The strong network orientation problem

Christophe Duhamel, Andréa Cynthia Santos

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

WILEY

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 0 (2022) 1–29
DOI: 10.1111/itor.13229

The strong network orientation problem

Christophe Duhamel and Andréa Cynthia Santos*

Normandie Université, UNIHAVRE, UNIROUEN, INSA Rouen, LITIS, 25 Rue Philippe Lebon, Le Havre 76600, France
E-mail: christophe.duhamel@univ-lehavre.fr [Duhamel]; andrea-cynthia.duhamel@univ-lehavre.fr [Santos]

Received 3 March 2021; received in revised form 4 October 2022; accepted 11 October 2022

Abstract

This study presents models and heuristics for solving the strong network orientation problem (SNOP), which can model several tactical optimization problems of setting directions in urban networks. The objective is to set an orientation for each edge in an undirected graph such that the resulting digraph is strongly connected and the total travel distance between any pair of nodes is minimized (or maximized). Investigating tactical optimization problems such as SNOP is motivated by several challenges in urban networks due to the growth of population in urban areas, large number of daily trips, increasing price of maintaining urban networks, and the need to reduce air pollution and passive noise. Thus, a new trend is to utilize the urban networks better. In this context, we first use a multicommodity flow formulation to model the minimization problem. The maximization version is modeled by using the dual formulation of the shortest path problem. Then, scalable heuristic strategies for solving SNOP are investigated. For such purpose, we first propose basic components such as constructive heuristics, perturbations and local searches. They are combined into several metaheuristics based on local searches, multi-start and evolutionary schemes, i.e. Multistart Local Search, Iterated Local Search (ILS), Relaxed ILS, Evolutionary Local Search (ELS), Relaxed ELS, and Variable Neighborhood Search. Computational experiments have been performed to analyze the proposed methods in terms of efficiency and quality of solutions, using grid instances and a graph from downtown Clermont-Ferrand in France.

Keywords: urban network; strong connectivity; network design; mixed integer linear programming; heuristics

1. Introduction

Tactical optimization problems in urban networks have recently received much attention due to the growth of population in urban areas, the increasing number of daily trips, the raising costs of maintaining urban networks, and the need to preserve agricultural areas and to reduce air pollution and background noise. Even if several strategies have been developed, such as improving public transportation, creating multimodal park areas, dedicated pedestrian areas, tolls, shared and

*Corresponding author.

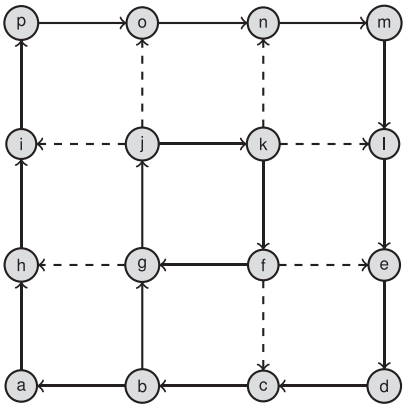


Fig. 1. Example of a deterrent policy strategy.

self-service vehicles (bikes, electrical cars, etc), the organization of urban areas remains challenging. This motivates the study of new proposals and solutions to use the road networks better and to improve them.

A growing trend consists in optimizing the use of existing urban networks rather than expanding the infrastructures. For instance, one strategy consists in setting orientations in urban networks or in an area (e.g. city centers) in order to minimize travel distances or travel times, while ensuring there is a path between each pair of nodes (strong orientation, for short). Another application consists of applying a deterrent policy strategy, where one may deter drivers entering city centers, historical areas and tourist zones, by imposing a topology that requires long paths between origins and destinations and takes time to go out of such areas. In such cases, one looks to maximize the distance between every pair of origin and destination. This strategy is typically associated with one-way route configurations as illustrated in Fig. 1. Let $\{a \dots p\}$ model a city center surrounded by a ring, called outer core. Dashed lines correspond to exits from the inner core $\{g, j, k, f\}$ to the outer core. Suppose a driver intending to drive from node b to node f. If (b, g, j, k, f), the path to f is not known, there is a probability of making a mistake. For instance, if the driver turns left at node g instead of going straight ahead to node j, he will have to restart his travel in node b due to the network configuration. This leads to a long loop back, $(h, i, p, o, n, m, l, e, d, c, b)$. When associated with other mechanisms, for instance car parks located close to the outer core and multimodal facilities such as nearby public transportation stations, to perform the last part of the trip, this approach provides an incentive for avoiding the city center without any restrictive measure (tolls, fees, few car parks slots, etc). Moreover, drivers needing to cross city will likely use the outer core, instead of entering the inner core. Yet, such topologies result also in longer paths for inhabitants willing to park their car close to home or for delivering furniture for downtown shops.

Grid networks with one-way routes shown by Ortigosa et al. (2015) are more suitable to avoid congestion since they present a better trade-off between intersection capacity and the travel distances. Readers are referred to Farahani et al. (2013) for a survey on urban networks application considering tactical, strategical and operational levels of decisions.

In this study, we investigate the strong network orientation problem (SNOP) that is defined in a simple, undirected, weighted and bridgeless connected graph $G = (V, E)$, where V is the

node-set and E is the edge-set. SNOP consists in setting a unique direction for each edge $e \in G$, such that the resulting digraph is strongly connected and the total distance between each pair of nodes is minimized (or maximized). The maximization is used to apply deterrence policies to areas with heavy traffic like center cities or historical zones. The idea is to orient arcs in such a way that long paths are designed between origins and destinations in a network area. In the urban context, this means in general areas with a unique flow circulation for each route (one-way routes). The idea is to discourage drivers from entering the aforementioned area. This problem is NP-hard (Burkard et al., 1999), even for graphs with only two origins and costs associated with every arc equal to one. The authors also presented relevant results on polynomial cases for specific topologies like series parallel graphs and regular grids.

Some closely related works are found in Huang et al. (2020a, 2020b) and Coco et al. (2020). In Huang et al. (2020b), the authors investigate the reconfiguration of urban networks after some disruptions (accidents, events, etc.). Two objectives are focused separately: minimizing arc reversals and minimizing distances. An arc reversal means in practice the modification of a route direction, and in graph theory it means orienting an arc in an opposite direction. The work of Huang et al. (2020b) is extended in Huang et al. (2020a) by considering a multi-objective approach for the two criteria. Coco et al. (2020) focus on a multi-period scheduling problem to handle medium and long-term disruptions in urban networks considering network design issues such as reversing arcs. A common feature of these studies is that arc orientation is considered, which is the core of the SNOP problem. However, Huang et al. (2020b, 2020a) and Coco et al. (2020) address an initial orientation, which will be modified for some arcs depending on disruptions and on the need to reestablish the strong connectivity. On the contrary, the initial graph in SNOP does not have an initial orientation and we look for the best one according to the optimization criterion. In crisis management, some works are also dedicated to restore partial (Yücel et al., 2018; Aksu and Ozdamar, 2014) (between origins and destinations) or complete (Sakuraba et al., 2016a, 2016b) network connectivity after major earthquakes, but route directions are not explicitly considered.

SNOP models are a relevant basis for several problems of orienting a graph and it opens several avenues of research in terms of graph theory. It is also of practical interest, for instance arc reconfiguration in urban networks due to disruptions, strong connectivity restoration after disaster, etc. The additional features of this study compared to the existing ones in the literature are the following. Specific mathematical formulations for the minimization and for the maximization versions of SNOP are proposed, together with several analyses on how to strengthen them in terms of number of commodities. In addition, a collection of metaheuristics are proposed and compared. For this purpose, a number of components such as a constructive heuristic, a perturbation operator and a local search using dedicated neighborhood structures have been developed. These components are combined and used in the following metaheuristics: Multistart Local Search (MS), Iterated Local Search (ILS), Relaxed ILS (RILS), Evolutionary LS (ELS), Relaxed ELS (RELS), and Variable Neighborhood Search (VNS). The methods have been tested on two kinds of networks, grid networks modeling downtowns from North America and a realistic graph from downtown Clermont-Ferrand, France.

The remainder of this paper is organized as follows. Initially, a bibliographical review is done in Section 2. Then, the problem is formally defined in Section 3, followed by a mathematical formulation. The proposed heuristics and metaheuristics are presented in Section 4, before showing

computational experiments in Section 5. Concluding remarks and perspectives for future research are provided in Section 6.

2. Related works

Components of the SNOP are found in the scientific literature of graph theory, Automated Guided Vehicles (AGV) systems and urban network applications. Below we review such studies and mention our contributions.

Most works of graph theory associated with strong orientations in graphs provide theoretical results for this problem. For instance, the pioneering work of Robbins (1939) established the conditions in which a graph can admit a strong orientation: an undirected graph $G = (V, E)$ can be strongly orientated if and only if G is connected and has no bridges (i.e. an edge whose removal breaks the network connectivity). The study is motivated by an application on an urban network which is two-way on week-days and the aim is to repair routes such that every route is set one-way and each point in the city is still reachable. Chung et al. (1985) extend the results of Robbins (1939) by considering strong orientations for mixed multigraphs, i.e. multigraphs with both undirected and directed connections. Chvátal and Thomassen (1978) investigate the conditions for bridge-less graphs to be strongly oriented considering a given upper bound on the radius. Nash–William' strong orientation theorem has been extended in Fukunaga (2012), by defining directions on an undirected graph, while ensuring connectivity requirements among pairs of vertices. Recently, Conte et al. (2016) proposed an algorithm to enumerate efficiently all strong orientations for a given undirected connected graph. It relies on the linear-time detection of all strong bridges proposed by Italiano et al. (2012). One may note that there is an exponential number of possible orientations. Our work considers graph orientations, but focuses on optimization models and methods to find an orientation that minimizes or maximizes distances.

Variations of the SNOP have been investigated in the context of AGV, where connecting requirements have to be ensured for every pair of pickup and delivery points. In AGV systems, solutions are applied at an operational level and the network design (layout) can be often reconfigured on the fly, which is not suitable in an urban context.

The well-known flow path design is closely related to the SNOP and it has been widely investigated for AGV systems. Introduced by Gaskins and Tanchoco (1987), the problem has been motivated by the use of AGV in production systems and aims at designing a unique path between pickup and delivery points. A Mixed Integer Linear Programming (MILP) formulation to set orientation in a given graph between all pairs of pickup and delivery points was proposed by Gaskins and Tanchoco (1987), such that the total traveling distances is minimized. The model is indexed on the possible paths.

Following the study of Gaskins and Tanchoco (1987), the amount of work dedicated to AGV systems has increased significantly for several applications in manufacturing, distribution, transshipment and transportation systems. Several aspects were addressed such as designing routes for AGV, balancing the charge, defining positioning policies and scheduling. They are surveyed in Vis (2006) and Qiu et al. (2002). Kaspi and Tanchoco (1990) raise the concept of connectivity between pickup and delivery points and model the problem using a multiflow formulation. The authors

also propose a Branch-and-Bound (B&B) method. Venkataramanan and Wilson (1991) explicitly define and apply the idea of strong connectivity. It is handled in a B&B approach, but the strong connectivity is not explicitly mathematically formalized. Moreover, computational results are not presented in Kaspi and Tanchoco (1990) and Venkataramanan and Wilson (1991), probably due to the limits of MILP solvers at that time. Here, our proposed formulations extend the ones presented in Kaspi and Tanchoco (1990), by ensuring the strong connectivity for the final network and by formalizing the idea introduced in Venkataramanan and Wilson (1991), by means of a multiflow formulation. In addition, a comparison and computational results are shown here for both models. A recent review on AGV problems is found in Vancea and Orha (2019), and for readers looking for more on AGV, Ullrich (2014) provides a very interesting retrospective of AGV.

The network configuration in AGV systems is more flexible than in the SNOP for urban networks since the topology can be easily changed by adding arcs, for example. On the contrary, adding arcs in urban networks means building new routes, which is very complicated. In addition, AGV system has quite often objectives related to path optimization instead of looking for a global network optimized topology, as the SNOP for urban networks.

In the context of urban networks, the problem of one-way orientation such that the total travel time is minimized for every pair of nodes has been introduced in Drezner and Wesolowsky (1997). The authors consider a smaller street travel time if it is set one-way since it increases the traffic capacity on such arc. However, the shortest distances also increase for some pairs of nodes in the network. A B&B, simple heuristics and a simulating annealing are presented. Three strategies to build initial solutions are proposed: (i) random generation, (ii) reparation procedure which starts from the graph with all orientations and set a unique direction per arc at a time, and (iii) reparation procedure for which triangulation rules are applied to generate a feasible solution. Computational results are presented for instances with up to 40 nodes and 99 arcs. Roberts and Xu (1988, 1994) investigate the case of cities where streets are represented by a grid graph, i.e. a graph where nodes are set on p rows and q columns. In Roberts and Xu (1988), the authors focus on the optimality conditions for grid graphs. They conclude that an optimal orientation for a 4×4 grid graph has the following weakness in an urban context: there is a vertex with two entering arcs coming from different directions, which can produce congestion in urban networks. The case with $p = 3$ is studied in Roberts and Xu (1994). Some works also investigate the efficiency of network configurations by means of simulation. For instance, Ortigosa et al. (2015) analyze the impact of various grid configurations modeling urban networks in terms of network exit function. One of the configurations is the No Turn Left (NTL) layout, which is safer and more efficient in the sense it does not interfere with the incoming traffic.

The signal settings problem also received much attention in the literature. It is another way to manage and to improve the use of urban networks. The SNOP and the signal settings are related in terms of practical objectives, optimizing the efficiency of the urban network. The signal settings consist of evaluating local delays induced by the green light frequency. Some interesting entry points are Cantarella et al. (2006) and Gallo et al. (2010). The former investigates the signal settings such that travel time is optimized, and tested the proposed methods using instances from Villa San Giovanni and Barcellona Pozzo di Gotto in Italy. For the latter, a stochastic strategy is used instead of a deterministic one applied to Benevento city in Italy. It should be mentioned that some signal setting problems are now handled by some real-time signal controls. They allow the green time period to be adapted in order to reduce congestion.

As can be observed in the bibliographical review, some variants of the SNOP have been investigated in the context of AGV. However, AGV systems are usually designed for workshops in industries, thus in a closed and reduced dimension. In up-to-date applications on urban networks, SNOP has an additional and relevant characteristic to consider: a large-scale dimension. This requires the development of scalable, robust and effective heuristics. This study encompasses a set of fundamental features for SNOP. In particular, two versions of the problem are proposed and modeled by means of graph theory and flow-based mathematical formulations. One may note that there is a lack of studies in the literature dedicated to the maximization version of SNOP. Moreover, a set of components like building feasible solutions and local moves are proposed, and combined to build modern and successful metaheuristics. Last but not least, extensive computational experiments to analyze the performance and robustness of the proposed methods are described. This study opens several avenues of research in terms of graph theory, mathematical programming, and applied research on urban networks.

3. Problem definition and mathematical formulations

Let $G = (V, E)$ be a simple, undirected, weighted and bridgeless connected graph, where V and E stand respectively for the set of $n = |V|$ vertices and the set of $m = |E|$ edges. Each edge $e \in E$ is associated with a length $c_e > 0$. Let $K = \{(o, d, q)\}$ be the set of traffic requests, also referred to as commodities. Each request $(o, d, q) \in K$ corresponds to a traffic demand q to be routed from its origin o to its destination d. It may correspond to users' shifts, such as commuting (e.g. daily home-work or work-home trips in heavy traffic) or commercial trips. We make the assumption of rational driver behavior, such that each request (o, d, q) will be routed on the shortest path from o to d. In addition, the set $P = \{(o, d) \in V \times V, o \neq d\}$ defines the set of all node pairs and $A = \{(i, j), (j, i) | [i, j] \in E\}$ is the set of all arcs that can be obtained by setting an orientation to the edges in E.

The Orientation Problem (OP) consists in setting a unique orientation (i, j) or (j, i) for each edge $e = [i, j] \in E$, leading to a digraph G'. Without loss of generality, we assume here $c_{ij} = c_{ji} = c_e$, i.e. the length does not depend on the orientation. G' distance matrix will not be symmetric. The Strong Orientation Problem (SOP) extends the OP by requiring G' to be strongly connected, even if the set of requests does not cover all the $n(n-1)$ pairs of nodes in P. This means there must be a path between all pairs of nodes in the network. Thus, the SNOP combines the SOP with the minimization (or maximization) of the total travel distance for traffic requests.

3.1. SNOP with minimization criterion

The mathematical formulation (M1a) from (1) to (6) relies on a classical arc formulation that ensures a path between each pair $(o, d) \in P$ of nodes. Thus, strong connectivity is ensured and, at the same time, it provides means for computing the total travel distance. The formulation uses the decision variable x_{ij} to state if arc $(i, j) \in A$ is chosen ($x_{ij} = 1$) or not ($x_{ji} = 0$). Moreover, f_{ij}^{od} is the flow variable which indicates if the flow from o to d uses the arc (i, j).

$$\min \sum_{(i,j)\in A} c_{ij} \sum_{(o,d,q)\in K} q f_{ij}^{od} \tag{1}$$

$$\text{s.t.} \sum_{l:(l,i)\in A} f_{li}^{od} - \sum_{l:(i,l)\in A} f_{il}^{od} = \begin{cases} -1, & \text{if } i = o \\ +1, & \text{if } i = d \\ 0, & \text{otherwise} \end{cases} \quad \forall (o,d) \in P, \forall i \in V \tag{2}$$

(M1a)

$$f_{ij}^{od} \leq x_{ij} \qquad \forall (i,j) \in A, \forall (o,d) \in P \tag{3}$$

$$x_{ij} + x_{ji} = 1 \qquad \forall [i,j] \in E \tag{4}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{5}$$

$$f_{ij}^{od} \geq 0 \qquad \forall (i,j) \in A, \forall (o,d) \in P. \tag{6}$$

The objective function (1) computes the total travel distance for all requests. Constraints (2) are the flow conservation constraints. Inequalities (3) allow flow to be routed only on selected arcs (i, j). Equations (5) guarantee a unique orientation is set for every edge $e = [i, j] \in E$. Variables are defined in (5) and (6).

Another version of a mathematical formulation for the SNOP, without the strong connectivity, was proposed for AGV by Kaspi and Tanchoco (1990). This formulation is referred to as (M1b) and it is presented in Equations (7) to (12).

One multicommmodity flow is set for each traffic request (set K) instead of each pair of nodes (set P). Thus, strong connectivity is not guaranteed anymore in (M1b) for any set K:

$$\min \sum_{(i,j)\in A} c_{ij} \sum_{(o,d,q)\in K} q f_{ij}^{od} \tag{7}$$

$$\text{s.t.} \sum_{l:(l,i)\in A} f_{li}^{od} - \sum_{l:(i,l)\in A} f_{il}^{od} = \begin{cases} -1, & \text{if } i = o \\ +1, & \text{if } i = d \\ 0, & \text{otherwise} \end{cases} \quad \forall (o,d,q) \in K, \forall i \in V \tag{8}$$

(M1b)

$$f_{ij}^{od} \leq x_{ij} \qquad \forall (i,j) \in A, \forall (o,d,q) \in K \tag{9}$$

$$x_{ij} + x_{ji} = 1 \qquad \forall [i,j] \in E \tag{10}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{11}$$

$$f_{ij}^{od} \geq 0 \qquad \forall (i,j) \in A, \forall (o,d,q) \in K. \tag{12}$$

Figure 2(a) illustrates the difference between (M1a) and (M1b) in terms of their optimal solutions. Let us consider a small graph with three vertices and three edges, see Fig. 2(a), and a set of three commodities $K = \{(1, 2, 4), (1, 3, 5), (3, 2, 1)\}$. For simplicity, all edges are of unit length. The optimal solution for model (M1a) is given in Fig. 2(b). Its cost is 14 and it is strongly connected. The optimal solution for model (M1b) is shown in Fig. 2(c). Its cost is 10 and it is not strongly connected. Obviously, enforcing the strong connectivity and not just the connectivity for the requests may lead to solutions with a higher cost.

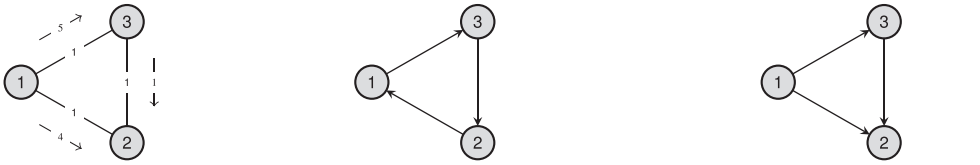


Fig. 2. Difference between solutions produced by models (M1a) and (M1b).

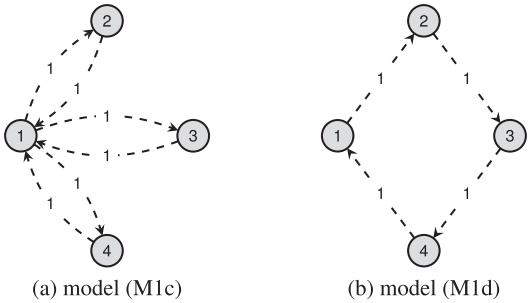


Fig. 3. Strong connectivity flows on a graph with four vertices.

Model (M1b) contains fewer variables and constraints than model (M1a), especially when $|K| \ll |P|$. For example, model (M1a) for the small instance in Fig. 2(a) uses six unit flows (hence 36 flow variables, 18 flow constraints (2) and 36 coupling constraints (3)). The model (M1b) uses three unit flows (hence 18 flow variables, 9 flow constraints (8) and 18 coupling constraints (9)). The strong connectivity property can be added to model (M1b) by forcing an additional unit flow between some pairs of nodes. This can be done in several ways:

1. One node, say node 1, sends one unit of flow to every other node and (ii) receives one unit flow from every other node. Thus, node 1 and every other node belong to the same strong component and the resulting digraph is strongly connected. This leads to model (M1c).
2. Each node i, $i = 1, \ldots, n-1$, sends one unit flow to node $i+1$. Node n sends one unit flow to node 1. Thus, a (non-elementary) circulation is set on all the nodes and the resulting digraph is strongly connected. This leads to model (M1d).

Figure 3 illustrates the added flow structure for models (M1c) and (M1d). Both models add $O(n)$ unit flows: model (M1c) adds at most $2(n-1)$ unit flows while model (M1d) adds up to n unit flows. For this reason, model (M1d) is preferred since it requires fewer unit flows.

3.2. SNOP with maximization criterion

Models (M1a), (M1b), (M1c) and (M1d) cannot be used for the total travel distance maximization. In fact, one wants to set the orientation in maximization such that the sum of the shortest travel distances is the largest possible. Switching from min to max is not sufficient as the model would

turn into a (incorrect) longest path computation. The primal formulation for the shortest paths is not compatible with maximization. Instead, the dual formulation can be used for the shortest path and then the maximization is used. Let us consider $u_i^k \geq 0$ be the distance label from node k at node i. Moreover, let Q be a large-enough constant. Then the generic model is as follows:

$$\max \sum_{(o,d,q)\in K} q u_o^d \tag{13}$$

$$\text{s.t.} \quad u_j^o \leq u_i^o + c_{ij} + (Q - c_{ij}) x_{ji} \qquad \forall (i,j) \in A, \forall (o,d) \in P \tag{14}$$

$$u_o^o = 0 \qquad \forall (o,d) \in P \tag{15}$$

(M2) $<$ strong connectivity constraints $>$

$$x_{ij} + x_{ji} = 1 \qquad \forall [i,j] \in E \tag{16}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \tag{17}$$

$$u_i^o \geq 0 \qquad \forall i \in V, \forall (o,d) \in P. \tag{18}$$

The objective function (13) and constraints (14) and (15) correspond to the dual model for shortest path computation. Strong connectivity constraints are any set of constraints that force x variables to define a strongly connected digraph G'. Constraints (14) have been adapted to take the arc orientation into account. Equations (16) are kept to ensure exactly one direction is selected for each arc. Constraints (17) and (18) are the variable definitions.

As mentioned before, a set of constraints enforcing the strong connectivity has to be included, otherwise the problem is unbounded. Two versions are proposed, using the same idea illustrated in Fig. 3 for models (M1c) and (M1d). In the first version, one unit flow is sent from node 1 to any other node and from any other node to node 1. Let $f_{ij}^{+k} \geq 0$ be the flow variable on arc $(i,j) \in A$ from node 1 to node k. Similarly, let $f_{ij}^{-k} \geq 0$ be the flow variable on arc $(i,j) \in A$ from node k to node 1. The first set of strong connectivity constraints is then as follows:

$$\sum_{l:(l,i)\in A} f_{li}^{+k} - \sum_{l:(i,l)\in A} f_{il}^{+k} = \begin{cases} -1, & \text{if } i = 1 \\ +1, & \text{if } i = k \\ 0, & \text{otherwise} \end{cases} \qquad \forall k \in V \setminus \{1\}, \forall i \in V \tag{19}$$

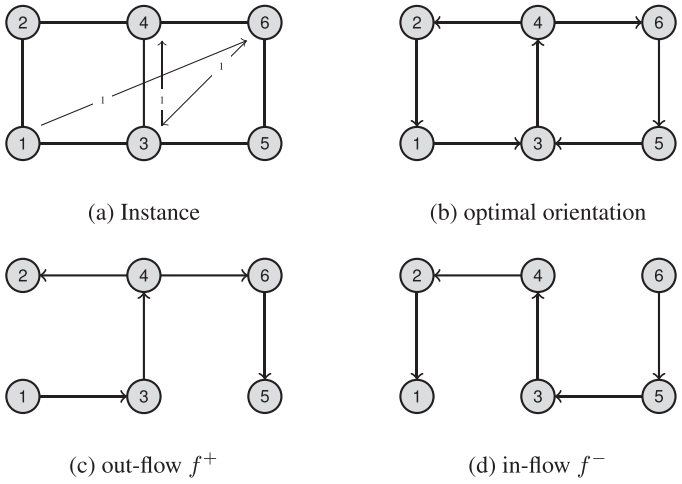
$$\sum_{l:(l,i)\in A} f_{li}^{-k} - \sum_{l:(i,l)\in A} f_{il}^{-k} = \begin{cases} +1, & \text{if } i = 1 \\ -1, & \text{if } i = k \\ 0, & \text{otherwise} \end{cases} \qquad \forall k \in V \setminus \{1\}, \forall i \in V \tag{20}$$

$$f_{ij}^{+k} \leq x_{ij} \qquad \forall (i,j) \in A, \forall k \in V \setminus \{1\} \tag{21}$$

$$f_{ij}^{-k} \leq x_{ij} \qquad \forall (i,j) \in A, \forall k \in V \setminus \{1\} \tag{22}$$

$$f_{ij}^{+k}, f_{ij}^{-k} \geq 0 \qquad \forall (i,j) \in A, \forall k \in V \setminus \{1\}. \tag{23}$$

Constraints (19) set the flow from node 1 to any other nodes. Constraints (20) set the flow from any node to node 1. Constraints (21) and (22) forbid non-selected arcs to carry the flow. Constraints (23) are the definition of the variables. Using these equations in model (M2) leads to model (M2a).



(a) Instance (b) optimal orientation

(c) out-flow f^+ (d) in-flow f^-

Fig. 4. Counterexample to stronger constraint $f_{ij}^{+k} + f_{ij}^{-k} \leq x_{ij}$.

Table 1
Models' main properties

models	obj	#flows	SO	#var	#0/1	#cstr
(M1a)	min	$n(n-1)$	yes	$mn(n-1)$	m	$(n^2+mn)(n-1)+m/2$
(M1b)	min	k	no	mk	m	$k(m+n)+m/2$
(M1c)	min	$\leq k+2(n-1)$	yes	$\leq m(k+2(n-1))$	m	$\leq (m+n)(2(n-1)+k)+m/2$
(M1d)	min	$\leq k+n$	yes	$\leq m(k+n)$	m	$\leq k(m+n)+n^2+m/2$
(M2a)	max	$2(n-1)$	yes	$(2m+n^2)(n-1)$	m	$(mn+2m+3n)(n-1)+m/2$
(M2b)	max	n	yes	$(m+n^2)(n-1)$	m	$(mn+n)(n-1)+mn+n^2+m/2$

The second version sets a unit flow circulation over all the nodes (one unit flow from node i to node $i+1$, $i = 1, \ldots, n-1$, and one unit flow from node n to node 1). This leads to model (M2b).

Note that the two sets of constraints (21) and (22) cannot be merged into a stronger constraint ($f_{ij}^{+k} + f_{ij}^{-k} \leq x_{ij}$) since it could forbid the optimal solution. This is illustrated in Fig. 4 for an instance with six vertices and seven edges with unit length. Three requests are considered: (1, 6, 1), (3, 4, 1) and (6, 3, 1). Vertex 1 is set as root node. The optimal solution is shown in Fig. 4(b) and the flow f^+ from node 1 (resp. f^- to node 1) is presented in Fig. 4(c) (resp. Fig. 4(d)). One can see arc (3,4) carries both kinds of flow.

3.3. Model characteristics summary

Given an instance with n vertices, m arcs and k requests, Table 1 summarizes the main properties of the six models presented in Sections 3.1 and 3.2. The column "#flows" corresponds to the number of unit flows used in the model while column "SO" indicates if the models guarantee a strong orientation of the optimal solution. The last three columns indicate the number of variables ("#var"), the number of 0/1 variables ("#0/1") and the number of constraints ("#cstr") for each model. Note

that the number of flows for models (M1c) and (M1d) is an upper bound since the original OD flows can be used as part of the strong connectivity flow scheme. As mentioned before, aggregated flow can be used. The resulting models are more compact but their linear relaxation is worse, due to big-M constants in flow-topology coupling constraints.

The two following properties trivially hold for the linear relaxations of the proposed models:

Property 1. *The linear relaxation of model (M1b) is weaker than the linear relaxation of models (M1a), (M1c) and (M1d). The last three linear relaxations are equal.*

Property 2. *The linear relaxation of model (M2a) is equal to the linear relaxation of model (M2b).*

4. Heuristics for the SNOP

Using multicommodity flow in models (M1a) to (M2b) involves a large number of variables and constraints. Thus, these models cannot handle medium-to-large problems and other approaches have to be investigated. Several metaheuristics are considered. Since each one provides its own mechanism to explore the search space, we intend to assess their behavior for different network configurations and select the ones with the best performance. These methods share the following components: constructive heuristics, a local search and perturbations. They are detailed in the next subsections.

4.1. Constructive heuristics

Let $G = (V, E)$ be an undirected connected graph with n vertices and m edges. An algorithm for computing a strong orientation of G has been proposed in Roberts (1978). It consists in performing a Depth-First Search (DFS) (Cormen et al., 2009) on G, starting from an arbitrary initial node. A rank is associated with each vertex. It corresponds to the order the vertex is visited in the DFS. The edges used to extend the exploration arborescence are oriented the way they are traversed. The other edges are oriented from the extreme vertex with the largest rank to the extreme vertex with the smallest rank. This deterministic algorithm runs in $O(m + n)$ time. It guarantees a feasible solution, i.e. a strong orientation.

We first propose its randomization in order to be able to produce several strong orientations. The idea consists in assigning a weight $w_i \geq 0$ to each vertex $i \in V$. Let $Adj_w(i)$ be the adjacency list of vertex i, sorted by increasing weight. Then, at each DFS iteration, the unvisited neighbor with the smallest weight is selected, see Algorithms 1 and 2.

Given a vector of vertices weight, this DFS algorithm ensures a strong orientation is computed since it extends Roberts' algorithm (Roberts, 1978) by only changing the order the neighbors are visited. It runs in linear time, provided the neighbors have been first sorted by increasing weight. The interest of randomizing the DFS is to be able to produce several feasible solutions by randomly generating several weight vectors w and calling the algorithm. Yet, some strong orientations cannot be obtained using this algorithm. For instance, the graph in Fig. 5 with five nodes and eight arcs is strongly connected but none of the 120 possible node sequences induced by weights can lead to this strong orientation. The other components of the metaheuristic are used to mitigate this limitation.

Algorithm 1. Randomized DFS orientation R-DFS(G, A)

Input: $G = (V, E)$: graph
Output: A: edges orientation
for $i \in V$ **do**
  color[i] $\leftarrow$ WHITE
end for
$A \leftarrow \emptyset$
$r \leftarrow \arg\min_{i \in V}\{w_i\}$
DFS_visit(G,A,i,-1)

Algorithm 2. Subroutine DFS_visit(G, A, i, f)

Input: $G = (V, E)$: graph; A: edges orientation; i: node to visit; f: ancestor
Output: A: edges orientation
color[i] $\leftarrow$ GRAY
for $j \in Adj_w(i)$ (increasing weight) **do**
  if color[j] = WHITE **then**
    $A \leftarrow A \cup (i, j)$
    DFS_visit(G,A,j,i)
  else if $j \neq f$ **and** color[j] $\neq$ BLACK **then**
    $A \leftarrow A \cup (i, j)$
  end if
end for
color[i] $\leftarrow$ BLACK

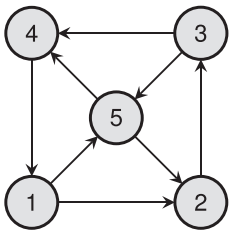
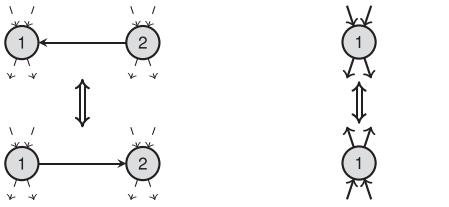


Fig. 5. Counterexample for the randomized DFS.

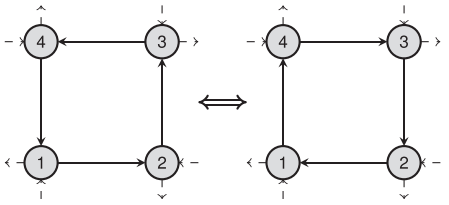
As a consequence, encoding a solution s as a node sequence v or as a vector w of node weights, and using Algorithm 1 to ensure the binding $v/w \rightarrow s$ cannot be applied extensively in a metaheuristic since it does not guarantee the optimal solution can be reached. However, it can be used to compute initial solutions in a multistart strategy by setting random node weights.

4.2. Local search

The Variable Neighborhood Descent (VND) of Mladenović and Hansen (1997) has been initially defined as a local search for the Variable Neighborhood Search (VNS). It relies on an ordered list $\mathcal{N} = \{\mathcal{N}_1, \ldots, \mathcal{N}_T\}$ of T neighborhood structures and it works in the following way: only one neighborhood structure is active in each iteration, with $\mathcal{N}_1$ being used initially. Then, given the current



(a) $\mathcal{N}_1$ move (b) $\mathcal{N}_2$ move



(c) $\mathcal{N}_3$ move

Fig. 6. Basic moves.

solution s and the active neighborhood structure $\mathcal{N}_t$, the set of neighbor solutions $\mathcal{N}_t(s)$ is explored. If an improving solution $s' \in \mathcal{N}_t(s)$ is found, then $s \leftarrow s'$ and $t \leftarrow 1$. Otherwise $t \leftarrow t + 1$. The VND stops as soon as $t = T + 1$, when no improving solution has been found in $\mathcal{N}_1(s) \cdots \mathcal{N}_T(s)$. Here, three neighborhood structures are proposed for the VND.

A move in the first neighborhood, $\mathcal{N}_1$, called *One arc reversal*, has been introduced by Drezner and Wesolowsky (1997). It consists of changing the orientation for a single arc, see Fig. 6(a). One may note that this does not guarantee a strong connectivity of the resulting solution s'. Thus, a basic $O(m + n)$ algorithm is used to check if s' is strongly connected. All arcs are considered for this neighborhood. In the *One vertex cocycle reversal* move $\mathcal{N}_2$, the orientation of all the edges connected to a vertex is modified, see Fig. 6(b). This move does not ensure strong connectivity. Thus, as for a $\mathcal{N}_1$ move, a check on strong connectivity is performed. All vertices are considered for this neighborhood. To the best of our knowledge, the next neighborhood $\mathcal{N}_3$ has not been proposed in the literature. A *One cycle reversal* move in $\mathcal{N}_3$ consists in reverting the arcs in a cycle, see Fig. 6(c). This move is especially interesting since the strong connectivity is ensured. As a consequence, the new solution is feasible. Only a subset of cycles is considered for this neighborhood. Namely, the smallest cycle including each arc is selected by performing a BFS (Breadth First Search). One may note that these three neighborhood structures $\mathcal{N}_1 \ldots \mathcal{N}_3$ require a global evaluation of the new solution, that is the computation of all the shortest paths.

4.3. Perturbation

The perturbation mechanism aims at modifying the current solution such that the subsequent use of a local search will drive the process into other areas of the solution space. The perturbation can be parametrized in order to control the amount of modifications from the current solution.

For the SNOP, the perturbation consists in changing the orientation of a given number of arcs, such that the resulting topology remains strongly connected. This is done iteratively. In each iteration, a subset of arcs whose orientation can be safely changed is computed as follows: a DFS is performed on the current solution. Following Cormen et al. (2009), the combination of tree arcs and back arcs defines cycles in the digraph. Thus, the forward arcs and the cross arcs can be reversed without breaking the strong connectivity. They can be identified in $O(m+n)$ time complexity to define the subset. Then, an arc is randomly chosen from this subset and it is reversed. In order to avoid as much as possible the arc being reverted, a null weight p_{ij} is initially assigned to each edge $[i, j] \in E$. It is increased each time the orientation of edge $[i, j]$ is changed and the arc to be changed is randomly chosen in the subset among the arcs whose weight is minimal.

4.4. Metaheuristics

Six metaheuristics are being used and compared: the multistart strategy (MS), the ILS proposed by Lourenço et al. (2002), the RILS of Afsar et al. (2014), the ELS of Wolf and Merz (2007), the RELS, and the VNS of Mladenović and Hansen (1997). They all make use of the components presented in the previous sections, but in different ways since each one defines its own exploration strategy, as described below.

At each iteration of the MS, an initial solution is randomly generated before being improved in the local search. The best solution obtained over the iterations is kept. This is the most basic metaheuristic since its global strategy consists in randomly sampling the solution space.

In the ILS, an initial feasible solution is computed and then submitted to a local search. In the following, and for a given number of iterations, a random perturbation is applied to the current solution before calling the local search to get a new local optimum. This new solution becomes the new current solution whenever it is better than the current solution. Otherwise, the current solution is kept. The best-known solution found so far is returned by the algorithm.

The ELS extends the ILS by creating, at each iteration, a given number of the current solution copies. A random perturbation is performed on each copy before improving it in the local search. The best new solution is kept and the current solution is updated the same way ILS does.

The VNS first improves the initial solution in the local search. Then, given a level τ of perturbation, one iteration consists in copying the current solution, applying a random perturbation of level τ, then improving the result with the local search. If the new local optimum is better than the current solution, the current solution is updated and τ is reset to its minimal value $\tau_{\min}$. Otherwise, the current solution is kept and $\tau \leftarrow \tau + 1$. The method stops when τ reaches its maximal value $\tau_{\max}$.

Note that a variation is also considered for both ILS and ELS. The "better" acceptance criterion can be replaced by the "random walk" acceptance criterion: the new solution always replaces the current solution, even if it is not better than the incumbent solution (Lourenço et al., 2002). In such a case, the incumbent solution has to be explicitly stored as it does not necessarily correspond to the current solution. This leads to the RILS and the RELS.

5. Computational experiments

The experiments are performed on an Intel Core i7 with 2.6 GHz clock and 8Gb of RAM. The mathematical formulations are tested using the CPLEX 12.7 solver under default parameters and the proposed heuristics are developed in C++ and compiled with GCC g++ 9.3.0.

The first computational experiment addresses the mathematical models (M1a) to (M2b) in order to measure the impact of enforcing or not the strong connectivity and the various unit flow sets, and to evaluate the running times and the linear relaxation, see Section 5.1. The metaheuristics are tested in the second set of experiments, see Section 5.2, in order to provide a comparison, to evaluate their convergence, robustness and running time.

Three sets of instances are used.[1] The first one relies on grid graphs of size 4×4, 5×5, up to 10×10. For each grid size, 10%, 25%, 50%, 75% and 100% of the possible number of requests (unit flow) is randomly selected. Thus, there is one instance for each size of grid and each size of set of requests. The second set of instances contains 100 randomly generated sets of requests for the same 4×4 grid and for each percentage — 10%, 25%, 50%, 75% — of the possible number of requests (unit flow). For 100%, there can be only one set of requests. Thus, there is only one instance. The third test set consists of a single instance corresponding to the center of Clermont-Ferrand city in France. It has been obtained from a Geographical Information System, followed by a data treatment to generate the corresponding graph. It contains 92 vertices, 304 edges and 992 requests, which corresponds to around 11.8% of the possible number of requests.

5.1. Results of mathematical formulations

The mathematical formulations (M1a) to (M2b) are tested on the first set of instances described above, with a running time limit set to 2 hours. Results for the minimization and the maximization are reported in Tables 2 and 4 respectively. Each line corresponds to an instance. Columns "size" and percentage of requests "%*req*." indicate the instance characteristic dimensions, resp. the grid size and the number of requests. The three next columns contain the optimal value, the linear relaxation value and the relative integrality gap in percentage for models (M1a), (M1c) and (M1d). Then for each model, the CPU time in seconds is provided. For model (M1b), the column "feas." indicates if the computed optimal solution if strongly connected ("y") or not ("n"). The symbol "-" indicates the solver has run out of memory, as a consequence no solution has been obtained.

As can be seen in Table 2, all instances with up to 36 nodes and 50% requests can be solved. The number of flows used in the model has significant impact on the time to find an optimal solution. Time consumed with model (M1a) grows faster than with smaller models (M1c) and (M1d). Model (M1b) consumes the least time, since it only uses the flows corresponding to the requests. It is worth mentioning that the four models have the same value of the linear relaxation for all these instances, even if Property 1 states that model (M1b) can have a weaker linear relaxation.

In Table 2, all the optimal solutions obtained with model (M1b) were strongly connected, even if the property is not guaranteed for this model. Intuitively, one may think the probability for model

[1]Instances will be available on https://cv.archives-ouvertes.fr/andrea-cynthia-santos, and https://pagesperso.litislab.fr/cduhamel/, or under reasonable request.

Table 2
Results for models (M1a), (M1b), (M1c) and (M1d) (minimization)

Instances					M1a	M1b		M1c	M1d
size	%req.	z^*	$\bar{z}$	%gap	time	time	feas.	time	time
4 × 4	10	3782	3782	0.00	0.74	0.03	y	0.04	0.03
	25	7791	7248	6.97	3.99	0.36	y	0.67	0.41
	50	13969	12691	9.15	6.24	1.60	y	2.18	2.11
	75	15244	14303	6.17	4.24	2.73	y	2.89	2.75
	100	43160	38352	11.14	7.33	7.33	y	7.33	7.33
5 × 5	10	12333	11734	4.86	287.33	1.09	y	2.48	1.64
	25	31332	28630	8.62	238.15	9.95	y	22.93	12.32
	50	49136	46908	4.53	78.69	16.13	y	19.11	19.70
	75	77522	73214	5.56	91.40	49.81	y	35.91	54.42
	100	109458	96620	11.73	1068.87	1068.87	y	1068.87	1068.87
6 × 6	10	29178	28227	3.26	2788.80	5.88	y	12.79	8.05
	25	72371	67149	7.22	–	286.56	y	634.72	609.54
	50	160299	147199	8.17	–	1450.12	y	–	–
	75	–	–	–	–	–	–	–	–
	100	–	–	–	–	–	–	–	–

Table 3
Impact of the number of requests on model (M1b)

Instances		M1a		M1b			
size	%req.	%gap	time	%gap	time (s)	#same	#feas.
4 × 4	10	2.51	2.16	2.44	0.07	65	84
	25	6.57	3.61	6.56	0.41	99	99
	50	9.84	5.48	9.84	1.88	100	100
	75	5.92	4.06	5.92	2.57	100	100
	100	11.14	5.30	11.14	5.30	100	100

(M1b) optimal solution to be feasible should depend on the number of requests. In order to check this intuitive idea, models (M1a) and (M1b) have been run on the second set of instances, i.e. 100 randomly generated sets of requests for each number of requests (save 100% requests). The results are reported in Table 3. For each model, "%gap" and "time" stand, respectively, for the average relative integrality gap and the average computing time in seconds. The columns "#feas." and "#same" report the number of times, on the 100 sets of requests, model (M1b) optimal solution was (i) similar to model (M1a) optimal solution and (ii) feasible.

It appears the empirical probability of computing a strongly connected solution with model (M1b) is close to 1, even with 25% requests. With only 10% requests, this probability is still 0.84. The probability of getting the same solution with models (M1a) and (M1b) is close to 1 with at least 25% requests. Below, it drops to 0.65 with 10% requests. This means with 10% requests, the solution may differ on arcs that are not used to define the shortest path for the requests. Yet, on some occasions, the solution provided by model (M1b) has a better optimal value, but it is not strongly

Table 4
Results for models (M2a) and (M2b) (maximization)

Instances				M2a	M2b
size	%req.	z^*	$\bar{z}$	time	time
4×4	10	9588	344401	14.96	9.82
	25	16743	834489	18.67	14.59
	50	28109	1683328	18.83	12.24
	75	37602	2502418	21.18	13.87
	100	71998	3216800	17.60	21.25
5×5	10	–	–	–	–
	25	–	–	–	–
	50	–	–	–	–
	75	–	–	–	–
	100	–	–	–	–

connected. This is highlighted by the %gap: 2.51% for model (M1a) versus 2.44% for model (M1b) for 10% requests.

Even if there seems to be a correlation between the percentage of requests and the probability of obtaining a strongly connected solution with model (M1b), counterexamples do exist. For example, if the instance contains n requests that define a circulation on the nodes, the optimal solution is guaranteed to be strongly connected. On the other hand, if there is at least one request from every node but one to every other node (hence $(n-1)^2$ requests), the optimal solution will not be strongly connected. However, the probability of such configurations is small.

Table 4 reports the results for the maximization on the first set of instances with models (M2a) and (M2b) respectively. Only instances with 16 nodes have been solved in the 2h time limit, while some 36 nodes instances have been solved to optimality in Table 2. This indicates experimentally that the SNOP with maximization criterion is more difficult than the SNOP with minimization criterion. The value of the linear relaxation is the same for both models, as stated by Property 2. The large integrality gaps that can be observed are a direct consequence of big-M in constraints (14). One can also note model (M2b), which uses a smaller number of flows than model (M2a), leads to lower computing times. However, the impact of the number of requests on the CPU time seems to be smaller than in minimization.

5.2. *Results for the proposed metaheuristics*

Tables 5 and 6 present the results for the proposed metaheuristics (MS, ILS, RILS, ELS, RELS and VNS), resp. applying the minimization and maximization criteria for SNOP, and considering the first and the third set of instances. A limit of 200 calls to the VND local search is set, which means 200 iterations for MS, ILS and RILS.

The ELS and the RELS rely on two parameters, the number of iterations and the number of copies in each iteration, whose product must be equal to 200. The calibration has been done with Iterated Racing for Automatic Algorithm Configuration package (IRACE) (López-Ibáñez et al., 2016) on the 5×5 instances. This leads to 20 iterations (and 10 copies) for ELS, both for

Table 5
Results for the metaheuristics (minimization)

Instances			MS			ILS			RILS			ELS			RELS			VNS		
Size	%req.	BKS	UB	%dev	time	UB	%dev	time	UB	%dev	time	UB	%dev	time	UB	%dev	time	UB	%dev	time
4 × 4	10	**3782**	**3782**	0.0	0.2	3842	1.6	0.1	3842	0.0	0.1	3842	1.6	0.1	**3782**	0.0	0.1	**3782**	0.0	0.2
	25	**7791**	**7791**	0.0	0.3	**7791**	0.0	0.1	**7791**	0.0	0.2	**7791**	0.0	0.1	**7791**	0.0	0.1	**7791**	0.0	0.3
	50	**13969**	**13969**	0.0	0.3	14323	2.5	0.1	**13969**	0.0	0.1	14323	2.5	0.1	**13969**	2.5	0.1	**13969**	0.0	0.3
	75	**15244**	**15244**	0.0	0.3	15340	0.6	0.1	**15244**	0.0	0.1	15314	0.5	0.1	**15244**	0.5	0.1	**15244**	0.0	0.4
	100	**43160**	**43160**	0.0	0.3	**43160**	0.0	0.1	**43160**	0.0	0.2	**43160**	0.0	0.2	**43160**	0.0	0.1	**43160**	0.0	0.3
5 × 5	10	**12333**	**12333**	0.0	2.0	**12333**	0.0	0.8	**12333**	0.0	0.9	**12333**	0.0	0.8	**12333**	0.0	0.7	**12333**	0.0	1.5
	25	**31332**	31376	0.1	2.0	**31332**	0.0	0.8	31376	0.1	1.0	**31332**	0.0	0.9	**31332**	0.0	0.6	31376	0.1	1.7
	50	**49136**	**49136**	0.0	2.0	49420	0.8	0.8	**49136**	0.0	0.9	**49136**	0.0	0.8	**49136**	0.0	0.6	**49136**	0.0	1.4
	75	**77522**	**77522**	0.0	2.5	**77522**	0.0	1.0	**77522**	0.0	1.1	**77522**	0.0	1.0	**77522**	0.0	0.7	**77522**	0.0	2.0
	100	**109458**	**109458**	0.0	2.6	**109458**	0.0	1.3	**109458**	0.0	1.2	109576	0.1	1.0	109576	0.1	0.9	**109458**	0.0	2.0
6 × 6	10	**29178**	**29178**	0.0	9.1	29218	0.0	3.2	**29178**	0.0	3.8	**29178**	0.0	3.2	**29178**	0.0	3.5	29210	0.1	7.0
	25	**72371**	**72371**	0.0	12.0	**72371**	0.0	3.7	**72371**	0.0	4.6	**72371**	0.0	3.7	**72371**	0.0	4.9	**72371**	0.0	10.0
	50	**160299**	160847	0.3	11.3	160655	0.2	4.3	160959	0.4	5.0	**160299**	0.0	4.7	160707	0.3	3.7	160959	0.4	8.5
	75	*143969*	145325	0.9	12.7	144051	0.2	5.8	144561	0.4	6.1	144317	0.2	5.8	144119	0.1	4.6	144685	0.5	9.4
	100	*234814*	237840	1.3	11.6	*234814*	0.0	4.9	237056	1.0	5.8	*234814*	0.0	4.7	*234814*	0.0	3.7	*234814*	0.0	9.2
8 × 8	10	*120528*	122166	1.4	167.7	*120528*	0.0	63.2	120800	0.2	67.6	121046	0.4	61.5	120690	0.1	65.7	121020	0.4	109.9
	25	*267713*	269909	0.9	179.9	*267713*	0.0	60.6	268089	0.1	71.0	*267713*	0.0	60.2	267809	0.0	52.4	267793	0.0	114.6
	50	*557686*	562058	0.8	177.5	*557686*	0.0	59.3	562210	0.8	75.6	558658	0.2	60.7	558474	0.1	52.4	559772	0.4	133.9
	75	*930992*	939114	0.9	168.0	933494	0.3	58.7	938174	0.8	65.6	931130	0.0	57.6	*930992*	0.0	46.9	935762	0.5	121.1
	100	*1479278*	1490268	0.7	188.2	1491574	0.8	65.3	1490620	0.8	76.6	1491588	0.8	67.1	1485382	0.4	63.3	1480154	0.1	125.6
10 × 10	10	*435610*	437472	0.4	1230.9	439038	0.8	442.5	*435610*	0.0	498.1	439026	0.8	431.9	436780	0.3	358.6	435612	0.0	564.1
	25	*757124*	761662	0.6	1237.3	*757124*	0.0	368.3	759942	0.4	416.4	760164	0.4	407.3	759520	0.3	399.6	757630	0.1	652.6
	50	*2042847*	2067149	1.2	1390.9	*2042847*	0.0	478.4	2059623	0.8	575.4	2048995	0.3	489.3	2055181	0.6	467.1	2053661	0.5	949.4
	75	*3327881*	3364629	1.1	1332.3	*3327881*	0.0	445.7	3353731	0.8	531.4	3339893	0.4	472.8	3331299	0.1	464.8	3360021	1.0	846.1
	100	*3809398*	3823346	0.4	1331.5	3819884	0.3	455.2	3825700	0.4	532.8	*3809398*	0.0	468.6	3810430	0.0	504.3	3820276	0.3	864.3
92	11.8	820460	826525	0.7	147.6	821413	0.1	61.7	823930	0.4	63.4	822624	0.3	60.1	**820460**	0.0	59.5	820504	0.0	279.8
nb_BKS			26	11		14			11			12			14			11		

Table 6
Results for the metaheuristics (maximization)

Instances			MS			ILS			RILS			ELS			RELS			VNS		
Size	%req.	BKS	UB	%dev	time	UB	%dev	time	UB	%dev	time	UB	%dev	time	UB	%dev	time	UB	%dev	time
4 × 4	10	**9588**	**9588**	0.0	0.2	**9588**	0.0	0.1	**9588**	0.0	0.1	**9588**	0.0	0.1	**9588**	0.0	0.1	**9588**	0.0	0.2
	25	**16743**	**16743**	0.0	0.2	**16743**	0.0	0.1	**16743**	0.0	0.1	**16743**	0.0	0.1	**16743**	0.0	0.1	**16743**	0.0	0.3
	50	**28109**	27609	1.8	0.2	27609	1.8	0.1	**28109**	0.0	0.1	27609	1.8	0.1	**28109**	1.8	0.1	**28109**	0.0	0.3
	75	**37602**	36934	1.8	0.2	35130	6.6	0.1	**37602**	0.0	0.1	36934	1.8	0.2	36934	1.8	0.1	**37602**	0.0	0.3
	100	**71998**	**71998**	0.0	0.2	69454	3.5	0.1	**71998**	0.0	0.1	**71998**	0.0	0.1	**71998**	0.0	0.1	**71998**	0.0	0.3
5 × 5	10	*31181*	*31181*	0.0	0.8	30829	1.1	0.7	31171	0.0	0.7	30829	1.1	0.7	31171	0.0	0.5	31171	0.0	1.3
	25	*78238*	*78238*	0.0	1.0	77246	1.3	0.7	77668	0.7	0.8	77246	1.3	0.7	76030	2.8	0.5	*78238*	0.0	1.5
	50	*150146*	140234	6.6	1.3	*150146*	0.0	0.8	*150146*	0.0	0.8	*150146*	0.0	0.8	*150146*	0.0	0.6	*150146*	0.0	1.6
	75	*209352*	206662	1.3	1.2	*209352*	0.0	0.9	206662	1.3	0.9	192154	8.2	0.8	207916	0.6	0.5	*209352*	0.0	1.5
	100	*246192*	236674	3.9	1.1	234364	4.8	0.8	234364	4.8	0.9	239256	2.8	0.8	*246192*	0.0	0.6	238286	3.2	1.5
6 × 6	10	*105304*	100474	4.7	4.2	93966	10.8	2.8	100774	4.3	2.9	93966	10.8	2.9	*105304*	0.0	3.2	104092	1.1	5.3
	25	*224339*	204435	8.9	4.5	206939	7.8	3.3	212097	5.5	3.8	206939	7.8	3.4	*224339*	0.0	2.4	223087	0.0	0.6
	50	*492467*	484073	1.7	5.1	475723	3.4	2.9	474025	3.7	3.9	486385	1.2	4.3	483285	1.9	2.6	*492467*	0.0	7.5
	75	*457213*	448421	1.9	5.6	455851	0.3	3.8	439583	3.9	4.2	452611	1.0	4.0	*457213*	0.0	3.0	453831	0.7	7.9
	100	*697236*	682600	2.1	5.2	*697236*	0.0	3.6	673680	3.4	4.2	687986	1.3	4.0	*697236*	0.0	2.9	668394	4.1	7.1
8 × 8	10	*525223*	449195	14.5	50.8	452267	13.9	33.6	461249	12.2	37.9	508023	3.3	35.6	512016	2.5	37.1	475937	9.4	65.1
	25	*1244047*	1105387	11.2	54.8	*1244047*	0.0	35.4	1119543	10.0	40.8	1198415	3.7	39.5	942265	24.3	30.8	1145677	7.9	84.9
	50	*2436482*	2142644	12.1	56.7	2271568	6.8	39.9	2177088	10.7	42.7	2338696	4.0	39.1	2242096	8.0	31.4	2153604	11.6	82.2
	75	*3482980*	3145328	9.7	59.6	3398188	2.4	36.5	3318676	4.7	44.2	3427392	1.6	41.8	*3482980*	0.0	31.0	3305072	5.1	88.8
	100	*5428182*	4685432	13.7	51.8	4685432	13.7	34.3	4903500	9.7	39.7	5384026	0.8	37.0	5261022	3.1	31.1	4907118	9.6	88.0
10 × 10	10	*1949546*	1859260	4.6	281.1	1881122	3.5	250.2	1822164	6.5	260.5	1947502	0.1	249.3	1862034	4.5	209.0	*1949546*	0.0	431.1
	25	*4262908*	3690164	13.4	352.5	4243588	0.5	257.4	3636788	14.7	299.9	3772352	11.5	302.8	3620462	15.1	275.3	4235338	0.7	260.8
	50	*10034431*	8330461	17.0	310.4	9238531	7.9	268.2	8872463	11.6	290.5	9964117	0.7	243.5	*10034431*	0.0	219.0	8988643	10.4	532.3
	75	*16492175*	14002133	15.1	330.3	15389023	6.7	299.3	14964623	9.3	288.5	*16492175*	0.0	283.8	16387161	0.6	234.7	16008913	2.9	542.6
	100	*16827816*	14835030	11.8	331.9	*16827816*	0.0	306.0	15173708	9.8	304.0	16171418	3.9	287.4	15067992	10.5	328.3	15382230	8.6	544.2
92	11.8	*3467395*	3132116	9.7	37.2	3465119	0.1	33.5	2984797	13.9	42.2	3373331	2.7	34.8	*3467395*	0.0	38.6	3446262	0.6	110.6
nb_BKS			26	5		7			6			5			13			10		

minimization and maximization. For RELS, the recommended value for the iterations was 100 in minimization and 40 in maximization, with quite a low sensitivity in the latter. Thus 100 iterations (and two copies) have been chosen for RELS. With these settings, RELS is quite close to RILS.

For each call of the shaking procedure in ILS, RILS, ELS and RELS, the number of iterations is randomly drawn in the interval $(0.01m, 0.05m)$ where m is the number of arcs. For VNS, $\tau_{\max}$ is set to $0.8m$, which means up to 80% of the arcs may be reversed.

The column BKS refers to the value of the best-known solution. Optimal values are in bold, whenever they have been found by models (M1a)–(M2b) in Tables 2 and 4. Otherwise, the value in italic corresponds to the best value obtained by the metaheuristics. For each method, the best upper bound (UB), the relative gap to the BKS (%dev) and the running time in seconds (time) are reported.

For the minimization criterion in Table 5, the MS, ILS, RILS, ELS, RELS and VNS found the BKS, respectively 11, 14, 11, 12, 14 and 11 times out of 26 instances (25 grids and 1 city center). In general, the gaps remain below 2% of the optimal solutions for small instances and 2% of BKS for the other ones for all methods. In terms of running time, the MS is the most time consuming one. This is an expected result since the MS method generates random initial solutions that do not benefit from optimized parts of the previous known local optima. As a consequence, the local search needs to improve each initial solution until a local optimum is reached. For the realistic instance, RELS is the only one to obtain the BKS, the ILS and the VNS being close behind.

For the maximization criterion in Table 6, the MS, ILS, RILS, ELS, RELS and VNS found the BKS, respectively 5, 7, 6, 5, 13 and 10 times. The computational times are significantly smaller in maximization, and MS time consumption is not much higher than the other methods. Yet, the instances seem harder to solve: the BKS has been found a total of 46 times in maximization as opposed to 73 times in minimization. In addition, relative gaps to the BKS in maximization are larger than in minimization. Even if the settings in RELS (100 iterations, two copies) is quite close to RILS, the results differ significantly. This means the copying mechanism in ELS/RELS provides a substantial benefit. For the realistic instance, RELS finds the BKS, closely followed by ILS. The other methods get significantly worse results, especially MS and RILS.

The next set of experiments is quite similar except that the number of calls to the local search is now unlimited. A time limit, corresponding to the time spent by the slower method (i.e. VNS) in Tables 5 and 6, is set instead for each method on the same instance. Thus, each method is given the same amount of time and uses at least the time it spent in these tables. The results for the minimization and for the maximization are presented in Tables 7 and 8, respectively. Optimal values are in bold and best-known results are in italic. The number of time each method got the best result is reported at the bottom of each table.

As can be seen, ILS and RILS get a significant improvement. In minimization, ILS finds 19 BKS, five more than with 200 calls to the LS. RILS finds five more BKS as well. It is worth noting the BKS is improved for two instances: 8×8 75% by RILS and 10×10 75% by ILS. MS, ELS, RELS do not benefit much from the increase of CPU time.

The situation is similar for maximization. ILS is the method benefiting the most from the increase of cpu time (12 BKS found instead of 7). The other methods improve their results too, to a smaller extent, with the exception of MS which seems to reach its limits. Yet, RELS still looks the most effective of these approaches in maximization. The BKS of 4 instances is improved (5×5 10% by RILS, 6×6 10% by ILS and RELS, 10x10 10% by ILS, and *clermont* 92 nodes 11.8% by ILS).

Table 7
Results for the metaheuristics (fixed time minimization)

Instances			MS	ILS	RILS	ELS	RELS	VNS
Size	%req.	cpu	UB	UB	UB	UB	UB	UB
4 × 4	10	0.2	**3782**	3842	**3782**	3842	**3782**	**3782**
	25	0.3	**7791**	**7791**	**7791**	**7791**	**7791**	**7791**
	50	0.3	**13969**	**13969**	**13969**	14297	**13969**	**13969**
	75	0.4	**15244**	15314	**15244**	15314	**15244**	**15244**
	100	0.3	**43160**	**43160**	**43160**	**43160**	**43160**	**43160**
5 × 5	10	1.5	**12333**	**12333**	**12333**	**12333**	**12333**	**12333**
	25	1.7	**31332**	**31332**	**31332**	**31332**	**31332**	31376
	50	1.4	**49136**	**49136**	**49136**	**49136**	**49136**	**49136**
	75	2.0	**77522**	**77522**	**77522**	**77522**	**77522**	**77522**
	100	2.0	**109458**	**109458**	**109458**	109576	**109458**	**109458**
6 × 6	10	7.0	**29178**	**29178**	**29178**	**29178**	**29178**	29210
	25	10.0	**72371**	**72371**	**72371**	**72371**	**72371**	**72371**
	50	8.5	160847	**160299**	160655	**160299**	160473	160665
	75	9.4	145133	*144051*	*144051*	144317	144119	144645
	100	9.2	236694	*234814*	*234814*	*234814*	*234814*	*234814*
8 × 8	10	109.9	122080	*120528*	120800	121046	120690	*120528*
	25	114.6	269909	*267713*	268089	*267713*	*267713*	267793
	50	133.9	561188	*557686*	561610	575554	559856	559772
	75	121.1	939114	932572	*929680*	931130	930992	931938
	100	125.6	1490269	*1479278*	1481160	1491588	1483714	1480154
10 × 10	10	564.1	437472	437246	*435610*	439026	436780	435612
	25	652.6	761662	*757124*	759942	760164	758676	757630
	50	949.4	2065205	*2042847*	2057581	2048995	2051431	2053661
	75	846.1	3364629	*3326229*	3353731	3333005	3331299	3346371
	100	864.3	3823346	3812642	3825700	*3809398*	3810430	3819432
92	11.8	279.8	825534	821413	822293	822624	*820460*	820504
nb_best			12	19	16	12	14	12

The best-known solution obtained by the metaheuristics on the realistic instance is displayed in Figs 7–8 for the minimization and the maximization respectively. These two solutions can be compared on their structural properties in order to assess their differences better. This is done in Table 9 for three properties: the node balance, the return length and the increase in travel distance for each request. The node balance is defined as the absolute difference between the number of incoming arcs and the number of outgoing arcs. Thus, it measures the equilibrium of a node in terms of entering/leaving arcs. The return length, for a node, is the length of the shortest cycle containing this node. It provides some information on the local arc configuration in terms of travel distance. For a request, the increase in travel distance is the relative difference between the length of the shortest path in the current solution and the length of the shortest path in the undirected graph. It shows how close to the best possible situation the current solution is, with respect to the request. For the sake of clarity, the BKS corresponding to the minimization is referred as to BKS min, respectively BKS max for the maximization.

Table 8
Results for the metaheuristics (fixed time maximization)

Instances			MS	ILS	RILS	ELS	RELS	VNS
Size	%req.	cpu	UB	UB	UB	UB	UB	UB
4 × 4	10	0.2	**9588**	**9588**	**9588**	**9588**	**9588**	**9588**
	25	0.3	**16743**	**16743**	**16743**	**16743**	**16743**	**16743**
	50	0.3	27609	27609	**28109**	27609	**28109**	**28109**
	75	0.3	36934	35130	**37602**	36934	36934	**37602**
	100	0.3	**71998**	70804	**71998**	**71998**	**71998**	**71998**
5 × 5	10	1.3	31231	31171	*32399*	30828	*32399*	*32399*
	25	1.5	*78238*	77246	*78238*	77246	*78238*	*78238*
	50	1.6	*150146*	*150146*	*150146*	*150146*	*150146*	*150146*
	75	1.5	206662	*209352*	*209352*	204208	*209352*	*209352*
	100	1.5	239256	241260	240336	239256	*246192*	*246192*
6 × 6	10	5.3	102688	*107900*	104092	93966	*107900*	104092
	25	7.8	204923	210831	223087	*224339*	*224339*	223087
	50	7.5	484073	478621	475929	486385	483285	*492467*
	75	7.9	448421	455851	447275	452611	*457213*	453831
	100	7.1	682600	*697236*	679788	687986	*697236*	681318
8 × 8	10	65.1	469296	503104	489652	508023	*512016*	507736
	25	84.9	1105387	*1244047*	1119543	1198415	1010967	1145677
	50	82.2	2142644	*2398622*	2177088	2345886	2242096	2254874
	75	88.8	3300174	3398188	3422536	3427392	*3578280*	3305072
	100	88.0	4992488	5102718	5166662	*5384026*	5261022	4907118
10 × 10	10	431.1	1859260	*2014004*	1896464	1947502	1924276	1949546
	25	260.8	3690164	*4243588*	3636788	3772352	3900512	4235338
	50	532.3	8330461	9238531	8872463	9964117	*10034431*	9303039
	75	542.6	15768273	15422191	15179123	*16492175*	16387161	16471941
	100	544.2	14931142	*16827816*	15173708	16171418	15408808	15759412
92	11.8	110.6	3132116	*3675395*	3026080	3373331	3467395	3446262
nb_best			5	12	9	7	16	11

In terms of node balance, both solutions contain balanced nodes (min value 0). The maximal imbalance is 2 for BKS min, while it is 4 for BKS max. The average node imbalance is 0.54 for BKS min, and twice as much (1.02) for BKS max. The next columns report the number of nodes with a given node imbalance. One can note nearly half the nodes (45 out of 92) are balanced in BKS min, while more than one quarter of the nodes (26 out of 92) has an imbalance greater or equal to 2 in BKS max. Thus, in terms of node balance, BKS min contains much more balanced nodes than BKS max.

In terms of return length, the minimal value is almost the same for the two BKS. However, the maximal value is six times higher for BKS max (6996) than for BKS min (1147), and the average length is nearly four times larger (2116.0 against 552.6). When comparing with the smallest possible return time (using each edge at most once), the same behavior happens. Thus, on average, the local traffic consumes more distance in BKS max than in BKS min.

The difference is even more evident with the travel distance for each request. Even if there is at least one request whose travel distance is the smallest possible for each BKS (min = 0.0 %), the

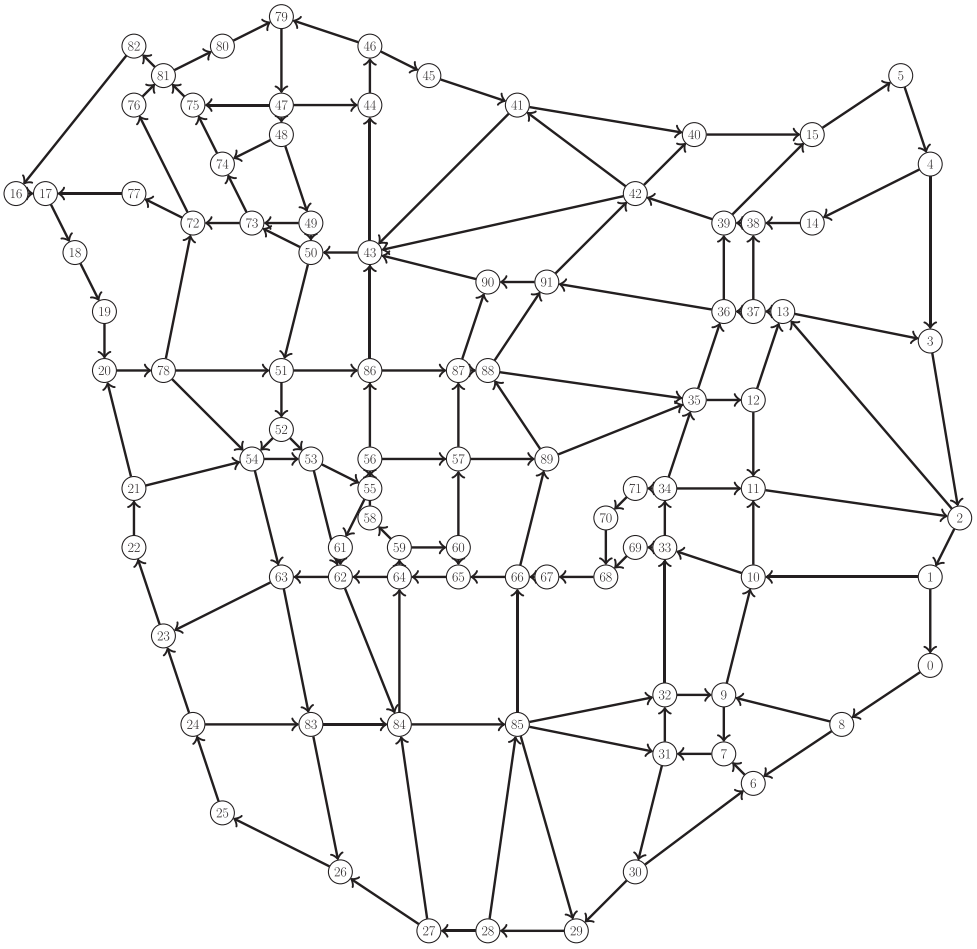


Fig. 7. Best found solution (minimization) for the realistic instance.

situation significantly differs when considering the average relative gap to the best distance: the travel distance is 53.8% higher than the smallest possible for BKS min while it is 707.2% larger for BKS max. The next columns report the number of requests whose travel distance gap belongs to a class: for instance, 192 requests have a gap between 0% and 10% in BKS min, almost the same (194) for BKS max. For the class 0%–10%, there is no significant difference. Starting from 10%–20%, BKS min obtains better results. This can also be seen for the class above 100% (at least twice the smallest possible distance): 115 requests fall within this class for BKS min, compared to 656 requests for BKS max. This is an expected result, since BKS min aims at providing the smallest total distance while BKS max does the opposite.

The convergence and the robustness have been analyzed in graphical terms, using a descent graph for solution values and TTTPlots (Aiex et al., 2007).

Figures 9–10 are representative examples of the trade-off between the objective values and running time on the 6×6 grid instance with 1260 requests. In Fig. 9, the evolution of the objective

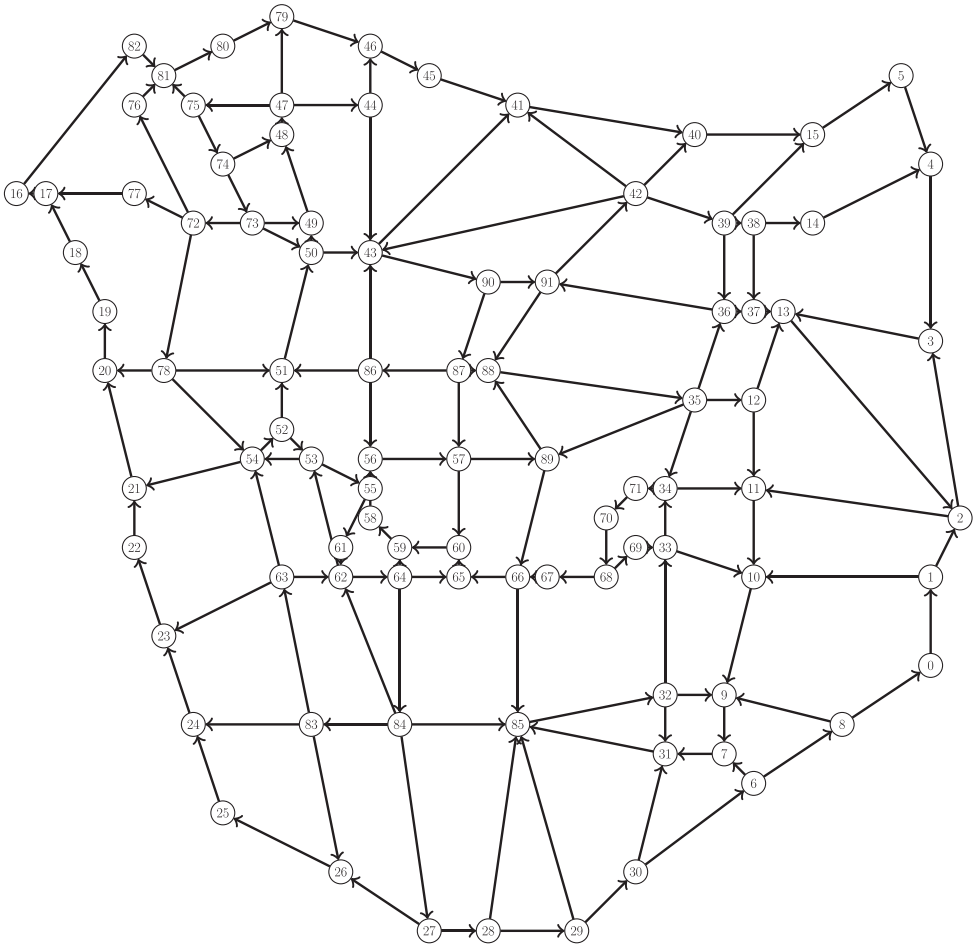


Fig. 8. Best found solution (maximization) for the realistic instance.

value for a single run of each method is reported. The horizontal axis indicates the running time in seconds, while the vertical one corresponds to the objective values. Thus, the closer a line is to the vertical axis, the faster is the method's convergence to better solutions in this run. Some interesting results can be noticed in these graphics, for instance, the VNS and the RELS show a fast convergence for SNOP with minimization criterion, and these two methods together with ILS are ones able to find the optimum solution. ILS is the heuristic with the fastest convergence for SNOP with maximization criterion and the only one able to meet the optimum solution. In both cases, SNOP with minimization and maximization criteria, the MS presents the worst convergence quality.

For the TTT plots in Fig. 10, 100 runs have been done for each method. The target value is set to 1.05BKS in minimization and to 0.95BKS in maximization. The horizontal and vertical axis stand respectively for the running time and the probability of finding a target value. Thus, the more vertical the plot is for a method, the more robust the method is. These graphics indicate ILS, ELS, RELS heuristics are the most robust ones for the SNOP with minimization criterion, while

Table 9
Properties of the BKS for the realistic instance

	Balance			# nodes with balance				
Node balance	Min	Avg.	Max	0	±1	±2	±3	±4
BKS min	0	0.54	2	45	40	7	–	–
BKS max	0	1.02	4	29	37	22	3	1

	Length			% gap to best		
Return length	Min	Avg.	Max	Min	Avg.	Max
BKS min	134	552.5	1147	0.00	38.71	259.59
BKS max	146	2116.0	6996	0.00	430.29	1494.20

	% gap to best			# requests with gap to best					
Request length	Min	Avg.	Max	0%	10%	20%	50%	100%	>100%
BKS min	0.0	53.8	2383.9	192	157	167	262	99	115
BKS max	0.0	707.2	17174.2	194	24	35	52	31	656

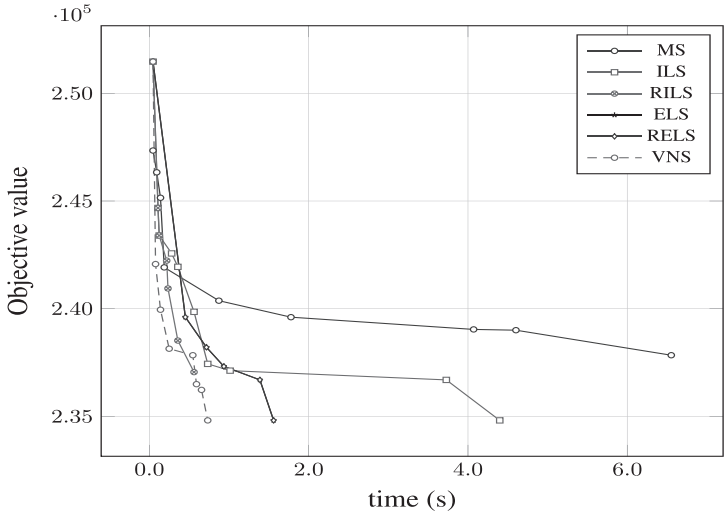
ILS, RILS, ELS, RELS present close results in terms of robustness for SNOP with maximization criterion. These results are more statistically meaningful since they rely on a larger number of runs.

Summarizing the analysis previously done in terms of best solutions found, running time, convergence and robustness the ILS seems a good compromise for SNOP with minimization and maximization criteria.

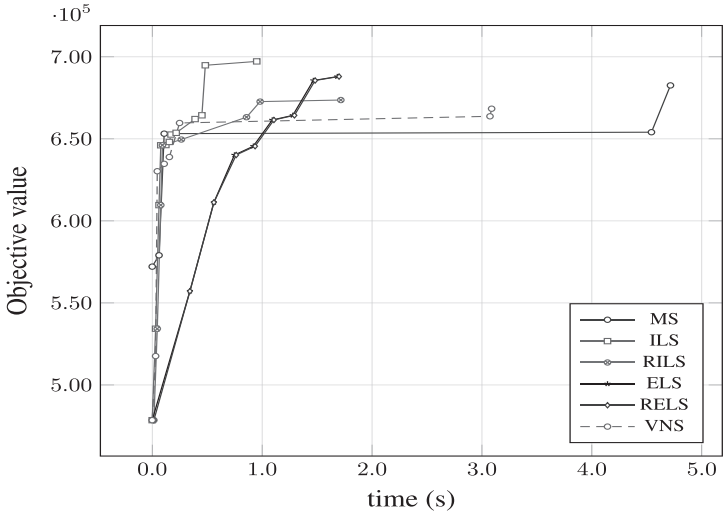
6. Concluding remarks

Several mathematical formulations and sophisticated heuristics were proposed and tested over grid instances and a realistic one. The former formalized the problem, while the latter allowed us to obtain high-quality results. In terms of mathematical formulation, models (M1d) and (M2b) are, respectively, the best ones for the SNOP with minimization and maximization. Considering the metaheuristics, the task of choosing one is not simple. Globally, ILS, ELS and RELS have the best performances considering the analysis done and the set instances used. However, ILS appears as a good compromise whenever one looks for a unique method for both SNOP with minimization and maximization criteria.

SNOP is a baseline problem for solving tactical optimization problems in urban networks. Thus, the contributions of this article open several avenues for further research. One direction is to investigate a more general case of SNOP on multigraphs (multiple directed arcs between two nodes) or in mixed graphs (directed and not-directed arcs between two nodes). These two cases generalize the SNOP to address an overall transportation network, differing the way the network is modeled in terms of a graph. Other avenues of research cover the development of other mathematical formulations, valid inequalities, new components for the heuristics methods, and exact algorithms. SNOP presents also theoretical interest in graph theory and network design.



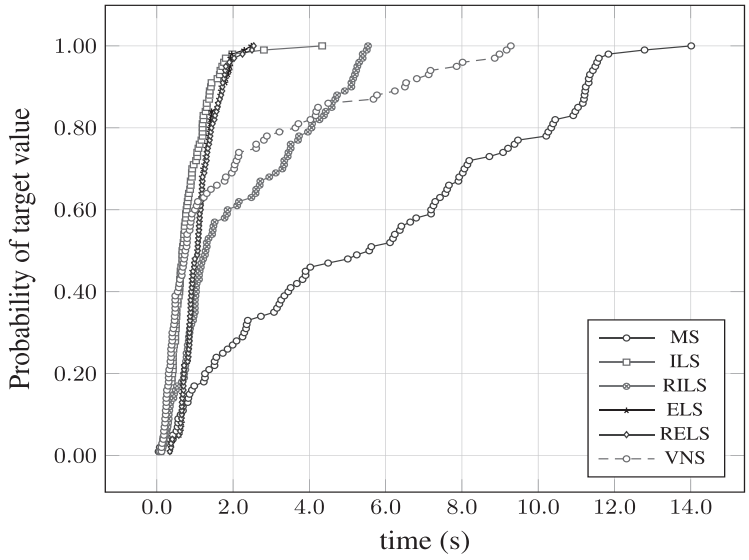
(a) Minimization



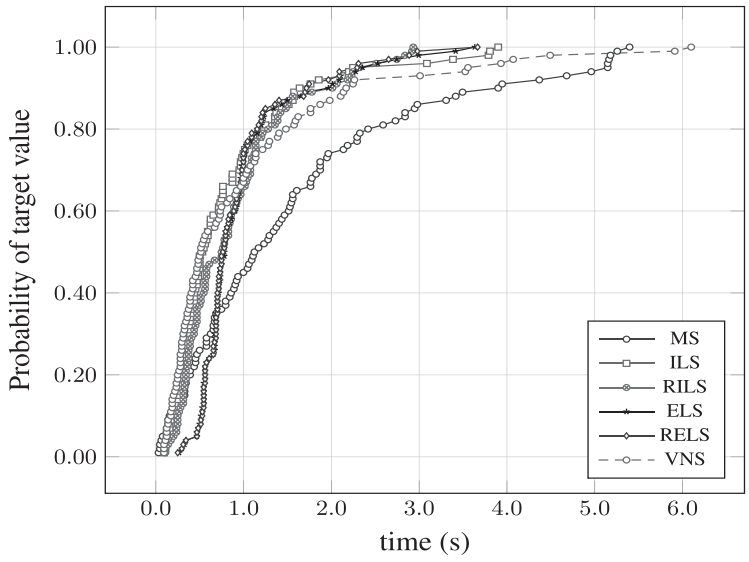
(b) Maximization

Fig. 9. Methods trajectory for the grid instance 6 × 6 with 1260 requests.

As mentioned before, SNOP is a key problem with extensions appearing in a number of practical applications. For instance, both mathematical models and methods can be adapted to handle other urban network problems such as the installation of green lanes (pathways for pedestrians, green pathways, bikeways, etc.) and dynamic routes (routes used for different objectives, and whose direction can also change over the day). Green lanes and dynamic routes appear for different reasons. Green lanes have been implemented to try to change the inhabitants' life style and to encourage more eco-responsible and healthier habits. Dynamic routes are justified by better use of the physical



(a) Minimization



(b) Maximization

Fig. 10. Methods TTTplot for the grid instance 6×6 with 1260 requests.

space in order to reduce daily congestion. Even if green lanes and dynamic routes are different, they share some common technical and modeling issues. In particular, which routes are more likely to adopt one of these systems? How should one organize the transport flow after such a modification? What are the impacts on the environment? Poor choices, local decisions or short-term decisions,

concerning the implantation of green lanes or dynamic routes can generate congestion in unexpected points of the urban network.

Acknowledgments

The authors would like to thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions. This study is financed by Normandie Region, FEDER-FSE/IEJ – Haute Normandie in the context of the emerging LIS project.

References

Afsar, H.M., Prins, C., Santos, A.C., 2014. Exact and heuristic algorithms for solving the generalized vehicle routing problem with flexible fleet size. *International Transactions in Operational Research* 21, 1, 153–175.

Aiex, R.M., Resende, M.G.C., Ribeiro, C.C., 2007. TTT plots: a perl program to create time-to-target plots. *Optimization Letters* 1, 4, 355–366.

Aksu, D., Ozdamar, L., 2014. A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review* 61, 56–67.

Burkard, R., Feldbacher, K., Klinz, B., Woeginger, G., 1999. Minimum-cost strong network orientation problems: classification, complexity, and algorithms. *Networks* 33, 1, 57–70.

Cantarella, G., Pavone, G., Vitetta, A., 2006. Heuristics for urban road network design: Lane layout and signal settings. *European Journal of Operational Research* 175, 3, 1682–1695.

Chung, F., Garey, M., Tarjan, R., 1985. Strongly connected orientations of mixed multigraphs. *Networks* 15, 4, 477–484.

Chvátal, V., Thomassen, C., 1978. Distances in orientations of graphs. *Journal of Combinatorial Theory, Series B* 24, 1, 61–75.

Coco, A., Duhamel, C., Santos, A., 2020. Modeling and solving the multi-period disruptions scheduling problem on urban networks. *Annals of Operations Research* 285, 1-2, 427–443.

Conte, A., Grossi, R., Marino, A., Rizzi, R., Versari, L., 2016. Directing road networks by listing strong orientations. *Lecture Notes in Computer Science*. Springer Nature, New York, pp. 83–95.

Cormen, T., Leiserson, C., Rivest, R., Stein, C., 2009. *Introduction to Algorithms* (3rd edn.). The MIT Press, Cambridge, MA.

Drezner, Z., Wesolowsky, G., 1997. Selecting an optimum configuration of one-way and two-way routes. *Transportation Science* 31, 4, 386–394.

Farahani, R., Miandoabchi, E., Szeto, W., Rashidi, H., 2013. A review of urban transportation network design problems. *European Journal of Operational Research* 229, 2, 281–302.

Fukunaga, T., 2012. Graph orientations with set connectivity requirements. *Discrete Mathematics* 312, 15, 2349–2355.

Gallo, M., D'Acierno, L., Montella, B., 2010. A meta-heuristic approach for solving the urban network design problem. *European Journal of Operational Research* 201, 1, 144–157.

Gaskins, R., Tanchoco, J., 1987. Flow path design for automated guided vehicle systems. *International Journal of Production Research* 25, 667–676.

Huang, Y., Santos, A., Duhamel, C., 2020a. Bi-objective methods for road network problems with disruptions and connecting requirements. *Journal of the Operational Research Society* 71, 12, 1959–1971.

Huang, Y., Santos, A., Duhamel, C., 2020b. Model and methods to address urban road network problems with disruptions. *International Transactions in Operational Research* 27, 6, 2715–2739.

Italiano, G., Laura, L., Santaroni, F., 2012. Finding strong bridges and strong articulation points in linear time. *Theoretical Computer Science* 447, 74–84.

Kaspi, M., Tanchoco, J., 1990. Optimal flow path design of unidirectional AGV systems. *International Journal of Production Research* 28, 6, 1023–1030.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Stützle, T., Birattari, M., 2016. The IRACE package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3, 43–58.

Lourenço, H., Martin, O., Stützle, T., 2002. Iterated local search. In Glover, F., Kochenberger, G. (eds), *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston, pp. 321–353.

Mladenović, N., Hansen, P., 1997. Variable neighborhood search. *Computers & Operations Research* 24, 11, 1097–1100.

Ortigosa, J., Menendez, M., Gayah, V., 2015. Analysis of network exit functions for various urban grid network configurations. *Transportation Research Record: Journal of the Transportation Research Board* 2491, 12–21.

Qiu, L., Hsu, W.J., Huang, S.Y., Wang, H., 2002. Scheduling and routing algorithms for AGVs: a survey. *International Journal of Production Research* 40, 3, 745–760.

Robbins, H., 1939. A theorem on graphs, with an application to a problem on traffic control. *The American Mathematical Monthly* 46, 281–283.

Roberts, F., 1978. Graph theory and its applications to problems of society, *CBMS-NSF Regional Conference Series in Applied Mathematics*. Vol. 29. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

Roberts, F., Xu, Y., 1988. On the optimal strongly connected orientations of city street graphs I: Large grids. *SIAM Journal on Discrete Mathematics* 1, 2, 199–222.

Roberts, F., Xu, Y., 1994. On the optimal strongly connected orientations of city street graphs IV: Four east-west avenues or north-south streets. *Discrete Applied Mathematics* 49, 1-3, 331–356. Special Volume Viewpoints on Optimization.

Sakuraba, C., Santos, A., Prins, C., 2016a. Work-troop scheduling for road network accessibility after a major earthquake. *Electronic Notes in Discrete Mathematics* 52, 317–324.

Sakuraba, C., Santos, A., Prins, C., Bouillot, L., Durand, A., Allenbach, B., 2016b. Road network emergency accessibility planning after a major earthquake. *EURO Journal Computational Optimization* 4, 381–402.

Ullrich, G., 2014. The history of automated guided vehicle systems. *Automated Guided Vehicle Systems*. Springer, Berlin Heidelberg, pp. 1–14.

Vancea, A., Orha, I., 2019. A survey in the design and control of automated guided vehicle systems. *Carpathian Journal of Electronic and Computer Engineering* 12, 41–49.

Venkataramanan, M.A., Wilson, K., 1991. A branch-and-bound algorithm for flow-path design of automated guided vehicle systems. *Naval Research Logistics* 38, 3, 431–445.

Vis, I.F., 2006. Survey of research in the design and control of automated guided vehicle systems. *European Journal of Operational Research* 170, 3, 677–709.

Wolf, S., Merz, P., 2007. Evolutionary local search for the super-peer selection problem and the p-hub median problem. In Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds), *Hybrid Metaheuristics*, Lecture Notes in Computer Science. Vol. 4771. Springer, Berlin Heidelberg, pp. 1–15.

Yücel, E., Salman, F., Arsik, I., 2018. Improving post-disaster road network accessibility by strengthening links against failures. *European Journal of Operational Research* 269, 2, 406–422.