: Results of heuristics on group3. Computational time is expressed in seconds. The time-limit for heuristics is set to 2h= 7200s.

We shall point out that, in 3 out of 6 of the low capacity instances, the best heuristic can find the optimal solution even within one hour. With a longer time-limit, the best value improves in 9 out of 18 instances. The worst heuristic improves in a few instances: janos-us-ca with both high and medium capacity and giul39 with medium capacity.

If we focus only on the best solution value, it is not easy to determine a clear winner. Indeed, for the high and medium capacity instances, the AFR-based heuristics get better results, in general. Nevertheless, in some instances, their performance is bad (see, for example, pioro40, where AFR-LA obtains a solution with value 40 and DFR-LA a solution with value 8). For low capacity instances, the DFR-based heuristics behave better than the AFR-based ones. The results obtained are never far from the best one, they are often the best (see, for example, janos-us-ca).

This reasoning leads to a very logical question: "can we determine the algorithm that, without being the best, performs well on the largest number of instances?" To answer this question, we introduce performance profiles as proposed in [25]. Performance profiles aim to assess the quality of a set of algorithms on a given data-set. More formally, we call:

- I the set of instances,
- $\mathcal{A}$ the set of algorithms (AFR-L, AFR-LA, ...),
- $f_i(j)$ a measure of the performance of algorithm $j \in \mathcal{A}$ on instance $i \in I$, in our case the obtained objective value,
- $f_i^\star = \min\{f_i(j),\ \forall j \in \mathcal{A}\}$.

Performance profiles are a graphical representation of the following functions:

$$p_j(x) = \left| \{i \in I : \frac{f_i(j)}{f_i^\star} \leq x\} \right| \ \forall j \in \mathcal{A}$$

For the value of $x = 1$, the function $p_j(x)$ corresponds to the number of instances where the heuristic j can find the best solution. For $x = 2$, it is the number of instances where the heuristic j produces a solution with a value at most twice the value found by the best algorithm, and so on.

A "good" algorithm may fail in finding the best solution for some instances, but it guarantees to have a limited error even in the worst case. In performance profiles, it occupies the upper-left part of the graph, and quickly asymptotizes over the maximum number of instances.

In Figure 6, we report the performance profiles for the four heuristics: the first plot reports the performance profiles of AFR-based heuristics, while the second plot reports

the profiles of the DFR-based ones. The AFR-based heuristics obtain the best solution on more instances, but their performance is very poor on the remaining ones. Indeed, in two instances, the obtained value is five times larger than the best one. On the other hand, the DFR-based heuristics find solutions that are, at worst, twice the overall best for all the instances, thus, guaranteeing a form of robustness.

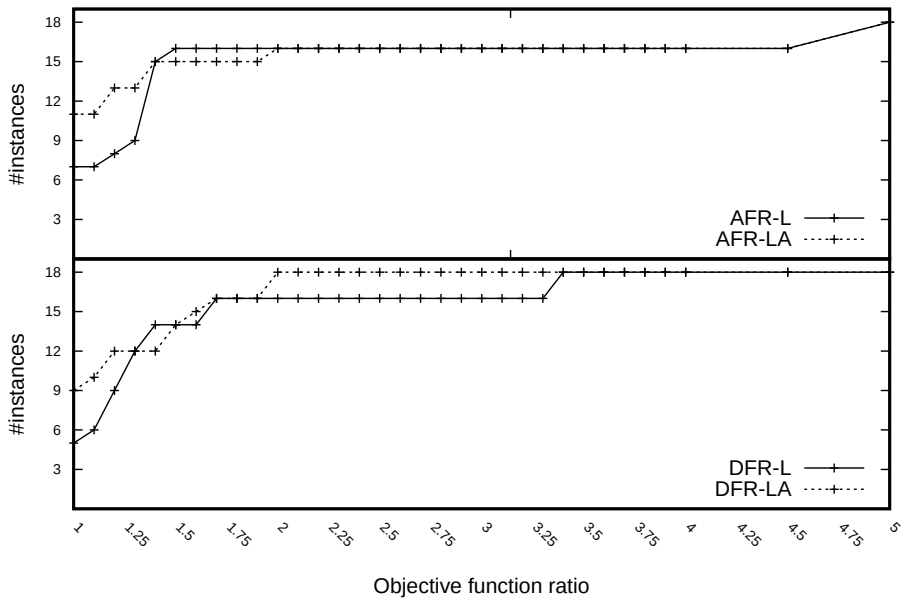
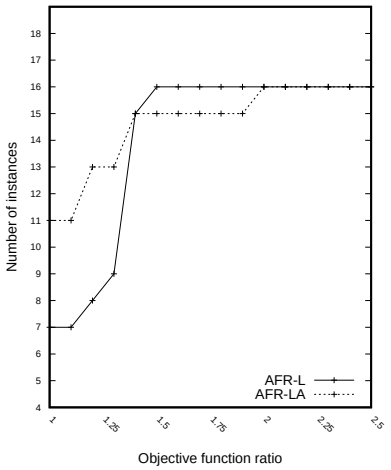


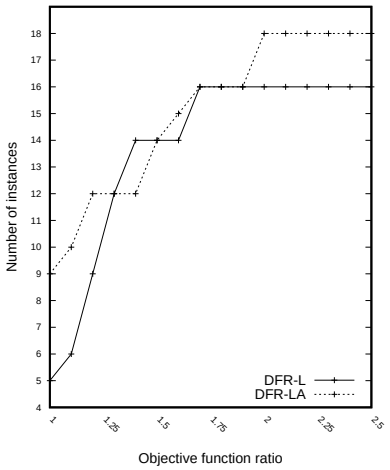
Figure 6: Comparing quality of solutions on group3 of different heuristic strategies.

In Figure 7, we report a detailed view of Figure 6 (in two separated sub-figures). We can observe that the L local search (both in AFR and DFR heuristics) is less effective in getting the best solution than the LA local search. Nevertheless, it allows to obtain good quality solutions (within 1.5 and 2 times the best algorithm's value) for a larger number of instances.

The complementary behavior of the AFR-based and DFR-based heuristics gave us the idea of combining them: we ran them sequentially and then kept the best solution obtained. We tested all possible versions with the different local search strategies (L and LA). In Figure 8, we report the best combination, namely AFR-LA/DFR-LA, and as



(a) AFR heuristics



(b) DFR heuristics capacity.

Figure 7: Detailed view of performance profiles (up to twice the best value).

a term of comparison the original AFR-LA and DFR-LA. All the heuristics run with a time-limit of two hours. In the combined heuristic, every single heuristic has a time-limit of an hour.

AFR-LA/DFR-LA and DFR-LA are more robust than AFR-LA. Indeed, they provide a solution for all the instances with an objective value no greater than twice the one achieved with the best algorithm. The combined algorithm behaves worse than the single heuristics in terms of number of best solutions. Nevertheless, its performance increases faster than the single heuristics, allowing to solve 14 instances within less than 1.5 times the best value.

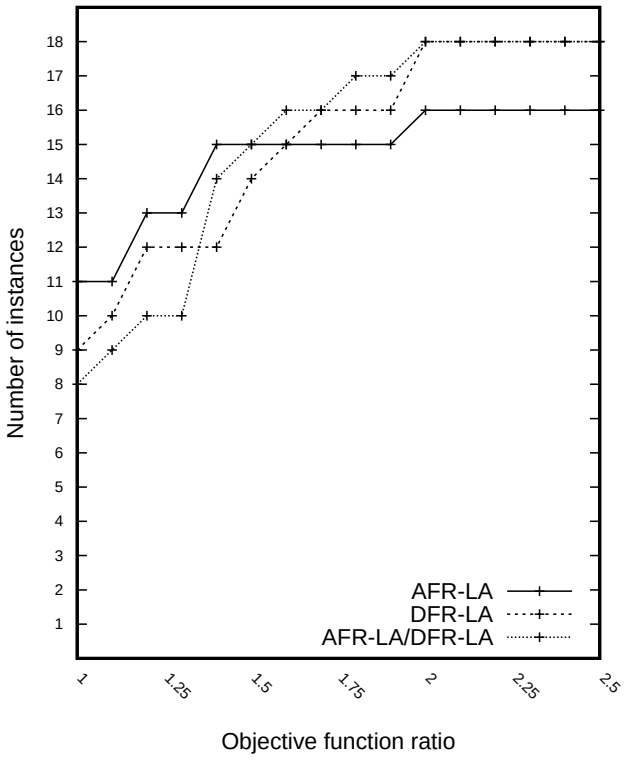


Figure 8: Comparison between AFR-LA, DFR-LA and their combination.

6. Conclusions

In this paper, we consider a VNF placement and routing problem with a single service type, and capacitated VNF instances and links. The problem arises in telecommunication applications and several variants of this problem have been addressed in the telecommunication literature. The main structure of the problem is a challenging combination of routing and location aspects. Rather than developing heuristics for a very specific version of the problem with several particular features, as is often proposed in the literature, we aimed at addressing its core structure.

We developed ILP-based heuristics, whose philosophy can be applied to tackle different variants of the problem with specific features. Indeed, our methods rely on the solution of a sequence of ILP models, and therefore they can be adapted to tackle different versions of the problem. We tested the heuristics on a significant data-set that includes instances with different sizes, network topologies and service capacity.

The computational tests show that the heuristics outperform the most promising formulation as the problem size increases, especially when the link capacity is not too tight. Although there is not a clear winner among the proposed approaches, they all show a good behavior and the gap obtained w.r.t. the best solution known is often reasonable. Furthermore, a simple combination of two "complementary" heuristics provides good quality results in almost all the instances.

The obtained results allow us to believe that it is worth extending our methods to more realistic versions of the problem, such as the multi-VNF case with no uniform resources.

References

[1] B. Addis, G. Carello, M. Gao, On a virtual network functions placement and routing problem: Some properties and a comparison of two formulations, Networks 75 (2) (2020) 158–182.

[2] M. Bouet, J. Leguay, V. Conan, Cost-based placement of vDPI functions in NFV infrastructures, in: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–9. doi:10.1109/NETSOFT.2015.7116121.

[3] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, S. Davy, Design and evaluation of algorithms for mapping and scheduling of virtual network functions, in: Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–9. doi:10.1109/NETSOFT.2015.7116120.

[4] B. Addis, D. Belabed, M. Bouet, S. Secci, Virtual network functions placement and routing optimization, in: 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), 2015, pp. 171–177. doi:10.1109/CloudNet.2015.7335301.

[5] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, L. P. Gaspary, Piecing together the NFV provisioning puzzle: Efficient placement and chaining of virtual network functions, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 98–106. doi:10.1109/INM.2015.7140281.

[6] Z. Allybokus, N. Perrot, J. Leguay, L. Maggi, E. Gourdin, Virtual function placement for service chaining with partial orders and anti-affinity rules, Networks 71 (2) (2018) 97–106. doi:10.1002/net.21768.

[7] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, O. C. M. B. Duarte, On orchestrating virtual network functions, IEEE Transactions on Network and Service Management 13 (2017) 725739.

[8] H. Moens, F. D. Turck, Vnf-p: A model for efficient placement of virtualized network functions, in: 10th International Conference on Network and Service Management (CNSM) and Workshop, 2014, pp. 418–423. doi:10.1109/CNSM.2014.7014205.

[9] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, On orchestrating virtual network functions, in: 2015 11th International Conference on Network and Service Management (CNSM), 2015, pp. 50–56. doi:10.1109/CNSM.2015.7367338.

[10] D. Dietrich, C. Papagianni, P. Papadimitriou, J. S. Baras, Network function placement on virtualized cellular cores, in: 2017 9th International Conference on Communication Systems and Networks (COMSNETS), 2017, pp. 259–266.

[11] M. Ghaznavi, N. Shahriar, S. Kamali, R. Ahmed, R. Boutaba, Distributed service function chaining, IEEE Journal on Selected Areas in Communications 35 (11) (2017) 2479–2489.

[12] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, T. Ahmed, Virtual network functions orchestration in wireless networks, in: 2015 11th International Conference on Network and Service Management (CNSM), 2015, pp. 108–116.

[13] M. C. Luizelli, W. L. da Costa Cordeiro, L. S. Buriol, L. P. Gaspary, A fix-and-optimize approach for efficient and large scale virtual network function placement and chaining, Computer Communications 102 (2017) 67 – 77. doi:https://doi.org/10.1016/j.comcom.2016.11.002.

[14] H. A. Alameddine, S. Sebbah, C. Assi, On the interplay between network function mapping and scheduling in vnf-based networks: A column generation approach, IEEE Transactions on Network and Service Management 14 (4) (2017) 860–874. doi:10.1109/TNSM.2017.2757266.

[15] N. Huin, B. Jaumard, F. Giroire, Optimal network service chain provisioning, IEEE/ACM Transactions on Networking 26 (3) (2018) 1320–1333. doi:10.1109/TNET.2018.2833815.

[16] M. Nguyen, M. Dolati, M. Ghaderi, Proactive service orchestration with deadline, in: 2019 IEEE Conference on Network Softwarization (NetSoft), 2019, pp. 369–377. doi:10.1109/NETSOFT.2019.8806703.

[17] S. Ayoubi, S. Sebbah, C. Assi, A logic-based benders decomposition approach for the vnf assignment problem, IEEE Transactions on Cloud Computing 7 (4) (2019) 894–906. doi:10.1109/TCC.2017.2711622.

[18] I. Ljubić, A. Mouaci, N. Perrot, ric Gourdin, Benders decomposition for a node-capacitated virtual network function placement and routing problem, Computers & Operations Research (2021) 105227doi:https://doi.org/10.1016/j.cor.2021.105227.

[19] T. Kuo, B. Liou, K. C. Lin, M. Tsai, Deploying chains of virtual network functions: On the relation between link and server usage, IEEE/ACM Transactions on Networking 26 (4) (2018) 1562–1576. doi:10.1109/TNET.2018.2842798.

[20] M. Ghaznavi, A. Khan, N. Shahriar, K. Alsubhi, R. Ahmed, R. Boutaba, Elastic virtual network function placement, in: 2015 IEEE 4th International Conference on Cloud Networking (CloudNet), 2015, pp. 255–260. doi:10.1109/CloudNet.2015.7335318.

[21] M. Gao, Models and Methods for Network Function Virtualization (NFV) Architectures, PhD thesis, Université de Lorraine (Mar. 2019).
URL https://hal.univ-lorraine.fr/tel-02132766

[22] B. Addis, M. Gao, G. Carello, On the complexity of a Virtual Network Function placement and routing problem, Electronic Notes in Discrete Mathematics 69 (2018) 197 – 204, joint EURO/ALIO International Conference 2018 on Applied Combinatorial Optimization (EURO/ALIO 2018). doi:https://doi.org/10.1016/j.endm.2018.07.026.

[23] M. Fischetti, A. Lodi, Local branching, Mathematical Programming 98 (1) (2003) 23–47. doi:10.1007/s10107-003-0395-5.

[24] S. Orlowski, R. Wessäly, M. Pióro, A. Tomaszewski, SNDlib 1.0 – Survivable Network Design library, Networks 55 (3) (2010) 276–286. doi:10.1002/net.v55:3.

[25] E. Dolan, J. Mor, Benchmarking optimization software with performance profiles. math. program, Mathematical Programming (2002) 201–213doi:https://doi.org/10.1007/s101070100263.

Appendix A. Impact of the local search on the presented matheuristics

In this section, we analyze the two steps of the proposed matheuristics. First, we show the results of the two proposed first steps (AFR or DFR). Second, we analyze the impact of the two local search procedures (L or LA).

	Instances	SP	AFR		DRF		$\frac{AFR}{DFR}$	
		LB	Sol	Time	Sol	Time	Sol	Time
high	cost266	3	37	213.2	20	418.8	1.9	0.5
	germany50	3.35	50	34.4	27	47.8	1.9	0.7
	giul39	-	39	914.9	36	2166.2	1.1	0.4
	india35	3.46	35	804.6	19	815.1	1.8	**1.0**
	janos-us-ca	2.84	39	644.2	21	918.1	1.9	0.7
	pioro40	3	40	342.3	31	1063.1	1.3	0.3
medium	cost266	3	37	206.0	20	406.8	1.9	0.5
	germany50	3.35	50	33.0	27	42.2	1.9	0.8
	giul39	-	39	893.4	36	2118.3	1.1	0.4
	india35	3.46	35	804.8	19	780.7	1.8	**1.0**
	janos-us-ca	2.84	39	608.8	21	832.8	1.9	0.7
	pioro40	3	40	341.7	31	1062.1	1.3	0.3
low	cost266	19	37	421.6	28	788.6	1.3	0.5
	germany50	26	50	139.5	38	55.1	1.3	**2.5**
	giul39	20	39	685.4	35	1292	1.1	0.5
	india35	18	35	226.3	27	321.1	1.3	0.7
	janos-us-ca	20	39	655.6	35	1313.7	1.1	0.5
	pioro40	21	40	217.6	31	580.5	1.3	0.4

Table A.1: Details of first phase heuristics AFR and DFR for the group3 instances.

In Table A.1, we compare the procedures used to find a starting feasible solution, namely AFR and DFR. For each instance, we report:

- the lower bound provided by the model (LB)
- for each method:
  - the value of the solution
  - the computational time
- the ratio between the AFR and DFR methods of:

649 – the values of the solution

650 – the computational times

651 AFR is faster, in general, with times that are half of those of DFR. The only excep-
652 tions are germany50 with low capacity, where AFR is slower than DFR (2.5 times) and
653 india35 with high and medium capacity, where the two methods need almost the same
654 computational time. Conversely, as expected, AFR produces worse results than DFR,
655 sometimes twice as large. An exception to this behavior is giul39 (for all the capacity
656 cases). In these instances, a significant gain in computational time is coupled to a small
657 loss in the solution quality (39 to 36/35).

658 Overall, DFR seems more promising as a stand-alone procedure, even if it may require
659 more than half an hour. AFR, on the other hand, is able to provide a good starting
660 point for the local search step, as shown in Tables A.2 and A.3. In these tables, we
661 report detailed results on the AFR-based and DFR-based methods. For each instance,
662 we report:

663 • the lower bound provided by the model (LB)

664 • for the first step:

665 – the value of solution (initial solution)

666 – the computational time

667 • for the two local search strategies (L and LA):

668 – the value of solution (final value of the algorithm)

669 – the computational time (of the local search alone)

670 – the number of local search iterations

671 – the percentage of improvement w.r.t. the initial solution.

672 The best heuristic values are reported in bold, and the best local search strategy, in terms
673 of quality of the solution and computational time, is highlighted in light gray. Note that
674 we imposed a time-limit of one hour to the overall heuristic procedure (first step plus
675 local search). When the full algorithm reaches the time-limit, we report the label 'TL',
676 to highlight when the algorithm is "prematurely" stopped by the time-limit condition.

	Instances	SP	AFR		AFR-L				AFR-LA			
			Init. solution		LS solution		Impr.		LS solution		Impr.	
		LB	Sol	Time	Sol	Time	#	%	Sol	Time	#	%
high	cost266	3	37	213.2	**4**	TL	11	89.2	**4**	TL	12	89.2
	germany50	3.35	50	34.4	7	1978.0	9	86.0	**5**	2510.2	10	90.0
	giul39	-	39	914.9	**27**	TL	4	30.8	**27**	TL	4	30.8
	india35	3.46	35	804.6	**8**	TL	9	77.1	**8**	TL	9	77.1
	janos-us-ca	2.84	39	644.2	13	TL	9	66.7	**12**	TL	9	69.2
	pioro40	3	40	342.3	**37**	1801.6	1	7.5	40	1800.3	0	0.0
medium	cost266	3	37	206.0	**4**	TL	11	89.2	**4**	TL	12	89.2
	germany50	3.35	50	33.0	7	1965.6	9	86.0	**5**	2510.9	10	90.0
	giul39	-	39	893.4	**27**	TL	4	30.8	**27**	TL	4	30.8
	india35	3.46	35	804.8	**8**	TL	9	77.1	**8**	TL	9	77.1
	janos-us-ca	2.84	39	608.8	**6**	TL	11	84.6	12	TL	9	69.2
	pioro40	3	40	341.7	**37**	1801.0	1	7.5	40	1801.1	0	0.0
low	cost266	19	37	421.6	28	1517.8	3	24.3	**19**	1513.3	6	48.6
	germany50	26	50	139.5	**26**	134.8	5	48.0	**26**	370.4	5	48
	giul39	20	39	685.4	36	1511.1	1	7.7	**33**	2914.6	2	15.4
	india35	18	35	226.3	**18**	1635.0	6	48.6	**18**	1337.5	6	48.6
	janos-us-ca	20	39	655.6	**27**	1747.7	4	30.8	29	2944.4	4	25.6
	pioro40	21	40	217.6	28	2659.4	3	30.0	**24**	TL	4	40.0

Table A.2: Details of heuristics on AFR based heuristics for the group3 instances with one hour time-limit.

For both AFR-based and DFR-based heuristics, the local search steps require in general way more computational time than the initial solution phase. However, local search steps are worth performing, as they tend to improve significantly upon the initial solution. The only exceptions being pioro40, with high and medium capacity for AFR-LA, and giul39 with low capacity for DFR-L.

In AFR-based heuristics, the improvement is slightly higher for the high and medium capacity instances. It is above 30%, with the only exception of pioro40, and it may rise to about 90%. For the low capacity instances, the percentage of improvement is between 8% and 48%. There is not a clear winner between the two neighborhoods. Starting from the AFR initial solution, they find the same solution in 8 instances, the L local search provides better results in four, while LA outperforms L in six. The number of iterations and the average per iteration time vary for the different instances.

			DFR		DFR-L				DFR-LA			
	Instances	SP	Init. solution		LS solution		Impr.		LS solution		Impr.	
		LB	Sol	Time	Sol	Time	#	%	Sol	Time	#	%
high	cost266	3	20	418.8	**5**	2546.6	5	75.0	6	2750.4	5	70.0
	germany50	3.3	27	47.8	7	1263.9	4	74.1	**6**	2535.7	5	77.8
	giul39	-	36	2166.2	**30**	TL	2	16.7	**30**	TL	3	16.7
	india35	3.5	19	815.1	**10**	2238.5	3	47.4	**10**	TL	3	47.4
	janos-us-ca	2.8	21	918.1	**6**	2464.9	5	71.4	12	TL	3	42.9
	pioro40	3	31	1063.1	27	1629.7	1	12.9	**11**	TL	5	64.5
medium	cost266	3	20	406.8	**5**	2485.7	5	75.0	6	2693.8	5	70.0
	germany50	3.3	27	42.2	7	1235.8	4	74.1	**5**	3300.0	6	81.5
	giul39	-	36	2118.3	30	TL	2	16.7	**24**	1011.9	2	33.3
	india35	3.5	19	780.7	**10**	2155.9	3	47.4	**10**	TL	3	47.4
	janos-us-ca	2.8	21	832.8	10	TL	4	52.4	**6**	TL	6	71.4
	pioro40	3	31	1062.1	27	1627.3	1	12.9	**11**	TL	5	64.5
low	cost266	19	28	788.6	**19**	874.8	3	32.1	**19**	1105.1	3	32.1
	germany50	26	38	55.1	**26**	192.5	3	31.6	**26**	331.5	3	31.6
	giul39	20	35	1292.0	35	872.0	0	0.0	**32**	TL	1	8.6
	india35	18	27	321.1	**18**	809.0	3	33.3	**18**	733.3	3	33.3
	janos-us-ca	20	35	1313.7	**23**	TL	4	34.3	29	TL	2	17.1
	pioro40	21	31	580.5	**23**	2382.8	2	25.8	**23**	TL	2	25.8

Table A.3: Details of heuristics on DFR based heuristics for the group3 instances with one hour time-limit.

A similar behavior is observed for the DFR-based heuristics, where the number of iterations (and thus improvements) is, in general, lower than for the AFR-based ones. This is not surprising, as the DFR procedure provides a better solution than the AFR one.