European Journal of Operational Research 224 (2013) 458–465

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

The accessibility arc upgrading problem

Pablo A. Maya Duque [a,c,*], Sofie Coene [b], Peter Goos [a,d], Kenneth Sörensen [a], Frits Spieksma [b]

[a] University of Antwerp, Faculty of Applied Economics, ANT/OR, Belgium
[b] KU Leuven, Faculty of Business and Economics, ORSTAT, Belgium
[c] Universidad de Antioquia, Faculty of Engineering, Colombia
[d] Erasmus University Rotterdam, Erasmus School of Economics, The Netherlands

ARTICLE INFO

Article history:
Received 8 February 2012
Accepted 2 September 2012
Available online 12 September 2012

Keywords:
Network upgrading problem
Knapsack problem
Variable neighbourhood search

ABSTRACT

The accessibility arc upgrading problem (AAUP) is a network upgrading problem that arises in real-life decision processes such as rural network planning. In this paper, we propose a linear integer programming formulation and two solution approaches for this problem. The first approach is based on the knapsack problem and uses the knowledge gathered from an analytical study of some special cases of the AAUP. The second approach is a variable neighbourhood search with strategic oscillation. The excellent performance of both approaches is demonstrated using a large set of randomly generated instances. Finally, we stress the importance of a proper allocation of scarce resources in accessibility improvement.

1. Introduction

Accessibility is formally defined by Donnges (2003) as the degree of difficulty people or communities have in accessing locations for satisfying their basic social and economic needs. This concept has been recognised to play an important role in the quality of life as well as the potential for development of communities and regions. The road network is one of the main elements that contributes to the accessibility. This is particularly true in rural areas of lesser-developed countries, where the road network ensures the accessibility to the economic and social infrastructure and to facilities, such as hospitals, usually located in regional centres or in more developed cities. In this paper, we study the accessibility arc upgrading problem (AAUP), a network upgrading problem in which resources have to be allocated in order to improve the accessibility to a set of vertices in a network. In the domain of rural road network planning, this problem arises when allocating resources to upgrade roads of a rural transport network, in order to improve the access that communities in small villages have to regional centres. We proceed by giving a precise description of this problem.

The AAUP can be described as follows: Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed connected graph in which the vertex set $\mathcal{V}$ is partitioned into two different sets $\mathcal{V}_1$ and $\mathcal{V}_2$. Vertices in $\mathcal{V}_1$ are called centres, while vertices in $\mathcal{V}_2$ are called regular vertices. Each arc e in $\mathcal{E}$ has a current level and a set of possible upgrading levels. The level of an arc determines the time required to traverse it. An upgrading cost is incurred when improving an arc from its current level to a specific upgrading level. There is a total budget B to upgrade the level of some arcs. For each vertex j in $\mathcal{V}_2$, a weight w_j (e.g., number of inhabitants) is given. We define as measure of the accessibility of regular vertex j the travel time from j to the closest centre i in $\mathcal{V}_1$. An *upgrading strategy* specifies a set of arcs to be upgraded and the level to which each of them has to be improved. The objective is to find an upgrading strategy that does not exceed the budget B and minimises the weighted sum of the accessibility measures, i.e., the weighted sum of the times required to travel from each vertex j in $\mathcal{V}_2$ to its nearest centre i in $\mathcal{V}_1$.

The rest of this paper is structured as follows. In Section 2, we propose a linear integer programming formulation of the AAUP problem. Section 3 reviews the literature, and, in Section 4, we analyse special cases. Section 5 proposes heuristic methods for the AAUP, and, in Section 6, we test these methods on randomly generated instances. Section 7 discusses the potential practical impact of the AAUP. Finally, Section 8 summarises the main contributions of this work and highlights some opportunities for future research on this topic.

2. Mathematical formulation

Based on the mathematical formulation described in Campbell and Lowe (2006), the AAUP can be formulated as a non-linear binary programming model, as shown by Maya Duque and Sörensen (2011). In this paper, we propose an alternative formulation in which the AAUP is defined as a special case of a more general problem called budget constrained minimum cost flow problem (BC-MCFP).

* Corresponding author. Address: University of Antwerp, Stadscampus S.B. 513, Prinsstraat 13, 2000 Antwerp, Belgium. Tel.: +32 32654061; fax: +32 32654901.
E-mail address: pmayaduque@gmail.com (P.A. Maya Duque).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.09.005

In the BC-MCFP, a given amount of flow has to be sent from a set of supply vertices or sources, through the arcs of a network, to a set of demand vertices or sinks. For each existing arc in the network, there is set of possible upgrading levels. Therefore, for each existing arc, we define one new arc per possible upgrading level connecting the same pair of vertices. Thus, in the BC-MCFP formulation, $\mathcal{E}$ represents the augmented set of arcs that contains all the original arcs and the arcs generated for each possible upgrading level. For each arc in $\mathcal{E}$, there is a cost per unit of flow, and a fixed cost associated with the use of the arc. In our particular setting of the BC-MCFP, there is no fixed cost for using an arc at its lowest level, but that cost increases with the upgrading level. The cost per unit of flow decreases as the arc is upgraded. The problem is to find a minimum cost flow, such that the sum of the fixed costs incurred by using some of the arcs at an upgraded level is limited to a fixed budget. Basically, this problem is a minimum cost flow problem that involves an additional set of decision variables related to the upgrading decisions.

Consider the variable x_e which is equal to the flow over arc e, and a binary variable y_e which is equal to 1 if the arc e is used, and 0 otherwise. Let $\delta^+(i)$ and $\delta^-(i)$ be the forward and backward stars of vertex i, respectively. Furthermore, let parameter d_i denote the demand or supply in vertex i, and let p_e and c_e represent the fixed cost of using arc e, and the cost per unit of flow over arc e, respectively. Note that d_i is positive for supply vertices and negative for demand vertices. A formulation for the BC-MCFP is as follows:

$$\min \sum_{e\in\mathcal{E}} c_e x_e \tag{1}$$

s.t.

$$\sum_{e\in\delta^+(i)} x_e - \sum_{e\in\delta^-(i)} x_e = d_i \qquad \forall i \in \mathcal{V} \tag{2}$$

$$x_e \leqslant M y_e \qquad \forall e \in \mathcal{E} \tag{3}$$

$$\sum_{e\in\mathcal{E}} p_e y_e \leqslant B \tag{4}$$

$$\sum_{e:e=(i,j)} y_e \leqslant 1 \qquad \forall i,j \in \mathcal{V} : (i,j) \in \mathcal{E} \tag{5}$$

$$0 \leqslant x_e \leqslant a_e \qquad \forall e \in \mathcal{E} \tag{6}$$

$$y_e \in \{0,1\} \qquad \forall e \in \mathcal{E} \tag{7}$$

The objective function (1) minimises the total flow cost. The constraints in (2) ensure that the demand for each sink vertex j is satisfied and that the supply of each source i is not exceeded. The constraints in (3), where M denotes a large number, enforce that flow can only pass through arcs that have been selected for use. Constraint (4) imposes an upper bound B on the total upgrading cost. The constraints in (5) ensure that at most one arc connecting each pair of vertices is chosen. Note that these constraints are not needed when the arcs are uncapacitated. Finally, constraints (6) and (7) define the type and the bounds for the decision variables. In constraints (6), a_e represents the capacity of arc e.

We now show that the AAUP is a special case of the BC-MCFP. Consider an instance of the AAUP as described in Section 1. Each regular vertex acts as a sink, while each centre is a supply vertex. The value of d_j for a regular vertex j is set to $-w_j$, while the value of d_i for each centre i is set to the total demand on the network (i.e., the sum of the w_j values for all j in $\mathcal{V}_2$). Then, we create one dummy demand vertex connected to each of the centres. The fixed cost and cost per unit of flow for the arcs connecting the dummy vertex and the centres are set to 0, while the d_i value of the dummy vertex is set to $-(|\mathcal{V}_1|-1)\sum_{j\in\mathcal{V}_2} w_j$. Solving the resulting instance of the BC-MCFP yields a solution for the corresponding instance of the AAUP. Fig. 1 shows the transformation of an AAUP into a BC-MCFP, schematically.

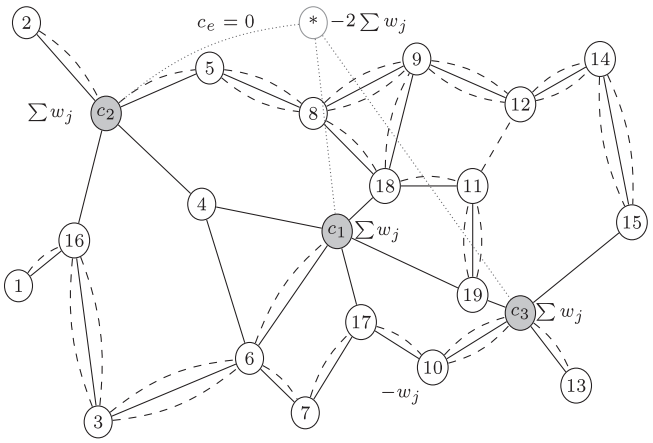

Fig. 1. Transformation of the AAUP into a BC-MCFP.

In Fig. 1, the vertices c_1, c_2 and c_3 are the centres, while the vertices 1–19 represent the regular vertices. The solid lines correspond to existing arcs of the network, while the dashed lines are possible upgrading levels of the existing arcs. The vertex labelled with an asterisk represents the dummy demand vertex and the grey dotted lines are the arcs that connect the dummy vertex to the centres.

3. Literature review

In this section, we review the literature that is relevant for the AAUP. We first concentrate on the network upgrading problem. Afterwards, we extend the review to consider the accessibility factor within the upgrading network problem.

Although several authors have addressed network upgrading problems, the literature is not as extensive as it is for other problems within the domain of network design. Krumke et al. (1998) distinguish two kinds of upgrading problems depending on whether the focus is on upgrading the arcs or upgrading the vertices. The authors propose a bi-objective approach for both types of problems. In that approach, a sub-class of graph $\mathcal{S}$ is considered (e.g., the set of spanning trees) and a budget or target value is defined for the first objective. The goal is to find a network within the fixed budget that belongs to $\mathcal{S}$ and minimises the second objective. Results on the complexity of a number of node-based and edge-based upgrading problems are presented. In particular, the case in which the objectives are defined as minimising the cost of improving the network and minimising the total length of the minimum spanning tree is shown to be NP-hard for trees and general networks. Drangmeister et al. (1998) study a related problem that looks for an optimal reduction strategy (i.e., shortening some of the edges) such that a budget constraint is satisfied and the total length of a minimum spanning tree in the modified network is minimised. Some NP-hardness results, even for simple classes of graphs, are presented, as well as some approximation algorithms.

Campbell and Lowe (2006) address two q-upgrading arc problems that involve finding the best q arcs to upgrade in a network. The q-upgrading arc diameter problem requires finding q arcs to upgrade such that the travel time on the maximum shortest path between any origin–destination pair (i.e., the diameter of the network) is minimised. The q-upgrading arc radius problem requires finding q arcs to upgrade and locating the vertex centre, i.e., the node for which the maximum shortest path to the other nodes in the network (i.e., the radius of the network) is minimised. The two problems are shown to be NP-hard on general graphs, but polynomially solvable on trees. A variant of the problems, which involves a budget constraint, is also studied. It is shown that these

problems are NP-hard for general graphs and even for a path graph. Three heuristic algorithms are proposed to deal with these kinds of problems.

Accessibility maximisation has been considered mainly in the domain of road network planning. Antunes et al. (2003) consider a inter urban road network long-term planning problem. They propose a non-linear combinatorial optimisation model that does not involve an evolution of the network over time but defines the final status of the network at the end of the planning horizon. Two different heuristic approaches to solve the model are described, based on local search and simulated annealing principles. Santos et al. (2010) extend the study to consider accessibility and robustness objectives simultaneously. A model to help road authorities in their strategic, system-wide decisions regarding the long-term evolution of an interurban road network is proposed. Scaparra and Church (2005) also tackle a road network planning problem, but focus on the rural case for developing countries. The authors propose a GRASP (Greedy Randomized Adaptive Search Procedure) and path relinking heuristic, and consider a bi-objective model, which minimises the sum of the weighted shortest paths between all pairs of nodes and maximises the traffic flow. The maximal covering network improvement problem is studied by Murawski and Church (2009) with the objective of improving accessibility to rural health service. The problem is formulated as an integer linear programming problem, and is applied to a real case in the Suhum District of Ghana. The convenience of using a metaheuristic approach for larger instances is pointed out by the authors.

4. Analytical analysis

We study the AAUP for two specific network topologies and present some basic results on the complexity of this problem. First, in Section 4.1, we consider the AAUP for the case in which the network topology is a star. Second, in Section 4.2, we study the case in which the network is a tree. Throughout this analysis, we denote by AAUP/p/q/$\mathcal{S}$ the AAUP problem that considers at most p upgrading levels for each arc, exactly q centres, and a network topology as indicated by $\mathcal{S}$. Additionally, we define $s^l(e)$ as the reduction in the cost per unit of flow when the arc e is upgraded to level l, compared to the cost when the arc is at its original level.

4.1. AAUP on star network topologies

For the case in which the network topology is a star, it is possible to show that:

Theorem 4.1. *The AAUP/p/q/star is NP-hard.*

In order to prove Theorem 4.1, consider first the case in which there is only one centre c, one possible upgrading level for each arc, and the network has a star topology centred in c. We denote this problem as AAUP/1/1/star. Define the binary variable x_e to indicate whether arc e is selected for upgrading ($x_e = 1$) or not ($x_e = 0$). Additionally, for each arc e, we define a benefit b_e and a cost p_e. The benefit b_e is defined as the product of $s^1(e)$ and the weight w_j of the vertex j that is connected to the centre by means of arc e. The cost p_e corresponds to the cost of upgrading arc e. Then, the AAUP can be reformulated as the following knapsack problem:

$$\max \sum_{e\in\mathcal{E}} b_e x_e \tag{8}$$

$$\sum_{e\in\mathcal{E}} p_e x_e \leqslant B \tag{9}$$

$$x_e \in \{0,1\} \qquad \forall e \in \mathcal{E} \tag{10}$$

Thus, the AAUP/1/1/star can be transformed into a knapsack problem and each instance of the knapsack problem can be transformed into an instance of the AAUP/1/1/star. Therefore, the two problems are polynomially equivalent, and, in particular, the AAUP/1/1/star is at least as hard as the knapsack problem. In other words, the AAUP/1/1/star is NP-hard.

For the more general case in which q centres are considered and each arc might have more than one upgrading level, AAUP/p/q/star, the star topology looks slightly different. While each arc still connects a centre and a regular vertex, a regular vertex may now be adjacent to multiple centres. An example of this case is presented in Fig. 2, where, again, c_1, c_2 and c_3 are centres, the vertices 1–13 represent the regular vertices, the solid lines correspond to existing arcs of the network, and the dashed lines are possible upgrading levels of the existing arcs.

By following an analysis similar to the one we did for the AAUP/1/1/star, the AAUP/p/q/star can be formulated as the multi-choice knapsack problem (MCKP) presented in (11)–(14). In this formulation, $\mathcal{E}(j)$ is the set of arcs adjacent to vertex j, $\mathcal{L}_e$ is the set of possible upgrading levels for arc e, x_{el} is a binary variable that indicates whether arc e is improved from its current level to level l, and p_{el} and b_{el} are the corresponding fixed cost and benefit, respectively. The MCKP has been shown to be NP-hard (Kellerer et al., 2005):

$$\max \sum_{e\in\mathcal{E}}\sum_{l\in\mathcal{L}_e} b_{el} x_{el} \tag{11}$$

$$\sum_{e\in\mathcal{E}}\sum_{l\in\mathcal{L}_e} p_{el} x_{el} \leqslant B \tag{12}$$

$$\sum_{e\in\mathcal{E}(j)}\sum_{l\in\mathcal{L}_e} x_{el} \leqslant 1 \qquad \forall j \in \mathcal{V}_2 \tag{13}$$

$$x_{el} \in \{0,1\} \qquad \forall e \in \mathcal{E}, \forall l \in \mathcal{L}_e \tag{14}$$

4.2. AAUP on tree network topologies

We study now the case in which the network is a tree. It is possible to show that:

Theorem 4.2. *The AAUP/p/q/tree is NP-hard.*

To prove Theorem 4.2, we initially consider the problem with only one centre and one possible upgrading level for each arc, AAUP/1/1/tree. This problem can be polynomially reduced to the case in which the network is a star. Therefore, the AAUP/1/1/tree problem is as hard to solve as the knapsack problem. As a result, it is also NP-hard.

In order to reduce an AAUP/1/1/tree to the problem on a star, we consider, without loss of generality, the tree rooted at the

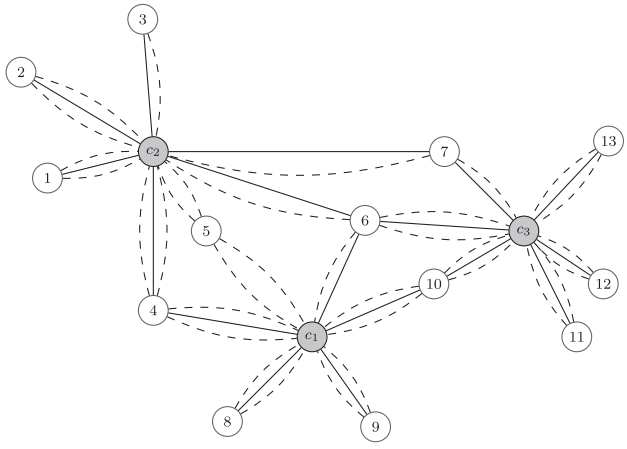

Fig. 2. Example of a graph induced by an AAUP with a star topology.

vertex corresponding to the centre. A cumulative weight $\hat{w}_j$ is defined for each regular vertex j. To compute this cumulative weight for vertex j, we identify the path connecting it to the root. This cumulative weight is the result of summing w_j and the weights w_k of all the vertices located in the same branch as vertex j but located further from the root. Additionally, for each arc e, there is an upgrading cost p_e, and a benefit $b_e = s^1(e) \cdot \hat{w}_j$, where j is the arc's vertex furthest from the root. The problem can then be formulated as a knapsack problem using the model in (8)–(10).

Consider now the case in which there are exactly two centres, c_1 and c_2. Under this assumption, the set of vertices can be divided into three groups. The first group is formed by c_1 and all vertices for which the path to c_2 passes through c_1. Similarly, the second group consist of c_2 and the vertices whose path to c_1 passes through c_2. The vertices for which the path to c_1 does not pass by c_2 and the path to c_2 does not pass through c_1 form the third group. An example of the AAUP/1/2/tree case is presented in Fig. 3. In this picture, the vertices c_1 and c_2 are the centres and vertices 1 to 9 represent regular vertices, while the numbers next to the arcs define the cost of using the arc at the given status. For this particular example, the set of vertices is partitioned into three groups. The first group is formed by vertices $\{c_1,1,2,3\}$, while the second group is $\{c_2,7,8,9\}$, and vertices $\{4,5,6\}$ form the third group.

For any solution, regardless of which arcs are upgraded, c_1 will be the closest centre for the vertices in the first group and c_2 will be the closest centre for the vertices in the second group. For the vertices in the third group, which centre is the closest may depend on the upgrading decisions. However, note that this third group can always be partitioned into two different sets of connected vertices. One of those sets will have c_1 as its closest centre, while c_2 will be the closest centre for the other set. The *frontier* that separates those two sets is always one of the arcs that form the shortest path between c_1 and c_2. For example, in Fig. 3, vertices $\{4,5\}$ have c_1 as their closest centre, while c_2 is the closest centre for vertex $\{6\}$. The arc $(5,6)$, depicted using a dotted line, is the frontier between those two sets.

When an arc r in the shortest path between the two centres is removed (e.g., arc $(5,6)$), the original graph is decomposed into two trees $\mathcal{T}_a = \{\mathcal{V}_a, \mathcal{E}_a\}$ (involving centre c_1) and $\mathcal{T}_b = \{\mathcal{V}_b, \mathcal{E}_b\}$ (involving centre c_2). As we previously showed, each of these trees can be transformed into a star. Therefore, finding the optimal upgrading strategy for a tree involving two centres requires the solution of the problem on two stars, which can be integrated into a single MCKP. By the definition of a tree, there can be at most $n-1$ arcs in the shortest path between the two centres. As a result, there are at most $n-1$ options for choosing the arc r to be removed. Thus, an optimal solution for the AAUP/1/2/tree can be found by solving $\mathcal{O}(n)$ (multi-choice) knapsack problems and choosing the best out of at most $n-1$ solutions.

Using a similar analysis, it is possible to show that, for the general case, in which there are q centres and each arc might have more than one upgrading level, an optimal solution for the AAUP on trees can be found by solving at most $\mathcal{O}(n^{q-1})$ MCKPs and choosing the best solution among them. By removing $q-1$ arcs, one in each shortest path that connects two consecutive centres, the tree can be decomposed into q independent sub-trees. The arc upgrading problem on each of those sub-trees can be reformulated as a MCKP by considering an approach similar to the one used for the AAUP/1/1/tree. As there are at most $n-(q-1)$ arcs connecting two centres, there are at most $(n-(q-1))^{q-1}$ possibilities for removing the $q-1$ arcs, that is, for defining the set of sub-trees.

5. Solution methods

In Section 2, we have described a formulation that can be used to solve instances of the AAUP using a dedicated MIP solver. In this section, we propose two alternative approaches. First, we describe a heuristic approach specifically designed for the AAUP that is based on the analytical analysis we have done in the previous section. Second, we outline a variable neighbourhood search approach (Hansen and Mladenovic, 2005) for the general BC-MCFP, and apply it to the AAUP, which, as we explained in Section 2, is a special case of the BC-MCFP.

5.1. A knapsack problem-based heuristic for the AAUP

We have shown that, for the cases in which the network is a star or a tree, the AAUP can be tackled by transforming it into either a knapsack problem or a multi-choice knapsack problem. Based on these basic cases, we outline an approach for the AAUP on general networks. In order to do that, note that, for any feasible solution of the AAUP, we can assign each vertex to its nearest centre. The paths linking each centre to the vertices assigned to it define a forest composed of $|\mathcal{V}_1|$ trees. Therefore, an (exponential time) exact algorithm for the AAUP enumerates all the forest sub-graphs that can be defined from the original graph and in which each tree of the forest is rooted in a different centre. Then, the problem on each of these forests is solved by the approach we describe in Section 4.2. The best among all those solutions corresponds to the optimal solution for the AAUP.

Clearly, from a practical point of view, this approach is not feasible for realistically sized instances. However, by using some key ideas from that exact algorithm, we have built a heuristic for the AAUP, which we refer to as the knapsack problem-based heuristic (KPBH). In that heuristic, we first create an initial solution. This solution specifies a particular status of the network in which each arc is fixed at a given upgrading level. Based on that particular network, each vertex is assigned to its closest centre by using the minimum cost flow formulation of the AAUP. The paths linking each regular vertex to its closest centre form a forest involving only a subset of the arcs of the complete graph. Then, the AAUP is solved on that forest. We call this sub-problem the restricted AAUP, and solve it by transforming it into a multi-choice knapsack problem (see Section 4). The resulting solution is a

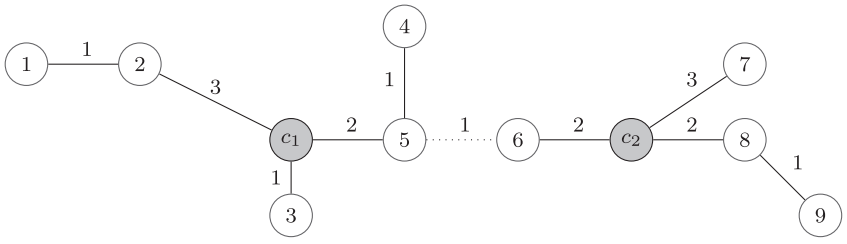

Fig. 3. Example of an AAUP/1/2/tree problem.

feasible solution for the original AAUP. In that solution, the arcs that are not part of the forest are set to their original level, while the upgrading level of the other arcs is dictated by the solution of the restricted AAUP. Based on this new solution, a new assignment of the regular vertices to the centres is made and the entire procedure is iterated. The algorithm stops when a local optimum is reached, i.e., when there is no improvement after iterating over a given feasible solution. Algorithm 1 presents a schematic overview of the KPBH.

Algorithm 1. KPBH for the AAUP

Initialize: Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2\}$
Construct an initial solution x for the AAUP
repeat
Define a forest $\mathcal{F}$ by solving the MCFP over x.
Solve the restricted AAUP on the sub-graph induced by $\mathcal{F}$.
Update x based on the solution of the restricted AAUP
until x is a local optimum

The solution obtained by the KPBH depends on the initial solution x. We consider three different initial solutions: (i) the present solution in which the level of each arc is set to its original (lowest) status; (ii) the ideal solution in which the arcs are assigned the best possible level; and (iii) a random feasible solution in which the level of each arc is set randomly. The algorithm is run once for the first two initial solutions while a given number of replications are performed using different initial random solutions.

5.2. Variable neighbourhood search for the AAUP

In this section, we describe a metaheuristic approach we developed for the BC-MCFP and used to solve instances of the AAUP. The approach is a variable neighbourhood search (VNS) with strategic oscillation that considers a feasible upgrading strategy (i.e., a particular status of the network in which each arc is fixed at a given upgrading level) at each iteration. Given an upgrading strategy, the accessibility value in the objective function of the AAUP can be obtained by solving a standard minimum cost flow problem. Two basic moves, upgrade and downgrade, are used to define two neighbourhoods. The first neighbourhood ($\mathcal{N}_1$) contains all solutions that can be reached from the current solution by upgrading the level of an arc. The second neighbourhood ($\mathcal{N}_2$) includes all the solutions that can be obtained from the current solution by considering upgrading and downgrading simultaneously. Whenever the search reaches a local optimum, a strategic oscillation is applied. This oscillation allows the search to temporarily consider solutions that exceed the budget limit (i.e., infeasible solutions) and restarts the VNS once the feasibility has been restored. Additionally, a shaking phase is applied when the strategic oscillation fails to move the search to a better solution. Algorithm 2 presents an outline of the VNS approach.

Algorithm 2. VNS with strategic oscillation

Initialization: Set the current network status as the initial solution x.
Consider the set of neighbourhood structures $\mathcal{N}_k(x),\ k = 1, 2$.
Set a stopping criterion.
while Stopping criterion has not been met **do**
repeat
Set $k \leftarrow 1$
while $k <= 2$ **do**
Explore neighbourhood $\mathcal{N}_k(x)$.
if a better solution $x' \in \mathcal{N}_k(x)$ is found
Set $\mathbf{x} \leftarrow \mathbf{x}'$
Set $k \leftarrow 1$
else
set $k \leftarrow k + 1$
end if
end while
Apply shake.
until x is a local optimum
Apply shake.
end while

5.2.1. Upgrading move

The upgrading neighbourhood $\mathcal{N}_1$ is formed by all the solutions that can be reached from the current solution by upgrading an arc by one level without violating any constraint. In order to estimate the cost-saving potential of an arc whose upgrading is feasible, we use the solution of the MCFP associated with the current solution to distinguish between basic and non-basic arcs. Basic arcs are those for which the associated decision variable is basic when solving the linear problem, while non-basic arcs are related to the non-basic decision variables. For the basic arcs, the saving is estimated as the product of the reduction in cost per unit of flow obtained by upgrading the arc and the flow that passes the arc in the current solution. The saving for a non-basic arc is estimated by its reduced cost. The arc to be upgraded is selected randomly among the α arcs that provide the best saving estimates.

After upgrading an arc, a procedure to remove redundant upgraded arcs is run. That procedure downgrades all the arcs that have been upgraded in previous iterations of the VNS algorithm but do not carry any flow in the current solution to their lowest level. We use the solution of the MCFP associated with the current upgrading strategy to identify the arcs that must be downgraded. The solution obtained by downgrading those arcs will have the same objective value but it will use less resources.

5.2.2. Combined upgrading-downgrading move

The neighbourhood $\mathcal{N}_2$ contains all feasible solutions that can be obtained from the current solution by downgrading one arc and upgrading at least one other arc. This neighbourhood requires a list of upgrading candidates and a list of downgrading candidates. The former list is formed by all the arcs that are basic in the current solution and can be upgraded. A saving cost is estimated for each of those arcs, in the way described in Section 5.2.1 for the upgrading move. That saving is used to rank the upgrading candidates. The list of downgrading candidates contains all the arcs that can be downgraded, i.e., the arcs for which the level in the current solution is not the lowest. This list is arranged in decreasing order of the ratio of the fixed cost for using the arc at its current level to the flow through the arc in the current solution. Only the β best candidates in this list are kept.

For every downgrading candidate, the upgrading list is traversed in decreasing order of saving potential searching for a feasible upgrading candidate. Whenever a feasible candidate is found, the remainder of the list is traversed in order to upgrade as many candidate arcs as possible. As a result, whenever a feasible move is identified, it involves one arc to be downgraded and either one arc or a set of arcs to be upgraded. Each of the feasible moves is evaluated by solving a minimum cost flow problem. In order to limit the computational effort, a first improvement approach is used, i.e., the exploration of the $\mathcal{N}_2$ neighbourhood is stopped as

soon as a feasible move is found that improves the current solution.

5.2.3. Strategic oscillation and shaking

The strategic oscillation component aims to help the search escape from local optima and explore different parts of the solution space. This is done by allowing the search to temporarily consider infeasible solutions by relaxing the budget constraint. Based on the solution of the MCFP associated with the current solution, we select arcs to be upgraded ignoring the budget constraints. To that end, the arcs are partitioned in two different lists groups, one containing the basic arcs and one containing the remaining arcs. A saving is estimated for each arc in the same way as in Section 5.2.1, and the arcs are ranked in decreasing order of the savings. Then, the best candidate in each list is upgraded. This is repeated as long as the total upgrading cost does not exceed γ times the original budget, where γ is a user-specified tuning parameter.

Next, the MCFP corresponding to the new upgrading strategy is solved and its solution is used to run a procedure to restore feasibility. That procedure calculates for all arcs the ratio of the fixed cost for using them at their current level to the flow that passes through them in the current solution. The arcs with the largest ratio are downgraded to their lowest level until the budget feasibility is restored. The VNS is then invoked again, starting from the resulting feasible solution. This procedure is iterated until it fails to generate a better solution. In that case, a shaking phase is applied to restart the search from a significantly different solution.

The shaking phase downgrades all the arcs that carry flow in the current solution to the lowest level, while, for arcs that do not carry flow, the level is set randomly. After ensuring that this new solution is feasible, the search is restarted, giving rise to a new iteration of the VNS. The number of iterations of the shaking phase is used as a stopping criterion for the VNS algorithm.

6. Computational results

In this section, we evaluate the performance of the two heuristic solution approaches using a set of 480 randomly generated AAUP instances which are available from http://antor.ua.ac.be/downloads. The mathematical model and the two algorithms for solving the AAUP were implemented in Java and ILOG CPLEX Concert Technology (IBM ILOG CPLEX Optimisation Studio Academic Research Edition V12.2).

6.1. Instance generation

To generate the 480 AAUP instances, we first created a set of 30 random instances for the minimum cost flow problem using GNETGEN, which is a modification of the widely used NETGEN generator proposed by Klingman et al. (1974). Table 1 shows the main parameters that were used for the generator. The number of sources (centres), transhipments and sinks (regular vertices) are expressed as percentages of the total number of vertices n. The number of arcs is defined as a percentage of the number of arcs in a complete graph. Finally, the total supply was set to 100 times the number of vertices.

Table 1
Parameters used for generating random MCFP instances.

Parameter	Values
Number of vertices (n)	100, 200, 400, 500, 1000
Percentage of sources (%)	2, 5
Percentage of transshipments (%)	30
Percentage of arcs (%)	3, 5, 15
Total supply/demand	$100n$
Minimum cost for arcs	1
Maximum cost for arcs	100

These minimum flow cost networks form the starting point to generate the AAUP instances. In those instances, $\mathcal{V}_1$ corresponds to the sources, the set $\mathcal{V}_2$ contains all sinks, and, for each vertex j in $\mathcal{V}_2$, the demand represents the weight w_j. For each arc of the original network, we generated a set of upgrading options or copies. In order to make the set of instances as diverse as possible, we use two different procedures to generate these copies: the number of copies for each arc is either a fixed number m or a random number between 1 and m. We used two values for m, namely 2 and 3. Each copy of an arc has a cost per unit of flow and a fixed cost for using the arc. The flow cost decreases with each extra copy of an arc, while the fixed cost increases. For each arc, the cost per unit of flow at the lowest level (copy 0 of the arc) is the cost generated by GNETGEN, while the fixed cost for using that arc is 0. We name these costs c_e^0 and p_e^0, respectively. For the rth copy of an arc e, the flow cost c_e^r and fixed cost p_e^r were generated using

$$c_e^r = c_e^{r-1}/2 + U(0, c_e^{r-1}/2) \tag{15}$$

and

$$p_e^r = p_e^{r-1} + U(0, p_e^0) \tag{16}$$

where $U(a,b)$ represents a continuous uniformly distributed random variable on the interval $[a,b]$.

Finally, four different budget values are considered for each AAUP network. To compute these budget values, we first solved the minimum cost flow problem over the original network, i.e., the network in which all arcs are set at their lowest level. Then, the total budget was defined as the sum of the cost of upgrading all the arcs that carry flow in the optimal solution to their best possible level. The four different budget values correspond to four different percentages of this total budget. In total, 480 instances were generated as a result of combining the 30 random minimum cost flow networks and the parameter values given in Table 2.

We used CPLEX to solve the 480 AAUP instances using the mathematical model described in Section 2 with a time limit of one hour. Table 3 shows the number of times the optimal solution was found within the time limit for the 24 instances at each combination of number of vertices and budget. The results show that it becomes harder to solve the instances to optimality when their size, as measured by the number of vertices, increases. The hardest type of problem is one involving a larger number of vertices and a small budget. For the instances for which the optimum was not found, the average gap to optimality was 2.3%, with a maximum of 30.1%.

6.2. Results for the KPBH

We first study the KPBH for the AAUP, as described in Section 5.1. As pointed out, three different kinds of initial solutions were considered: (i) one based on the present status of the network, (ii) one based on the ideal status of the network, and (iii) one based on random initial solutions. For the latter option, the algorithm was run for 100 different random initial solutions and the best result was kept. Table 4 presents the average percentage difference between the best solutions obtained with the KPBH

Table 2
Parameters used for generating the AAUP instances.

Parameter	Values
Copy procedure	Fixed, random
Number of copies (m)	2, 3
Budget values (%)	20, 50, 70, 100

Table 3
Number of instances (out of 24) solved to optimality within one hour using the mathematical model in section 2.

Budget (%)	Number of vertices				
	100	200	400	500	1000
20	23	19	11	6	2
50	23	20	13	12	4
70	21	16	12	11	4
100	24	23	15	12	12

Table 4
Average % gap for the feasible solutions obtained with the KPBH.

Budget (%)	Number of vertices				
	100	200	400	500	1000
20	0.30	0.77	2.46	2.81	4.90
50	0.58	1.27	2.18	2.41	3.35
70	0.53	1.00	1.35	1.50	1.74
100	0.13	0.21	0.32	0.41	0.53

(considering the three different options for the initial solution) and the optimal value or best lower bound reported by CPLEX. That difference is less than 5% for each combination of number of vertices and budget, and the maximum values are observed for instances with a large number of vertices. When only the instances that were solved to optimality are considered, the average gap is 0.39% and the maximum gap is 3.33%. Furthermore, for 12 instances with 1000 vertices, the KPBH generates a better solution than CPLEX. The average computing times (expressed in seconds) for the heuristic using the three options for the initial solution are shown in Table 5. The running time increases with the size of the instances and is shortest for the instances with a budget level of 100%. Additionally, the computing time required by the heuristic is, on average, only 4.3% of that required by CPLEX.

The results reveal an excellent performance of the KPBH in terms of finding good feasible solutions for the AAUP in acceptable running times. For about 67% of the instances, the heuristic generates the best results when the initial solution is created based on the ideal status of the network, while, for almost 30% of the instances, the best solution is found when the algorithm starts from 100 random solutions.

6.3. Results for the VNS heuristic

We used a designed experiment both to study the usefulness of the different components of the VNS algorithm and to tune its parameters α, β and γ. According to the results, the two neighbourhoods, the strategic oscillation, and the shaking phase contribute significantly to the performance of the VNS algorithm. It also shows that the parameter settings influence the computing time rather than the solution quality.

Based on the experimental results, we set $\alpha = 4$, $\beta = 3$, and $\gamma = 1$. Finally, we set the stopping criterion for the VNS to 100. In other words, we perform the shaking procedure 100 times.

Table 5
Average computing time, expressed in seconds, for the KPBH.

Budget (%)	Number of vertices				
	100	200	400	500	1000
20	27.51	32.69	80.31	87.94	102.01
50	38.79	48.50	100.59	103.55	117.67
70	39.85	44.35	84.35	85.53	108.23
100	27.61	34.29	60.72	64.04	95.50

Table 6
Average % gap for the best feasible solutions obtained with the VNS heuristic.

Budget (%)	Number of vertices				
	100	200	400	500	1000
20	0.80	1.46	2.85	2.62	5.74
50	0.84	1.66	2.78	2.42	4.65
70	0.77	1.41	2.03	1.88	2.98
100	0.09	0.26	0.45	0.43	0.77

Table 7
Average computing time, expressed in seconds for the VNS heuristic.

Budget (%)	Number of vertices				
	100	200	400	500	1000
20	1.47	8.82	74.58	127.45	954.99
50	2.06	11.38	95.11	166.40	1178.07
70	2.41	13.51	104.32	180.84	1283.43
100	2.90	17.97	142.06	241.99	1497.67

Table 8
Average % difference between the best solutions generated with the VNS approach and the KPBH.

Budget (%)	Number of vertices				
	100	200	400	500	1000
20	0.49	0.68	0.37	0.37	0.77
50	0.27	0.39	0.58	0.60	1.22
70	0.24	0.40	0.67	0.68	1.21
100	−0.04	0.05	0.13	0.02	0.24

Table 6 shows, for each combination of budget and number of vertices, the average gap between the best solution found by the VNS algorithm and the optimal solution or best lower bound provided by CPLEX. That gap is largest for the instances with budget levels 20% and 50% and increases with the number of vertices. When only the instances solved to optimality by CPLEX are taken into account, the average percentage gap is 0.60%, with a maximum of 3.60%. Furthermore, for eight of the instances with 1000 vertices, the solution generated by the VNS heuristic is better than the solution found by CPLEX. The average computing times are presented in Table 7. The computing times increase both with the number of vertices and the budget level, but the impact of the number of vertices is substantially larger than the impact of the budget level.

6.4. Comparison of the two heuristics

In order to compare the two different heuristic approaches, we present in Table 8, for each combination of budget level and number of vertices, the average percentage difference between the best solution found by the VNS algorithm and the best solution generated by the KPBH. On average, the KPBH generates better solutions, particularly for the instances with a tight budget and a large number of vertices. Additionally, while, for the small-sized instances, the computing times for the KPBH are on average greater than those required by the VNS, for the instances with 1000 vertices, the computing time of the former is on average only 16.59% of the computing time required by the latter.

7. Potential practical impact

After having verified the excellent performance of the two heuristic approaches to tackle the AAUP, we would like to highlight their potential for solving real-life decision problems. A matter of

Table 9
Percentage of accessibility improvement achieved.

Budget (%)	Number of vertices				
	100	200	400	500	1000
20	69.52	70.66	71.66	72.58	70.80
50	90.01	90.29	90.41	90.89	90.39
70	95.85	95.57	95.68	95.88	95.91
100	99.29	98.99	98.97	99.01	99.00

fact, large improvements in accessibility can be obtained by allocating scarce resources properly. To show how the algorithms we have discussed contribute to this end, we measured for each experiment the improvement in accessibility. First, we determine the maximum possible improvement as the difference between the measure of accessibility for the network at its present status and the measure of accessibility for the best possible network, in which all arcs are upgraded to their best level while ignoring the budget constraint. That estimate represents the target improvement for each instance. Once the target had been fixed, we calculated the percentage improvement that is achieved for each given budget level. Table 9 presents the average percentage improvement in accessibility for each combination of budget level and number of vertices. From this table, it can be observed that, on average, approximately 70% and 90% of the target improvement is obtained when the budget level is set to 20% and 50% of the total budget, respectively. Note that the target improvement is not achieved with 100% the total budget. This is due to the fact that, when generating the instances, the budget was computed as the sum of the cost of upgrading the arcs that carry flow in the solution based on the current network status, while the optimal solution might use arcs that are not part of this current solution.

8. Conclusions

We have defined the accessibility arc upgrading problem (AAUP), a network upgrading problem for which real-life applications can be found in several domains such as transportation, telecommunications and logistics. The problem was formulated as an special case of the budget constrained minimum cost flow problem (BC-MCFP).

An analytical study of some special cases of the AAUP provided some theoretical results on the complexity of the problem. Furthermore, based on the insights obtained from the analytical study, we generated a solution approach (which we called the knapsack problem-based heuristic or KPBH) that performs excellently in terms of efficiency and solution quality. Although this approach might lack the flexibility to deal with additional constraints, it could easily act serve as a building block for an algorithm designed for more complicated problems.

We also proposed a second solution approach for the AAUP, namely a variable neighbourhood search (VNS) with strategic oscillation. The VNS algorithm exploits the underlying network flow structure of the problem when defining and evaluating the neighbourhoods. At each iteration, the VNS algorithm considers a feasible upgrading strategy that can be evaluated by solving a minimum cost flow problem. Therefore, by properly defining a function that evaluates the feasibility of a solution, this approach can be easily extended to situations in which additional constraints have to be considered. The experiments illustrate the good performance of our VNS algorithm.

Finally, we pointed out the potential of this study for real-life decision processes. It was shown how the algorithms we have proposed lead to large improvements in accessibility by allocating scarce resources properly.

References

Antunes, A., Seco, A., Pinto, N., 2003. An accessibility-maximization approach to road network planning. Computer-Aided Civil and Infrastructure Engineering 18, 224–240.
Campbell, A., Lowe, T., 2006. Upgrading arcs to minimize the maximum travel time in a network. Networks 47, 72–80.
Donnges, C. 2003. Improving Access in Rural Areas, Technical report, International Labour Office, Bangkok.
Drangmeister, K., Krumke, S., Marathe, M., Noltemeier, H., Ravi, S., 1998. Modifying edges of a network to obtain short subgraphs. Theoretical Computer Science 203, 91–121.
Hansen, P., Mladenovic, N., 2005. Variable neighborhood search. In: Search Methodologies, Introductory Tutorials in Optimization and Decision Support Techniques. Springer, pp. 211–238.
Kellerer, H., Pferschy, U., Pisinger, D., 2005. Knapsack Problems. Springer-Verlag.
Klingman, D., Napier, A., Stutz, J., 1974. NETGEN: a program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. Management Science 20, 814–821.
Krumke, S., Marathe, M., Noltemeier, H., Ravi, R., Ravi, S., 1998. Approximation algorithms for certain network improvement problems. Journal of Combinatorial Optimization 2, 257–288.
Maya Duque, P., Sörensen, K., 2011. A GRASP metaheuristic to improve accessibility after a disaster. OR Spectrum 33, 525–542.
Murawski, L., Church, R., 2009. Improving accessibility to rural health services: the maximal covering network improvement problem. Socio-Economic Planning Sciences 43, 102–110.
Santos, B., Antunes, A., Miller, E., 2010. Interurban road network planning model with accessibility and robustness objectives. Transportation Planning and Technology 33, 297–313.
Scaparra, M., Church, R., 2005. A GRASP and path relinking heuristic for rural road network development. Journal of Heuristics 11, 89–108.