Les Cahiers du GERAD ISSN: 0711–2440

A tabu search for the design of capacitated rooted survivable planar networks

A. Hertz,
T. Ridremont

G–2018–64

August 2018

La collection *Les Cahiers du GERAD* est constituée des travaux de recherche menés par nos membres. La plupart de ces documents de travail a été soumis à des revues avec comité de révision. Lorsqu'un document est accepté et publié, le pdf original est retiré si c'est nécessaire et un lien vers l'article publié est ajouté.

Citation suggérée: A. Hertz, T. Ridremont (Août 2018). A tabu search for the design of capacitated rooted survivable planar networks, Rapport technique, Les Cahiers du GERAD G–2018–64, GERAD, HEC Montréal, Canada.

Avant de citer ce rapport technique, veuillez visiter notre site Web (https://www.gerad.ca/fr/papers/G-2018-64) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

The series *Les Cahiers du GERAD* consists of working papers carried out by our members. Most of these pre-prints have been submitted to peer-reviewed journals. When accepted and published, if necessary, the original pdf is removed and a link to the published article is added.

Suggested citation: A. Hertz, T. Ridremont (August 2018). A tabu search for the design of capacitated rooted survivable planar networks, Technical report, Les Cahiers du GERAD G–2018–64, GERAD, HEC Montréal, Canada.

Before citing this technical report, please visit our website (https://www.gerad.ca/en/papers/G-2018-64) to update your reference data, if it has been published in a scientific journal.

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2018
– Bibliothèque et Archives Canada, 2018

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2018
– Library and Archives Canada, 2018

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

A tabu search for the design of capacitated rooted survivable planar networks

Alain Hertz [a]

Thomas Ridremont [a,b]

[a] *GERAD & Department of Mathematics and Industrial Engineering, Polytechnique Montréal (Québec) Canada*

[b] *ENSTA ParisTech & CEDRIC-CNAM, Paris, France*

alain.hertz@gerad.ca
thomas.ridremont@gerad.ca

August 2018
Les Cahiers du GERAD
G–2018–64

Copyright © 2018 GERAD, Hertz, Ridremont

Les textes publiés dans la série des rapports de recherche *Les Cahiers du GERAD* n'engagent que la responsabilité de leurs auteurs. Les auteurs conservent leur droit d'auteur et leurs droits moraux sur leurs publications et les utilisateurs s'engagent à reconnaître et respecter les exigences légales associées à ces droits. Ainsi, les utilisateurs:

- Peuvent télécharger et imprimer une copie de toute publication du portail public aux fins d'étude ou de recherche privée;
- Ne peuvent pas distribuer le matériel ou l'utiliser pour une activité à but lucratif ou pour un gain commercial;
- Peuvent distribuer gratuitement l'URL identifiant la publication.

Si vous pensez que ce document enfreint le droit d'auteur, contactez-nous en fournissant des détails. Nous supprimerons immédiatement l'accès au travail et enquêterons sur votre demande.

The authors are exclusively responsible for the content of their research papers published in the series *Les Cahiers du GERAD*. Copyright and moral rights for the publications are retained by the authors and the users must commit themselves to recognize and abide the legal requirements associated with these rights. Thus, users:

- May download and print one copy of any publication from the public portal for the purpose of private study or research;
- May not further distribute the material or use it for any profit-making activity or commercial gain;
- May freely distribute the URL identifying the publication.

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Abstract: Given a directed graph $G = (V, A)$, capacity and cost functions on A, a root r, a subset $T \subset V$ of terminals, and an integer k, we aim at selecting a minimum cost subset $A' \subseteq A$ such that for every subgraph of $G' = (V, A')$ obtained by removing k arcs from A' it is possible to route one unit of flow from r to each terminal while respecting the capacity constraints. We focus on the case where the input graph G is planar and propose a tabu search algorithm whose main procedure takes advantage of planar graph duality properties.

Keywords: Capacitated rooted survivable networks, planar graphs, tabu search

Acknowledgments: This work was done with the support of the Gaspard Monge Program for Optimization and operations research (PGMO) http://www.fondation-hadamard.fr/fr/pgmo.

1 Introduction

The design of efficient networks is crucial in many fields such as transport, telecommunications or energy. In this paper, we focus on the design of networks which are resilient to one or several breakdowns. More precisely, let $G = (V, A)$ be a directed graph with vertex set V and arc set A, and let c and u be respectively a cost and a capacity function on A. Also, consider a subset $T \subset V$ of vertices called *terminals*, and let $r \in V \setminus T$ be a *root* in G (i.e., there is a path from r to each vertex of T). We say that G has a feasible flow if it is possible to route one unit of flow from the root to each terminal while respecting the arc capacity constraints. Given an integer $k \geq 0$, a subgraph $G' = (V, A')$ of G is said *k-survivable* if every subgraph obtained from G' by removing at most k arcs of A' contains a feasible flow. We aim at selecting a minimum cost subset $A' \subseteq A$ of arcs such that $G' = (V, A')$ is k-survivable. We call this problem the Capacitated Rooted Survivable Network Problem (CRSNP).

Bentz et al [1] give complexity and approximability results for the case $k = 0$ (i.e., there is no breakdown) and the additional constraint that the selected network must be a tree. Grötschel et al [9] study a problem similar to the CRSNP, but without capacity constraints. They consider values r_{vw} for all ordered pair (v, w) of vertices, and impose that the selected network must contain a path from v to w after the removal of any subset of at most $r_{vw} - 1$ arcs.

Many authors [3, 11, 15] propose algorithms for the design of capacitated survivable networks, but they aim to assign capacities to the arcs to make the network survivable, while we assume that all capacities are fixed. Other authors [4, 5] consider hop-constraints which limit the number of arcs on a path from the root to a terminal. Also, multi-commodity survivable network design problems are studied, for example, in [6, 16]. For surveys on survivable networks, we refer the reader to [8, 12]

The main motivation for this work is the design of a survivable wind farm collection network. One of the nodes of the network is the sub-station where the energy produced by the turbines must be transported. The potential network links have already been determined, and the aim is to select a subset of these links. Each link has a cost and a capacity. By defining each turbine as a terminal, the sub-station as the root, and reversing the direction of each arc, the problem becomes a delivery problem from the root to the terminals. Since the wind turbines are all identical, we can assume that each one produces one unit of energy, and the problem is therefore to select a subset of links so that there is a feasible flow in the corresponding subgraph, even if k arcs are removed from the chosen links. More details about this application can be found in [10].

In this paper, we focus on planar graphs which are useful in practice since many underlying graphs in real-world networks are planar. In the next section, we describe a procedure that determines in polynomial time whether a given planar graph is k-survivable. In Section 3, we embed this procedure in a tabu search. Computational experiments and comparisons with an exact algorithm [2] are reported in Section 4.

2 Testing survivability

Given a subset of A' of the original arc set A , we show in this section how to test whether $G' = (V, A')$ is k-survivable. Every arc $a = (i, j) \in A'$ has an associated upper bound $u_a = u_{ij}$ (called capacity) on the flow that can traverse it, while the lower bound $l_a = l_{ij}$ is equal to zero since there is no imposed flow on any arc of G'.

The first step of our procedure consists in determining a tree $\mathcal{T}$ rooted at r and spanning all terminals of T, which is an easy task. Note that such a tree necessarily exists since r is a root. For every vertex v in $\mathcal{T}$, let n_v be the number of terminals in the subtree rooted at v. For each arc (i, j) in $\mathcal{T}$, we add the reverse arc (j, i) in G' and set $l_{ji} = u_{ji} = n_j$. We denote by A_R the set of arcs added to G' and by $\widetilde{G'}$ the resulting graph. An example is depicted in Figure 1. The left graph G' has two terminals t_1 and t_2. The directed tree $\mathcal{T}$ with arc set $\{(r, u), (u, t_1), (u, v), (v, t_2)\}$ induces the graph $\widetilde{G'}$ on the right, the arcs in A_R being represented with bold lines. Since $n_r = n_u = 2$ while $n_v = n_{t_1} = n_{t_2} = 1$, all arcs (j, i) in A_R have $l_{ji} = u_{ji} = 1$, except the arc (u, r) for which $l_{ur} = u_{ur} = 2$.

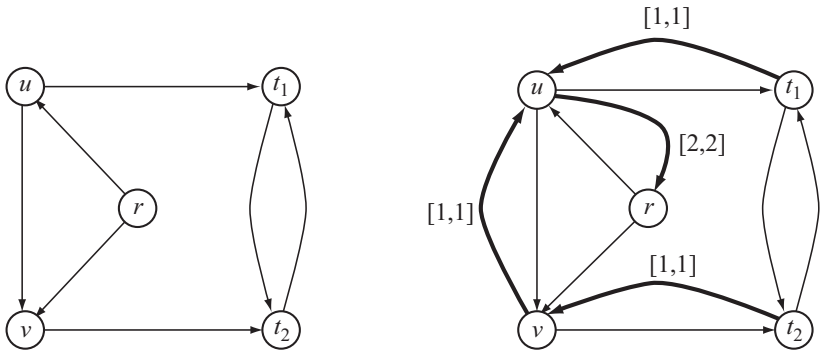

Figure 1: Construction of $\widetilde{G'}$ from G'.

The circulation problem is a generalization of the network flow problem where flow conservation is required for all vertices (i.e. there are no special vertices). It is easy to observe that the original graph G' has a feasible flow if and only if the extended graph $\widetilde{G'}$ has a feasible circulation: the flow sent from the root to the terminals travels back to r using the arcs in A_R.

The problem of determining whether G' is k-survivable was shown to be NP-complete in the general case [17], but can be solved in polynomial time if the graph $G' + t$ obtained from G' by adding a sink t to which all terminals are linked is planar [14]. Since the addition of a sink to a planar graph G' does not necessarily preserves planarity, Zenklusen [18] has shown how to solve the problem in polynomial time when G' (and not necessarily $G' + t$) is planar. His procedure is described here below.

Notice first that if G' is planar, then $\widetilde{G'}$ is too. We can therefore consider its dual. By convention (see [13] for more details), the dual of a directed planar graph is obtained by creating a vertex for each face of the original graph, and by connecting two vertices by an arc if they correspond to faces in the original graph sharing an arc. Every dual arc is oriented so that it corresponds to a 90° anticlockwise turn from the corresponding primal arc. As proposed by Zenklusen [18], we consider here an extended dual graph that contains not only the standard dual arcs, but also the arcs with the opposite direction. More precisely, we build a graph $D_{\widetilde{G'}}$ from $\widetilde{G'}$ as follows. We first create a vertex in $D_{\widetilde{G'}}$ for every face in $\widetilde{G'}$. Then, for every arc (i, j) in $\widetilde{G'}$, we consider the two faces F_1 and F_2 that (i, j) separates, where F_1 is the face below (i, j) when (i, j) is drawn horizontally with an arrow going from left to right. We create an arc (F_1, F_2) of length u_{ij} and an arc (F_2, F_1) of length $-l_{ij}$ in $D_{\widetilde{G'}}$. Note that (F_1, F_2) is the standard dual arc associated with (i, j), while (F_2, F_1) is the opposite one. We also denote a^d_{ij} the standard dual arc (F_1, F_2) in order to distinguish it from its opposite. Note that the arc (F_2, F_1) has length 0 if $(i, j) \in A'$ (since there is no imposed flow on any arc of G'), while its length is $-n_j$ if $(i, j) \in A_R$. The construction of $D_{\widetilde{G'}}$ from $\widetilde{G'}$ is illustrated in Figure 2. The general arc creation procedure appears on the left, while the dual graph associated with the graph $\widetilde{G'}$ of Figure 1 is drawn on the right with dashed lines. To simplify the drawing, arcs with arrows on both sides represent two oppositely directed arcs.

Zenklusen [18] has shown that there exists a feasible circulation in $\widetilde{G'}$ if and only if there is no negative circuit in $D_{\widetilde{G'}}$. Indeed, there is a correspondence between the circuits in $D_{\widetilde{G'}}$ and the cutsets in $\widetilde{G'}$. In a circuit of $D_{\widetilde{G'}}$, the total length of the standard dual arcs corresponds to the available capacity, while the absolute value of the total length of the other arcs corresponds to the demand. A circuit of negative total length means that the demand cannot be satisfied. For example, consider the circuit (f, d, h, g, f) in the graph $D_{\widetilde{G'}}$ of Figure 2: the standard dual arcs (f, d) and (h, g) give a total capacity $u_{u,_1} + u_{t_2t_1}$, while the other arcs (d, h) and (g, f) induce a demand of 1 unit (since $|l_{dh} + l_{gf}| = |-1 + 0| = 1$), which means that $u_{ut_1} + u_{t_2t_1}$ must be at least equal to 1 to satisfy the demand of terminal t_1.

The above construction shows how to determine if $G' = (V, A')$ has a feasible flow, which is equivalent to deciding whether $\widetilde{G'}$ has a feasible circulation. By definition, G' is k-survivable if and only if there is no subgraph G'' of G', obtained by removing at most k arcs of A', so that $\widetilde{G''}$ has no feasible circulation. Every arc removal is equivalent to paying one unit of a *budget* of k units, and we therefore try to find a

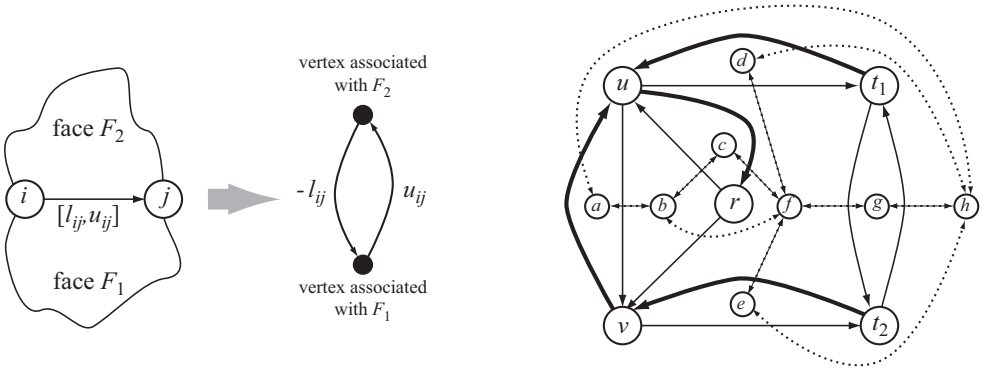

Figure 2: Illustration of the onstruction of $D_{\widetilde{G'}}$ from $\widetilde{G'}$.

subgraph G'' of G' so that $\widetilde{G''}$ has no feasible circulation, without exceeding the budget. Zenklusen [18] (who considers more general budget constraints) shows how this can be done by solving a multi-objective shortest path problem. In our simpler case, we propose to consider a graph $(k+1)D_{\widetilde{G'}}$ that is built as follows. We make $k+1$ copies of $D_{\widetilde{G'}}$, and for every arc (i,j) in A', we consider its standard dual arc $a^d_{ij} = (x,y)$ in $D_{\widetilde{G'}}$, and add links of length 0 from the s-th copy of x to the $(s+1)$-th copy of y $(s = 1, \ldots, k)$. Note that if a vertex v is not the tail of any standard dual arc (v,w) in $D_{\widetilde{G'}}$, then all its neighbors in $D_{\widetilde{G'}}$ are the tail of at least one standard dual arc. The resulting graph is denoted by $(k+1)D_{\widetilde{G'}}$. An example for $k = 1$ and the graph $D_{\widetilde{G'}}$ of Figure 2 is given in Figure 3. Since A' contains seven arcs, there are seven arcs linking the first to the second copy of $D_{\widetilde{G'}}$.

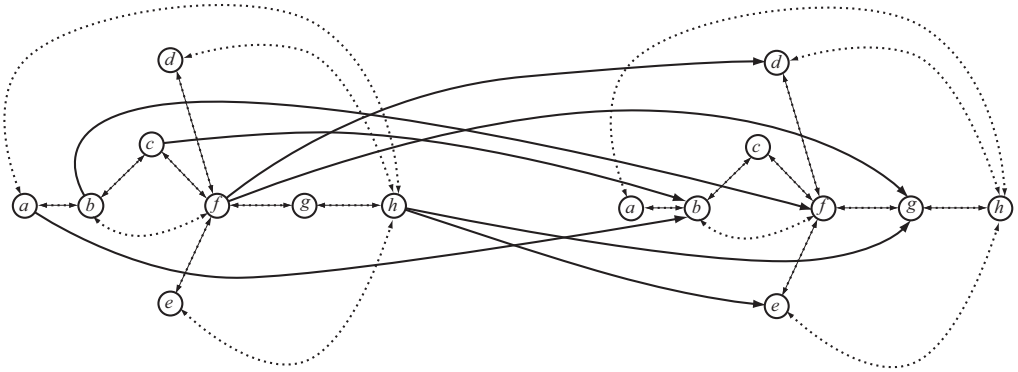

Figure 3: Construction of $2D_{\widetilde{G'}}$ for the graph $\widetilde{G'}$ of Figure 1.

We now prove that it is possible to test the k-survivability of G' by solving a series of shortest path problems in $(k+1)D_{\widetilde{G'}}$.

Theorem 1 *A graph $G' = (V, A')$ is k-survivable if and only if all shortest paths linking a first to a $(k+1)$-th copy of a vertex in $(k+1)D_{\widetilde{G'}}$ have a non-negative length.*

Proof. Assume $G' = (V, A')$ is not k-survivable. As explained above, this means that there is a circuit C in $D_{\widetilde{G'}}$ and a subset A_C of its arc set such that $|A_C| \leq k$, all arcs in A_C are standard dual arcs, and the total length L of the other arcs on C is strictly negative. Let v be a vertex on C which is the tail of at least one standard dual arc (not necessarily on C) in $D_{\widetilde{G'}}$. Consider the path P in $(k+1)D_{\widetilde{G'}}$ that links the first to the $(|A_C|+1)$-th copy of v, and uses arcs (x,y) that link consecutive copies of $D_{\widetilde{G'}}$ when (x,y) belongs to A_C. Clearly, the total length of P is L. Consider any standard dual arc (v,w) in $D_{\widetilde{G'}}$. Note that its opposite arc (w,v) exists and has length 0. We now extend P by adding an arc from the s-th copy of v to the $(s+1)$-th copy of w and an arc for the $(s+1)$-th copy of w to the $(s+1)$-th copy of v, $s = |A_C|+1, \cdots, k$. Clearly the resulting path P' links the first to the $(k+1)$-th copy of v and has length $L < 0$ in $(k+1)D_{\widetilde{G'}}$.

Suppose now there is a path P of strictly negative length L linking a first to a $(k+1)$-th copy of a vertex v in $(k+1)D_{\widetilde{G'}}$. If P contains a proper subpath linking two copies of a vertex w, then consider a minimal (inclusion-wise) such subpath P' (i.e., no proper subpath of P' links to copies of a vertex). If the total length L' of P' in $(k+1)D_{\widetilde{G'}}$ is strictly negative, then P' corresponds to a circuit C in $D_{\widetilde{G'}}$, and the set A_C of arcs on P' that link two vertices in different copies of $D_{\widetilde{G'}}$ is such that $|A_C| \leq k$, all arcs in A_C are standard dual arcs, and the total length of the other arcs on C is $L' < 0$. Hence G' is not k-survivable. So assume all proper subpaths of P that link two copies of a vertex w have a non-negative length. Let $(v = x_1, x_2, \ldots, x_p = v)$ be the sequence of vertices visited by P. If P contains two copies x_i and x_j $(j > i)$ of a vertex with $i \neq 1$ or/and $j \neq p$, then assume x_i belongs to the s-th copy of $D_{\widetilde{G'}}$, while x_i belongs to the s'-th copy $(s' \geq s)$. We build a new path P' from P by considering the new sequence $(v = x_1, \ldots, x_i, x_{j+1}, \ldots, x_p = v)$ where all x_q belonging to the s''-copy of $D_{\widetilde{G'}}$, with $s'' > s$, are moved to the $(s'' + s - s')$-th copy. Hence, P' is a path linking the first to the $(k + s - s' + 1)$-copy of v, and its length is at most L since we have removed a subpath of P of non-negative length. By repeating this process , we obtain a path P^* linking the first to the k'-th copy of v such that $k' \leq k$, P^* has length at most $L < 0$ in $(k+1)D_{\widetilde{G'}}$, and P^* does not contain any proper subpath linking two copies of a vertex. Hence, P^* corresponds to a circuit C in $D_{\widetilde{G'}}$ and the set A_C of arcs on P^* that link two vertices in different copies of $D_{\widetilde{G'}}$ is such that $|A_C| \leq k$, all arcs in A_C are standard dual arcs, and the total length of the other arcs on C is strictly negative, which means that G' is not k-survivable. □

For illustration of the above theorem, consider the path (f, d, h, g, f) in the graph $2D_{\widetilde{G'}}$ of Figure 3, with (h, g) as unique arc linking the first to the second copy of $D_{\widetilde{G'}}$. The length of this path in $2D_{\widetilde{G'}}$ is $u_{ut_1} - 1 + 0 + 0 = u_{ut_1} - 1$, while the corresponding circuit (f, d, h, g, f) in $D_{\widetilde{G'}}$ has length $u_{ut_1} - 1 + u_{t_2t_1} + 0 = u_{ut_1} + u_{t_2t_1} - 1$. Requiring that the total length of the arcs on P must be non-negative is equivalent to impose $u_{ut_1} \geq 1$, which corresponds to the fact that the arc (t_2, t_1) is possibly removed from A', and the flow reaching t_1 can then only come from vertex u.

Theorem 1 provides an easy way to test k-survivability. Indeed, it is sufficient to determine all shortest paths in $(k+1)D_{\widetilde{G'}}$ linking a first to a $(k+1)$-th copy of a vertex. If one of these paths has a strictly negative length, then the second part of the proof of Theorem 1 provides a procedure to determine a set A'' of arcs whose removal transforms G' into a graph with no feasible flow. This is summarized in Procedure TestSurvivability which, given an arc set A', either determines a subset A'' of A' such that $G' = (V, A' \backslash A'')$ has no feasible flow, or produces the message "The graph is k-survivable".

Procedure TestSurvivability(A')

Input : A subgraph $G = (V, A')$ of G;
output: A subset $A'' \subseteq A'$ such that $G' = (V, A' \setminus A'')$ has no feasible flow, or the message "$G' = (V, A')$ is k-survivable"
Construct $(k+1)D_{\widetilde{G'}}$ and set $survivable = true$;
foreach *vertex* x *in* $(k+1)D_{\widetilde{G'}}$ **do**
  if *survivable* **then**
    Determine a shortest path P linking the first to the $(k+1)$-th copy of x in $(k+1)D_{\widetilde{G'}}$;
    if the total cost of the arcs on P is strictly negative **then**
      Apply the procedure used in the second part of the proof of Theorem 1 to determine a set A'' such that $G' = (V, A' \setminus A'')$ has no feasible flow;
      set $survivable = false$
    end
  end
end
if *survivable* **then** write the message "$G' = (V, A')$ is k-survivable";
else Return A'';

3 Tabu search

As shown in [2], the CRSNP is NP-hard, and exact methods can only solve instances of relatively small size. This justifies the use of metaheuristics for larger instances. We propose to apply a tabu search which is one of

the most frequently used metaheuristics in combinatorial optimization [7]. The method can be summarized as follows. Let S be the *solution space* to a combinatorial optimization problem, and let F a function to be minimized over S. For a solution $s \in S$, let $N(s)$ denote the *neighborhood* of s which is defined as the set of solutions in S obtained from s by performing a local change, called *move*. A tabu search generates a sequence $s_0, \cdots, s_q$ of solutions in S where s_0 is an initial solution and each s_i $(i = 1, \cdots, q)$ belongs to $N(s_{i-1})$. In order to avoid cycling, the algorithm uses a *tabu list* that contains forbidden moves. Hence, a move m from s_i to s_{i+1} can only be performed if m does not belong to the tabu list, unless $F(s_{i+1})$ is strictly smaller than the value $F(s^*)$ of the best solution encountered so far. A move in the tabu list has a *tabu status* until is is removed from it. As stopping criterion, one may use a fixed amount of CPU time, a fixed number of iterations, or a fixed number of consecutive iterations without improvement of the value $F(s^*)$. More details are given in [7].

The proposed adaptation of tabu search to the CRSNP can be roughly described as follows. A k-survivable graph is said *inclusion-wise minimal* if the removal of any of its arcs makes it non k-survivable. The solution space S visited by the tabu search is the set of inclusion-wise minimal k-survivable subgraphs G' of the input graph $G = (V, A)$. A move from a solution $G' = (V, A')$ to a neighbor one is performed as follows. We first remove an arc from A'. Since G' is inclusion-wise minimal, this means that the resulting graph is not k-survivable, and we therefore repair it by adding arcs until we obtain a k-survivable subgraph of G. We then remove arcs in order to get a new inclusion-wise minimal k-survivable subgraph. More details about each phase of such a move are given in the next sections. We will then be ready to give a more precise description of the proposed tabu search.

3.1 A repair procedure

In what follows, we denote by c_a the cost of arc a in the input graph G, and by $G' + t$ the graph obtained from a subgraph G' of G by adding a sink t to which all terminals are linked. Let $G' = (V, A')$ be a subgraph of G which has to be repaired in order to become k-survivable. We use the TestSurvivability procedure in order to determine a subset A'' of k arcs in A' whose removal transforms G' into a subgraph $G'' = (V, A' \setminus A'')$ of G with no feasible flow. We then determine a maximum flow f in $G'' + t$. This maximum flow has a value $f_{r \to t}$ strictly smaller than $|T|$ (since it is not feasible). As next step, we build a graph H from $G + t$ as follows:

- all arcs $a \in A' \setminus A''$ for which the flow $f(a)$ is equal to the capacity u_a of a are removed from H; the other arcs of $A' \setminus A''$ are kept in H, but with a zero cost $c_H(a) = 0$.
- all arcs $a \in A \setminus A'$ are kept in H. Those with no tabu status have their original cost $c_H(a) = c_a$, while the others have a cost $c_H(a) = c_a + M$, where M is an integer at least equal to the total cost of the arcs in G.
- all arcs a linking a terminal to the sink t have cost $c_H(a) = 0$.

We next determine a path P of minimum cost linking r to t in H, using cost function c_H. Every arc a on P which does not belong to A' is added to G', which means that by sending one unit of flow on P, the value of the new flow in $G'' + t$ (where $G'' = (V, A' \setminus A'')$ has the updated arc set A') is exactly one unit larger than the previous one. The idea behind the definition of cost function c_H is to only pay for the addition of arcs not in A': if an arc $a \in A \setminus A'$ has no tabu status, then the cost of its addition is its original cost c_a, while a penalty M is added to c_a if a has a tabu status. We repeat this process until we get a flow of $|T|$ units in $G'' + t$, which means that with this new set A' of arcs, the removal of the arcs in A'' does not transform $G' = (V, A')$ into a graph with no feasible flow. There is however possibly another set A''' so that $G'' = (V, A' \setminus A''')$ has no feasible flow. We therefore repeat the complete process until we obtain a k-survivable subgraph of G. The whole repair procedure is summarized in Procedure Repair. Note that if the original graph G is k-survivable, then Repair necessarily produces a set A' so that (V, A') is a k-survivable subgraph of G since the graph $(V, A \setminus A'')$ has a feasible flow for all subsets A'' containing k arcs of A.

3.2 Inclusion-wise minimal solutions

Given a k-survivable subgraph $G' = (V, A')$ of G, we now explain how to remove arcs from A' in order to obtain an inclusion-wise minimal solution. Following the notations of Section 2, let $D_{\widetilde{G'}}$ be the graph that

Procedure Repair(A')

Input : A set $A' \subseteq A$ of arcs such that $G' = (V, A')$ is not k-survivable;
output: An updated set $A' \subseteq A$ of arcs such that $G' = (V, A')$ is k-survivable;
Set $ToBeRepaired = true$;
Set $A'' =$TestSurvivability(A') and $G'' = (V, A' \setminus A'')$;
while $ToBeRepaired$ **do**
  Determine a maximum flow f in $G'' + t$ and set H equal to $G + t$;
  foreach $a \in A'$ **do**
    if ($a \in A''$ **or** $f(a) = u_a$) **then** remove a from H;
    else set $c_H(a) = 0$;
  end
  foreach $a \in A \setminus A'$ **do**
    if a has a tabu status **then** set $c_H(a) = c_a + M$;
    else set $c_H(a) = c_a$;
  end
  for $i = 1$ **to** $|T| - f_{r \to t}$ **do**
    Compute a minimum cost path P (using c_H) linking r to t in H
    foreach $a \in P$ **do**
      if $a \notin A'$ **then**
        Add a to A';
        if $u_a > 1$ **then** set $c_H(a) = 0$ and $f(a) = 1$;
        else remove a from H;
      else
        if $f(a) < u_a - 1$ **then** set $f(a) = f(a) + 1$;
        else remove a from H;
    end
  end
  if the output of TestSurvivability(A') is an arc set A'' **then** set $G'' = (V, A' \setminus A'')$;
  else set $ToBeRepaired = false$;
end

contains a negative circuit if and only if G' has no feasible flow. For every arc $(i, j) \in A'$, we consider its standard dual arc $a^d_{ij} = (x, y)$ and determine a shortest path P linking the first copy of y to the $(k + 1)$-th copy of x in $(k + 1)D_{\widetilde{G'}}$. Let P' be the path in $(k + 2)D_{\widetilde{G'}}$ obtained from P by adding the arc linking the $(k + 1)$-th copy of x to the $(k + 2)$-th copy of y. Clearly, P and P' have the same length. Hence, if P has a negative length, this means that there is a path of negative length in $(k + 2)D_{\widetilde{G'}}$ linking the first to the last copy of y, and this path contains an arc linking x to y in different copies of $D_{\widetilde{G'}}$. In other words, the negative length of P' means that there is a graph obtained from G' by removing (i, j) and k additional arcs which does not contains a feasible flow. This means that if (i, j) is removed from G', then the resulting graph is not k-survivable.

If follows that a procedure that transforms G' into an inclusion-wise minimal k-survivable subgraph of G simply consists in considering all standard dual arcs $a^d_{ij} = (x, y)$ in $D_{\widetilde{G'}}$ and to determine a shortest path P linking the first copy of y to the $(k + 1)$-th copy of x in $(k + 1)D_{\widetilde{G'}}$: if P has a negative length, this means that (i, j) must stay in A' since its removal would make G' not k-survivable; if P has a positive length, then (i, j) can be removed from G'. The procedure that transforms G' into an inclusion-wise minimal solution is called MakeMinimal and is described here below.

Procedure MakeMinimal(A')

Input : A subset $A' \subseteq A$ of arcs so that $G' = (V, A')$ is k-survivable;
output: An updated subset A' so that $G' = (V, A')$ is k-survivable and inclusion-wise minimal;
foreach $(i, j) \in A'$ **do**
  Consider the standard dual arc $a^d_{ij} = (x, y)$ of (i, j) and compute a shortest path P from the first copy of y to the $(k + 1)$-th copy of x in $(k + 1)D_{\widetilde{G'}}$;
  if P has a non-negative total length **then** remove (i, j) from A' ;
end

Note that there are possibly more than one inclusion-wise minimal subgraphs of G', and the order in which the arcs are considered for possible deletion may therefore have an impact on the resulting graph. We

suggest an ordering that gives preference to large costs, but with a random component to bring diversity. More precisely, we suggest to first multiply each cost c_a by a random number ρ_a uniformly chosen in $]0,1]$, which gives a new cost $c'_a = \rho_a c_a$, and to then order the arcs by non-increasing value c'_a.

3.3 The proposed tabu search for the CRSNP

We now describe in more details the proposed tabu search procedure. In what follows, we denote by $c(A) = \sum_{a \in A} c_a$ the total cost of an arc set A.

We first determine whether the input graph $G = (V, A)$ is k-survivable, using the TestSurvivability procedure. If it turns out that $G = (V, A)$ is not k-survivable, then there is obviously no feasible solution to the CRSNP, and we therefore stop the procedure. Otherwise, we start the tabu search. The initial solution $G' = (V, A')$ is generated by applying the MakeMinimal procedure to $A' = A$.

Then, given any solution $G' = (V, A')$ visited by the tabu search, we select a subset A'' of $\lceil \lambda |A'| \rceil$ arcs in A', where λ is a parameter whose value belongs to $]0, 1]$. For every arc a in A'', we create a new set A''_a by first removing a from A', then applying the Repair procedure, and finally running the MakeMinimal procedure to get a set A''_a so that $(V, A''_a))$ is an inclusion-wise minimal k-survivable subgraph of G. In order to avoid the presence of arc a in A''_a, we add a temporary penalty M to its cost c_a, this penalty being removed when moving to the next arc in A''. Note that it may happen that a necessarily belongs to A''_a. For example, if $G = (V, A)$ contains only two vertices, one being the root r and the other a terminal t_1, and if A contains only two parallel arcs of capacity 1 linking r to t_1, then G is 1-survivable since it contains a feasible flow, even if one of its arcs is removed. However, no proper subgraph of G is 1-survivable. Hence, if one of its arc is removed, the Repair procedure has to add it again to recover 1-survivability. This is the reason why we penalize the insertion of a in A''_a instead of forbidding it. For the same reason, the arcs with a tabu status can be required to repair $A \backslash \{a\}$, and they are therefore also considered for insertion into A''_a, but with a penalty M.

Once all arcs in A'' are treated, the algorithm moves from A' to the set A''_a with lowest total cost, and the arc a that was removed from A' to produce A''_a gets a tabu status for the next τ iterations, where τ is a parameter of the method. The algorithm stops when a stopping criterion is met (which will be a time limit in our case). A detailed description of the whole process appears in Procedure TabuSearchCRSNP.

In order not to choose always the same subset A'' of arcs while giving preference to arcs with a large cost, we suggest to use the same kind of technique as the one proposed for ordering the vertices before applying the MakeMinimal procedure. More precisely, we suggest to compute a value $c'_a = \rho_a c_a$ for each $a \in A'$, where ρ_a is a random number uniformly generated in $]0, 1]$. We then sort the arcs a of A' by non-increasing value c'_a, and include the $\lceil \lambda |A'| \rceil$ first ones in A''.

4 Computational experiments

An exact algorithm for the CRSNP based on an integer programming formulation is described in [2]. It applies to all graphs, hence also to non planar ones. We compare it to our TabuSearchCRSNP procedure. All experiments are performed on a computer with a 2.40GHz Intel(R) Core(TM) i7-5500U CPU and 16Gb of RAM, and the integer programs are solved using CPLEX (v12.2).

Tests are performed on instances with $|V| \in \{20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100\}$ vertices, and with $|T| \in \{\lceil 0.1|V| \rceil, \lceil 0.2|V| \rceil, \lceil 0.3|V| \rceil, \lceil 0.6|V| \rceil, |V| - 1\}$ terminals. To create them, we have generated $|V|$ random vertices in the Euclidean plane $[0, 1] \times [0, 1]$, and the edges are those of a Delauney triangulation. One vertex, chosen at random is defined as the root r, while $|T|$ other vertices, chosen at random, are considered as terminals. This gives a total of 60 undirected planar graphs. To get oriented graphs, we have replaced every edge linking two vertices i and j with two arcs (i, j) and (j, i).

Since arcs that are close to the root r typically require larger capacities than the other ones, the capacity u_{ij} of an arc (i, j) is chosen at random in $\{\lceil 0.8|T| \rceil, \lceil 0.6|T| \rceil\}$ if there is path with at most 2 arcs linking r to i, and in $\{\lceil 0.8|T| \rceil, \lceil 0.6|T| \rceil, \lceil 0.4|T| \rceil, \lceil 0.2|T| \rceil\}$ otherwise. According to our experiments, these capacities are high enough to assure, in most cases, the existence of a 1-survivable subgraph, but are also low enough to make the CRSPN difficult to solve.

Procedure TabuSearchCRSNP

Input : A graph $G = (V, A)$ with capacity and cost functions on the arc set, two parameter λ and τ, a positive integer k;
Output: A set A^* of arcs so that (V, A^*) is inclusion-wise minimal k-survivable, or the message "$G = (V, A)$ is not k-survivable";
if the output of TestSurvivability(A) is an arc set **then**
  write the message "$G = (V, A)$ is not k-survivable"
end
else
  Set $A' = A$, apply MakeMinimal(A') and set $A^* = A'$;
  while no stopping criterion is met **do**
    Generate a subset A'' of $\lceil \lambda |A'| \rceil$ arcs in A' and set $BestCost = +\infty$;
    foreach $a \in A''$ **do**
      Set $A''_a = A' \setminus \{a\}$ and $c_a = c_a + M$;
      Apply Repair(A''_a);
      Apply MakeMinimal(A''_a);
      if $c(A''_a) < BestCost$ **then** set $BestCost = c(A''_a)$ and $a_{best} = a$;
      Set $c_a = c_a - M$;
    end
    Set $A' = A''_{a_{best}}$ and assign a tabu status to a_{best} for τ iterations;
    if $c(A') < c(A^*)$ **then** set $A^* = A'$;
  end
end

For the costs, we have chosen values that depend on the length (in the Euclidean plane) and the capacity of the arcs. More precisely, for two vertices i and j with coordinates (x_i, y_i) and (x_j, y_j), we have set $c_{ij} = \frac{1}{|T|} u_{ij} \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$, which means that the Euclidian distance is multiplied by 0.8, 0.6, 0.4 or 0.2.

Seven of these 60 instances did not pass the test of 1-survivability at the first line of of Procedure TabuSearchCRSNP. These instances therefore do not appear in our experiments. The tabu search was run 10 times on the 53 remaining instances and each execution was stopped after 45 seconds, while we have allocated 3000 seconds to each run of the exact method described in [2]. Based on preliminary experiments, we have used $\lambda = 0.4$ and $\tau = \left\lceil \sqrt{|A|}/2 \right\rceil$ at lines 7 and 15 of TabuSearchCRSNP, respectively.

Results for $k = 1$ appear in Table 1. The $|V|$ and $|T|$ columns indicate the number of vertices and terminals of each instance. Columns C_{opt} and T_{opt} indicate the cost of an optimal solution and the time (in seconds) needed by the exact method to get it. If no proof of optimality was obtained after 3000 seconds, we indicate the cost of the best found feasible solution as well as, in brackets, the best lower bound on the value of an optimal solution. Column C_b indicates the best cost obtained after 10 runs of the tabu search, while column T_b gives the best time needed to get such a solution, and column n_b shows the number of runs for which a solution of that cost was found. Gray boxes in column C_b mean that we could reach the proven optimal solution, while gray boxes in column n_b indicate that the 10 runs of our tabu search all ended with the same cost. Column C_a gives the average cost produced by our tabu search, over the 10 runs. While each run was stopped after 45 seconds, column T_a indicates the average time at which the best solution A^* was last improved (line 16 of Procedure TabuSearchCRSNP). Column I_a indicates the average number of iterations of our tabu search in 45 seconds.

We observe that TabuSearchCRSNP could find the optimal solution for all instances solved to optimality by the exact method. Moreover, these optimal solutions were obtained for each of the 10 runs, except in one case. Indeed, for the instance with $|V| = 45$ vertices and $|T| = 44$ terminals, our tabu search has found the optimal solution in only 3 of the 10 runs, but the average cost $C_a = 28.03$ is very close to the optimal value $C_{opt} = 27.90$. We also note that when the exact method has produced a proven optimal solution (i.e., $T_{opt} < 3000$), the average time T_a needed by TabuSearchCRSNP to obtain the same optimal cost is significantly smaller than T_{opt}. For the instances not solved to optimality (i.e., $T_{opt} = 3000$), the average cost produced by our tabu search is always strictly better than the best cost found by the exact method. We observe that the average number I_a of iterations of the tabu search decreases when the number $|T|$ of terminals or the number $|V|$ of vertices increases. For graphs with 100 vertices, less than 100 iterations could

be performed, in average, which indicates that more time should probably be allocated so that the tabu search has more chance to find solutions of good quality.

Similar results appear in Tables 2 and 3 for $k = 2$ and $k = 3$, respectively. Among the 53 1-survivable instances, only 31 instances passed the 2-survivability test, and 8 of them passed the 3-survivability test of line 1 in Procedure TabuSearchCRSNP. Again, we observe that our tabu search could find the optimal solution for all instances solved to optimality by the exact method, and these optimal solutions were obtained for each of the 10 runs, except in one case : the instance with $|V| = 45$ vertices, $|T| = 14$ terminals and $k = 2$, where our tabu search has found the optimal solution in only 7 of the 10 runs. But also in this case, the average cost $C_a = 25.71$ is very close to the optimal value $C_{opt} = 25.63$. For these larger values of k, we see that TabuSearchCRSNP is able, in 45 seconds, to get very big improvements when compared to the best solutions produced by the exact method in almost one hour. For example, for the instance with $|V| = 100$ vertices, $|T| = 20$ terminals and $k = 3$, the exact method stopped with a best upper bound equal to 54.79, while our tabu search has generated a solution of value $C_b = 35.82$, and the average cost C_a is equal to 36.43.

Up to this point, all instances had an arc (i, j) in A if and only if $(j, i) \in A$, while in practice, it often happens that only one of them exists in the given graph $G = (V, A)$. We have thus decided to also test our tabu search on instances where each edge of the original undirected graphs is oriented in exactly one of the two possible directions. For each of the 53 1-survivable instances ot Table 1, we have deleted one of the two arcs $(i, j), (j, i)$ for each pair of adjacent vertices. The chosen directions were made so that the resulting oriented graphs remained 1-survivable. We have then run our algorithm on these new instances. Clearly, the optimal value for each new instance is at least as large than the value of the same instance where all edges were associated with two arcs with opposite directions. Results appear in Tables 4, 5 and 6. While 53 new instances were 1-survivable, only 26 of them were 2-survivable, and 4 were 3-survivable. We can observe that the removal of exactly one the two arcs (i, j), (j, i) for each pair of adjacent vertices has a big impact on the optimal value. For example, for $|V| = 35$ vertices, $|T| = 10$ terminals, and $k = 3$, the optimal solution in Table 3 has value 34.76, while the optimal cost C_{opt} is 55.91 in Table 3. When comparing C_{opt} with C_b we again observe that our tabu search was always able to determine an optimal solution when the exact method stopped with a proven optimal one, except in one case : for the instance with $|V| = 90$ vertices, $|T| = 54$ terminals and $k = 2$, the optimal cost C_{opt} is 93.78, while the best solution produced by our tabu search has cost $C_b = 94.05$, and the average cost is $C_a = 94.25$. Note however that both T_a and T_b are larger than 40, which means that our tabu search improved the best solution A^* less than 5 seconds before the process was stopped. For this particular instance, we have determined that 25 additional seconds would have been sufficient to reach the optimal solution. By allocating 120 seconds instead of 45, we have even been able to reach the optimum in 7 of the 10 runs.

For four instances with $k = 1$, four with $k = 2$, and one with $k = 3$, our tabu search had at least one run out of 10 which did not end with the proven optimal cost. But for all these instances, the mean cost C_a is always very close to the optimal cost C_{opt}.

5 Conclusion

Given a directed rooted planar graph $G = (V, A)$ with capacity and cost functions on A, a subset $T \subset V$ of terminals, and an integer k, we are interested in solving the CRSNP which is to determine a minimum cost subset $A' \subseteq A$ of arcs such that $G' = (V, A')$ is k-survivable. In other words, every subgraph obtained from G' by removing at most k arcs must admit a feasible flow that routes one unit of flow from the root to every terminal. The problem of determining whether a graph is k-survivable is NP-complete in the general case. We have shown how planar duality properties allow to solve this problem in polynomial time in planar graphs, by determining a series of shortest paths.

Exact methods for the CRSNP can only solve relatively small size instances. We have described a tabu search that can handle much larger instances, and we have shown that it typically produces optimal solutions when these are known. Our algorithm has very low computing times, which makes it particularly interesting in practice, for example for the design of survivable wind farm collection networks with hundreds of wind turbines.

Table 1: Results for $k = 1$ where $(i, j) \in A$ if and only if $(j, i) \in A$.

—V—	—T—	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
20	4	9.61	1.0	9.61	0.0	10	9.61	0.1	8613
20	6	14.26	5.6	14.26	0.0	10	14.26	0.1	3666
20	12	18.29	29.9	18.29	0.0	10	18.29	0.7	1912
20	19	21.79	13.1	21.79	0.2	10	21.79	0.9	1867
25	5	12.43	12.1	12.43	0.1	10	12.43	0.2	2776
25	8	14.27	13.9	14.27	0.2	10	14.27	0.4	2482
25	15	17.50	42.8	17.50	0.6	10	17.50	1.5	1126
30	3	8.49	1.7	8.49	0.0	10	8.49	0.1	5653
30	9	12.49	14.1	12.49	0.1	10	12.49	0.9	1603
30	18	16.40	47.7	16.40	0.5	10	16.40	2.3	1067
30	29	20.65	40.2	20.65	0.8	10	20.65	3.3	401
35	4	5.91	1.6	5.91	0.0	10	5.91	0.1	3998
35	7	14.25	127.7	14.25	0.3	10	14.25	1.2	1114
35	10	15.49	44.4	15.49	0.4	10	15.49	1.0	1063
35	34	19.18	16.8	19.18	2.1	10	19.18	3.7	256
40	4	11.43	48.6	11.43	0.1	10	11.43	0.8	2174
40	8	14.44	27.0	14.44	0.8	10	14.44	3.5	805
40	12	19.96	1971.7	19.96	2.3	10	19.96	16.2	501
40	24	17.14	104.0	17.14	0.8	10	17.14	6.3	350
40	39	20.45	55.5	20.45	6.0	10	20.45	16.4	183
45	4	7.90	112.8	7.90	0.2	10	7.90	1.0	1919
45	9	12.72	848.0	12.72	1.4	10	12.72	14.3	739
45	14	16.54	91.6	16.54	1.8	10	16.54	4.1	389
45	27	24.84 (24.34)	3000	24.78	6.5	4	24.83	22.4	219
45	44	27.90	186.5	27.90	11.5	3	28.03	19.9	131
50	5	12.65	1507.1	12.65	0.9	10	12.65	5.8	700
50	10	18.07 (13.73)	3000	17.31	4.8	8	17.31	13.6	329
50	15	13.78	1090.7	13.78	2.2	10	13.78	10.6	419
50	30	17.19	420.5	17.19	3.5	10	17.19	10.7	192
60	6	20.49 (10.45)	3000	16.75	2.1	7	16.81	15.8	452
60	12	20.04 (15.83)	3000	18.20	2.9	10	18.20	15.6	225
60	18	18.70 (15.33)	3000	17.14	12.6	9	17.14	23.1	192
60	36	25.39 (21.58)	3000	24.16	5.7	2	24.21	18.7	103
70	7	12.98 (10.12)	3000	12.35	2.4	9	12.35	9.7	404
70	14	22.09 (9.28)	3000	18.14	25.6	2	18.25	19.3	147
70	21	19.79 (17.02)	3000	19.21	31.1	1	19.34	21.5	108
70	42	26.60 (18.62)	3000	23.34	14.3	3	23.47	28.4	78
70	69	29.99 (21.59)	3000	26.34	40.9	1	26.61	35.2	41
80	8	14.82 (9.04)	3000	12.34	4.2	9	12.34	16.3	208
80	16	23.72 (9.49)	3000	16.83	9.5	6	16.95	24.5	125
80	24	27.70 (11.91)	3000	21.96	32.9	1	22.12	27.8	82
80	48	25.92 (14.77)	3000	21.75	39.3	1	21.88	30.8	59
80	79	32.76 (18.02)	3000	27.50	30.6	1	28.78	39.4	31
90	9	25.24 (9.59)	3000	18.85	40.2	1	19.09	33.5	103
90	18	28.10 (9.84)	3000	18.61	17.1	2	18.77	28.2	99
90	27	33.90 (7.80)	3000	23.14	34.8	1	23.48	34.0	62
90	54	34.49 (12.07)	3000	24.92	41.5	1	25.41	40.8	45
90	89	32.34 (21.02)	3000	29.36	43.0	1	30.25	41.6	36
100	10	26.21 (9.70)	3000	16.45	25.3	1	16.75	26.5	89
100	20	21.34 (11.10)	3000	16.60	15.1	1	16.66	27.0	82
100	30	31.27 (8.76)	3000	19.96	34.3	1	20.40	34.7	44
100	60	33.68 (14.15)	3000	25.03	40.7	1	25.53	40.6	27
100	99	34.04 (12.59)	3000	31.38	41.2	2	32.69	43.6	19

Table 2: Results for $k = 2$ where $(i, j) \in A$ if and only if $(j, i) \in A$.

—V—	—T—	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
20	4	15.23	1.0	15.23	0.0	10	15.23	0.1	4929
20	6	20.93	5.6	20.93	0.0	10	20.93	0.1	2069
25	8	22.03	17.6	22.03	0.2	10	22.03	0.9	1309
25	15	26.49	94.6	26.49	0.4	10	26.49	1.1	616
30	9	19.46	17.1	19.46	0.2	10	19.46	1.6	1021
35	10	23.06	38.6	23.06	1.2	10	23.06	2.7	567
40	4	16.59	154.2	16.59	0.2	10	16.59	1.3	914
40	12	29.05	894.4	29.05	2.5	10	29.05	6.8	257
40	39	35.57	13.3	35.57	2.1	10	35.57	15.4	119
45	4	12.69	164.9	12.69	0.8	10	12.69	2.3	703
45	9	19.00	1425.4	19.00	0.5	10	19.00	4.1	327
45	14	25.63	138.2	25.63	5.9	7	25.71	15.0	175
50	5	17.56	991.7	17.56	2.2	10	17.56	5.5	338
50	10	28.57 (19.43)	3000	25.53	6.5	6	25.57	12.2	139
50	15	23.95 (22.27)	3000	23.33	2.0	10	23.33	7.0	177
60	6	25.72 (16.98)	3000	23.25	11.0	3	23.31	25.2	142
60	18	29.29 (21.97)	3000	26.92	17.4	7	26.92	28.1	100
60	36	40.59 (35.31)	3000	39.29	24.6	2	39.55	32.6	52
70	7	21.19 (10.81)	3000	17.59	3.9	9	17.60	23.7	189
70	14	33.25 (16.96)	3000	27.05	9.1	3	27.16	24.3	74
80	8	26.17 (12.56)	3000	18.84	11.1	8	18.86	21.8	89
90	9	41.56 (12.47)	3000	25.10	23.6	7	25.12	30.9	59
90	18	38.52 (11.56)	3000	28.05	29.0	1	28.40	36.1	47
90	27	47.35 (10.42)	3000	34.85	34.5	1	35.38	40.4	38
90	54	51.67 (10.85)	3000	41.16	41.5	1	41.79	41.3	22
90	89	52.89 (29.09)	3000	51.03	42.8	1	52.35	43.2	17
100	10	36.55 (12.06)	3000	23.29	15.1	1	23.64	33.6	47
100	20	39.24 (11.77)	3000	25.37	29.5	1	25.51	30.3	48
100	30	42.82 (11.99)	3000	32.04	39.6	1	32.80	39.7	31
100	60	50.52 (14.65)	3000	40.51	44.3	1	41.82	42.9	22
100	99	53.51 (26.76)	3000	51.02	47.4	1	52.55	44.6	16

Table 3: Results for $k = 3$ where $(i, j) \in A$ if and only if $(j, i) \in A$.

—V—	—T—	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
35	10	34.76	42.4	34.76	1.2	10	34.76	3.2	320
40	4	23.51	356.1	23.51	0.9	10	23.51	3.6	423
45	4	18.64	335.1	18.64	1.0	10	18.64	2.4	335
60	6	35.78 (22.18)	3000	32.83	7.0	9	32.89	19.0	95
70	7	29.54 (15.36)	3000	25.96	14.9	3	26.03	17.1	74
70	14	44.53 (23.60)	3000	37.93	13.8	3	37.99	22.8	53
100	10	42.54 (15.31)	3000	32.12	38.8	2	32.60	40.7	27
100	20	54.79 (15.89)	3000	35.82	36.9	1	36.43	39.8	24

Table 4: Results for $k = 1$ where $(i, j) \notin A$ if $(j, i) \in A$.

—V—	—T—	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
20	4	12.63	11.5	12.63	0.0	10	12.63	0.0	4912
20	6	17.48	9.4	17.48	0.1	10	17.48	0.3	2293
20	12	24.94	16.0	24.94	0.0	10	24.94	1.6	1004
20	19	42.57	11.3	42.57	0.0	10	42.57	0.1	963
25	5	14.33	20.8	14.33	0.0	10	14.33	0.1	1748
25	8	20.45	28.2	20.45	0.0	10	20.45	0.5	1713
25	15	24.24	21.9	24.24	0.3	10	24.24	0.7	665
30	3	8.61	2.3	8.61	0.0	10	8.61	0.0	5479
30	9	20.18	77.0	20.18	0.3	10	20.18	0.9	907
30	18	30.40	23.5	30.40	0.1	10	30.40	0.4	520
30	29	41.76	20.0	41.76	0.3	10	41.76	0.8	312
35	4	9.03	10.2	9.03	0.1	10	9.03	0.2	2984
35	7	20.76	32.2	20.76	0.2	10	20.76	0.4	669
35	10	24.68 (22.46)	3000	24.07	0.4	10	24.07	3.7	538
35	34	50.56	57.2	50.56	1.2	3	50.56	1.6	213
40	4	13.28	116.2	13.28	0.1	10	13.28	0.9	1649
40	8	25.23	199.9	25.23	0.3	10	25.23	0.8	635
40	12	26.82	850.1	26.82	1.1	4	26.83	1.3	352
40	24	31.97	120.3	31.97	1.5	10	31.97	2.3	207
40	39	55.09	62.0	55.09	3.0	10	55.09	4.6	131
45	4	9.06	78.0	9.06	0.2	10	9.06	1.2	1036
45	9	18.14	241.3	18.14	0.8	10	18.14	1.7	449
45	14	26.20	440.5	26.20	5.4	10	26.20	10.9	267
45	27	44.85	162.6	44.85	2.3	10	44.85	4.1	191
45	44	59.17	126.8	59.17	5.5	10	59.17	8.6	102
50	5	13.87	373.5	13.87	0.1	10	13.87	1.1	690
50	10	25.40 (21.34)	3000	24.93	1.4	8	24.93	11.8	296
50	15	26.9 (19.37)	3000	25.85	2.3	7	25.96	3.4	238
50	30	37.74	183.3	37.74	2.5	10	37.74	6.2	141
60	6	24.75 (13.12)	3000	21.73	8.4	9	21.74	17.4	286
60	12	29.93 (26.64)	3000	29.41	5.0	2	29.53	9.1	198
60	18	32.33 (17.13)	3000	29.91	8.1	5	29.93	18.2	169
60	36	44.04	2487.4	44.04	14.3	7	44.06	25.6	101
70	7	19.03 (10.85)	3000	16.93	1.9	10	16.93	10.1	239
70	14	35.52 (7.81)	3000	28.71	19.7	3	28.79	29.4	121
70	21	34.28 (24.00)	3000	32.18	9.9	8	32.19	15.8	94
70	42	54.11 (45.45)	3000	52.23	19.6	1	52.45	24.6	63
70	69	64.24	731.2	64.24	35.6	2	64.32	39.1	52
80	8	20.51 (11.69)	3000	17.04	5.0	6	17.06	18.8	177
80	16	33.52 (5.36)	3000	26.20	22.0	1	26.75	16.6	99
80	24	45.90 (30.29)	3000	39.77	16.1	1	40.49	21.2	68
80	48	49.84 (38.38)	3000	48.61	39.6	1	48.77	37.5	46
80	79	65.31 (64.00)	3000	65.76	44.1	1	66.58	42.4	37
90	9	30.81 (8.32)	3000	23.40	9.7	2	23.51	27.8	83
90	18	41.73 (12.94)	3000	32.29	38.1	1	32.67	32.2	63
90	27	44.92 (17.07)	3000	38.07	26.9	1	38.43	34.0	48
90	54	51.14 (32.73)	3000	48.83	41.1	1	49.68	41.3	31
90	89	73.08 (73.08)	3000	73.94	40.7	1	74.92	42.4	27
100	10	24.27 (9.71)	3000	21.84	11.4	3	22.00	24.8	95
100	20	35.93 (8.98)	3000	27.38	34.3	1	27.71	30.9	70
100	30	46.31 (12.97)	3000	36.78	33.7	1	37.55	39.9	44
100	60	57.19 (37.17)	3000	56.38	40.5	1	57.12	42.7	37
100	99	75.05 (70.55)	3000	76.98	46.2	1	78.03	44.1	24

Table 5: Results for $k = 2$ where $(i, j) \notin A$ if $(j, i) \in A$.

—V—	—T—	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
20	4	26.99	15.8	26.99	0.1	10	26.99	1.8	2084
20	6	29.39	5.9	29.39	0.0	10	29.39	0.0	1343
25	8	34.98	12.0	34.98	0.0	10	34.98	0.1	817
25	15	47.06	18.9	47.06	0.0	10	47.06	0.1	425
30	9	35.88	46.5	35.88	0.4	10	35.88	0.7	451
35	10	41.52 (38.61)	3000	41.34	1.0	4	41.34	2.6	292
40	4	21.51	515.6	21.51	0.5	10	21.51	1.2	749
40	12	43.32	75.0	43.32	1.6	4	43.43	5.7	216
45	4	17.40 (17.23)	3000	17.40	0.5	10	17.40	2.2	527
45	9	35.64	281.2	35.64	1.9	10	35.64	3.3	222
45	14	47.78	190.5	47.78	2.8	8	47.78	13.9	179
50	5	24.84	2313.6	24.84	1.0	10	24.84	3.4	325
50	10	43.90 (38.19)	3000	41.97	2.9	9	42.04	8.6	181
60	6	38.68 (17.79)	3000	33.81	12.2	1	33.91	19.2	124
60	18	52.55 (50.97)	3000	52.23	9.0	10	52.23	18.2	99
60	36	85.78	438.2	85.78	10.4	6	85.80	21.9	51
70	7	32.38 (18.46)	3000	29.82	6.8	4	29.93	11.8	119
70	14	56.24 (9.56)	3000	47.52	16.6	2	47.66	27.7	71
80	8	33.42 (19.38)	3000	29.58	7.1	7	29.63	15.2	84
90	18	61.59 (22.79)	3000	51.37	33.9	1	51.83	38.7	45
90	27	71.56 (46.51)	3000	68.61	40.8	1	68.94	41.8	36
90	54	93.78	1983.3	94.05	42.4	2	94.25	41.4	25
100	10	52.01 (12.48)	3000	36.29	34.6	1	36.94	37.3	36
100	20	58.46 (6.43)	3000	44.51	41.6	1	45.06	41.4	28
100	30	77.10 (40.09)	3000	65.17	43.1	1	65.55	43.3	19
100	60	106.62 (105.55)	3000	106.45	45.9	1	107.35	46.3	10

Table 6: Results for $k = 3$ where $(i, j) \notin A$ if $(j, i) \in A$.

V	T	C_{opt}	T_{opt}	C_b	T_b	n_b	C_a	T_a	I_a
35	10	55.91	121.8	55.91	18.8	1	56.92	16.5	222
40	4	33.59	988.4	33.59	1.0	9	33.64	11.2	379
45	4	28.54	2165.7	28.54	0.6	10	28.54	1.8	311
60	6	52.69 (32.67)	3000	49.12	13.2	1	49.24	12.8	109

References

[1] C. Bentz, MC. Costa, A. Hertz, On the edge capacitated Steiner tree problem, CoRR, abs/1607.07082 (2016).

[2] C. Bentz, MC. Costa, P. Poirion, T. Ridremont, Formulations for designing robust networks. An application to wind power collection, to appear in the Proceedings of the 8th International Network Optimization Conference (INOC), February 2017, Lisbonne, Portugal, Electronic Notes in Discrete Mathematics Special Issue.

[3] D. Bienstock, G. Muratore, Strong inequalities for capacitated survivable network design problems, Mathematical Programming, 89(1)(2000) 127–147.

[4] Q. Botton, B. Fortz, L. Gouveia, M. Poss, Benders decomposition for the hop-constrained survivable network design problem, INFORMS journal on computing, 25(1)(2013) 13–26.

[5] Q. Botton, B. Fortz, L. Gouveia, On the hop-constrained survivable network design problem with reliable edges, Computers & Operations Research, 64(2015) 159–167.

[6] G. Dahl, M. Stoer, A cutting plane algorithm for multicommodity survivable network design problems, INFORMS Journal on Computing, 10(1)(1998) 1–11.

[7] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, Norwell, MA, 1997.

[8] M. Goemans, D. Bertsimas, Survivable networks, linear programming relaxations and the parsimonious property, Mathematical Programming, 60(1–3)(1993) 145–166.

[9] M. Grotschel, CL. Monma, M. Stoer, Design of survivable networks, Handbooks in Operations Research and Management Science, 7(1995) 617-672.

[10] A. Hertz, O. Marcotte, A. Mdimagh, M. Carreau, F. Welt, Optimizing the design of a wind farm collection network, INFOR: Information Systems and Operational Research, 50(2)(2012) 95–104.

[11] H. Kerivin, D. Nace, J. Geffard, Design of survivable networks with a single facility, Universal Multiservice Networks, 2002. ECUMN 2002. 2nd European Conference on, IEEE (2002) 208–218.

[12] H. Kerivin, AR. Mahjoub, Design of survivable networks: A survey, Networks, 46(1)(2005) 1–21.

[13] EL. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart and Winston, 1976.

[14] C. Phillips, The network inhibition problem, Proceedings of the twenty-fifth annual ACM symposium on Theory of computing, 1(1993) 776–785.

[15] D. Rajan, A. Atamtürk, A directed cycle-based column-and-cut generation method for capacitated survivable network design, Networks, 43(4)(2004) 201–211.

[16] M. Stoer, G. Dahl, A polyhedral approach to multicommodity survivable network design, Numerische Mathematik, 68(1)(1994) 149–167.

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

[18] R. Zenklusen, Network flow interdiction on planar graphs, Discrete Applied Mathematics, 158(13)(2010) 1441–1455.