A Branch-and-Cut Algorithm for the Ring Spur Assignment Problem

Paula Carroll, Bernard Fortz, Martine Labbé and Seán McGarraghy
Quinn and Smurfit Schools of Business, University College Dublin, Ireland
and Département d'Informatique, Faculté des Sciences, Université Libre de Bruxelles

Abstract: The Ring Spur Assignment Problem (RSAP) arises in the design of Next Generation Telecommunications Networks (NGNs) and has applications in location-allocation problems. The aim is to identify a minimum cost set of interconnected ring spurs. We seek to connect all nodes of the network either on a set of bounded disjoint local rings or by a single spur edge connected to a node on a local ring. Local rings are interconnected by a special ring called the tertiary ring. We show that the problem is NP-Hard and present an Integer Programming formulation with additional valid inequalities. We implement a branch-and-cut algorithm and present our conclusions with computational results.

Keywords: Survivable Network Design, NP-Hardness Proof, IP Formulation, Branch-and-Cut Algorithm

1 Introduction

Modern economies demand *always on* information networks, yet the underlying physical medium on which the information economy depends is vulnerable. Duplicating every transmission path would of course create a reliable network but this demand for reliability competes almost directly with shareholder demand for return on investment. De-regulation of the Irish telecommunication market in 1998 opened up the market to competition and the need for efficient usage of resources.

The Ring Spur Assignment Problem (RSAP) is motivated by just such a practical situation: a network operator seeks to identify an economical fault tolerant Next Generation Network (NGN). Two IP formulations for the RSAP were presented and compared in Carroll et al. (2011). In this paper we show that the problem is NP-hard and strengthen the formulations with additional valid inequalities.

In our work, we focus on selecting a survivable topology in the physical layer of the Open systems Interconnection (OSI) model. We seek to identify a 2-connected (ring) topology to ensure survivability. If a suitable topology can be identified in the physical layer, the logical network can be implemented and managed at the logical layer. We seek a series of bounded disjoint local rings that are interconnected by a special ring called the tertiary ring. This eliminates the possibility of implementing mesh based networks described in Grover (2003). We wish to use pre-installed capacity in the physical Synchronous Digital Hierarchy (SDH) network to minimise costs. Our aim of identifying a highly resilient topology at minimum cost is achieved by assigning locations to rings.

The RSAP is a complement of the SDH Ring Assignment Problem (SRAP) problem described in Goldschmidt et al. (2003). We are identifying rings that can carry the estimated demand. However, no SRAP solution is possible in some real world instances. As an alternative, where no SRAP solution exists, we allow locations that have insufficient spare capacity or no possible physical route due to limitations of geography, to be connected to Self Healing Rings (SHRs) by spurs off the local rings. Spur nodes must be connected to a local ring by a single edge, i.e. we do not allow a chain of edges to connect spur nodes. A feasible topology for RSAP is illustrated in Figure 1.

Soni et al. (1999) propose a taxonomy to classify the many survivable network design problems described in the literature. They suggest the layer in the OSI model where network survival is provided as the most appropriate classification characteristic. Protection can be provided at the physical layer for networks by specifying a topology with either low or high connectivity requirements, as, for example, in (Grötschel et al., 1995). Protection can then be provided at the logical layer assuming the underlying physical network is survivable by specifying the minimum number of node or edge disjoint paths between demand pairs, as, for example, in Huygens et al. (2007). Our work can be classified amongst approaches providing physical survivability but differs from other works since we aim to exploit existing physical infrastructure where possible.

In addition, we note that the solution topology we propose may have practical applications in the area of Facility Location (Korte and Vygen, 2008) and Rapid Transit Network Design (Laporte et al., 2007).

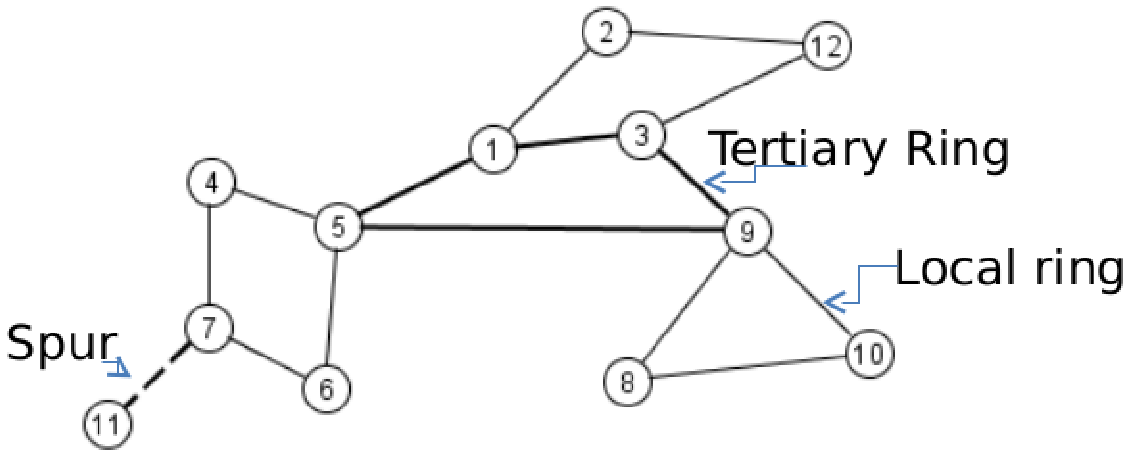


Figure 1: An example of feasible RSAP topology

The layout of our paper is as follows: In Section 2 we give a formal statement of the RSAP, then we review some Survivable Network Design Problems described in the literature to set the context for the RSAP in Section 3. In Section 4 we describe an IP formulation with additional valid inequalities. In Section 5 we describe our branch-and-cut implementation. Computational results are presented in Section 6. We summarise our conclusions in Section 7 and include a detailed NP-hardness proof of the RSAP in the Appendix.

2 The Ring Spur Assignment Problem (RSAP)

Communities of interest, defined in Cosares et al. (1995) as geographically close nodes that have high traffic demands between them, are identified and their traffic demands are estimated. If such communities can be clustered on node disjoint rings, no wavelength conversion is required, eliminating the cost of wavelength conversion and/or opto-electronic conversion equipment for intra-ring demand; this is an important cost consideration in any network upgrade plan. We call these rings *local rings*.

Local rings can then be connected by a special ring, which we call the *tertiary ring*, often called the federal or backbone ring in the literature. *Tertiary* is a legacy naming convention used by this operator to signify the highest level in the physical infrastructure. The tertiary ring facilitates inter-ring demand; wavelength converters are required where local rings connect to the tertiary ring.

So far the problem described is similar to the SRAP problem; we are identifying rings that can carry the estimated demand. However, as shown in Carroll and McGarraghy (2009b), in some real world instances, no SRAP solution is possible. We note that in the SRAP, demand pairs are grouped together so that there is as little inter-ring traffic as possible subject to capacity constraints on rings. The SRAP problem addresses how to design a ring based network by selecting the link capacities to install. In contrast, in the RSAP, we assess an existing network and the problem is to impose a ring topology over existing links at the logical level.

As an alternative, where no SRAP solution exists, we allow locations that have insufficient spare capacity or no possible physical route due to limitations of geography, to be connected to SHRs by spurs off the local rings. Spur nodes must be connected to a local ring by a single edge, i.e. we do not allow a chain of edges to connect spur nodes. We call this problem the *Ring Spur Assignment Problem* (RSAP). A solution to the RSAP is a set of disjoint bounded ring stars interconnected by a tertiary ring. Note, if a ring only solution exists, a ring-spur solution is not accepted.

We now state the RSAP formally as follows:

Instance:

- an undirected graph $G = (V, E)$ defined on a set V of nodes, labelled from 1 to n where $n = |V|$, $n \geq 6$ to exclude degenerate cases and a set of undirected edges E; the underlying set A of oriented arcs contains, for each edge $\{i, j\} \in E$, two arcs (i, j) and (j, i), one in each direction,

- a pair of positional co-ordinates for each node $i \in V$ that allow us to assign positional reference values for the end nodes of each edge $\{i,j\}$. These reference values signify the position of end nodes with respect to each other,
- a non-negative routing cost giving the cost of a link dependent on its length and capacity. Let $c_{ij} \geq 0$ be the cost coefficient of edge $\{i,j\} \in E$ for ring edges. If arc (i,j) is assigned as a spur, then it will be given a cost bc_{ij}, where b is a penalty weighting factor.
- An integer ring bound $R \geq 3$.

Problem: Find a minimum cost RSAP network $G' \subseteq G$ of disjoint bounded local ring spur partitions interconnected by a tertiary ring. The number of nodes on local rings is restricted by R, the Ring Bound. We wish to select $F \subseteq E$ constituting the RSAP solution topology network $N = (V, F)$. All nodes of G are either assigned to be part of a local ring or are connected by a spur edge to a node on a local ring. In addition, all local rings are interconnected by a tertiary ring. At least one node from each local ring represents that local ring on the tertiary ring. The tertiary ring forms a single simple cycle connecting the local ring representatives which may share edges but not spurs of the local rings.

SONET standards set the recommended maximum number of Add-Drop Multiplexers (ADMs) per ring at sixteen but, in this practical application, local rings are restricted to having no more than eight nodes. Since we wish to foster high resilience by having locations assigned to rings where possible, we assign a sufficiently high weight, b, to links that are spurs. We use a similar approach to Carroll and McGarraghy (2009a) to quantify a penalty weighting value in terms of other network parameters sufficient to ensure the creation of ring solutions if they exist. For simplicity, we set the coefficient of each arc $(i,j) \in A$, to be bc_{ij} in our objective function, i.e. the cost of using a spur edge is the network cost of that edge, c_{ij}, multiplied by the penalty weighting value of b for the network.

We summarise our use of some graph theory concepts that will be used to support the IP formulation described in Section 4.

The set of nodes on the ring of index k is denoted by $N(k)$ and the set of edges that form the ring of index k is denoted by $E(k)$. For simplicity, we refer to ring k, meaning the ring of index k. The set of nodes adjacent to node i is denoted by $\mathrm{adj}(i)$ and the cut of $S \subset V$ is denoted by $\delta(S) := \{\{i,j\} \in E : i \in S, j \notin S\}$, i.e. the set of edges having only one endpoint in S. We say a cut is *odd* if $|\delta(S)|$ is odd, *even* otherwise. Details of the decision variables of our formulations are given later. The support graph G_k associated with ring k is $G_k := (N(k), E(k))$ i.e. those edges for which the local ring edge decision variables are non-zero. The support graph G_T associated with the tertiary ring is $G_T := (N(T), E(T))$ i.e. those edges for which the tertiary ring edge decision variables are non-zero.

3 Survivable Network Design Problems

In this section, we summarise some literature on problems that have similarities to the RSAP.

The two-connected network with bounded ring (meshes), $2CNBR$ ($2CNBM$), design problem is to design a minimum cost network N such that N contains at least two node disjoint paths between every pair of nodes (two-connectivity constraints) and that each edge of N belongs to at least one cycle whose length is bounded by a given integer K (this gives us the ring constraints). The $2CNBM$ problem is described in detail by Fortz et al. (2000) and the $2CNBR$ in Fortz and Labbé (2002). The problem arises in topology design of backbone telecommunications networks, specifically networks with low connectivity requirements.

Fortz et al. (2000) describe their work on 52 zonal centres of the Belgian backbone network and note that modern fibre optic networks are characterised by high capacity, with high reliability per cable, leading to hierarchal sparse networks. Unfortunately sparse networks are more sensitive to damage so that designs need to ensure survivability. By imposing constraints on the maximum number of edges in a ring, the impact of failure can be limited.

Fortz and Labbé (2004) note that a minimum cost two-connected network is often a Hamiltonian cycle. This is undesirable in practice for a telecommunications network, as re-routed traffic would have to traverse all edges of the network. Ring Constraints, (cycles of length $\leq K$, an integer), that limit the region of influence of rerouted traffic in the event of a node/edge failure are introduced. This corresponds in practice to the limited number of hops allowed in a self-healing ring network.

They note that SHRs can be classified as unidirectional with dedicated spare capacity set aside for protection or bi-directional where working and spare capacity are combined on one fibre. Their work focuses on the selection of a set of edges that create a topology with a specified number of node (edge) disjoint paths between source and sink nodes where each edge belongs to a cycle of bounded length. Their objective is to find such a set of edges at minimum cost; we note that they have no requirement that the cycles be node (edge) disjoint. On the left of Figure 2 we show an example of a 2CNBR topology.

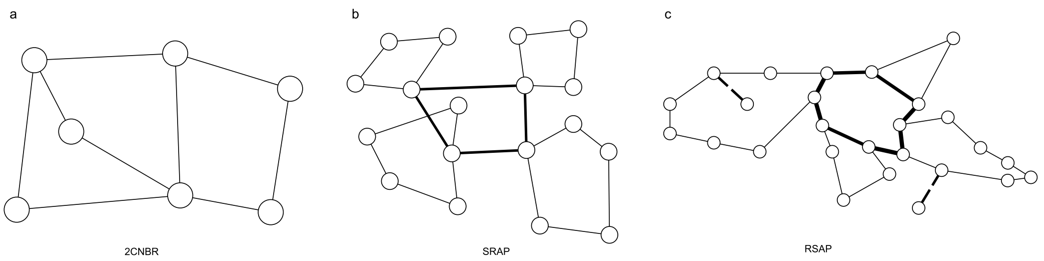


Figure 2: Topology examples: 2CNBR (left), SRAP (middle), RSAP (right)

The SONET Ring Assignment Problem (SRAP) is described in Goldschmidt et al. (2003). The SRAP is a design problem that seeks to identify which bi-directional SHR rings should be built to interconnect a set of customers and to identify a special ring called the *federal* ring that interconnects the customer rings. The authors minimise network costs by minimising the number of rings while satisfying customer demand, ring capacity and topology constraints. In the SRAP all Add Drop Multiplexers (ADMs) are the same size. Having solved the ring assignment problem, they use heuristics to solve TSP problems for each ring. We note that the resulting customer rings are disjoint. In the middle of Figure 2 we show an example of an SRAP topology. We see customer rings interconnected with a federal ring structure.

Goldschmidt et al. (2003) note that in practice, SDH Planning problems are decomposed into a sequence of manageable sub-problems. They describe the NP-Hard SDH Ring Loading problem (SRLP) as a low level problem of partitioning and routing traffic on a bidirectional SHR so as to minimise the ring capacity while satisfying all traffic demands. The SRLP can be tackled after the high level SRAP has been solved. They show this problem to be NP-hard. They note that using a commercial solver on the complete ILP is not practical and describe their approach in solving a series of related problems.

In practice, real world networks may contain some nodes that represent geographical locations that cannot facilitate two-connection. A town at the end of a peninsula or an area blocked by a mountain range may have a linear connection to the main backbone network. In other cases, a highly utilised network may not have sufficient spare capacity to allow the completion of a ring.

Labbé et al. (2004) describe the Ring Star problem (RSP) and an exact solution method. The Ring Star type of telecommunications network is used to connect terminals to concentrators by point-to-point links and is useful where not all nodes are required to be two connected. This type of topology is often used for Local Area Networks (LANs). A single node is a designated hub node that must be connected to the ring, all other nodes can either be assigned to the ring using concentrators or assigned to a node on the ring. The objective is to minimise the total cost of the ring and star assignments. The authors give an exact algorithm and a polyhedral analysis of the RSP. They also present computational results of a branch-and-cut implementation.

An extension to the RSP is the Capacitated m-Ring-Star Problem (CmRSP) described in Baldacci et al. (2007) in relation to the design of an optical network in an urban area, in this instance in an Italian city. The problem consists of finding m node disjoint ring stars that visit a central depot, the number of customers allocated or visited in a ring cannot be greater than some capacity Q. The objective is to minimize the total ring star assignment costs. The authors give two ILP formulations, a two index edge and a two commodity flow formulation and they develop a branch-and-cut algorithm. They assess the effectiveness of the two formulations in the branch-and-cut framework and compare the respective LP relaxations of the models. The LP relaxations will provide solutions of equal cost. The authors report that their algorithms can handle instances up to about 100 nodes. In Naji-Azimi et al. (2010), the authors give a heuristic method for solving larger CmRSP problem instances. While in Hoshino and de Souza (2009), the authors give an exact branch-and-cut-and-price algorithm for the CmRSP which they report outperforms the branch-and-cut implementation of Baldacci et al. (2007).

A solution to the RSAP is a set of interconnected ring stars. On the right of Figure 2 we show an example of an RSAP topology. Local rings are shown as light lines, spurs as dashed lines and the tertiary ring in heavy lines. For clarity, network edges that are not part of the solution topology are not shown.

RSAP ring solutions are not necessarily 2CNBR solutions since we only require local rings to have at least one representative node on the tertiary ring. RSAP ring solutions do not necessarily have two node disjoint paths between every pair of nodes.

In Carroll and McGarraghy (2009b) the authors argue that since SRAP is a special case of RSAP, it follows that RSAP is also NP-Hard. However, the SRAP differs from the RSAP in a crucial aspect. The SRAP problem addresses how to design a ring based network by selecting the link capacities to install. It focuses on clustering nodes that can be placed together on a ring subject to capacity restrictions on the volume of intra-ring traffic on each ring, and the total volume of inter-ring traffic carried by the tertiary ring. While ring solutions to the RSAP satisfy the SRAP topology requirements, they do not necessarily satisfy the SRAP capacity restrictions. In the RSAP, we assess an existing network and the problem is to impose a ring topology over existing links at the logical level. In this sense, the RASP is a complement of the SRAP. We include a more detailed NP-Hardness proof of the RSAP in the Appendix.

4 Ring Representatives Formulation

Having shown that the RSAP is NP-hard, we seek to implement an efficient polyhedral algorithm to solve RSAP instances. In Carroll et al. (2011) we describe two IP formulations for the RSAP, the double indexed formulation described here and a triple indexed formulation. We compare the two formulations and note that both formulations have $O(n^3)$ variables but that the triple indexed formulation tends asymptotically towards having four times as many variables as the double indexed formulation.

We note that the constraints in both formulations are mostly similar with $O(n^3)$ constraints in both formulations. The total number of constraints is higher in the triple indexed formulation. In Carroll et al. (2011), we show that there is no empirical difference in the LP bound of the two formulations and show that the triple indexed formulation is at least as strong as double indexed formulation. We leave the question of the equivalence of the LP relaxations open. For practical purposes, we work in this paper with the more compact double indexed formulation.

We now give the details of our double indexed formulation for the RSAP. We first concentrate on the local rings and come back to the tertiary ring later. The formulation uses variables to assign nodes and edges to rings, and a third set of variables for spurs. The main problem with such a choice of variables is that it induces a lot of symmetry in the formulation, as rings are interchangeable.

One way to break this inherent symmetry is to use the same kind of symmetry breaking that was used e.g. in Campêlo et al. (2008) for graph colouring problems. Each ring is identified by a representative node belonging to the ring. If we decide that the representative node is the node with smallest index in the ring, the symmetry is completely broken. Note that since each ring is composed of at least 3 nodes, only nodes $k \in K := \{1, \ldots, n-2\}$ can be ring representatives.

Let x_{ijk} be a binary variable equal to 1 if and only if edge $\{i, j\}$ appears on ring k, and equal to 0 otherwise; i.e. both i and j are assigned to the same ring k. Also, we implicitly assume that $i < j$ when writing an edge $\{i, j\}$. For each arc $(i, j) \in A$, let y_{ij} be a binary variable equal to 1 if vertex i is assigned to vertex j as a spur; we set $y_{ii} = 1$ for any vertex i that is on a ring. Let z_{ik} be a binary variable equal to 1 if vertex i is assigned to ring k as a ring node.

The tertiary ring variables are α_{ik}, a binary variable equal to 1 if and only if node i connects ring k to the tertiary ring, 0 otherwise. β_{ij} is a binary variable equal to 1 if and only if edge $\{i, j\}$ appears on the tertiary ring, and equal to 0 otherwise.

With these sets of variables, the initial relaxation of the RSAP, omitting subtour elimination constraints, can be formulated as follows:

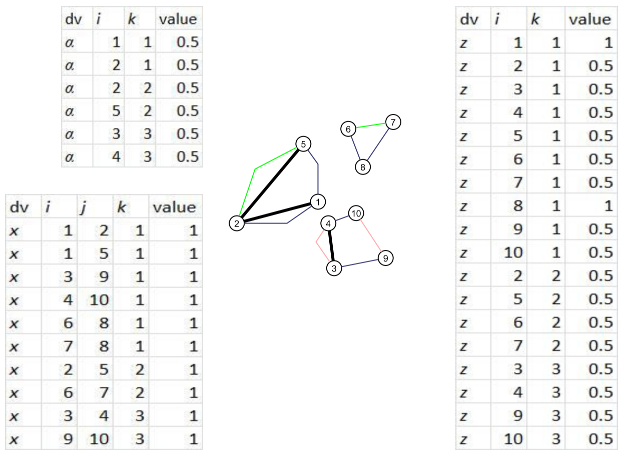
$$\begin{aligned}
\min \quad & \sum_{\{i,j\}\in E}\sum_{k\in K} c_{ij}x_{ijk} + \sum_{(i,j)\in A} bc_{ij}y_{ij} + \sum_{\{i,j\}\in E} c_{ij}\beta_{ij} && && (1)\\
\text{s.t.} \quad & \sum_{k=1}^{i} z_{ik} = y_{ii} && \forall\, i \in V && (2)\\
& \sum_{j\in \mathrm{adj}(i)} y_{ij} + y_{ii} = 1 && \forall\, i \in V && (3)\\
& \sum_{j\in \mathrm{adj}(i), j>i} x_{ijk} + \sum_{j\in \mathrm{adj}(i), j<i} x_{jik} = 2z_{ik} && \forall\, i \in V, k\in K && (4)\\
& \sum_{l\in \mathrm{adj}(i), l>i, l\neq j} x_{ilk} + \sum_{l\in \mathrm{adj}(i), l<i} x_{lik} \geq x_{ijk} && \forall\, \{i,j\} \in E, k\in K && (5)\\
& \sum_{l\in \mathrm{adj}(j), l<j, l\neq i} x_{ljk} + \sum_{l\in \mathrm{adj}(j), l>j} x_{jlk} \geq x_{ijk} && \forall\, \{i,j\} \in E, k\in K && (6)\\
& \sum_{i\in V, i>k} z_{ik} \leq (R-1)z_{kk} && \forall\, k\in K && (7)\\
& z_{ik} \leq z_{kk} && \forall\, i \in V, k\in K && (8)\\
& \sum_{k\in K} x_{ijk} + y_{ji} \leq y_{ii} && \forall\, i \in V, j\in adj(i) && (9)\\
& \sum_{k\in K} x_{ijk} + y_{ij} + y_{ji} \leq 1 && \forall\, \{i,j\} \in E && (10)\\
& \sum_{i\in V, i\geq k} \alpha_{ik} \geq z_{kk} && \forall\, k\in K && (11)\\
& \alpha_{ik} \leq z_{ik} && \forall\, i \in V, k\in K && (12)\\
& \sum_{j\in V, i<j} \beta_{ij} + \sum_{j\in V, j<i} \beta_{ji} = 2\sum_{k\in K}\alpha_{ik} && \forall\, i \in V && (13)\\
& \sum_{l\in \mathrm{adj}(i), l>i, l\neq j} \beta_{il} + \sum_{l\in \mathrm{adj}(i), l<i} \beta_{li} \geq \beta_{ij} && \forall\, \{i,j\} \in E && (14)\\
& \sum_{l\in \mathrm{adj}(j), l<j, l\neq i} \beta_{lj} + \sum_{l\in \mathrm{adj}(j), l>j} \beta_{jl} \geq \beta_{ij} && \forall\, \{i,j\} \in E && (15)\\
& \beta_{ij} + y_{ij} + y_{ji} \leq 1 && \forall\, \{i,j\} \in E && (16)\\
& x_{ijk}, y_{ij}, z_{ik}, \alpha_{ik}, \beta_{ij} \in \{0,1\} && \forall\, \{i,j\} \in E, k\in K && (17)
\end{aligned}$$

We use constraints (2) to link the y and z variables. Constraints (3) ensure every node is assigned. Constraints (4) say that every node i on ring k has exactly two incident local ring edges on ring k. We note that variables y_{ii} are not needed but are included to simplify the understanding of the model.

The connectivity constraints (4) allow the edges incident with the local ring node to belong to different rings in the LP relaxation. An example is shown in Figure 3. We need to ensure that each local ring edge on ring k is coincident with two local ring edges on the same ring. We dis-aggregate this requirement and look separately at the head and tail of each local ring edge. Constraints (5) and (6) are dis-aggregated edge connectivity constraints and ensure that the head (tail) of a ring edge on ring k each have an incident ring edge on ring k. This form of connectivity constraints ensure there are at least three edges (three nodes) on each local ring.

Rings are restricted to having no more than R nodes by the Ring Bound Constraints (7), if ring k is active, node k must be active and no more than $R-1$ other nodes. Constraints (8) ensure node i is assigned to ring k if and only if node k is assigned to ring k. Constraints (9) say that j is assigned to i as a spur or adjacent to i on a ring if and only if i is a ring node. Constraints (10), which are only needed in the relaxed LP, limit the usage of any edge (or its corresponding arcs) to 1.

The next set of constraints define the tertiary ring. Constraints (11) ensure every active local ring k is represented on the tertiary ring by at least one of its nodes. Constraints (12) link the α to the z



dv	i	k	value
α	1	1	0.5
α	2	1	0.5
α	2	2	0.5
α	5	2	0.5
α	3	3	0.5
α	4	3	0.5

dv	i	j	k	value
x	1	2	1	1
x	1	5	1	1
x	3	9	1	1
x	4	10	1	1
x	6	8	1	1
x	7	8	1	1
x	2	5	2	1
x	6	7	2	1
x	3	4	3	1
x	9	10	3	1

dv	i	k	value
z	1	1	1
z	2	1	0.5
z	3	1	0.5
z	4	1	0.5
z	5	1	0.5
z	6	1	0.5
z	7	1	0.5
z	8	1	1
z	9	1	0.5
z	10	1	0.5
z	2	2	0.5
z	5	2	0.5
z	6	2	0.5
z	7	2	0.5
z	3	3	0.5
z	4	3	0.5
z	9	3	0.5
z	10	3	0.5

Figure 3: $F2$ invalid IP solution

variables and ensure that node i only represents ring k if it is active on ring k. Constraints (13) ensure that nodes on the tertiary ring are two connected while constraints (14) and (15) are similar to the local ring edge dis-aggregated connectivity constraints (5) and (6). Again, these connectivity constraints ensure there are at least three edges on the tertiary ring. Constraints (16) ensure that no spur edge is used as a tertiary ring edge (this ensures the robustness of the tertiary ring) while constraints (17) are the binary integer constraints.

The z_{ik} and α_{ik} decision variables act as ancillary variables to enforce the RSAP topology requirements and link the local to the tertiary ring.

We note that the formulation thus far allows subtours on both the local and tertiary rings. In Section 4.1 we describe Subtour Elimination Constraints (SECs) that are added as required in the branch-and-cut algorithm described in Section 5.

4.1 Subtour Elimination Constraints

As noted, we need to add SECs to ensure a valid formulation. The traditional SECs force at least one edge of a subtour to be dropped. In the case of the RSAP, we can strengthen the traditional SEC. Recall that decision variable z_{kk} is equal to 1 if vertex k is assigned to ring k where k is the ring representative. In Carroll et al. (2011) the authors describe modified SECs (M-SECs). Suppose a subtour is detected on ring k, defined by a subset of nodes S, with node $k \notin S$. No nodes can be assigned to ring k if the ring representative node k is not active, i.e. if $z_{kk} = 0$. Then the following modified version of the traditional SEC (M-SEC) is valid:

$$\sum_{\{i,j\}\in E(S)} x_{ijk} \leq (|S|-1)z_{kk} \quad \forall\, S \subset V, k \notin S \quad \text{M-SECs} \tag{18}$$

Eq. (18) says that the set of edges of the subset S cannot all be on ring k. Indeed, such a constraint can be added for all $k \leq \text{minindex}(S)$, the minimum index of the subset S. For example, if the edges of the subset $S := \{5, 6, 7\}$ form a local ring, they must be on ring 5 and cannot be on ring 4 or lower. Also note, since constraints (5) and (6) force the number of nodes on an active local ring to be at least three, we consider subtours on $S \subset V$ where $|S| \geq 3$. This form of subtour elimination was used in Carroll et al. (2011).

4.1.1 Generalised Subtour Elimination constraints

Another way to eliminate subtours is to use the Generalised Subtour Elimination constraints (G-SECs) which are applicable to vehicle routing problems. G-SECs are described in Laporte (1986) and are used to ensure that no vehicle route is cut off from the depot. In the case of the local rings of the RSAP, we wish to ensure that no node on local ring k is cut off from the ring representative k using a modified G-SEC:

$$\sum_{i,j \in E(S)} x_{ijk} \leq \sum_{i \in S \setminus \{l\}} z_{ik} \quad \forall \quad k < minindex(S), S \subset V \quad k \notin S, l \in S \quad \text{G-SECs} \tag{19}$$

In Eq. (19) S is a subtour of size at least three, $|S| \geq 3$, the inequality says that at most $|S| - 1$ nodes can be on ring k when $k \notin S$, otherwise the nodes of S on ring k are cut off from node k. As in the modified SECs, this is true for all $k < minindex(S)$. The G-SECs are are a stronger form of subtour elimination for multiple ring problems than the M-SECs proposed in Carroll et al. (2011) as evidenced in our computational results below.

4.1.2 Tertiary ring SECs

SECs are also required for subtours on the tertiary ring. If a subtour consisting of subsets S_a and S_b is detected, since it is not known which local ring edges will be nonzero in the optimal solution, it is possible that the optimal tertiary ring could consist of all edges of one tertiary ring subtour, either S_a or S_b. However, not all edges of both subtours could be non-zero in the optimal solution. An aggregated version of the usual SECs is SEC-T:

$$\sum_{\{i,j\} \in E(S_a) \cup E(S_b)} \beta_{ij} \leq |S_a \cup S_b| - 2 \quad \forall \text{ Tertiary Subtours } S_a \text{ and } S_b \quad \text{SEC-T} \tag{20}$$

4.2 Additional Valid Inequalities

We also identify some additional inequalities to improve the LP bound and performance of the branch-and-cut algorithm.

4.2.1 2-matching constraints

Our formulation also gives rise to fractional results with invalid 2-matchings, i.e. cuts across an odd number of edges. As with TSP-like problems, the cut across rings must be even. $H \subset V$ is called a handle and $T \subset E$ is an odd set of disjoint edges with exactly one end in H, known as teeth. By taking linear combinations using weights of 0.5 on the connectivity constraints, Eq. (4) for all nodes in the handle and weights of 0.5 on the teeth edge trivial upper bounds, $x_{ijk} \leq 1$, and then rounding we get 2-matching constraints given by:

$$\sum_{\{i,j\} \in E(H)} x_{ijk} + \sum_{\{i,j\} \in T} x_{ijk} \leq \sum_{i \in H} z_{ik} + \left\lfloor \frac{T}{2} \right\rfloor \qquad \text{2-matching on ring } k \tag{21}$$

Nemhauser and Wolsey (1988) note that 2-matching constraints are dominated by the subtour inequalities when the number of teeth is one, hence $|T| \geq 3$ in traditional TSP-like problems. However, we note that in our formulation with the use of the dis-aggregated connectivity constraints, an odd cut with a single tooth may arise in a fractional solution hence $|T| \geq 1$ and odd in our modified 2-matching constraints.

We mention 2-matchings on the tertiary ring. As before, a violated 2-matching on the tertiary ring can exist on only one tooth, we use a modified 2-matching inequality on the tertiary ring of the following form:

$$\sum_{\{i,j\} \in E(H)} \beta_{ij} + \sum_{\{i,j\} \in T} \beta_{ij} \leq \sum_{i \in H} \sum_{k \in K} \alpha_{ik} + \left\lfloor \frac{T}{2} \right\rfloor \qquad \text{2-matching on Tertiary ring} \tag{22}$$

An example of a violated 2-matching on the tertiary ring is given in Figure 4. Fractional tertiary ring edges are shown in heavy lines, $H = \{4, 5, 6\}$, there is one tooth $\{3, 4\}$, the weights of $E(H) = 0.3$ while the tooth edge weight is 0.6. The handle nodes all partially represent ring 3 on the tertiary ring, $\alpha_{43} = 0.5, \alpha_{53} = 0.3, \alpha_{63} = 0.3$, while node 4 also partially represents ring 1, $\alpha_{41} = 0.1$.

4.2.2 Infeasible Ring constraints

The edges of a fractional solution may form an infeasible ring, i.e. the ring bound constraints are satisfied but there are too many nodes on the local ring. An example is shown in Figure 4. If the number of edges in a simple cycle on the fractional ring edges is greater than the ring bound, we have found a cycle that is too big. We call these infeasible rings *ghosts*. Ghost rings can be eliminated by observing that at least two edges of a cycle that exceeds the ring bound must be dropped, the remaining edges can be used to form valid rings.

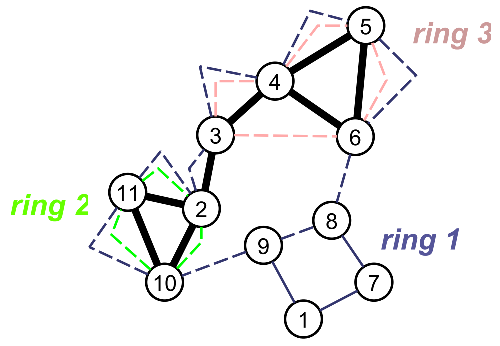


Figure 4: Fractional solution

In the example shown in Figure 4, all $y_{ii} = 1$, $x_{ijk} \in \{0.5, 1\}$ for the local ring edges shown in colour. The local ring assignment variables $z_{ik} \in \{0.5, 1\}$. Some nodes are partially assigned to two rings, e.g., node 2 is equally assigned to rings 1 and 2 with $z_{2,1} = z_{2,2} = 0.5$. Other nodes are completely assigned to a ring, e.g., node 1 is assigned to ring 1 with $z_{1,1} = 1$. Tracing a cycle around the 11 outer edges of the graph, we see than the sum of the local ring edges over all rings adds to 9.5. We define a ghost ring G_r as a simple cycle on the fractional ring edges of the LP solution with more than the ring bound number of edges.

Standard SECs break a subtour by forcing at least one edge of the subtour to be dropped. In the case of bounded rings, dropping one edge would leave a path, in our example of 10 edges. These 10 edges forming a path cannot form a bounded ring. We observe that $\lceil |G_r|/R \rceil$ is a lower bound on the number of rings that would be required to cover the nodes of a ghost ring. Thus, we obtain a stronger constraint by forcing at least $\lceil |G_r|/R \rceil$ edges of G_r to be dropped, as follows:

$$\sum_{\{i,j\} \in E(G_r), k \in K} x_{ijk} \leq |G_r| - \lceil |G_r|/R \rceil \qquad \forall \text{ Ghost ring } G_r \tag{23}$$

4.2.3 Lower ring bounds

Finally, although our formulation ensures there are at least two local rings in a solution, for small problems, it may be useful to add a constraint that forces the sum of the ring assignment variables to be at least two:

$$\sum_{i \in V, k \in K} z_{ik} \geq 2 \qquad \text{local ring lower bound} \tag{24}$$

Similarly, although constraints Eq. (14) and Eq. (15) ensure there are at least three edges on the tertiary ring, the sum of the value on the tertiary ring fractional edge variables may be less than three. If necessary, we can add a constraint to ensure the sum is greater than three.

$$\sum_{\{i,j\} \in E} \beta_{ij} \geq 3 \qquad \text{Tertiary ring lower bound} \tag{25}$$

5 A Branch-and-Cut Algorithm

This section describes the branch-and-cut parameters of our algorithm and the min cut algorithm used in our separation procedures.

5.1 Cut Separation

Finding the minimum cut for a graph has been described as the bottleneck in the polyhedral approach by Applegate et al. (2003). Since the graphs we consider are undirected and we only need to identify violated subtours on any local ring (or on the tertiary ring) rather than generate all min-cuts in the solution, we use a modified version of the min cut algorithm of Stoer and Wagner (1997). The authors describe a deterministic non-flow based min cut algorithm based on the principle of maximal adjacency. It takes as input a graph adjacency matrix and returns the min cut value. It runs through a number of iterations, finding a cut at each iteration, the smallest of which is the min cut of the graph.

We modified the Stoer Wagner algorithm to run on the support graph for a local ring (or tertiary ring) and track the cuts at each iteration using data structures to store the number of edges in a cut, the cut value and the resulting partitions. We interrupt the algorithm if a zero value min cut is detected. Otherwise, the Stoer Wagner algorithm runs to completion and in both cases we have access to the stored cuts. Note that the Stoer Wagner algorithm finds a number of cuts, we note that unless an odd cut detected is the min cut of the graph, it is not necessarily the minimal odd cut but any odd cut found on the fractional ring edges is a violation of some sort and is worthy of investigation. We check if any odd cut gives a violated 2-matching. A subtour is separated when the min cut of the support graph is zero and either a local ring M-SEC or G-SEC is added. In the case of the tertiary ring; an SEC-T is added.

Since there are potentially an exponential numbers of both feasible and infeasible rings on a fractional ring solution, we use a simple heuristic to detect and add GECs. For each fractional ring k in the LP solution, if the number of nodes on the ring exceeds the ring bound, we create the support graph G_k and select any 2-connected node as the start node and attempt to trace a path that returns to the start node to form a simple cycle along the edges of G_k. At any intermediate node of degree greater than two, we select the next node as the one that makes the biggest angle on our current path using the nodes' positional co-ordinates and continue tracing the path. If the path returns to the start node using more than the ring bound number of edges, we have detected an infeasible ring (ghost) and add the GEC. If we failed to form a cycle, no GEC has been detected on this fractional ring and we proceed to check the next ring with a fractional solution.

5.2 The algorithm

The branch-and-cut algorithm models the initial problem using constraints Eq. (2) to Eq. (17) and the lower bound inequalities Eq. (24) and Eq. (25) for small problems.

We solve this initial problem and separate and add cuts at the root node to improve the LP bound. If the solution is still fractional we call a branch-and-cut search. We turn off the solver presolve, automatic cuts and default heuristics. We synchronise the parallel threads and set our own cut directives. We choose between the Modified or Generalised Subtour Elimination Constraints, M-SECs or G-SECs. These cuts are separated at all nodes of the branch-and-cut tree. We add in the other additional valid inequalities specifying at what stages in the cutting plane algorithm they are separated and added. In Section 6 we report computational results using different combinations of the additional inequalities.

The methodology we have used is to add cuts to the branch-and-cut node as global cuts after the call to the separation procedure and clear the cut pool. This means that the cut applies to the node and all its descendants. It also means that the cut pool is kept small. In choosing this methodology it is possible that the same cut will be identified at some other branch of the branch-and-cut tree. We have to weigh up the cost of duplicating the separation procedure against the cost of maintaining a larger cut pool and checking whether cuts in the pool should be applied to the current node.

We save all integer solutions found during the branch-and-cut search for possible later scenario analysis.

6 Computational Results

Algorithms were implemented with code written in ANSI C, using the Xpress-MP suite 7.2 with Xpress-BCL version 4.4.0 Builder Component library (BCL) routines and Xpress-Optimizer 22.01, and run on a 32 bit Toshiba Satellite Pro with Intel Dual Core Pentium 1.86GHz processors and 2 GB of RAM under Windows Vista allowing a maximum of three hours computation time.

The release of the Xpress 7.2 suite (May 2011) includes a facility to exploit parallel processing in the branch-and-bound tree. This is achieved by synchronising the BCL and optimiser problems at the start and end of each callback, access to the BCL problem is locked to the particular thread in between these two function calls.

The test data used was SNDlib (Orlowski et al., 2010), since it provides many real world problem instances with both a network model and positional co-ordinates for each node. In earlier testing three problems, *nobel-eu*, *janos-us-ca* and *zib54*, were integer infeasible for RSAP solutions, and so were omitted from further testing. These networks have a small number of nodes of very high degree making them unsuitable for the RSAP topology. We give computational results for the remaining problems using different combinations of the additional cuts to compare their effectiveness. If the branch-and-cut algorithm has not completed within our three hour time limit, we report the best integer solution (if any).

We also include results comparing the impact of varying the ring bound, using values of four and twelve compared to the preferred value used in practice of eight.

Table 1 shows a summary of the SNDLib problem instances using a ring bound of eight. From left to right we show:

1. the problem name,
2. the problem size in terms of the number of nodes, n, and edges, m,
3. the penalty weighting b calculated for the problem instance,
4. the LP relaxation objective function value,
5. the computational time in seconds to obtain the optimal LP relaxation solution.

problem name	size n, m	penalty weight b	LP objective function value	LP time (s)
dfn-bwin	10,45	3	105,686	0.024
pdh	11,34	4	1313202	0.032
di-yuan	11,42	16	*412,300	0.030
dfn-gwin	11,47	6	15,392	0.030
polska	12,18	3	3,187	0.026
atlanta	15,22	17	38,476,500	0.047
newyork	16,49	7	1,415,120	0.069
ta1	24,51	7	8,712,416	0.108
france	25,45	10	14,600	0.134
janos-us	26,42	4	13,381	0.394
norway	27,51	6	464,198	0.536
sun	27,51	15	466	0.498
cost266	37,57	13	7,629,382	0.766
giul39	39,86	6	736	4.630
pioro40	40,89	9	6,235	1.186
germany	50,88	7	348,920	4.353
ta2	65,108	20	32,971,118	10.852

Table 1: Problem and LP summary, $R = 8$, * indicates an integer solution

The problem instances vary in size from the 10 node, 45 edge *dfn-bwin* instance to the 65 node, 108 edge *ta2* instance. We see that the LP relaxation of the 11 node *di-yuan* problem is integer. It has no subtours, so is the optimal RSAP solution. The LP relaxations for all other problem instances are fractional and contain subtours as well as violated 2-matchings and the infeasible rings described in Section 4.2.2.

In Table 2 we show a summary of results of the branch-and-cut algorithm using various cut combinations for the problem instances which are solved to optimality. These problems have 37 or fewer nodes. We see that optimal solutions for small problems are found in a short time. We omit *di-yuan* from this table since its linear relaxation is optimal. From left to right we show:

1. the problem name,
2. the cut combination,
3. the optimal IP objective function value,
4. the computational time in seconds to obtain the optimal IP solution,
5. the number of M-SECs on local rings,
6. the number of G-SECs on local rings,
7. the number of 2-matching cuts on local rings,
8. the number of GECs,
9. the number of SECs on the tertiary ring,
10. the number of 2-matching cuts on the tertiary ring,
11. the best bound on termination,
12. the number of nodes in the branch-and-cut tree.

The cut combinations reported are as follows

1. M-SECs: Modified SECs on local rings (L), Eq. (18) plus SECs on the tertiary ring (T), Eq. (20),
2. G-SECs: Generalised SECs on local rings (L), Eq. (19) plus SECs on the tertiary ring (T), Eq. (20),
3. G-SECs+2M: cut combination 2 plus 2-matching constraints on the local and tertiary rings, Eq. (21) and Eq. (22)
4. G-SECs+2M+GEC: cut combination 3 plus GECs, Eq. (23)
5. G-SECs+2M+GEC-P: cut combination 4 with prioritisation on z_{kk} variables.

Table 2: Branch-and-cut summary, small problems $R = 8$

problem name	Cut combination	IP obj value	IP (s)	M-SEC L	G-SEC L	2M L	GEC	SEC T	2M T	bestbound	Nodes in tree
dfn-bwin	M-SEC	105,810	0.47	16	0	0	0	0	0	105,810	38
dfn-bwin	G-SEC	105,810	0.48	0	16	0	0	0	0	105,810	14
dfn-bwin	G-SEC+2M	105,810	0.49	0	14	2	0	0	2	105,810	18
dfn-bwin	G-SEC+2M+GEC	105,810	0.46	0	14	2	0	0	2	105,810	18
dfn-bwin	G-SEC+2M+GEC-P	105,810	0.35	0	14	0	0	0	10	105,810	15
pdh	M-SEC	1,355,139	0.75	16	0	0	0	0	0	1,355,139	58
pdh	G-SEC	1,355,139	0.75	0	24	0	0	0	0	1,355,139	73
pdh	G-SEC+2M	1,355,139	0.57	0	15	4	0	0	4	1,355,139	18
pdh	G-SEC+2M+GEC	1,355,139	0.56	0	16	5	1	0	4	1,355,139	17
pdh	G-SEC+2M+GEC-P	1,355,139	0.62	0	13	2	1	0	4	1,355,139	24
dfn-gwin	M-SEC	15,724	0.33	3	0	0	0	0	0	15,724	3
dfn-gwin	G-SEC	15,724	0.46	0	3	0	0	0	0	15,724	10
dfn-gwin	G-SEC+2M	15,724	0.50	0	3	1	0	0	0	15,724	10
dfn-gwin	G-SEC+2M+GEC	15,724	0.45	0	3	1	0	0	0	15,724	10
dfn-gwin	G-SEC+2M+GEC-P	15,724	0.43	0	3	1	0	0	0	15,724	10
polska	M-SEC	3,487	0.37	3	0	0	0	0	0	3,487	14
polska	G-SEC	3,487	0.34	0	4	0	0	0	0	3,487	10
polska	G-SEC+2M	3,487	0.46	0	7	1	0	0	2	3,487	13
polska	G-SEC+2M+GEC	3,487	0.45	0	7	1	0	0	2	3,487	13
polska	G-SEC+2M+GEC-P	3,487	0.39	0	15	1	0	0	2	3,487	15
atlanta	M-SEC	55,452,500	1.18	18	0	0	0	11	0	55,452,500	117
atlanta	G-SEC	55,452,500	0.63	0	26	0	0	7	0	55,452,500	32
atlanta	G-SEC+2M	55,452,500	0.80	0	26	1	0	7	4	55,452,500	34
atlanta	G-SEC+2M+GEC	55,452,500	0.72	0	26	1	1	7	4	55,452,500	33
atlanta	G-SEC+2M+GEC-P	55,452,500	0.52	0	11	2	0	7	4	55,452,500	28
newyork	M-SEC	1,512,400	4.20	229	0	0	0	4	0	1,512,400	257
newyork	G-SEC	1,512,400	3.90	0	197	0	0	0	0	1,512,400	183
newyork	G-SEC+2M	1,512,400	5.69	0	329	89	0	24	56	1,512,400	368
newyork	G-SEC+2M+GEC	1,512,400	5.40	0	254	88	2	15	42	1,512,400	259
newyork	G-SEC+2M+GEC-P	1,512,400	2.75	0	224	41	1	0	24	1,512,400	170

Continued on next page

Table 2 – *Continued from previous page*

problem name	Cut combination	IP obj value	IP (s)	M-SEC L	G-SEC L	2M L	GEC	SEC T	2M T	bestbound	Nodes in tree
ta1	M-SEC	11,410,169	18.79	333	0	0	0	148	0	11,410,169	1,300
ta1	G-SEC	11,410,169	25.05	0	1675	0	0	164	0	11,410,169	1,297
ta1	G-SEC+2M	11,410,169	28.57	0	1674	331	0	165	288	11,410,169	1,344
ta1	G-SEC+2M+GEC	11,410,169	27.63	0	1,572	314	1	164	300	11,410,169	1,318
ta1	G-SEC+2M+GEC-P	11,410,169	25.23	0	2,203	519	3	146	266	11,410,169	1,329
france	M-SEC	20,800	70.32	4,573	0	0	0	497	0	20,800	5,438
france	G-SEC	20,800	20.74	0	1,787	0	0	496	0	20,800	1,646
france	G-SEC+2M	20,800	13.75	0	1,010	19	0	384	206	20,800	1,130
france	G-SEC+2M+GEC	20,800	13.92	0	1,010	19	0	384	206	20,800	1,130
france	G-SEC+2M+GEC-P	20,800	21.77	0	3,993	102	0	556	148	20,800	1,776
janos-us	M-SEC	16,672	66.20	1,981	0	0	0	182	0	16,672	3,474
janos-us	G-SEC	16,672	24.28	0	1,041	0	0	4	0	16,672	483
janos-us	G-SEC+2M	16,672	16.74	0	491	30	0	17	30	16,672	291
janos-us	G-SEC+2M+GEC	16,672	17.29	0	491	30	0	17	30	16,672	291
janos-us	G-SEC+2M+GEC-P	16,672	13.87	0	589	30	0	15	90	16,672	400
norway	M-SEC	596,070	59.45	2,688	0	0	0	35	0	596,070	1,450
norway	G-SEC	596,070	49.28	0	3,505	0	0	14	0	596,070	784
norway	G-SEC+2M	596,070	47.37	0	2,851	298	0	60	114	596,070	798
norway	G-SEC+2M+GEC	596,070	57.02	0	3,803	375	8	72	216	596,070	858
norway	G-SEC+2M+GEC-P	596,070	19.39	0	1,511	149	1	30	120	596,070	527
sun	M-SEC	695	31.91	887	0	0	0	14	0	695	675
sun	G-SEC	695	35.11	0	2,136	0	0	39	0	695	784
sun	G-SEC+2M	695	57.05	0	4,070	300	0	86	168	695	931
sun	G-SEC+2M+GEC	695	40.57	0	3,702	334	2	55	124	695	813
sun	G-SEC+2M+GEC-P	695	22.93	0	2,207	226	4	28	104	695	373
cost266	M-SEC	12,262,590	463.95	3,828	0	0	0	2,532	0	12,262,332	25,272
cost266	G-SEC	12,262,590	197.22	0	3,884	0	0	701	0	12,262,590	5,160
cost266	G-SEC+2M	12,262,590	98.39	0	2,498	132	0	432	132	12,262,590	3,062
cost266	G-SEC+2M+GEC	12,262,590	90.00	0	2,334	134	3	328	106	12,262,320	2,720
cost266	G-SEC+2M+GEC-P	12,262,590	139.29	0	4,324	315	12	468	184	12,262,590	4,253

In Table 2 we see that for most of the problem instances that are solved to optimality, there is some benefit derived from the stronger G-SECs. However, on the slightly larger problems, such as the 37 node *cost266* instance, we see a significant reduction in both the number of nodes in the branch-and-cut and the computational time. Only 5,160 nodes are processed in 197.22 seconds in order to find the optimal solution using the G-SECs, compared to 25,272 in 463.95 seconds using the M-SECs.

The addition of the 2-matching constraints decreases the computational time in some instances. Recall that we do not explicitly separate 2-matching constraints but check for any odd cut detected while separating the SECs. It is possible that an explicit 2-matching separation procedure would identify additional 2-matching violations that could improve the overall performance.

In most cases, the addition of the GECs, where a GEC is separated, causes a decrease in the computational time. For example, the run time for *cost266* is reduced to 90.00 seconds by the addition of 3 GECs. Where no GEC is separated the additional time spent trying to separate a GEC simply adds to the overall run time. This occurs in the case of *janos-us* where the run time is increased from 16.74 to 17.29 seconds with no GECs separated. Recall that we use a heuristic to separate GECs at the root and every fifth node of the branch-and-cut search.

Recall also that the z_{ik} and α_{ik} ancillary decision variable are used to enforce the RSAP topology requirements. We tested prioritising on these variables as they effectively control the local and tertiary ring. We found the best results were achieved by prioritising on the z_{kk} variables and these results are also shown in Table 2.

In most cases, by prioritising on the z_{kk} variables and using all the additional cuts, we decrease the run time. We see for example, that *sun* is solved to optimality in only 22.93 seconds using this approach.

In Table 3 we show a summary of results of the branch-and-cut algorithm for the bigger problem instances which are not solved to optimality. We add an additional column to show the gap between the best integer solution and the best bound.

For the bigger problem instances, we see that the G-SECs outperform the M-SECS, the gap in all such cases is reduced. In these cases, more G-SECs are separated (compared to M-SECs) and the overall number of nodes in the branch-and-cut search tree are reduced. In some of these instances, we encounter a limitation on the hardware memory availability. In the case of *pioro40*, the search is interrupted before the three hour time limit because the available memory is fully utilised. All such results are flagged with an * in Table 3.

The benefit of using of the 2-matching constraints on these bigger problems is less clear cut. Only in the case of *germany* do we see a reduction in the gap to 2.3%. Nor do the addition of the GECs close the gap. However, the combination of all the additional cuts and prioritising on the z_{kk} variables yields the best set of results for these problems.

Table 3: Branch-and-cut summary, large problems, $R = 8$, * indicates *out-of-memory*

problem name	Cut combination	IP obj value	IP (s)	M-SEC L	G-SEC L	2M L	GEC	SEC T	2M T	bestbound	Nodes in tree	gap %
giul39	M-SEC	1,051	10,801.94	136,181	0	0	0	941	0	813	37,933	22.69
giul39	G-SEC	995	10,807.13	0	165,337	0	0	302	0	827	16,533	16.86
giul39	G-SEC+2M	1,112	10,806.54	0	132,492	10,706	0	289	1,974	796	10,128	28.42
giul39	G-SEC+2M+GEC	1,050	10,810.81	0	133,789	11,442	35	311	2,286	800	10,808	23.81
giul39	G-SEC+2M+GEC-P	946	10,832.23	0	186,729	12,588	42	477	3,410	845	21,815	10.65
pioro40	M-SEC	10,857	*7,200.00	70,732	0	0	0	1,371	0	6,980	58,043	35.71
pioro40	G-SEC	10,193	*8,625.87	0	177,829	0	0	1,083	0	7,330	24,851	28.09
pioro40	G-SEC+2M	10,395	*6,600.00	0	102,367	23,893	0	909	1,568	7,188	14,166	30.85
pioro40	G-SEC+2M+GEC	10,414	*5,719.98	0	89,859	21,065	43	598	1,704	7,141	11,448	31.43
pioro40	G-SEC+2M+GEC-P	9,648	*6,120.00	0	129,242	21,649	27	1,543	2,792	7,387	20,884	23.44
germany	M-SEC	564,440	10,800.94	51,456	0	0	0	1,374	0	512,070	62,647	9.28
germany	G-SEC	558,140	*9,389.14	0	90,318	0	0	1,920	0	536,492	44,797	3.88
germany	G-SEC+2M	549,150	10,807.50	0	63,973	5,428	0	2,394	4,144	536,492	36,723	2.30
germany	G-SEC+2M+GEC	706,030	*6,000.00	0	14,502	2,430	39	302	13,627	470,993	2,981	33.29
germany	G-SEC+2M+GEC-P	549,150	*8,117.78	0	47,163	6,111	72	5,331	8,756	546,370	84,703	0.51
ta2	M-SEC	74,387,218	10,801.07	17,570	0	0	0	12,559	0	71,556,679	62,151	3.81
ta2	G-SEC	73,782,804	10,803.21	0	56,984	0	0	11,615	0	71,969,561	35,030	2.46
ta2	G-SEC+2M	73,782,804	10,806.61	0	55,301	3,823	0	13,564	662	71,895,458	36,902	2.56
ta2	G-SEC+2M+GEC	73,959,706	10,803.36	0	60,824	3,286	15	12,046	1,058	71,851,626	36,869	2.85
ta2	G-SEC+2M+GEC-P	73,782,804	10,803.47	0	26,489	1,513	7	12,764	366	72,139,091	43,182	2.23

6.1 Ring bound impact

In our final sets of results, we assess the impact of varying the ring bound. We use the G-SECs on the local rings(Eq. (19)) with T-SECs (Eq. (20)), 2-matchings on both the local and tertiary rings (Eq. (21) and Eq. (22)), the Ghost Elimination constraints, GECs (Eq. (23)) and prioritise on the z_{kk} variables. Figure 5 shows an example of the RSAP topology under varying ring bound values for the *janos-us* problem instance. When the ring bound is low, $R = 4$, we see that optimal RSAP solution has a topology with six local rings, five spurs and a tertiary ring of 12 edges. When $R = 8$ the RSAP topology consists of four local rings, two spurs and a smaller tertiary ring of size eight. When we relax the ring bound to 12, we get an RSAP topology of five local rings with one spur and a tertiary ring on five edges.

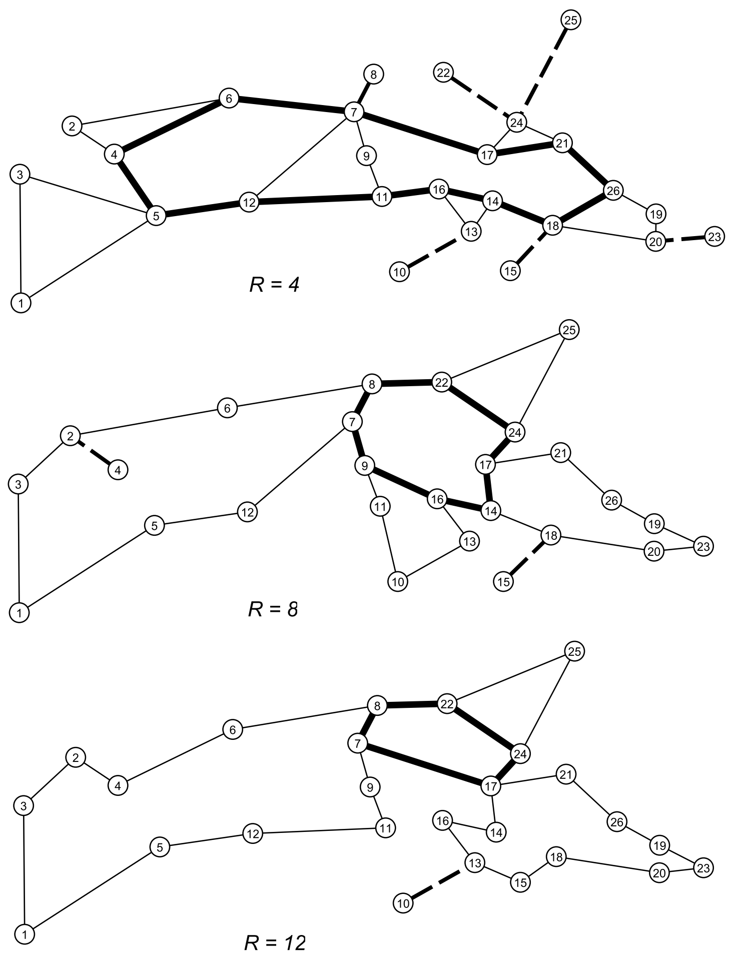


Figure 5: Topology examples for varying ring bound values

Table 4 shows a summary of the ring bound impact results. From left to right we show the problem name, the value of the ring bound R, the IP objective function value and computational time. Then we show the best bound on termination or interruption of the algorithm followed by the number of nodes in the branch-and-cut tree and the gap between the integer solution and best bound. The final columns show a summary of the RSAP topology: the size of the tertiary ring, the number of local rings and the number of spurs.

We see that restricting the ring bound in general makes the problem more difficult to solve. Run times are longer and in some instances no feasible solutions are possible. By lowering the ring bound, we may be unable to find rings that satisfy the ring bound resulting in solutions with more spurs which in turn increases the objective function cost.

In contrast, relaxing the ring bound to 12 highlights the combinatorial nature of the RSAP problem.

Solutions have less spurs and correspondingly, lower cost.

problem	R	IP obj	IP time (s)	bestbound	Nodes	gap	$\|T\|$	num rings	num spurs
dfn-bwin	4	105,882	0.88	105,882	38	0.00	3	3	0
dfn-bwin	8	105,810	0.35	105,810	15	0.00	3	2	0
dfn-bwin	12	105,810	0.37	105,810	15	0.00	3	2	0
pdh	4	1,429,570	1.28	1,429,570	94	0.00	3	3	0
pdh	8	1,355,139	0.62	1,355,139	24	0.00	3	2	0
pdh	12	1,355,139	0.48	1,355,139	17	0.00	3	2	0
di-yuan	4	467,900	0.92	467,900	69	0.00	3	3	0
di-yuan	8	412,300	0.03	NA	0	0.00	3	2	0
di-yuan	12	412,300	0.03	NA	0	0.00	3	2	0
dfn-gwin	4	17,616	0.76	17,616	87	0.00	3	3	0
dfn-gwin	8	15,724	0.43	15,724	10	0.00	3	2	0
dfn-gwin	12	15,724	0.45	15,724	10	0.00	3	2	0
polska	4	4,154	0.42	4,154	14	0.00	5	3	1
polska	8	3,487	0.39	3,487	15	0.00	4	2	0
polska	12	3,487	0.35	3,487	10	0.00	4	2	0
atlanta	4	INT INF	0.63	NA	NA	NA	NA	NA	NA
atlanta	8	55,452,500	0.52	55,452,500	28	0.00	5	3	0
atlanta	12	55,452,500	0.81	55,452,500	59	0.00	5	3	0
newyork	4	1,798,800	15.57	1,798,800	2,578	0.00	5	4	0
newyork	8	1,512,400	2.75	1,512,400	170	0.00	3	2	0
newyork	12	1,466,800	2.62	1,466,800	116	0.00	3	2	0
ta1	4	17,395,348	28.78	17,395,348	2,813	0.00	8	6	3
ta1	8	11,410,169	25.23	11,410,169	1,329	0.00	8	4	0
ta1	12	9,744,343	2.90	9,744,343	78	0.00	6	3	0
france	4	25,800	42.44	25,800	3,132	0.00	11	6	4
france	8	20,800	21.77	20,800	1,776	0.00	12	6	2
france	12	20,800	22.40	20,800	1,260	0.00	12	6	2
janos-us	4	24,728	18.28	24,728	602	0.00	12	6	5
janos-us	8	16,672	13.87	16,672	400	0.00	8	4	2
janos-us	12	15,492	15.43	15,492	536	0.00	5	3	1
norway	4	888,130	71.35	888,093	2,086	0.00	13	6	4
norway	8	596,070	19.39	596,070	527	0.00	4	4	2
norway	12	524,160	34.79	524,160	1,513	0.00	3	3	2
sun	4	1,225	48.21	1,225	1,553	0.01	13	6	4
sun	8	695	22.93	695	373	0.00	4	4	1
sun	12	528	30.11	528	1,562	0.01	7	3	0
cost266	4	INT INF	0.53	NA	NA	NA	NA	NA	NA
cost266	8	12,262,590	139.29	12,262,590	4,253	0.00	19	7	1
cost266	12	10,723,770	151.45	10,723,443	4,367	0.00	15	6	1
giul39	4	1,367	4,058.57	1,367	38,031	0.00	17	10	3
giul39	8	946	10,832.23	845	21,815	10.65	9	6	2
giul39	12	813	10,811.90	798	21,031	1.90	6	4	0
pioro40	4	10,673	8,542.56	9,556	51,898	10.47	21	11	2
pioro40	8	9,648	6,120.00	7,387	20,884	23.44	21	8	1
pioro40	12	9,058	6,383.09	6,819	16,176	24.72	11	5	0
germany	4	INT INF	808.98	NA	NA	NA	NA	NA	NA
germany	8	549,150	8,117.78	546,370	84,703	0.51	16	7	4
germany	12	451,670	9,363.76	420,470	50,361	6.91	9	6	1
ta2	4	INT INF	13.41	NA	NA	NA	NA	NA	NA
ta2	8	73,782,804	10,803.47	72,139,091	43,182	2.23	22	10	13
ta2	12	39,567,043	6,916.23	39,563,767	34,623	0.01	25	8	6

Table 4: Ring bound impact

7 Conclusion

We have described the RSAP and set it in context relative to survivable network design problems described in the literature. The main contributions of our paper are a proof of the NP-hardness of the RSAP, a valid IP formulation with and a set of additional valid inequalities that can be used in a cutting plane approach to solve problem instances of reasonable size.

We have described a cutting plane framework and given results that justify our choice of methodology. The G-SECs outperform the M-SECs as expected and empirical results suggest that the combination of 2-matching and GEC inequalities play a role in solving larger problem instances.

Future avenues of research suggested by our research are to identify other additional valid inequalities to further strengthen the formulation.

8 Acknowledgments

The first author thanks the Quinn and Smurfit Schools of Business at UCD for their support. Bernard Fortz and Martine Labbé were supported by an "Actions de Recherche Concertées" (ARC) project of the "Communauté française de Belgique" and a FRFC program "Groupe de Programmation Mathémathique".

Appendix - Complexity of the RSAP

Before trying to identify an efficient algorithm to solve any problem, we need to first decide whether an efficient algorithm can exist for this problem, that is, we need to determine its computational complexity. In this section we show using local replacement that every instance of EXACT COVER by 3-SETS can be transformed into an instance of RSAP. The EXACT COVER by 3-SETS (X3C) problem is shown to be NP-complete in Garey and Johnson (1979). It is referenced as problem $SP2$ in their list of NP-complete problems and can be stated as follows:

Instance: A finite set X with $|X| = 3q$ and a collection C of 3-element subsets of X.

Question: Does C contain an exact cover for X, that is a subcollection $C' \subseteq C$ such that every element of X is in exactly one member of C'?

We consider the special case of RSAP where b the spur weighting penalty is sufficiently high to ensure the selection of a ring solution if one exists and the ring bound R is set to three. Moreover, since each ring must contain at least three vertices, in this special case each ring is a triangle. We formally state the decision version of this special case of RSAP, RSAP-3D as follows:

Instance: Graph $G = (V, E)$ with $|V| = 3q'$ and $|V| \geq 6$ for some integer q' and ring bound $R = 3$.

Question: Can the vertices of G be partitioned into q' disjoint sets $V_1, V_2, \dots V_{q'}$ each containing exactly three vertices such that for each $V_i = \{u_i, v_i, w_i\}$, all three edges $\{u_i, v_i\}$, $\{v_i, w_i\}$ and $\{u_i, w_i\}$ belong to E to form the local rings and can the local rings be interconnected by a tertiary ring?

Theorem 8.1. *RSAP-3D is NP-Complete.*

Proof. Using a technique similar to that in Garey and Johnson (1979) to show that Partition into Triangles is NP-complete, we transform X3C to RSAP-3D. Let the set X with $|X| = 3q$ and the collection C of 3-element subsets of X be an instance of X3C. We use a local replacement on X3C to create G with $|V| = 3q'$ in such a way that ensures an RSAP-3D solution exists for G if and only if C contains an exact cover. That is, a YES answer to X3C gives a YES answer to RSAP-3D, similarly, a NO answer to X3C means there is a NO answer to RSAP-3D. We order the collection of 3-element subsets of C as follows: $c_1, c_2, \dots c_n$

For each 3-element subset in $c_i \in C$ we first replicate the local replacement of Garey and Johnson (1979) by adding the collection of 18 edges E_i. Figure 6 shows an example of the construction for c_i. The graph $G = (V, E)$ is defined by:

$$V = X \cup \bigcup_{i=1}^{n} \{a_i[j] : 1 \leq j \leq 9\}$$
$$E = \bigcup_{i=1}^{n} E_i + E'$$

where E' are additional edges added to connect the local replacement constructs for each c_i as shown in Figure 7. We add the two edges to join $a_i[2]$ to $a_i[4]$ and $a_i[5]$ to $a_i[7]$ for each c_i. We also add an edge joining each c_i to the next 3-element subset c_j by joining $a_i[9]$ to $a_j[3]$. We link the last 3-element subset, c_n, back to the first one, c_1, by adding the edge that joins $a_n[9]$ to $a_1[3]$.

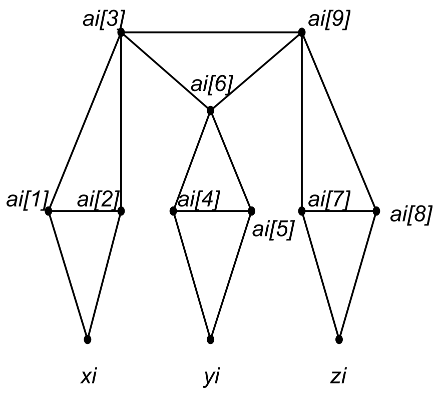


Figure 6: Local replacement for $c_i = \{x_i, y_i, z_i\} \in C$

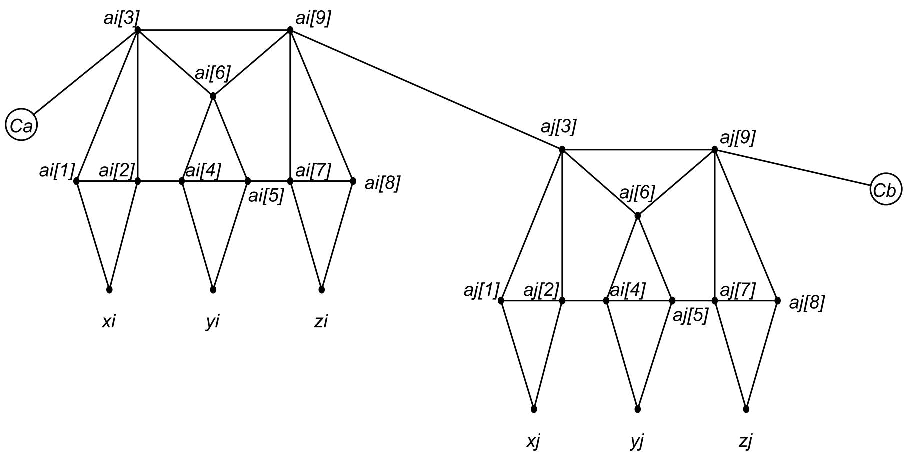


Figure 7: Local replacement; additional E' edges

We verify the number of vertices in the construction of G,

$$|V| = |X| + 9n = 3q + 9n = 3(q + 3n) = 3q'$$
$$\text{so } q' = q + 3n$$

The number of edges of G is given by $18n + 2n + n = 21n$. So G can be constructed in polynomial time, that is the transformation from an X3C instance to an RSAP-3D instance can be implemented in polynomial time. Some vertices of X may appear in more than one 3-element subset but the vertices $a_i[1]$ to $a_i[9]$ are unique to each c_i.

Next we need to see how to transform a YES answer to X3C to a YES answer to RSAP-3D. Let $c_1, c_2, ...c_q$ be the 3-element subsets of C in an X3C solution for X. Then we can read a partition into local rings from $V = V_1 \cup V_2 \cup ... \cup V_{q'}$ as follows:

$$\{a_i[1], a_i[2], x_i\}, \{a_i[4], a_i[5], y_i\}$$
$$\{a_i[7], a_i[8], z_i\}, \{a_i[3], a_i[6], a_i[9]\}$$

That is, these are the four local rings (triangles, since the ring bound $R = 3$) corresponding to c_i when c_i is in the exact cover. For each c_k not in the cover, we can take the alternative set of local rings (triangles):

$$\{a_k[1], a_k[2], a_k[3]\}, \{a_k[4], a_k[5], a_k[6]\} \text{ and } \{a_k[7], a_k[8], a_k[9]\}$$

That is, these three local rings (triangles) correspond to c_k when c_k is not in the cover of X3C and our construction ensures that each element of X is included in exactly one local ring in the partition. Examples of the local rings that cover G are shown in heavy black edges in Figure 8.

Lastly, we need to connect the tertiary ring of the RSAP-3D solution to ensure a YES to RSAP-3D when we have a YES to X3C. We start the tertiary ring at the local replacement for c_1. If c_1 is in the cover of X3C, we select the sequence of vertices and the edges that join them $a_1[3], a_1[2], a_1[4], a_1[5], a_1[7], a_1[9]$. We use this type of sequence for any c_i in the cover. We next join $a_1[9]$ to $a_2[3]$, joining each $a_i[9]$ to $a_j[3]$ where c_i and c_j are next to each other in the ordered list of $c_1, \ldots c_n$.

For any c_k not in the cover, to connect the tertiary ring we select the sequence of vertices and the edges that join them as follows: $a_k[3], a_k[6], a_k[9]$. Using these sequences of vertices of G we can form the tertiary ring that interconnects the local rings.

So the full sequence of vertices (and corresponding edges) of the tertiary is given by:

$a_1[3], a_1[2], a_1[4], a_1[5], a_1[7], a_1[9]$	c_1 is an example in the cover
$a_2[3], \ldots, a_2[9]$	
....	
$a_k[3], a_k[6], a_k[9]$	c_k is an example not in the cover
$\ldots$	
$a_n[3], \ldots, a_n[9], a_1[3]$	

where c_n is the final 3-element subset of C.

Part of the tertiary ring is shown in figure 8 for a sample instance with tertiary ring edges shown in green. In this instance c_i and c_j are in the X3C cover while c_k is not. Since c_k is not in the cover its elements must be covered by some other of the 3-element subsets in the X3C cover. The dashed edges indicate that the elements of c_k are covered elsewhere.

Conversely, if there is a YES answer to RSAP-3D, there is a YES answer to X3C. We can derive the X3C solution from the RSAP-3D solution by choosing $c_i \in C$ such that $\{a_i[3], a_i[6], a_i[9]\} = V_j$ for some $j, 1 \leq j \leq q'$.

Each triangle $\{a_i[3], a_i[6], a_i[9]\}$ in the RSAP-3D solution corresponds to a 3-element subset of C, $c_i = \{x_i, y_i, z_i\}$ in the exact cover solution. Where the triangle $\{a_k[3], a_k[6], a_k[9]\}$ corresponding to $c_k = \{x_k, y_k, z_k\}$ is not in the RSAP-3D solution, c_k is not in the exact cover. Since the RSAP-3D solution is a partition of all nodes of G into triangles, the individual elements of c_k must be included in some triangle of the RSAP-3D solution. So the individual elements are covered by some other 3-element subsets of C that are in the cover.

So there is a YES answer to RSAP-3D iff there is a YES answer to X3C. Hence RSAP-3D is NP-complete and its corresponding optimisation problem, RSAP-3, is NP-hard. Since RSAP contains RSAP-3 as a special case, it follows that RSAP is NP-hard.

$\square$

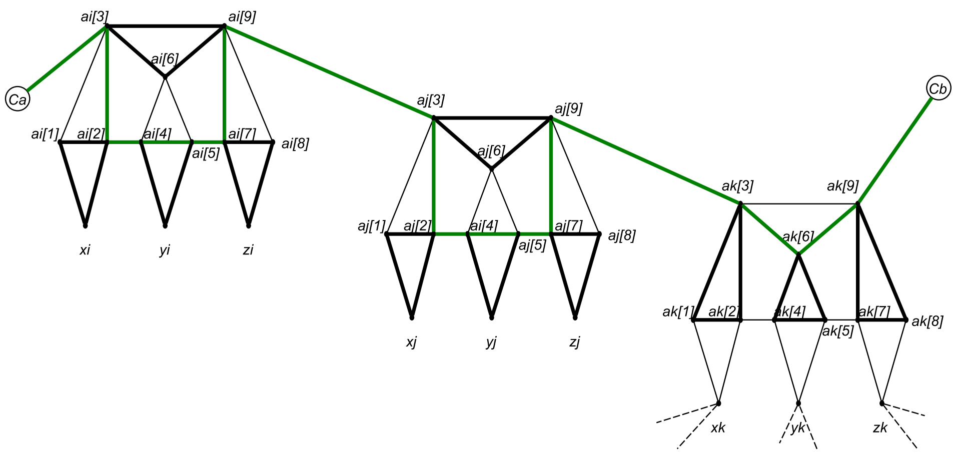


Figure 8: RSAP-3D solution extract

References

Applegate, D., R. Bixby, V. Chvátal, W. Cook. 2003. Implementing the dantzig-fulkerson-johnson algorithm for large travelling salesman problems. *Math Program. Ser.* **B** 91–153.

Baldacci, R., M. Dell'Amico, L. Salazar González. 2007. The capacitated m-ring-star problem. *Operations Research* **55** 1147–1162.

Campêlo, M., V.A. Campos, R.C. Corrêa. 2008. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics* **156** 1097 – 1111. doi:DOI:10.1016/j.dam.2007.05.058. GRACO 2005 - 2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics.

Carroll, P., B. Fortz, M. Labbé, S. McGarraghy. 2011. Improved formulations for the ring spur assignment problem. Pahl, Reiners, Voß, eds., *Network Optimization: International Network Optimization Conference (INOC 2011)*, vol. 6701. Lecture Notes in Computer Science (LNCS), Hamburg, Germany, 24–36.

Carroll, P., S. McGarraghy. 2009a. An algorithm for the ring spur assignment problem. Nina Papova, Micheál Ó hÉigeartaigh, eds., *Proceedings of the International Eugene Lawler PhD Summer School 2009 held at WIT, Ireland, June 6–10, 2009*. Scientific Computing, WIT, 180–197.

Carroll, P., S. McGarraghy. 2009b. Investigation of the ring spur assignment problem. Pisa, Italy, MB1–3. URL `http://www.di.unipi.it/optimize/Events/proceedings/M/B/1/MB1-3.pdf`.

Cosares, S., D.N. Deutsch, I. Saniee, O.J. Wasem. 1995. Sonet toolkit: A decision support system for designing robust and cost-effective fiber-optic networks. *Interfaces* **25** 20–40.

Fortz, B., M. Labbé. 2002. Polyhedral results for two-connected networks with bounded rings. *Math. Program. Ser.* **A** 27–54.

Fortz, B., M. Labbé. 2004. Two-connected networks with rings of bounded cardinality. *Computational Optimization and Applications* **27** 123–148.

Fortz, B., M. Labbé, F. Maffioli. 2000. Solving the two-connected network with bounded meshes problem. *Operations Research* **48** 866–877.

Garey, M.R., D.S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences, W.H. Freeman, San Fancisco.

Goldschmidt, O., A. Laugier, E.V. Olinick. 2003. SONET/SDH ring assignment with capacity constraints. *Discrete Applied Mathematics* **129** 99 – 128.

Grötschel, M., C.L. Monma, M. Stoer. 1995. Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research* **43** 1012–1024.

Grover, W. 2003. *Mesh Based Sruvivable Networks, Options and Strategies for Optical, MPLS, Sonet and ATM Networking*. Prentice Hall.

Hoshino, E.A., C.C. de Souza. 2009. A branch-and-cut-and-price approach for the capacitated m-ring-star problem. *Electronic Notes in Discrete Mathematics* **35** 103–108.

Huygens, D., M. Labbé, R. Mahjoub, P. Pesneau. 2007. The two-edge connected hop-constrained network design problem: Valid inequalities and branch-and-cut. *Networks* **49** 116–133.

Korte, B., J. Vygen. 2008. *Combinatorial Optimization: Theory and Algorithms*, *Algorithms and Combinatorics*, vol. 21. forth ed. Springer, electronic resource.

Labbé, M., G. Laporte, I.R. Martin, J.J. Salazar-Gonzalez. 2004. The Ring Star Problem: Polyhedral analysis and exact algorithm. *Networks* **43** 177–189.

Laporte, G. 1986. Generalized subtour elimination constraints and connectivity constraints. *Journal of the Operational Research Society* **37** 506–514.

Laporte, G., A. Marin, J. Mesa, F. Ortega. 2007. An integrated methodology for the rapid transit network design problem. Frank Geraets, Leo Kroon, Anita Schoebel, Dorothea Wagner, Christos Zaroliagis, eds., *Algorithmic Methods for Railway Optimization*, *Lecture Notes in Computer Science*, vol. 4359. Springer Berlin / Heidelberg, 187–199.

Naji-Azimi, Z., M. Salari, P. Toth. 2010. A heuristic procedure for the capacitated m-ring-star problem. *European Journal of Operational Research* **207** 1227–1234.

Nemhauser, G.L., L.A. Wolsey, eds. 1988. *Integer and Combinatorial Optimization*.

Orlowski, S., M. Pióro, A. Tomaszewski, R. Wessäly. 2010. SNDlib 1.0–Survivable Network Design Library. *Networks* **55** 276–286. doi:10.1002/net.20371. URL `http://sndlib.zib.de`.

Soni, S., R. Gupta, H. Pirkul. 1999. Survivable network design: The state of the art. *Information Systems Frontiers* **1** 303–315.

Stoer, M., F. Wagner. 1997. A simple min-cut algorithm. *Journal of the ACM* **44** 585–591.