Algorithmic Operations Research Vol.4 (2009) 19–35

Compatible Weights and Valid Cycles in Non-spanning OSPF Routing Patterns

Peter Broström and Kaj Holmberg

Department of Mathematics, Linköping Institute of Technology, SE-581 83 Linköping, Sweden

Abstract

Many IP (Internet Protocol) networks use OSPF (Open Shortest Path First) for determining the routing of traffic. OSPF routers compute routing paths using link weights set by the network administrator, and the routers send traffic on all shortest paths to the destination. An interesting question is whether or not a set of prespecified routing patterns can be realized in an OSPF network. If not, we seek structural properties that explain why no such weights exist. Mathematical models for finding weights and for combining routing patterns are presented. We show that two possibly non-spanning routing patterns forming a "valid cycle" cannot simultaneously be obtained in an OSPF network. Two new methods for finding valid cycles are presented, illustrated by numerical examples, and shown to be faster than those previously known.

Key words: Internet Protocol, OSPF, routing, compatible weights, valid cycle.

1. Introduction

The Internet consists of a huge number of routing domains, also called autonomous systems, and each domain is supervised by an administrator. The administrator has several responsibilities, and one of the most important is to determine how traffic is routed through the domain. The routing is determined by choosing a routing protocol, and by setting appropriate values on routing parameters. There are several different routing protocols available for autonomous systems, for example OSPF (Open Shortest Path First), RIP (Routing Internet Protocol) and IS-IS (Intermediate-System to Intermediate-System), see [19], [18], and [12].

We will here study networks where OSPF is used as routing protocol. In such networks, each link is assigned a positive integer link weight by the network administrator, and the routers send traffic on the paths that have a minimal sum of weights to each destination (called the shortest paths). If several paths have the same minimal sum of weights, the router splits the traffic addressed to the destination evenly on all outgoing links that belong to a shortest path. A "routing pattern" contains all the paths that are used for routing.

It is an easy task to compute the routing paths if the weights are known, and this is in fact exactly what the routers do when the routing paths are computed. However, from an administrator's point of view, it is not always as easy to find link weights that give a certain set of desired routing paths. In fact it depends on the structure of the desired routing paths if such weight exists. The administrator's problem is studied in this paper, i.e. a set of desired routing paths is given, and the task is to find weights such that all desired paths are shortest paths with respect to the weights, while any other path between the same pairs of routers should have a larger sum of weights. If there are no such weights, we wish to explain which parts of the routing paths that are in conflict. Since this problem consists of finding a set of weights with certain properties, it will be called the Weight Finding Problem (WFP).

[15], [3] and [9] present mathematical models for different versions of the WFP, and the differences originate mainly from the structure of the prespecified routing paths. Some models require that at most one path is specified between a pair of nodes, while other models allow more than one. Load balancing can only be used if several paths are allowed between two nodes. Another variation is that the desired routing paths sometimes are directed, and sometimes undirected. The case with undirected routing paths yields symmetric routing, and this case is treated in [15] and [3]. The case with spanning and directed routing patterns is studied in [9]. Directed routing paths allow non-symmetric routing since it is possible to specify different paths in the different directions between two nodes. Symmetric routing can also be obtained using directed routing paths, simply by specifying pairs of oppositely directed paths, see [2].

In previous research, the OSPF protocol has also been modeled as side constraints in network design models and flow allocation models, see e.g. [4], [17], [23] and [14]. A common approach is to introduce integer variables for the link weights, and let the side constraints use the weights for ensuring that traffic is routed only on shortest paths to each destination. There are however disadvantages with this approach, one of which is a constraint coefficient that must be larger than the sum of weights on any path. This makes the LP-relaxation very weak, which is a disadvantage when solution methods are considered.

These difficulties may be avoided if the design model is not based on the weights. An alternate approach is therefore to develop network design models based on routing patterns which are obtainable from OSPF weights, but not explicitly including the weights. The OSPF protocol is then modeled by ensuring that the routing patterns satisfies necessary conditions for the existence of compatible weights. Such conditions have been developed by investigating infeasible instances of the WFP. Necessary conditions for undirected routing paths are presented in [3], and necessary conditions for directed and spanning routing patterns are presented in [11]. (A spanning routing pattern contains paths to/from all nodes, while in a non-spanning routing pattern, all nodes are not included.)

In this paper we consider the case with directed routing patterns and load balancing, and discuss several models for the WFP. Compared to the model in [11], it is not required that the routing patterns are spanning. We show that routing patterns with certain properties can be combined into a single routing pattern, and describe the advantages of doing so. We prove that two possibly non-spanning routing patterns can not be obtained simultaneously if they contain a structure called "valid cycle". In [11] this was shown only for spanning routing patterns. Two new methods for finding valid cycles are proposed, and computational results indicate that these new methods are faster than the one previously known, [6], [11].

The next section contains a detailed description of the problem and mathematical formulations of the weight finding problem defined for sets of routing paths. Section 3. investigates how non-spanning routing patterns can be combined into so called SP-graphs. Valid cycles are discussed in Section 4., and methods for finding valid cycles are presented in Section 5.. The case with symmetric routing paths is discussed in Section 6.. Section 7. contains numerical examples and Section 8. contains results from computational tests. The last section contains conclusions.

2. Problem formulation

We consider the directed graph $G = (N, A)$ with the set of nodes N and the set of arcs A. A number of node pairs, $(o_k, d_k),\ k = 1, \ldots, K$, is given, and for each k, a set S_k of paths from node o_k to node d_k is given. S_k contains the desired shortest paths between these two nodes. S_k can either contain one single path or several paths (corresponding to splitting of the traffic), and each path in S_k is represented by the included arcs. The case when each S_k contains one path is called "the single path case". We will also use the set of all arcs in any of the paths in S_k, i.e. $R_k = \bigcup_{p \in S_k} p = \{(i,j) : \exists p \in S_k : (i,j) \in p\}$. Each arc is included at most once in R_k, even if it is present in more than one of the paths, so $|R_k| \leq |A|$. We assume S_k contains all paths given by the arcs in R_k.

In an OSPF-network, integral weights $w_{ij} \geq 1$ are associated with each link $(i,j) \in A$. The paths used by the routers are those with minimal sum of weights w, i.e. the routers find the shortest paths to the destinations. We will use the notation $W(p) = \sum_{(i,j) \in p} w_{ij}$, i.e. $W(p)$ denotes the total weight of path p. If $W(p) \leq W(q)$ for all paths q between the same pair of nodes, p is called a *minimal weight path*. If $W(p) < W(q)$ for all paths $q \neq p$ between the same pair of nodes, p is called a *unique minimal weight path*. This notation is also extended to sets of paths as follows. Let P be a set of paths between a pair of nodes. If $W(p) = W(p')$ for all paths $p \in P, p' \in P$ and $W(p) < W(q)$ for all paths $p \in P$ and for all paths $q \notin P$ between the same pair of nodes, P is called a *unique minimal weight path set*.

The goal is to find weights w such that S_k becomes a unique minimal weight path set from node o_k to node d_k, for all k. Such a set of weights is said to be compatible with the given paths, so we use the term *compatible weights*. Obviously, a difficulty is that the same set of weights are used for all the sets S_k. The difference between two different sets of compatible weights is unimportant, so the question of whether or not a set of compatible weights exist is more important than which to choose. If compatible weights do not exist, we wish to find a small set of paths or links that prohibits the existence of compatible weights. Our practical goal then is to extract information about how to modify the paths, in order to enable the existence of compatible weights.

Lemma 1 *If a desired routing path contains a directed cycle, no compatible weights exist.*

Proof: Since a cycle leads to an already visited node, it can only have minimal weight if its total weight is zero or less. However, since no weight may be less than one, the total weight of a cycle is positive. □

Let $p(s,t)$ denote a path p (represented by the included arcs) from node s to node t, $N(p)$ the set of nodes included in path p, and $N(S_k)$ the set of nodes included in any path in S_k.

Definition 1 *If p and q are two paths and if $q \subset p$, then q is called a subpath of p.*

Lemma 2 *If p is a minimum weight path with respect to w and if $q(s,t)$ is a subpath of p, then $q(s,t)$ is a minimum weight path from s to t.*

Proof: If not, there exists another path $r(s,t)$ with $W(r(s,t)) < W(q(s,t))$. Then we can replace the subpath $q(s,t)$ by $r(s,t)$ in p, and get a path with less weight, which contradicts the assumption that p is a minimum weight path. □

Thus any subpath of any path in S_k should be a minimal weight path.

There cannot be two different unique shortest paths between two nodes, so if two desired paths pass the same pair of nodes, the paths between these two nodes must be identical. This property has previously been called *suboptimality* in for example [3]. In [3], it is shown that in the undirected single path case, a necessary condition for the existence of compatible weights is that all pairs of desired paths are suboptimal.

We will now generalize the concept of suboptimality for the case with directed paths and load balancing, under the name "subpath consistency" (since it is a requirement that certain subpaths shall be consistent). We say that two sets of desired paths are *subpath inconsistent* if both sets contain directed paths from one node to another, and if these sets are not identical. Two sets of desired paths which are not subpath inconsistent with respect to any pair of nodes are called *subpath consistent*. If we let $S_k^S(s,t)$ be the set of all subpaths from node s to node t of paths in S_k, we can make a precise definition of subpath consistency.

Definition 2 *S_k and S_l are subpath consistent if $S_k^S(s,t) = S_l^S(s,t)$ for all $s \in N$ and $t \in N$ such that $S_k^S(s,t) \neq \emptyset$ and $S_l^S(s,t) \neq \emptyset$.*

Thus, if S_k and S_l are subpath inconsistent, $S_k^S(s,t) \neq \emptyset$, $S_l^S(s,t) \neq \emptyset$, and $(S_k^S(s,t) \setminus S_l^S(s,t)) \cup (S_l^S(s,t) \setminus S_k^S(s,t)) \neq \emptyset$ for some $s,t \in N$. Clearly, if $|N(S_k) \cap N(S_l)| < 2$, there does not exist two such nodes, so two sets of paths containing at most one node in common are always subpath consistent. As the definition of subpath consistency really concerns subpaths, the following is not surprising.

Lemma 3 *If the paths p_1 and p_2 are subpath consistent, then any two subpaths $q_1 \subset p_1$ and $q_2 \subset p_2$ are subpath consistent.*

Corollary 1 *If S_k and S_l are subpath consistent, then for any subpath $p(s,t)$ of any path in S_k, either s and t does not both belong to $N(S_l)$, or $p(s,t)$ is a subpath of some path in S_l.*

We will now show that subpath consistency is a necessary condition for the existence of compatible weights.

Lemma 4 *If two sets of desired paths are subpath inconsistent, there exist no compatible weights.*

Proof: If S_k and S_l are subpath inconsistent, then there exists a subpath $p(s,t) \in (S_k^S(s,t) \setminus S_l^S(s,t)) \cup (S_l^S(s,t) \setminus S_k^S(s,t))$ for some $s \in N$ and $t \in N$ such that $S_k^S(s,t) \neq \emptyset$ and $S_l^S(s,t) \neq \emptyset$. If $p(s,t) \in S_k^S(s,t) \setminus S_l^S(s,t)$, then $p(s,t)$ should be a minimal weight path according to S_k but not according to S_l. If $p(s,t) \in S_l^S(s,t) \setminus S_k^S(s,t)$, then $p(s,t)$ should be a minimal weight path according to S_l but not according to S_k. Since none of these cases can be true, there exist no compatible weights. □

Let us now set up a mathematical model for finding compatible weights, if they exist. The model should give weights w such that all paths in S_k should be the shortest paths from the node o_k to the node d_k. All other paths should be more expensive. Let us denote the set of *all* paths in the graph from node o_k to node d_k by P_k. We require that all paths in $P_k \setminus S_k$ should have a larger sum of weights than the paths in S_k. Since the weights will be integral, the least difference will be one, so we get constraints (1) below. Furthermore, all shortest paths between a pair of nodes must obviously have the same cost, which yields constraints (2). (In the single path case, constraints (2) are not present, as there is only one path in each S_k.)

P1:

$$\sum_{(i,j)\in q} w_{ij} - \sum_{(i,j)\in p} w_{ij} \geq 1 \ \forall p \in S_k, \quad \forall q \in P_k \setminus S_k, \ \forall k \tag{1}$$

$$\sum_{(i,j)\in r} w_{ij} - \sum_{(i,j)\in p} w_{ij} = 0 \ \forall p \in S_k : p \neq r, \quad \forall r \in S_k, \ \forall k \tag{2}$$

$$w_{ij} \geq 1, \text{ integer } \ \forall (i,j) \in A \tag{3}$$

By construction it is obvious that any feasible solution

to P1 is a set of compatible weights. On the other hand, any set of compatible weights must satisfy (1), (2) and (3), and must therefore be a feasible solution to P1. Thus we have the following.

Lemma 5 *P1 has a feasible solution if and only if there exists a compatible set of weights.*

A feasible solution to P1 is not unique. For example, multiplying a feasible set of weights with a positive integer, yields another feasible solution. We consider all feasible sets of weights as equivalent, although in practice, unnecessarily large weights might be avoided. Lemmas 4 and 5 yield the following.

Lemma 6 *If two sets of paths are not subpath consistent, then P1 has no feasible solution.*

A disadvantage of the model P1 is the large number of constraints, which comes from the large number of possible paths in a network. Each path in S_k is compared to all other paths from o_k to d_k, either in constraint(s) (4) or in (5).

Letting C_k denote the (unknown) minimal weight of a path from o_k to d_k, P1 can be rewritten as follows, where C_k is variable.

P2:

$$\sum_{(i,j)\in q} w_{ij} \geq C_k + 1 \ \forall q \in P_k \setminus S_k, \ \forall k \quad (4)$$

$$\sum_{(i,j)\in p} w_{ij} = C_k \forall p \in S_k, \ \forall k \quad (5)$$

$$w_{ij} \geq 1, \text{ integer } \forall (i,j) \in A \quad (6)$$

The number of constraints is less in P2 than in P1 (unless $|S_k| = 1 \ \forall k$). By simply using (5) one can easily verify that any feasible solution of P1 is feasible in P2, and that any feasible solution of P2 is feasible in P1. Thus P2 has a feasible solution if and only if P1 has a feasible solution.

According to [16], there exists a node potential, $\pi_i \ \forall i \in N$, satisfying $w_{ij} + \pi_i - \pi_j \geq 0 \ \forall (i,j) \in A$ if all directed cycles in a graph have a non-negative sum of weights. Furthermore, the node potential π can be chosen integer-valued if w is integer-valued. It is also shown that p is a minimum weight path if and only if there is a potential satisfying $w_{ij} = \pi_j - \pi_i \ \forall (i,j) \in p$. If we sum up these equalities for a minimal weight path p starting in node s and ending in node t, we get $W(p(s,t)) = \pi_t - \pi_s$, since all other node potentials cancel out. A path q from s to t which is not a minimum weight path must therefore satisfy $W(q(s,t)) > \pi_t - \pi_s$.

In the weight finding problem, we need different node potentials for each set S_k, so we introduce $\pi_i^k \ \forall i \in N, \ \forall k = 1, \ldots, K$. We have

$$W(p) = \pi_{d_k}^k - \pi_{o_k}^k \ \forall p \in S_k, \ \forall k,$$

and according to Lemma 2, this applies to any subpath of p, i.e.

$$W(p(s,t)) = \pi_t^k - \pi_s^k \text{ for all}$$

$$p(s,t) \in S_k^S(s,t) \ \ \forall s \in N(S_k), t \in N(S_k) \ \forall k.$$

Especially, this applies to subpaths consisting of single links.

$$w_{ij} = \pi_i^k - \pi_j^k \ \forall (i,j) \in p, \ \forall p \in S_k, \forall k$$

Here, $(i,j) \in p, p \in S_k$ is more efficiently written as $(i,j) \in R_k$, since this avoids representing links more than once. Furthermore we should have

$$W(q) \geq \pi_{d_k}^k - \pi_{o_k}^k + 1 \ \forall q \in P_k \setminus S_k, \ \forall k,$$

(since the weights and π_i^k are integer-valued). We thus get the following model.

P3:

$$w_{ij} + \pi_i^k - \pi_j^k = 0 \forall (i,j) \in R_k, \forall k \quad (7)$$

$$\sum_{(i,j)\in q} w_{ij} + \pi_{o_k}^k - \pi_{d_k}^k \geq 1 \forall q \in P_k \setminus S_k, \forall k \quad (8)$$

$$w_{ij} \geq 1, \text{ integer } \forall (i,j) \in A \quad (9)$$

Theorem 1 *P3 has a feasible solution if and only if P1 has a feasible solution.*

Proof: Assume that we have a feasible solution to P1. Now let π_i^k be equal to the minimum sum of weights from o_k to i. We then get $w_{ij} = \pi_j^k - \pi_i^k \ \forall (i,j) \in R_k, \forall k$, so (7) is satisfied. The paths in $P_k \setminus S_k$ are not minimum weight paths, so all constraints in (8) are satisfied. Constraints (9 are identical to (3), so we conclude that P3 has a feasible solution.

Assume now that we have a feasible solution to P3. Summing up constraints (7) over any path p in S_k yields $W(p) = \pi_{d_k}^k - \pi_{o_k}^k$. This ensures that constraints (2) are satisfied. Inserting $W(p) = \pi_{d_k}^k - \pi_{o_k}^k$ into (8) immediately yields constraints (1). Thus a feasible solution to P3 is also feasible in P1. □

We conclude that P3 has a feasible solution if and only if there exists a set of compatible weights.

3. Combining paths into SP-graphs

P3 may have a large number of constraints in (8), due to the large number of paths in the sets $P_k \setminus S_k$. Each path in $P_k \setminus S_k$ contains at least one subpath (possibly a single arc) that starts and ends at nodes spanned by R_k, and does not included any arc in R_k. These subpaths are not included in any path $p \in S_k$, and if we ensure that these subpaths are not part of a minimal weight path from node o_k to node d_k, we have ensured that the paths in $P_k \setminus S_k$ have a larger sum of weights than the paths in S_k. Since many paths in $P_k \setminus S_k$ include the same subpath, this can be used for reducing the size of the model.

We may also decrease the size of the model by combining sets of minimum weight paths into *shortest path graphs*, SP-graphs. We will use the term "origin" ("destination") to denote any node in an SP-graph without predecessors (successors). An SP-graph is defined to be a set of arcs which contains at least one path from each origin to each destination, and does not contain any directed cycle. (Consequently each SP-graph must contain at least one origin and at least one destination.)

R_k is an example of an SP-graph with a single origin and a single destination. Since an SP-graph no longer is associated with a single node pair (o_k, d_k), we introduce the index l for SP-graphs and let A_l denote SP-graph l.

A set of weights is compatible with an SP-graph if all paths in the SP-graph are minimum weight paths and any other path from an origin to a destination is not a minimum weight path. A set of weights is called *compatible* if it is compatible with each SP-graph. Let us also extend the meaning of subpath consistency to SP-graphs. Let $A_l^S(s,t) \subseteq A_l$ denote the set of arcs included in any path from s to t in A_l.

Definition 3 *$A_{l'}$ and $A_{l''}$ are subpath consistent if $A_{l'}^S(s,t) = A_{l''}^S(s,t)$ for all $s \in N$ and $t \in N$ such that $A_{l'}^S(s,t) \neq \emptyset$ and $A_{l''}^S(s,t) \neq \emptyset$.*

We will now give a model for finding compatible weights to a set of SP-graphs (possibly with multiple origins and destinations). Let V_l be the set of node pairs spanned by each A_l, i.e. let $V_l = \{(s,t) : s \in N(A_l)$ and $t \in N(A_l)\}$. For each $(s,t) \in V_l$, let T_{st}^l be the set of paths from node s to node t completely outside A_l. This means that each path in T_{st}^l starts at node s, ends at node t, and does not pass any other node spanned by A_l. We can now formulate the following model.

P4:

$$w_{ij} + \pi_i^l - \pi_j^l = 0 \ \forall (i,j) \in A_l, \forall l \tag{10}$$

$$\sum_{(i,j)\in q} w_{ij} + \pi_s^l - \pi_t^l \geq 1 \ \forall q \in T_{st}^l, \forall (s,t) \in V_l, \forall l \tag{11}$$

$$w_{ij} \geq 1, \text{ integer } \ \forall (i,j) \in A \tag{12}$$

The first set of constraints is in principle identical to (7), and ensures that all paths in A_l are minimum weight paths. The second set of constraints ensures that each subpath that starts and ends in A_l and does not pass any other node spanned by A_l is not a part of a minimum weight path from an origin of A_l to a destination of A_l.

Theorem 2 *P4 has a feasible solution if and only if compatible weights exist.*

P4 is in principle a modification of P3. If each SP-graph is equal to an arc set R_k, the result follows from Lemma 5 and Theorem 1. Below we show how SP-graphs can be combined into larger and fewer SP-graphs without changing the feasibility of P4. As an alternative, a general proof of Theorem 2 is found in [10].

The number of constraints (11) could be large if there are many nodes not spanned by each SP-graph. However, if each SP-graph spans all nodes in the node set, set (11) consists of only $\sum_l (|A| - |A_l|)$ constraints, so here we see an incentive to make the SP-graphs as large as possible. There are however restrictions on which sets of paths can be combined. Starting from the case when each SP-graph has one origin and one destination, in which case P4 has a feasible solution if and only if P3 has one, we will increase the sizes of the SP-graphs, while maintaining the property that P4 has a feasible solution if and only if compatible weights exist.

Assume that a number of SP-graphs are given. We now seek the answer to the following question. Does combining two of them, $A_{l'}$ and $A_{l''}$, into one SP-graph, i.e. letting $A_C = A_{l'} \cup A_{l''}$, yield the same result in P4 as using $A_{l'}$ and $A_{l''}$ separately? Let P4s denote the case when $A_{l'}$ and $A_{l''}$ are treated separately, and P4c the case when they are combined. We allow this combination if either compatible weights exist for both P4s and P4c, or compatible weights do not exist for any of P4s and P4c.

We will consider the case when $A_{l'}$ and $A_{l''}$ has one and the same origin, denoted by o. Let $\delta^-(\bar{N})$ denote all arcs leaving node set $\bar{N}$, $\delta^+(\bar{N})$ all arcs entering node set $\bar{N}$, and $\gamma(\bar{N})$ all arcs with both endpoints in

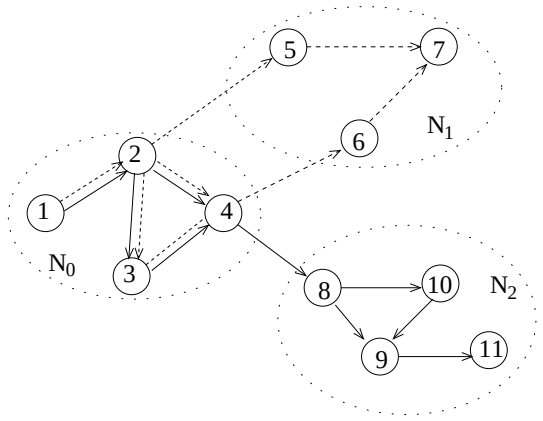

Fig. 1. A partition allowing combination.

node set $\bar{N}$. $N(\bar{A})$ denotes the set of nodes spanned by the arc set $\bar{A}$.

We find that two SP-graphs can be combined if they have the same (single) origin, are identical in a subgraph containing the origin, do not contain any arcs entering the identical subgraph, and are completely separate in the rest of the graph. More specifically, let $N_0 \cup N_1 \cup N_2 = N(A_{l'} \cup A_{l''})$, i.e. the set of nodes spanned by $A_{l'}$ and $A_{l''}$ is partitioned into the sets N_0, N_1 and N_2. Now assume that $o \in N_0$ and $\gamma(N_0) \cap A_{l'} = \gamma(N_0) \cap A_{l''}$, i.e. that $A_{l'}$ and $A_{l''}$ are identical in N_0. Also assume that $N(A_{l'}) \cap N_2 = \emptyset$ and $N(A_{l''}) \cap N_1 = \emptyset$, i.e. that no part of $A_{l'}$ is in N_2 and no part of $A_{l''}$ is in N_1. Furthermore, we require that $A_{l'} \cap \delta^+(N_0) = \emptyset$ and $A_{l''} \cap \delta^+(N_0) = \emptyset$, i.e. that no arc in $A_{l'}$ or $A_{l''}$ enters the node set N_0. See Figure 1 for an example of such a partition. Here $A_{l'}$ is indicated with dashed lines and $A_{l''}$ by solid lines. Both have node 1 as origin. The partition is given by $N_0 = \{1, 2, 3, 4\}$, $N_1 = \{5, 6, 7\}$ and $N_2 = \{8, 9, 10, 11\}$. An important consequence of the assumptions is that $A_{l'}$ consists of arcs in $\gamma(N_0)$, arcs in $\gamma(N_1)$ and arcs directed from nodes in N_0 to nodes in N_1. So if a path in $A_{l'}$ leaves the node set N_0, the path enters and stays inside N_1 until the destination is reached. Similarly, $A_{l''}$ consists of arcs in $\gamma(N_0)$, arcs in $\gamma(N_2)$ and arcs directed from nodes in N_0 to nodes in N_2. The paths in $A_{l''}$ first visit a sequence of nodes in N_0 and then a sequence of nodes in N_2.

Lemma 7 *If there exists a partitioning $N_0 \cup N_1 \cup N_2 = N(A_{l'} \cup A_{l''})$, such that $o \in N_0$ is the only origin in $A_{l'}$ and $A_{l''}$, $\gamma(N_0) \cap A_{l'} = \gamma(N_0) \cap A_{l''}$, $N(A_{l'}) \cap N_2 = \emptyset$, $N(A_{l''}) \cap N_1 = \emptyset$, $A_{l'} \cap \delta^+(N_0) = \emptyset$ and $A_{l''} \cap \delta^+(N_0) = \emptyset$, then P4c has a feasible solution if and only if P4s has one. Therefore the SP-graphs may be combined.*

Proof: We study how P4 changes as a result of combining $A_{l'}$ and $A_{l''}$. The w-variables are the same in P4s and P4c, but the two sets of node prices π^1 and π^2 in P4s will be combined into one set, π^C, in P4c.

Assume first that P4s has a feasible solution, $(\bar{w}, \bar{\pi}^1, \bar{\pi}^2, \ldots, \bar{\pi}^l)$. The solution remains feasible if we increase/decrease all components in $\bar{\pi}^2$ with a constant. Letting $\hat{\pi}_i^2 = \bar{\pi}_i^2 + (\bar{\pi}_o^1 - \bar{\pi}_o^2)\ \forall i \in N$ yields $\bar{\pi}_o^1 = \hat{\pi}_o^2$, which implies $\bar{\pi}_i^1 = \hat{\pi}_i^2\ \forall i \in N_0$, since $A_{l'}$ and $A_{l''}$ are identical in N_0 and since (10) yields $\bar{w}_{ij} = \bar{\pi}_j^1 - \bar{\pi}_i^1\ \forall (i,j) \in \gamma(N_0) \cap A_{l'}$ and $\bar{w}_{ij} = \hat{\pi}_j^2 - \hat{\pi}_i^2\ \forall (i,j) \in \gamma(N_0) \cap A_{l''}$.

P4s and P4c have one constraint in set (10) for each arc in A_C. For arcs in $\gamma(N_0)$, the constraints are identical, so the same π-solution can be used. We set $\pi_i^C = \bar{\pi}_i^1\ \forall i \in N_0$. Next we note that $\bar{\pi}^1$ together with $\bar{w}$ will ensure that (10) is satisfied for arcs in $A_{l'} \setminus \gamma(N_0)$, so we set $\pi_i^C = \bar{\pi}_i^1\ \forall i \in N_1$. Similarly, $\hat{\pi}^2$ together with $\bar{w}$ satisfies (10) for arcs in $A_{l''} \setminus \gamma(N_0)$, so we set $\pi_i^C = \hat{\pi}_i^2\ \forall i \in N_2$.

What remains is constraint set (11), stating that the paths completely outside A_C should not be minimum weight paths. We have $\pi_i^C = \bar{\pi}_i^1\ \forall i \in N_0 \cup N_1$, so P4c satisfies all constraints from set (11) concerning paths starting and ending in $N_0 \cup N_1$ without passing nodes in N_2. Similarly, we have $\pi_i^C = \bar{\pi}_i^2\ \forall i \in N_0 \cup N_2$, so P4c satisfies all constraints from set (11) concerning paths starting and ending in $N_0 \cup N_2$ without passing nodes in N_1.

Now we study paths starting in N_2 and ending in N_1. Suppose that one constraint in set (11) is not satisfied in P4c, for example the constraint defined for the path $q(s,t)$ from $s \in N_2$ to $t \in N_1$. Since π can be chosen integer-valued if w is integer-valued, see [16], we then have $\sum_{(i,j) \in q(s,t)} \bar{w}_{ij} + \pi_s^C - \pi_t^C \leq 0$. Now consider a path $q = \{q(o,s), q(s,t)\}$, where $q(o,s)$ is an arbitrary path from o to s completely inside A_C. We then get

$$0 = \sum_{(i,j) \in q(o,s)} (\bar{w}_{ij} + \pi_i^C - \pi_j^C) = \sum_{(i,j) \in q(o,s)} \bar{w}_{ij} + \pi_o^C - \pi_s^C \geq \sum_{(i,j) \in q(o,s)} \bar{w}_{ij} + \pi_o^C - \pi_s^C + \sum_{(i,j) \in q(s,t)} \bar{w}_{ij} + \pi_s^C - \pi_t^C = \sum_{(i,j) \in q} \bar{w}_{ij} + \pi_o^C - \pi_t^C = \sum_{(i,j) \in q} \bar{w}_{ij} + \bar{\pi}_o^1 - \bar{\pi}_t^1.$$

The first equality holds since (10) is satisfied for all $(i,j) \in q(o,s)$. The node prices cancel out in the second and the fourth equality, and the last equality holds since

$\pi_i^C = \bar{\pi}_i^1 \ \forall i \in N_0 \cup N_1$. The inequality follows from the assumption that (11) is not satisfied for $q(s,t)$.

We thus get $\sum_{(i,j)\in q} \bar{w}_{ij} + \bar{\pi}_o^1 - \bar{\pi}_t^1 \leq 0$, i.e. $W(q) \leq \bar{\pi}_t^1 - \bar{\pi}_o^1$, so the path q does not have a larger sum of weights than the path from o to t in $A_{l'}$. The path q is not completely inside $A_{l'}$ since $s \in N_2$, so this contradicts that the solution is feasible in P4s. Thus, assuming that (11) is not satisfied yields a contradiction, so we conclude that (11) is satisfied for all paths starting in N_2 and ending in N_1. This reasoning can be repeated for paths starting in N_1 and ending in N_2, so we conclude that P4c has a feasible solution if P4s has a feasible solution.

Finally we note that if P4c has a feasible solution $(\bar{w}, \bar{\pi}^C, \bar{\pi}^3, \ldots \bar{\pi}^l)$, we immediately get a feasible solution to P4s by using $\pi^1 = \bar{\pi}^C$ and $\pi^2 = \bar{\pi}^C$. □

The result also holds if the two SP-graphs are identical in a subgraph containing the common destination, and completely separate in the rest of the graph. This means that two SP-graphs can be combined if there exists a partitioning $N_0 \cup N_1 \cup N_2 = N(A_{l'} \cup A_{l''})$ such that $d \in N_0$ is the only destination of $A_{l'}$ and $A_{l''}$, $\gamma(N_0) \cap A_{l'} = \gamma(N_0) \cap A_{l''}$, $N(A_{l'}) \cap N_2 = \emptyset$, $N(A_{l''}) \cap N_1 = \emptyset$, $A_{l'} \cap \delta^-(N_0) = \emptyset$ and $A_{l''} \cap \delta^-(N_0) = \emptyset$. This can be shown by simply reversing the roles of origins and destinations in the previous proof.

Corollary 2 *If two SP-graphs are subpath consistent and have the same node as the only origin, they may be combined.*

Proof: Letting $N_0 = N(A_{l'}) \cap N(A_{l''})$, it is easy to see that the assumptions of Lemma 7 are satisfied. □

If $N_1 = \emptyset$ (i.e. if all destinations of $A_{l'}$ are in N_0) in the partitioning in Lemma 7 then $A_{l'}$ is a subset of $A_{l''}$. In that case all sets of weights that are compatible with $A_{l''}$ will also be compatible with $A_{l'}$, so $A_{l'}$ does not add anything to P4. $A_{l'}$ and $A_{l''}$ may be combined, but the combination of them will be identical to $A_{l''}$. Therefore we can simply delete $A_{l'}$. The same applies to $A_{l''}$ if $N_2 = \emptyset$.

An SP-graph can sometimes be combined with a part of another SP-graph. This could be useful when two SP-graphs are similar, but do not have the same origin or the same destination. Recall that $A_l^S(s,t)$ is the set of arcs included in any path from node s to node t in A_l.

Lemma 8 *P4 remains feasible/infeasible if $A_l^S(s,t)$ is defined as a separate SP-graph and added to the set of SP-graphs, for any pair of nodes s and t.*

Proof: The lemma is only interesting if $A_l^S(s,t) \neq \emptyset$. If P4 has a feasible solution, it follows from Theorem 2 that some set of weights is compatible with each original SP-graph. $A_l^S(s,t)$ contains exactly the same paths from s to t as A_l, so the weights are also compatible with $A_l^S(s,t)$. Thus P4 remains feasible if the SP-graph $A_l^S(s,t)$ is added. P4 is infeasible when no set of weights is compatible with each of the original SP-graphs. Adding an SP-graph cannot change this fact. □

Lemma 7 can be used repeatedly, since several destinations are allowed. If we start from $A_{l'} = R_1$, $A_{l''} = R_2$, etc., and repeatedly combine SP-graphs with the same origin, the resulting SP-graphs will contain an arborescence. If all resulting SP-graphs are spanning, each path in T_{st}^l consists of a single arc, so P4 can be simplified to the following problem.

P5:

$$w_{ij} + \pi_i^l - \pi_j^l = 0 \ \forall (i,j) \in A_l, \ \forall l \quad (13)$$

$$w_{ij} + \pi_i^l - \pi_j^l \geq 1 \ \forall (i,j) \notin A_l, \forall l \quad (14)$$

$$w_{ij} \geq 1, \text{ integer } \ \forall (i,j) \in A \quad (15)$$

Letting m denote the number of SP-graphs, we find that P5 has only $|A|(m+1)$ constraints since each arc either is included in (13) or (14). In [11] we prove that, if all SP-graphs are spanning, P5 has a feasible solution if and only if there exists a set of compatible weights.

Let us also mention a case where SP-graphs may not be combined. Assume that $o_1 \in N(A_{l'}) \setminus N(A_{l''})$, $d_1 \in N(A_{l'}) \setminus N(A_{l''})$, $o_2 \in N(A_{l''}) \setminus N(A_{l'})$, $d_2 \in N(A_{l''}) \setminus N(A_{l'})$, and $N(A_{l'}) \cap N(A_{l''}) \neq \emptyset$. If we combine $A_{l'}$ and $A_{l''}$, this SP-graph will not only contain the paths from o_1 to d_1 and from o_2 to d_2, but also from o_1 to d_2 and from o_2 to d_1. A feasible solution of P4s ensures that the paths from o_1 to d_1 and from o_2 to d_2 are minimal weight paths, but not that the paths from o_1 to d_2 and from o_2 to d_1 are of minimal weight. Therefore, P4s being feasible does not imply that P4c is feasible. Our conclusion is the following.

Lemma 9 *Two SP-graphs cannot be combined if they create a new origin-destination pair.*

Thus, if the two sets of paths have at least one node in common, but different origins and different destinations, they may not be combined.

4. Valid cycles

Our main goal is to analyze instances that lack compatible weights. In [7] this is done by finding unbounded solutions to the LP-dual of the weight finding problem. These unbounded solutions are represented by cycles, in which dual variables can be changed infinitely. Here

we will not use duality, but verify the effect of such cycles in another way.

Recall that each A_l contains at least one path from each origin to each destination, and no A_l contains a directed cycle. Now consider two SP-graphs, $A_{l'}$ and $A_{l''}$, and a *cycle* $C \subseteq A$, $C = F \cup B$, where F is the set of arcs considered to be used *forward* and B the set of arcs considered to used *backward*, in relation to SP-graph $A_{l'}$.

Definition 4 *A cycle $C = F \cup B$ is called feasible if $B \subseteq A_{l'}$ and $F \subseteq A_{l''}$.*

Definition 5 *The arc (i,j) is called eligible if $(i,j) \in (F \setminus A_{l'}) \cup (B \setminus A_{l''})$.*

In words, an eligible arc lies in F but not in $A_{l'}$ or in B but not in $A_{l''}$.

Definition 6 *The cycle $C = F \cup B$ is called improving if it contains at least one eligible arc.*

Definition 7 *A cycle $C = F \cup B$ is called valid if there exist two indices l' and l'' such that the cycle is feasible and improving.*

Theorem 3 *If there exists a valid cycle, then there exists no compatible set of weights.*

The proof for the spanning case (i.e. when all SP-graphs span all nodes) is given in [11]. Here we give a more general proof, covering both the spanning and non-spanning cases.

Proof: Theorem 3 will be proved by summing the weights around a valid cycle, once in view of SP-graph $A_{l'}$ and once in view of SP-graph $A_{l''}$. Obviously these two sums should be equal, but we will get a contradiction, verifying that P4 cannot have a feasible solution.

Let $\Gamma_l(C)$ denote the sum of weights around the valid cycle $C = F \cup B$ in view of SP-graph A_l. Now we split C into subpaths, depending on whether an arc is in F or B and also whether it is in $A_{l'}$ or not. Let $P^F_{l'}$ be the set of subpaths in $F \cap A_{l'}$, $P^B_{l'}$ the set of subpaths in $B \cap A_{l'}$, $Q^F_{l'}$ be the set of subpaths in $F \setminus A_{l'}$, and $Q^B_{l'}$ be the set of subpaths in $B \setminus A_{l'}$. Furthermore, let $p^F_{rl'}$ denote subpath r in the set $P^F_{l'}$, etc.

We require that each subpath starts and ends in nodes spanned by $A_{l'}$, and does not pass any other node spanned by $A_{l'}$. This means that each subpath in $P^F_{l'} \cup P^B_{l'}$ consists of a single arc, while the subpaths in $Q^F_{l'} \cup Q^B_{l'}$ may consist of more than one arc. The cycle C consists of all these subpaths, and we have the following.

$$F \cap A_{l'} = \bigcup_{r \in P^F_{l'}} p^F_{rl'}, \quad B \cap A_{l'} = \bigcup_{r \in P^B_{l'}} p^B_{rl'}, \quad F \setminus A_{l'} = \bigcup_{r \in Q^F_{l'}} q^F_{rl'}, \quad B \setminus A_{l'} = \bigcup_{r \in Q^B_{l'}} q^B_{rl'}.$$

We now get the following sum of weights around the cycle.

$$\Gamma_{l'}(C) = \sum_{(i,j)\in F} w_{ij} - \sum_{(i,j)\in B} w_{ij} = \sum_{(i,j)\in F\cap A_{l'}} w_{ij} + \sum_{(i,j)\in F\setminus A_{l'}} w_{ij} - \sum_{(i,j)\in B\cap A_{l'}} w_{ij} - \sum_{(i,j)\in B\setminus A_{l'}} w_{ij} = \sum_{r\in P^F_{l'}} \sum_{(i,j)\in p^F_{rl'}} w_{ij} + \sum_{r\in Q^F_{l'}} \sum_{(i,j)\in q^F_{rl'}} w_{ij} - \sum_{r\in P^B_{l'}} \sum_{(i,j)\in p^B_{rl'}} w_{ij} - \sum_{r\in Q^B_{l'}} \sum_{(i,j)\in q^B_{rl'}} w_{ij}$$

Now introduce s by letting $w_{ij} = \pi^l_j - \pi^l_i + s^l_{ij}$ $\forall (i,j) \in A$. We have $s^l_{ij} = 0$ $\forall (i,j) \in A_l$, due to constraint 4.1. We also note that for any path $p(s,t)$ for any s and t, we have

$$W(p(s,t)) = \sum_{(i,j)\in p(s,t)} (\pi^l_j - \pi^l_i + s^l_{ij}) = \pi^l_t - \pi^l_s + \sum_{(i,j)\in p(s,t)} s^l_{ij}.$$

For $\Gamma_{l'}(C)$, the π's cancel if we make this substitution, so we get

$$\Gamma_{l'}(C) = \sum_{r\in P^F_{l'}} \sum_{(i,j)\in p^F_{rl'}} s^{l'}_{ij} + \sum_{r\in Q^F_{l'}} \sum_{(i,j)\in q^F_{rl'}} s^{l'}_{ij} - \sum_{r\in P^B_{l'}} \sum_{(i,j)\in p^B_{rl'}} s^{l'}_{ij} - \sum_{r\in Q^B_{l'}} \sum_{(i,j)\in q^B_{rl'}} s^{l'}_{ij}.$$

Now we note that $Q^B_{l'} = \emptyset$, since $B \subseteq A_{l'}$, so the last term is equal to zero. Furthermore $s^l_{ij} = 0$ for each arc in $A_{l'}$, i.e. for each arc in $p^F_{rl'}$ and in $p^B_{rl'}$. That makes terms one and three equal to zero, so only term two remains. Now we note that if $F \subseteq A_{l'}$, we have $Q^F_{l'} = \emptyset$, which means that $\Gamma_{l'}(C) = 0$. The other possibility is $F \not\subseteq A_{l'}$, which implies $Q^F_{l'} \neq \emptyset$. Each path $r \in Q^F_{l'}$ starts and ends at nodes in $N(A_{l'})$ and does not pass any other node spanned by $A_{l'}$, so there is one constraint in 4.2 for each $r \in Q^F_{l'}$. Therefore, we have $\sum_{(i,j)\in q^F_{rl'}} s^{l'}_{ij} \geq 1$ $\forall r \in Q^F_{l'}$, i.e. $\Gamma_{l'}(C) > 0$. There are thus two possibilities, namely $Q^F_{l'} = \emptyset$, which yields $\Gamma_{l'}(C) = 0$, and $Q^F_{l'} \neq \emptyset$, which yields $\Gamma_{l'}(C) > 0$.

Let us now calculate $\Gamma_{l''}(C)$, which is the sum of weights around the same cycle, but taking $A_{l''}$ into account. Doing the same kind of partitioning of the arcs in the cycle, we get

$$\Gamma_{l''}(C) = \sum_{r\in P^F_{l''}} \sum_{(i,j)\in p^F_{rl''}} s^{l''}_{ij} + \sum_{r\in Q^F_{l''}} \sum_{(i,j)\in q^F_{rl''}} s^{l''}_{ij} -$$

$$\sum_{r\in P^B_{l''}}\sum_{(i,j)\in p^B_{rl''}} s^{l''}_{ij} - \sum_{r\in Q^B_{l''}}\sum_{(i,j)\in q^B_{rl''}} s^{l''}_{ij}.$$

Now we know that $F \subseteq A_{l''}$, so $Q^F_{l''} = \emptyset$. Furthermore, $s^{l''}_{ij} = 0$ for all $(i,j) \in A_{l''}$, i.e. for all (i,j) in $p^F_{rl''}$ or in $p^B_{rl''}$. This makes the first three terms above equal to zero, so only the last term remains. As above we can show that the this term is zero if $Q^B_{l''} = \emptyset$, and negative if $Q^B_{l''} \neq \emptyset$. So again there are two possibilities, namely $Q^B_{l''} = \emptyset$, which yields $\Gamma_{l''}(C) = 0$, and $Q^B_{l''} \neq \emptyset$, which yields $\Gamma_{l''}(C) < 0$.

Thus $\Gamma_{l'}(C) = \Gamma_{l''}(C) = 0$ if and only if $Q^F_{l'} = \emptyset$ and $Q^B_{l''} = \emptyset$. A valid cycle has $F \not\subseteq A_{l'}$ and/or $B \not\subseteq A_{l''}$, while $Q^F_{l'} = \emptyset$ if $F \subseteq A_{l'}$ and $Q^B_{l''} = \emptyset$ if $B \subseteq A_{l''}$, so for a valid cycle, at least one of the sets $Q^F_{l'}$ and $Q^B_{l''}$ is non-empty. If $\Gamma_{l'}(C) = 0$ then $Q^F_{l'} = \emptyset$, which implies that $Q^B_{l''} \neq \emptyset$ and $\Gamma_{l''}(C) < 0$. On the other hand, if $\Gamma_{l''}(C) = 0$ then $Q^B_{l''} = \emptyset$, which implies that $Q^F_{l'} \neq \emptyset$ and $\Gamma_{l'}(C) > 0$. A third possibility is that $Q^F_{l'} \neq \emptyset$ and $Q^B_{l''} \neq \emptyset$, which yields $\Gamma_{l'}(C) > 0$ and $\Gamma_{l''}(C) < 0$.

It is in neither of these cases possible that $\Gamma_{l'}(C) = \Gamma_{l''}(C)$. Thus there is a contradiction, which proves the theorem. □

This verifies that the existence of a valid cycle implies that a compatible set of weights does not exist, regardless of whether or not the SP-graphs span all nodes. However, the fact that no valid cycle exists is not enough to guarantee that compatible weights exist. Since valid cycles are obtained by comparing only two SP-graphs, it is important to include as much information as possible in each SP-graph. This is done by combining as many sets of desired paths as possible into each SP-graph.

There are instances that do not have compatible weights or valid cycles. In such cases, there are more complicated structures that can be used to explain the lack of compatible weights, see [8]. However, such structures cannot be found as efficiently as valid cycles. By combining sets of paths, one might transform such a complicated structure into a valid cycle, thereby enabling the usage of the more efficient method in Section 5..

Lemma 10 *A valid cycle must contain at least three nodes and three arcs.*

A proof of Lemma 10 is given in [11]. A stronger result is true if subpath consistency holds.

Lemma 11 *If the SP-graphs are subpath consistent, a valid cycle must contain at least four nodes and four arcs.*

Proof: Since an SP-graph may not contain a directed cycle, a valid cycle with three arcs must contain two arcs in F and one in B (or two arcs in B and one in F). Then the two arcs of the same type, F (or B), must be adjacent, and thus form a path from one node, i, to another, j. Furthermore, the remaining arc, in B (or F), must also lead from i to j, in order to complete the cycle. So there are two different desired paths from i to j. Now either both of these paths are contained in both SP-graphs, in which case no arc in this cycle can be eligible, or both of these paths are not contained in both SP-graphs, in which case the SP-graphs are not subpath consistent. □

The same reasoning can be used to show that, under subpath consistency, a valid cycle with four arcs and nodes cannot contain adjacent arcs in F (or B). However, as shown in Section 7., a valid cycle can contain four arcs and nodes if it is alternating between arcs in F and in B. One might want to use this in the method presented in Section 5., but unfortunately there is probably no way of checking subpath consistency that is significantly faster than the methods in Section 5.. If this is true, it is best to use our methods for simultaneously checking for valid cycles and subpath consistency.

Lemma 12 *If two SP-graphs are subpath inconsistent, there exists a valid cycle.*

Proof: If A_k and A_l are subpath inconsistent there exists some $s \in N(A_k) \cap N(A_l)$ and $t \in N(A_k) \cap N(A_l)$ such that $A^S_k(s,t) \neq \emptyset$, $A^S_l(s,t) \neq \emptyset$ and $(A^S_k(s,t) \setminus A^S_l(s,t)) \cup (A^S_l(s,t) \setminus A^S_k(s,t)) \neq \emptyset$. A valid cycle is now obtained by letting B be a path from s to t in $A^S_k(s,t)$ and F a path from s to t in $A^S_l(s,t)$, where at least one arc in B lies in $A^S_k(s,t) \setminus A^S_l(s,t)$ or at least one arc in F lies in $A^S_l(s,t) \setminus A^S_k(s,t)$. □

Furthermore, there are examples (see Section 7. and [11]) with SP-graphs that are subpath consistent, but still have valid cycles. Therefore we can draw the following conclusion.

Lemma 13 *The absence of a valid cycle is a stronger necessary condition for the existence of compatible weights than subpath consistency.*

5. Methods for finding valid cycles

A practical method for finding valid cycles must consider each pair of SP-graphs, $l' = 1, \ldots, m-1$ and $l'' = l'+1, \ldots, m$. The arcs in one of the SP-graphs will be used backwards, if they are included in the cycle, and are therefore labeled with B. The arcs in the other SP-graph are used forwards, if they are included in the cycle, and are therefore labeled with F. Arcs with

one label, B or F, are eligible, while arcs labeled with both are not, and arcs with no label are not used at all. This can be seen as constructing a graph $\bar{G}$ which contains the arcs in one of the SP-graphs and the reversed arcs in the other SP-graph.

After this, one can either remove parts of the graph that cannot be a part of a cycle, or start to search for a valid cycle. The first approach is used in **VC0**, the method given in [7], and in **VC1**, the first of our new methods, while the second approach is used in **VC2**, the second of our new methods.

After removing arcs and nodes that cannot be a part of a feasible cycle, VC0 proceeds as follows. An eligible arc is chosen, and we try to find a feasible cycle containing this arc. If there exists no such cycle, the eligible arc is removed. If an isolated subgraph (a subgraph with in-degree or out-degree zero containing no eligible arc) is found in the process, this subgraph is also eliminated. This kind of graph reduction is repeated until the whole graph is eliminated, or a valid cycle is found.

The new variations of the method are based on *strongly connected components* of $\bar{G}$ (i.e. subgraphs containing a directed path between each pair of nodes). Clearly each feasible cycle corresponds to a directed cycle in $\bar{G}$, and all nodes in a directed cycle must belong to the same strongly connected component of $\bar{G}$. Arcs between two strongly connected components cannot be part of a feasible cycle.

Lemma 14 *A valid cycle lies within one strongly connected component of $\bar{G}$.*

A valid cycle must contain an eligible arc, so if a strongly connected component contains no eligible arc, it contains no valid cycle. On the other hand, if it contains an eligible arc, this arc is included in at least one directed cycle of $\bar{G}$ (i.e. in at least one feasible cycle), so a valid cycle exists. We have thus proved the following.

Lemma 15 *A valid cycle exists if and only if a strongly connected component of $\bar{G}$ contains an eligible arc.*

Lemma 10 states that a valid cycle contains at least three nodes, so strongly connected components that consist of less than three nodes cannot contain valid cycles. One may also note that all arcs in $\bar{G}$ are reversed if the order of the SP-graphs are changed. The strongly connected components are the same in the reversed graph as in the original graph, so it suffices to investigate each pair of SP-graphs once.

We now specify the two new methods for finding valid cycles. The first method, VC1, finds strongly connected components using Tarjan's method, [22], and is similar to VC0 in that it uses reductions in order to reduce the size of the graph.

The following parts of $\bar{G}$ cannot be part of a valid cycle, and can therefore be removed.

• All arcs with endpoints in two different strongly connected components.

• All strongly connected components with less than three nodes.

• All strongly connected components that does not contain any eligible arc.

Comparing to VC0, we find the following. The main difference is that we do not need to return to the reduction phase. Eliminating a whole strongly connected component does not affect the other components. Furthermore, VC0 needs to use a shortest path method in order to try to find a valid cycle, starting from one endpoint of a certain eligible arc and searching for a path to the other endpoint of the eligible arc. If no path exists between these two nodes, the arc is removed, and the original reduction phase is reentered. In the new methods, however, we know that a valid cycle exists and it can easily be found by a simple depth first search (DFS) between the endpoints of the eligible arc. Just as in VC0, we have the following result.

Lemma 16 *If the graph is completely eliminated by the reduction phase, there exists no valid cycle with the two SP-graphs considered.*

Let us now present this method in an algorithmic form. First we give Tarjan's method. S is a stack.

Tarjan's method

(1) Set $k = 1$. Set $v(i) = 1\ \forall i \in N$. Set $S = \emptyset$.
(2) For each $i \in N$, if $v(i) = 1$ then do SEARCH(i).

Procedure SEARCH(k)

(1) Set $v(i) = 0$, $n(i) = k$ and $k = k + 1$.
(2) Set $l(i) = n(i)$, and push i onto S.
(3) For each node j such that $(i, j) \in A$ do:
  If $v(j) = 1$ then
  (a) SEARCH(j).
  (b) Set $l(i) = \min(l(i), l(j))$.
  else if $n(j) < n(i)$ and j is on S then $l(i) = \min(n(j), l(i))$.
(4) If $l(i) = n(i)$ then repeat pop x from S until $x = i$.

The main algorithm will be as follows.

Algorithm VC1

(1) **Choice of SP-graphs:** If all pairs of SP-graphs have been compared, go to 6. Otherwise choose two SP-graphs $A_{l'}$ and $A_{l''}$ not previously compared.
(2) **Graph construction:** Construct a graph $\bar{G}$ by adding the arcs in $A_{l'}$ to the reversed arcs of $A_{l''}$.

Arcs in $(A_{l'} \setminus A_{l''}) \cup (A_{l''} \setminus A_{l'})$ are marked eligible.

(3) **Reduction phase:** Find the strongly connected components in $\bar{G}$ with Tarjan's method.
Remove all arcs between two different strongly connected components.
Remove all strongly connected components with less then three nodes.
Remove all strongly connected components that do not contain an eligible arc.
Remove nodes with only one adjacent arc.

(4) **Cycle check:** If all nodes are eliminated: No valid cycle found. Go to 1. Otherwise, there exists a valid cycle. (Optionally: Terminate the method.)

(5) **Find valid cycle:** Form a valid cycle by finding an eligible arc $(i, j) \in A_{l''}$ and a path from node j to node i, or an eligible arc $(i, j) \in A_{l'}$ and a path from node i to node j. Terminate the method. No compatible weights exist.

(6) **No valid cycle found:** No valid cycle exists. Terminate the method. (Compatible weights may exist.)

If the optional stopping criterion in step 4 is used, we will know that there exists a valid cycle, but we will not find it. In step 5, the path is found by a simple depth-first search.

The second method, VC2, uses Kosaraju's method, [20], for finding strongly connected components. The method is based on two depth-first searches. The first DFS starts at an arbitrary node and number the nodes in postorder (i.e. the nodes are numbered in the order the DFS backs up from the nodes). The graph $\bar{G}$ is then reversed, and the next DFS starts at the node with highest number. A strongly connected component is found each time the second DFS terminates, and the nodes reached belong to the same component. As long as the second DFS have not reached all nodes, the DFS is restarted at the unreached node with highest number. When all strongly connected components are found, we investigate if some component contains an eligible arc. If this is the case, a valid cycle can be found by a DFS. The other possible result is that no strongly connected component contains an eligible arc, and Lemma 15 then tells us that no valid cycle exists.

Kosaraju's method

(1) Perform a DFS of $\bar{G}$ and number the nodes in order of completion of the recursive calls.

(2) Reverse the directions of every arc in $\bar{G}$.

(3) Perform a DFS on the reversed graph, starting the search from the highest numbered node according to the numbering assigned in step 1. If the DFS does not reach all nodes, start the next DFS from the highest numbered remaining node.

(4) Each tree in the resulting spanning forest is a strong component of $\bar{G}$

Algorithm VC2

Replace steps 3 and 4 in VC1 by the following:

3. **Find strongly connected components:** Find the strongly connected components of $\bar{G}$ with Kosaraju's method.
4. **Cycle check:** If no strongly connected component contains an eligible arc: No valid cycle found. Go to 1. Otherwise, there exists a valid cycle. (Optionally: Terminate the method.)

Theorem 4 *After a finite number of steps, algorithms VC1 and VC2 will terminate, either with a valid cycle or a proof that no valid cycle exists.*

The complexity of finding strongly connected components in a graph with n nodes and a arcs is $O(n + a)$ both with Tarjan's method and with Kosaraju's method, see [1] and [13]. These methods are used on graphs with $|N|$ nodes and at most $2|A_L|$ arcs, where $|A_L|$ denotes largest number of arcs in an SP-graph. An eligible arc can be found in $O(|A_L|)$, so if m is the number of SP-graphs, the complexity of VC1 and VC2 is $O(m^2(|N|+|A_L|))$. For complete SP-graphs (i.e. extremely much splitting) we would have $O(|A_L|) = O(|N|^2)$, while for simple spanning trees we get $O(|A_L|) = O(|N|)$, and for non-spanning SP-graphs $|A_L|$ might be even less.

If a valid cycle is found, modification of the SP-graphs can be made as described in [7], in order to produce compatible weights. In short, one can either make a feasible cycle infeasible, by removing an arc from one SP-graph, or make an improving cycle non-improving, by adding an arc to one SP-graph (and thereby making an eligible arc non-eligible). Moving an arc within an SP-graph may sometimes yield both effects at the same time. In [7], we only considered spanning SP-graphs, which often made removing arcs from SP-graphs impossible. Allowing SP-graphs that are not spanning, makes removing arcs from SP-graphs a possible alternative to adding or moving arcs.

6. The symmetric single path case

Some network operators require that the routing paths should be symmetric, which means that the same path should be used in both directions between two nodes. Some operators also require that the routing paths should be unique. We call this case the *symmet-*

ric single path case. Our models have been developed for the more general case, i.e. when the desired paths not necessarily have to be symmetric and unique, but we can deal with the symmetric single path case as well. For our models, the symmetric single path case is obtained by including a single path in each set S_k, and by ensuring that the reversed path is included in some other set $S_{k'}$.

The symmetric single path case has previously been treated by using undirected graphs. The desired routing paths are represented by undirected paths, and we let ϕ_l denote the desired path between node o_l and node d_l. Mathematical models for the symmetric single path case in undirected graphs are presented in [3]. A directed model also allows the same weight to be used, but may also give different weights for the different directions. Some operators require that the same weight should be used in both directions of a link, which is possible by using an undirected model (or a directed model with the constraints $w_{ij} = w_{ji}\ \forall (i,j) \in A$, see e.g. [5]).

Let us now describe how undirected paths can be converted to symmetric SP-graphs, so that our methods can be used. Two SP-graphs are introduced for each undirected path ϕ_l, and $A_{l'}$ consists of the arcs in the directed path from o_l to d_l visiting the same nodes in the same order as ϕ_l, and $A_{l''}$ contains the reversed arcs in $A_{l'}$. This way we will consider both "directions" of each undirected path, so all information from the given undirected paths is retained. Furthermore, the directed path in each SP-graph can be found in an undirected path, so no additional requirements have been introduced. The symmetric single path case has now been converted from undirected paths to directed SP-graphs, so P4 can be used for finding compatible weights and a valid cycle method can be used for explaining why no compatible weights exists. We may also use Lemma 7 and Lemma 8 in order to combine the SP-graphs as much as possible.

If this approach is used on undirected paths that are suboptimal, the constructed SP-graphs will be subpath consistent. It then follows from Corollary 2 that two SP-graphs with the same origin can be combined to an out-tree with two branches. The combined SP-graph contains the same paths as the original two SP-graphs, so it is subpath consistent with any other SP-graph. We can therefore use Lemma 7 repeatedly and combine all SP-graphs with the same origin to an out-tree with several branches. (Due to Lemma 8, we may also use subpaths of the original paths.) The same holds for SP-graphs with the same destination, so we may also construct SP-graphs formed as in-trees.

In [3] two necessary conditions for the existence of compatible weights, stronger than subpath consistency, are given for the undirected case, namely the *cyclic compatibility condition* and the *generalized cyclic compatibility*. In Section 7., examples from [3] are treated by the above approach. We find valid cycles in instances not satisfying the cyclic compatibility condition, in instances satisfying the cyclic compatibility condition and not satisfying the generalized cyclic compatibility condition, and in instances satisfying the generalized cyclic compatibility condition. In other words, these cyclic conditions are not stronger necessary conditions for the existence of compatible weights than the absence of valid cycles. We have not been able to prove that there exists a valid cycle for each instance without compatible weights where the cyclic compatibility condition or the generalized cyclic compatibility condition is not satisfied, but have not found any instance satisfying the cyclic compatibility condition and the generalized cyclic compatibility condition that has no valid cycle either.

In [3], no efficient method for checking if the cyclic compatibility condition or the generalized cyclic compatibility condition is satisfied is given. The straightforward way would be to enumerate for each edge all cycles containing the edge and checking all the desired paths with both endpoints on the cycle. Doing this, as described here, can obviously not be done in polynomial time. We believe that a more efficient approach is to convert the undirected routing paths to directed SP-graphs, and then search for valid cycles. This can be done in polynomial time.

7. Examples

Let us first study an example from [7]. Two subpath consistent SP-graphs are shown in Figures 2a-b. We construct the graph $\bar{G}$ by adding the arcs in A_2 and the reversed arcs in A_1, see Figure 2c. Note that arc $(1,6)$ is the only arc which is included in both SP-graphs, so all arcs except $(1,6)$ are eligible. The strongly connected components of $\bar{G}$ are $\{1,6\}$ and $\{2,3,4,5\}$, see Figure 2d. The first strongly connected component has only two nodes, and in addition contains no eligible arc, so it is removed. The second component contains several eligible arcs, so we know that a valid cycle exists. We get the cycle 2 - 4 - 3 - 5 - 2, and conclude that there do not exist any compatible set of weights for these two SP-graphs.

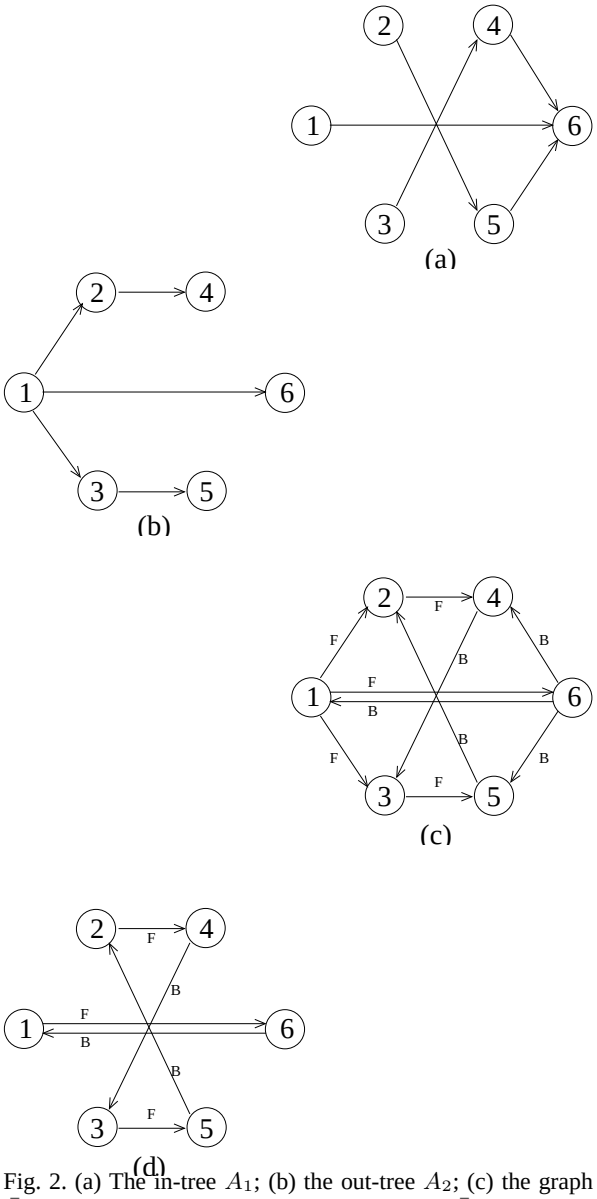

Fig. 2. (a) The in-tree A_1; (b) the out-tree A_2; (c) the graph $\bar{G}$; (d) the strongly connected components of $\bar{G}$.

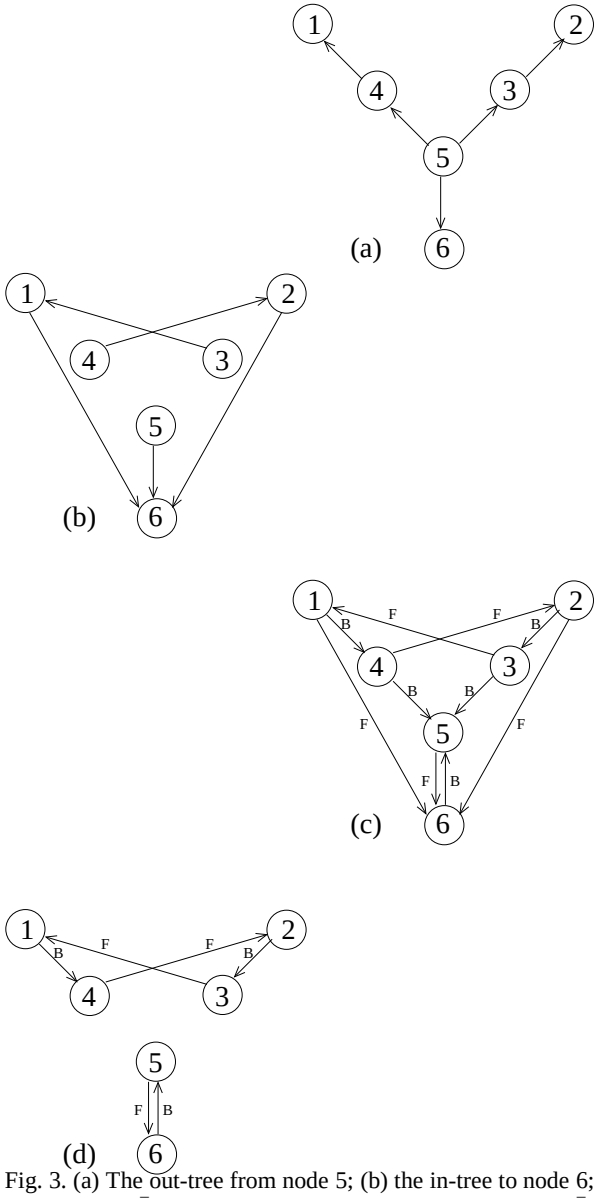

Fig. 3. (a) The out-tree from node 5; (b) the in-tree to node 6; (c) the graph $\bar{G}$; (d) the strongly connected components of $\bar{G}$.

The next example comes from [3]. Five undirected paths are given; 5-6, 1-4-5, 2-3-5, 3-1-6 and 4-2-6. The paths do not satisfy the cyclic compatibility conditions, so we know that there do not exist any compatible weights. The directed SP-graphs are constructed as described in Section 6.. Figures 3a-b show those that give a valid cycle. When a valid cycle method is applied, these SP-graphs are combined to the directed graph in Figure 3c, and its strongly connected components are $\{1, 2, 3, 4\}$ and $\{5, 6\}$, see Figure 3d. The last one does not contain any eligible arc, but the first one does, so a valid cycle is found.

In [3] another, similar, example is given, which satisfies the cyclic compatibility condition, but not the generalized cyclic compatibility condition. However, exactly the same valid cycle as found in the previous example is present. So, at least in these two examples, the difference between the cyclic compatibility condition and the generalized cyclic compatibility condition seems to be unrelated to the possible existence of a valid cycle.

Our next example also comes from [3]. It is based on the undirected graph and the undirected routing paths in

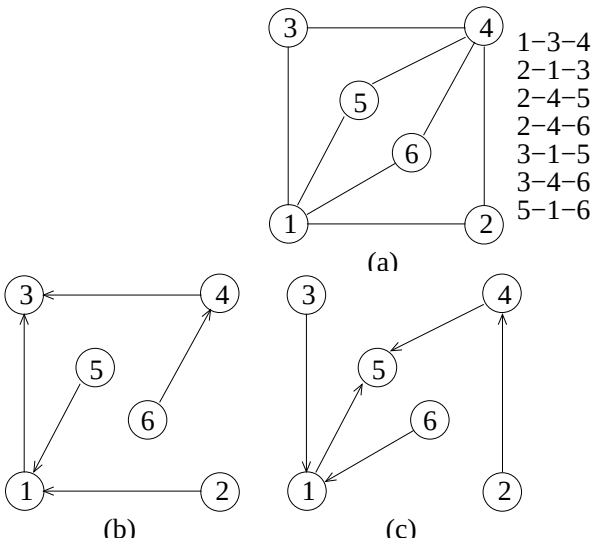

Fig. 4. (a) Undirected graph and routing paths; (b) the in-tree to node 3; (c) the in-tree to node 5.

Figure 4a. These paths satisfy all necessary conditions presented in [3], but there exists no compatible set of weights.

A set of SP-graphs is constructed as described in Section 6.. Now we apply a valid cycle method, and a valid cycle is found when the in-tree to node 3 and the in-tree to node 5 are investigated, see Figures 4b-c. The valid cycle method finds three strongly connected components, namely $\{1, 2, 4, 6\}$, $\{3\}$ and $\{5\}$. The first component consists of four eligible arcs, and a depth first search results in the valid cycle 1-6-4-2-1. If we analyze the valid cycle, we find that the eligible arcs originate from the routing paths 2-1-3, 3-4-6, 2-4-5 and 5-1-6. It can be verified that discarding any of these routing paths eliminates this particular conflict.

Let us finally consider an example which shows the advantages of combining SP-graphs. There is no valid cycle for the SP-graphs $A_1 = \{(1,3),(3,2)\}$, $A_2 = \{(1,4),(4,5)\}$, $A_3 = \{(6,4),(4,2)\}$ and $A_4 = \{(6,3),(3,5)\}$. Now combine A_1 and A_2 into A_5, and A_3 and A_4 into A_6, which is possible according to Lemma 7. Applying VC1 or VC2 to A_5 and A_6 yields the strongly connected components $\{1\}$, $\{6\}$ and $\{2, 3, 4, 5\}$. The last component contains several eligible arcs, forming the valid cycle 2-3-5-4-2.

8. Implementation and computational tests

Computational tests are performed on random networks with 10 to 90 nodes (RAND) and on real life networks from the survivable network design data library [21]; COST266, DI-YUAN, FRANCE, GERMANY50, and PIORO40. We use five different groups of test instances; RC, TR, RND, RM1 and RM5. The computer used is a 2x750 MHz Sun-Blade-1000 with 3Gb physical memory.

The test instances have been generated by computing spanning shortest path graphs to each node in the network with respect to a set of weights, and then changing one or several SP-graphs such that it is not certain that compatible weights exist. The instances in group RC are generated by adding one arc to a random SP-graph. (The added arc belongs to a shortest path if the weight is decreased by one.) A random arc is added to a random SP-graph in TR, and a random number (between 1 and $|N|$) of random arcs are added to random SP-graphs in RND. The instances in group RM1 are generated by removing one arc from a random SP-graph, and the instances in group RM5 are generated by repeating this five times. We ensure that each SP-graph is connected. Furthermore, an instance is discarded if a modified SP-graph contains a directed cycle, and this is why groups TR and RND sometimes contain fewer instances.

We have also generated test instances with non-spanning SP-graphs. A spanning SP-graph contains routing paths from one or several origins to one destination, and we generate non-spanning SP-graphs by computing the sets $A_l^S(o_l, d_l)$ for each origin and destination of each SP-graph. The non-spanning SP-graphs can be combined back to the spanning SP-graphs using Lemmas 7 and 8, so a non-spanning instance has compatible weights if the spanning instance it was generated from has compatible weights. This also means that the spanning and non-spanning versions contain the same information, so they constitute two different ways of solving the same basic problem.

Computational results for RAND networks and instances with spanning SP-graphs are summarized in Table 1. We use instances from groups RC and TR. N denotes the number of nodes in the group, I denotes the number of instances in the group, and N_A denotes the average number of arcs in the SP-graphs. Column N_V shows the number of instances with valid cycles, and the remaining two columns show the solution times for VC1 and VC2. These columns show the total solution time (in seconds) for all instances in the different

groups, so by dividing these numbers with the numbers of instances, the average time for one instance can be obtained.

Table 1

	N	I	N_A	N_V	VC1	VC2
RAND	10	40	9.8	18	0.02	0.08
	15	40	15.1	34	0.04	0.06
	20	40	20.6	33	0.09	0.10
	30	40	32.6	33	0.45	0.32
	40	40	42.8	39	1.10	0.78
	50	40	53.3	39	1.97	1.39
	90	40	101.4	38	11.30	8.48

Computational results for RAND *networks and spanning SP-graphs*

If we compare the solution times for VC1 and VC2, we find that VC1 is faster for smaller instances and that VC2 is faster for larger instances. VC0 was applied to the same random instances (with 10-30 nodes) in [6]. Comparing the solution times for VC0 with the ones obtained for VC1 and VC2, we find that the newer methods are up to 500 times faster then the original method. VC1 and VC2 has time complexity $O(m^2(|A_L| + |N|))$ and VC0 has time complexity $O(m^2|N|^2|A_L|)$ (reduced to $O(m^2|N|)$ for the single path case), where m is the number of SP-graphs and A_L is the SP-graph with the largest number of arcs. However, VC0 was implemented in Tcl/Tk (www.tcl.tk) within the framework of the graphical package VINEOPT (www.vineopt.com). Tcl/Tk is a scripting language and much slower than an implementation in C (VC1) and C++ (VC2).

Table 2 shows computational results obtained for spanning SP-graphs and real-life networks. The solution times for the two methods are very similar. The average solution times per instance are smaller than 0.04 seconds for all groups, so we conclude that both methods work well for these types of instances.

In [6], the LP-problem P5 was solved for the same instances, with the code LPSOLVE (lpsolve.sourceforge.net). We find, for example, that the COST266 instances took approximately 40 seconds per instance on average, the PIORO40 instances approximately 200 seconds and the GERMANY50 instances approximately 500 seconds. We thus conclude that it is *much* quicker to use VC1 or VC2 than to try to solve P5. We also recall that P5 cannot be used for non-spanning instances; instead we would have to use P3, which has more constraints and will take even longer time to solve.

Table 3 shows computational results obtained for the RAND networks and the non-spanning SP-graphs. Column N_S shows the average number of SP-graphs in each group. As for the previous groups of test instances, we find that VC1 is faster for smaller instances and that VC2 is faster for larger instances.

We also find that test instances with non-spanning SP-graphs require *much* more solution time than the spanning ones. (Recall that these non-spanning instances contain the same paths as the spanning, and that they are separated as much as possible.) For an instance with 90 nodes, a valid cycle method investigates approximately 4000 pairs of spanning SP-graphs or 5600000 pairs of non-spanning SP-graphs when no valid cycle can be found. However, investigating a pair of spanning SP-graphs require only 13% more solution time than a pair of the non-spanning ones. Our conclusion is clearly to combine SP-graphs as much as possible.

9. Conclusions

The problem of finding weights that gives prespecified routing patterns in IP networks using the OSPF protocol is considered. We discuss models that yield compatible weights, and use them to analyze the situation when no compatible weights exist. We show that the absence of valid cycles is a necessary condition for the existence of compatible weights for possibly non-spanning SP-graphs. We present two new methods for finding valid cycles by investigating strongly connected components. Computational experiments indicate that the new methods are faster than the previously known method. We show that SP-graphs with certain properties can be combined into a single SP-graph. Computational experiments indicate that the methods are faster if SP-graphs are combined as much as possible. Furthermore, we show how to apply our methods to the symmetric single path case (instead of using undirected models).

An interesting direction for future research is to include a valid cycle method into a framework for finding the optimal design of an OSPF network, or into a framework for optimizing the performance of existing OSPF networks.

Acknowledgment: This work has been financed by the Swedish Research Council.

Table 2

Computational results for real-life networks and spanning SP-graphs.

	COST266: 37 nodes, 114 arcs					DI-YUAN: 11 nodes, 84 arcs				
	I	N_A	N_V	VC1	VC2	I	N_A	N_V	VC1	VC2
RC	30	38.56	25	0.47	0.34	30	10.94	3	0.02	0.05
TR	29	38.57	27	0.35	0.20	30	10.94	7	0.02	0.04
RND	30	38.96	30	0.08	0.03	30	11.33	15	0.01	0.06
RM1	30	38.51	26	0.36	0.26	30	10.76	2	0.03	0.05
RM5	30	38.40	30	0.03	0.03	30	10.45	7	0.02	0.03
	FRANCE: 25 nodes, 90 arcs					GERMANY50: 50 nodes, 176 arcs				
	I	N_A	N_V	VC1	VC2	I	N_A	N_V	VC1	VC2
RC	30	26.28	28	0.11	0.10	30	54.26	27	0.99	0.75
TR	30	26.28	28	0.10	0.12	29	53.91	29	0.50	0.27
RND	29	26.57	29	0.03	0.04	27	54.35	27	0.06	0.06
RM1	30	26.20	23	0.18	0.10	30	54.22	27	1.06	0.77
RM5	30	26.04	29	0.03	0.03	30	54.14	30	0.20	0.10
	PIORO40: 40 nodes, 178 arcs									
	I	N_A	N_V	VC1	VC2					
RC	30	42.18	27	0.39	0.32					
TR	30	42.18	27	0.49	0.27					
RND	30	42.64	30	0.05	0.04					
RM1	30	42.13	23	0.75	0.59					
RM5	30	42.03	30	0.11	0.10					

Table 3

Computational results for RAND *networks and non-spanning SP-graphs.*

	N	I	N_S	N_A	N_V	VC1	VC2
RAND	10	40	59.9	2.09	18	0.36	1.16
	15	40	115.4	2.89	34	1.55	3.11
	20	40	210.2	3.25	33	5.83	9.43
	30	40	422.8	4.10	33	48.44	59.36
	40	40	762.0	4.93	39	243.67	245.67
	50	40	1099.7	5.84	39	610.71	549.67
	90	40	3344.0	8.66	38	14422.50	10690.67

References

[1] Aho, A. V., Hopcroft, J. E., and Ullman, J. D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley 1974.

[2] Ben-Ameur, W., Michel, N., Liau, B., and Gourdin, E., "Routing strategies for IP networks", *Telektronikk* 2/3 (2001) 145–158.

[3] Ben-Ameur, W. and Gourdin, E., "Internet routing and related topology issues", *SIAM Journal on Discrete Mathematics* 17 (2003) 18–49.

[4] Bley, A., Grötschel, M., and Wessäly, R., "Design of broadband virtual private private networks: Model and heuristics for the B-WiN", in: Dean, N., Hsu, D. F., and Rav, R. (eds.), *Robust Communication Networks : Interconnection and Survivability* volume 53 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–16, AMS, 2000.

[5] Bley, A. and Koch, T., "Integer programming approaches to access and backbone IP-network planning", Technical Report ZR-02-41, Konrad-Zuse-Zentrum für Informationstechnik, Berlin 2002.

[6] Broström, P., *Optimization Models and Methods for Telecommunication Networks using OSPF*, Dissertation, Linköping University, Sweden 2006. Linköping Studies in Science and Technology. Dissertation no. 1032.

[7] Broström, P. and Holmberg, K., "Determining the non-existence of a compatible OSPF metric", Research Report LiTH-MAT-R-2004-06, Department of Mathematics, Linköping Institute of Technology,

Sweden 2004.

[8] Broström, P. and Holmberg, K., "Stronger necessary conditions for the existence of a compatible OSPF metric", Research Report LiTH-MAT-R-2004-08, Department of Mathematics, Linköping Institute of Technology, Sweden 2004.

[9] Broström, P. and Holmberg, K., "A new derivation of valid cycles", Research Report LiTH-MAT-R-2005-03, Department of Mathematics, Linköping Institute of Technology, Sweden 2005.

[10] Broström, P. and Holmberg, K., "Compatible weights and valid cycles in non-spanning OSPF routing patterns", Research Report LiTH-MAT-R-2007-04, Department of Mathematics, Linköping Institute of Technology, Sweden 2007.

[11] Broström, P. and Holmberg, K., "Valid cycles: A source of infeasibility in OSPF routing", Accepted for publication in *Networks* 7 (2008) 206-215.

[12] Callon, R. W. "Use of OSI IS-IS for routing in TCP/IP and dual environments". RFC 1195, 1990.

[13] Cormen, T., Leiserson, C., Rivest, R., and Stein, C., *Introduction to Algorithms*, The MIT Press 2001.

[14] De Giovanni, L., Fortz, B., and Labbé, M., "A lower bound for the internet protocol network design problem", in: Gouveia, L. and Mourao, C. (eds.), *INOC 2005*, pages 401–408, University of Lisbon, Lisbon, Portugal 2005.

Received 21 May 07; revised 1 Sept 08; accepted 15 Sept 08

[15] Farago, A., Szentesi, A., and Szviatovszki, B., "Allocation of administrative weights in PNNI", in: *8th International Telecom. Network Planning Symposium (NETWORKS'98)*, pages 621–626, 1998.

[16] Frank, A., "Connectivity and network flows", in: Graham, A., Grötschel, M., and Lovász, L. (eds.), *Handbook of Combinatorics* volume 1, pages 111–177, North-Holland, 1995.

[17] Holmberg, K. and Yuan, D., "Optimization of Internet Protocol network design and routing", *Networks* 43 (2004) 39–53.

[18] Malkin, G. "RIP version 2". RFC 2453, 1998.

[19] Moy, J. "OSPF version 2". RFC 2328, 1998.

[20] Sharir, M., "A strong-connectivity algorithm and its application in data flow analysis", *Computers and Mathematics with Applications* 7 (1981) 67–72.

[21] SNDlib 1.0. "SNDlib 1.0 – Survivable network design data library". http://sndlib.zib.de 2005.

[22] Tarjan, R., "Depth first search and linear graph algorithms", *SIAM Journal of Computing* 1 (1972) 146–160.

[23] Tomaszewski, A., Pioro, M., Dzida, M., and Zagozdzon, M., "Optimization of administrative weights in IP networks using the branch-and-cut approach", in: Gouveia, L. and Mourao, C. (eds.), *INOC 2005*, pages 393–400, University of Lisbon, Lisbon, Portugal 2005.