The Complexity of Restricted Variants of the Stable Paths Problem

Kevin Donnelly
Google Chicago
Chicago, USA
kdonn@google.com

Assaf Kfoury
Boston University
Boston, USA
kfoury@bu.edu

Andrei Lapets
Boston University
Boston, USA
lapets@bu.edu

March 15, 2010

Abstract

Interdomain routing on the Internet is performed using route preference policies specified independently and arbitrarily by each autonomous system (AS) in the network. These policies are used in the border gateway protocol (BGP) by each AS when selecting next-hop choices for routes to each destination. Conflicts between policies used by different ASs can lead to routing instabilities that, potentially, cannot be resolved regardless of how long BGP runs.

The stable paths problem (SPP) is an abstract graph theoretic model of the problem of selecting next-hop routes for a destination. A solution to this problem is a set of next-hop choices, one for each AS, that is compatible with the policies of each AS. In a stable solution each AS has selected its best next-hop if the next-hop choices of all neighbors are fixed. BGP can be viewed as a distributed algorithm for solving an SPP instance.

In this report we consider a family of restricted variants of SPP, which we call $\overline{f}$-SPP. We show that several natural variants of $\overline{f}$-SPP are NP-complete. This includes a variant in which each AS is restricted to one of only two policies, and each of these two policies is based on a monotonic path weight aggregation function. Furthermore, we show that for networks with particular topologies and edge weight distributions, there exist efficient centralized algorithms for solving $\overline{f}$-SPP.

1 Introduction

The internet is a collection of thousands of autonomous systems (ASs) connected in a complex graph. Each AS is administratively separate and has its own policies about what routes through the graph should to be used for traffic to each destination AS. The border gateway protocol (BGP) is a distributed algorithm used for selecting interdomain paths to each destination, and it makes use of policy preferences in making these selections.

However, routers within the network generally route based only on the destination address of a packet. Consequently, the choice of routes that may be taken by packets originating from any AS are constrained by the routes chosen by each neighbor of that AS, which are further constrained by the choices of their neighbors, and so on. It is possible that the policies of different ASs conflict in a way that causes permanent instability in the network. It may be that there is always some AS that wants to change its route selection in response to the current set of routes offered by its neighbors.

The stable paths problem, introduced by others in earlier work [5], is an abstract graph theoretic model of internet route selection. An instance of SPP is a graph of ASs with arbitrary policies about

preferences between routes. The problem of whether an SPP instance has a solution has been shown to be NP-complete [5].

In this paper we consider a restricted, though still natural, version of the stable paths problem in which routing policies are based on the weights assigned to each route path to a destination, and in which the weight assigned to each path is computed by applying some fixed aggregate function (e.g. addition, minimum, multiplication) to the individual weights of the edges that constitute the path. We show that for several choices of aggregate function, the SPP problem is NP-complete.

Our results show that even if policies are not selected arbitrarily but based on a small set of synthetic link metrics, SPP is still NP-complete. In particular, even if there are only two policies used, minimum latency and maximum bottleneck bandwidth, SPP is NP-complete.

2 BGP and the Stable Paths Problem

2.1 Conventions

Given a set S, we denote by S^* the set of finite sequences (i.e. tuples, vectors) with elements drawn from S (e.g. $\langle 1,0,1,1,0,0\rangle \in \{0,1\}^*$). As a convention, we use bar notation when writing a variable $\overline{v}$ that represents a tuple $\overline{v} \in S^*$. Sequence concatenation is denoted by the binary operator $\odot$, and individual components of a sequence or vector $\overline{v}$ are denoted by v_i where $i \in \{1,\ldots,|\overline{v}|\}$.

Given a directed graph $G = (V,E)$ with any single special or distinguished node $d \in V$ and $n = |V|$, we assume for convenience that $V = \{d, 1, ..., n-1\}$ and that there exists at most one edge between any two nodes. The set of node paths in G is a subset of V^* of tuples such that each adjacent pair of nodes v, v' in the tuple is an edge $(v, v') \in E$.

2.2 Border Gate Protocol Convergence: the Stable Paths Problem

We review a commonly used abstract formulation of the general border gate protocol convergence problem: the stable paths problem (SPP). The abstract is based on one that is used in recent work by others on BGP convergence and the complexity thereof [8]. It is similar to a formulation described in related work [5] except that it uses a directed graph.

Definition 2.1. We define an instance (V,E,d,r) of the *stable paths problem* (SPP) as follows. We are given an arbitrary network of ASs represented as a directed graph (V,E) where each node $v \in V$ represents an AS and each edge $(v,v') \in E$ represents a link from v to v'. We are also given a distinguished node $d \in V$, representing the particular *destination* to which each AS wants to send traffic. Finally, we are given a ranking map $r : V \times V^* \to \mathbb{N} \uplus \{\infty\}$ that defines for each node $v \in V$ a strict ranking of all cycle-free paths that lead from v to d (we use the weight ∞ to denote that a path is prohibited by the routing policy). This ranking represents the path preference policies of each node. For $v \in V$, $\overline{u}, \overline{u}' \in V^*$, if $r(v,\overline{u}) < r(v,\overline{u}')$ then v prefers path $\overline{u}$ to path $\overline{u}'$.

To solve the problem, we must determine whether there is a spanning tree subgraph of (V,E) with root at d such that no node in the graph prefers a different path (according to the ranking determined for it by r) leading to d than the one specified by the tree subgraph.

Definition 2.2. Given an SPP instance (V,E,d,r), a *configuration* $C \subseteq V^*$ of (V,E,d,r) is a directed spanning tree of (V,E) rooted at d such that the paths in C satisfy the following conditions:

1. if $\langle v \rangle \odot \overline{u} \in C$ then $r(v, \langle v \rangle \odot \overline{u}) \neq \infty$,
2. if $\langle v \rangle \odot \overline{u} \in C$ then $\overline{u} \in C$, and
3. if $\langle v \rangle \odot \overline{u} \in C$ and $\langle v' \rangle \odot \overline{u}' \in C$ and $\langle v \rangle \odot \overline{u} \neq \langle v' \rangle \odot \overline{u}'$ then $\overline{u} \neq \overline{u}'$.

In other words, C contains only permitted paths, the suffix of each path in C is also in C, and for each node $v \in V$, C contains at most one path starting from v. For every node $v \in V$, we define

$$C(v) = \begin{cases} \langle v \rangle \odot \overline{u} & \text{if } \langle v \rangle \odot \overline{u} \in C \\ \perp & \text{otherwise} \end{cases}$$

So $C(v)$ is the unique path from v to d that is an element of C, or else $\perp$ if there is no such path. The configuration can be interpreted as a set of routing next-hop choices and $C(v) = \perp$ means that v has no route to d.

The set $\mathsf{choices}(C, v)$ represents the legal path choices for v, given the current choices for his neighbors. This is defined to be

$$\mathsf{choices}(C, v) = \begin{cases} \{\langle v \rangle \odot C(v') \mid (v, v') \in E\} & \text{if } v \neq d \text{ and } r(v, \langle v \rangle \odot C(v')) \neq \infty \\ \{d\} & \text{otherwise} \end{cases}$$

Given the next-hop choices available to a node v, the weight of the best path from v is fixed. We define $\mathsf{best}(C, v)$ to be

$$\mathsf{best}(C, v) = \min(\{\infty\} \cup \{r(v, \overline{u}) \mid \overline{u} \in \mathsf{choices}(C, v)\})$$

Definition 2.3. The configuration C is *stable* if for (V, E, d, r) if for all $v \in V, \mathsf{best}(C, v) = r(v, C(v))$ (that is, no node prefers to change its next-hop choice, given that the choices of neighbors are fixed).

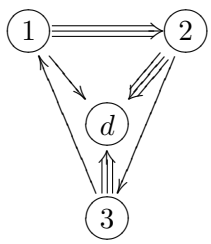
Definition 2.4. An instance (V, E, d, r) of SPP is *solvable* if there exists at least one stable configuration C for (V, E, d, r), and *unsolvable* if none of the configurations for (V, E, d, r) are stable.

Theorem 2.5 (SPP is NP-complete)**.** *The problem of determining whether an instance of* SPP *is solvable is NP-complete.*

Proof. It is clear that a configuration can be checked for stability in polynomial time, so SPP is in NP. In related work it is shown that 3-SAT can be reduced to SPP [5]. □

2.3 Examples

An instance of SPP can be succinctly described as a graph and a list of the permitted paths for each node in order of preference. Figure 1 shows a network with three nodes and a destination. The permitted paths for each node are listed in order of decreasing rank in the table to the right of the graph. This network has a unique solution that is highlighted with heavy edges. Figure 2 shows a network with the same topology and set of permitted paths, but this network has no stable configurations.

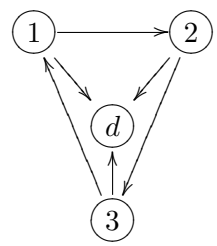


$1 \mapsto \langle 2, d\rangle, \langle d\rangle, \langle 2, 3, d\rangle$

$2 \mapsto \langle d\rangle, \langle 3, d\rangle, \langle 3, 1, d\rangle$

$3 \mapsto \langle 1, d\rangle, \langle d\rangle, \langle 1, 2, d\rangle$

Figure 1: Solvable instance of SPP (stable configuration highlighted with think arrows).



$1 \mapsto \langle 2, d\rangle, \langle d\rangle, \langle 2, 3, d\rangle$

$2 \mapsto \langle 3, d\rangle, \langle d\rangle, \langle 3, 1, d\rangle$

$3 \mapsto \langle 1, d\rangle, \langle d\rangle, \langle 1, 2, d\rangle$

Figure 2: Unsolvable instance of SPP.

3 A Restriction of the Stable Paths Problem

3.1 $\overline{f}$-SPP definition

Given that the stable paths problem is NP-complete, it is natural to ask whether there is a more restricted formulation of the problem that has a polynomial time solution and is still useful in practice. We approach this question by considering restrictions on the sorts of policies that are allowed. We define a family of restrictions, referred to as $\overline{f}$-SPP, and investigate the properties of some very simple (i.e. highly restricted), yet potentially computationally difficult, problems in this family.

$\overline{f}$-SPP restricts the possible routing policies of every node to those that can be generated by assigning weights to each edge in the network graph, and by combining the weights using one of a limited set of aggregation functions. Depending on the aggregation functions considered, such policies can still be very flexible. For example, it is possible to encode preference metrics such as minimum latency, maximum bottleneck bandwidth, minimum end-to-end packet loss probability, and even simpler metrics that have only a preferred next-hop for each destination.

In the restricted version of SPP, the provided graph (V, E) is coupled with a vector $\overline{w}$ of functions $w_i : E \to \mathbb{R}$ so that each edge is associated with a vector of n non-negative weights. Each dimension of weights corresponds to one of n different policies, each represented by an entry in a vector of functions $\overline{f}$, also of length n. Each node $v \in V$ is assigned a policy by a map $\pi : V \to \{f_i \mid 1 \leq i \leq |\overline{f}|\}$.

Each of the possible policies is represented by an aggregation function f_i in the vector $\overline{f}$. Each f_i is an aggregation function on sequences of non-negative integers that combines the weights of the edges in a path to compute an overall weight of the path. As an example, a policy f_i may prefer lowest latency paths:

$$w_i(v, v') = \text{“the latency of the link from } v \text{ to } v'\text{”}$$

where $f_i = +$. We provide a general definition below.

Definition 3.1. An *aggregation function* is an associative, commutative binary function $f \in \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ that is *computable in polynomial time*.[1] We abuse notation slightly and also use f to represent

[1]Note that the computability in polynomial time of the functions is essential for the NP-completeness results presented in this work. If the aggregate functions in $\overline{f}$-SPP instances are allowed to be non-polynomial or non-computable, then only NP-hardness results can be derived.

the application of the function to paths where if $e, e' \in E$ are edges and i is the index of f in $\overline{f}$ then

$$\begin{aligned} f(w_i(e)) &= w_i(e) \\ f(w_i(e_1), \ldots, w_i(e_n)) &= w_i(e_1) \oplus \ldots \oplus w_i(e_n) \quad \text{where} \quad x \oplus y = f(x, y) \end{aligned}$$

Definition 3.2. An instance $(V, E, d, \pi, \overline{w}, \overline{f})$ of the restricted stable paths problem $\overline{f}$-SPP is like an instance of the general SPP problem, with a few differences. The graph (V, E) is accompanied by a policy map $\pi : V \to \{f_i\}$ that assigns one of n policies to each node, a vector $\overline{w}$ of weight functions $w_i : E \to \mathbb{R}$ that collectively associate each edge with n non-negative weights (one for each policy), and by an n-component vector $\overline{f}$ of weight aggregation functions.

In other words, an instance of $\overline{f}$-SPP is a directed graph representing ASs and connections between them, an assignment of path weighting policies to each AS, and an assignment of edge weights for each policy. In this restricted version of the problem, any simple path from a node to the destination is permitted. The definitions for a *configuration* C for an instance $(V, E, d, \pi, w, \overline{f})$, and the definitions of choices and best are as before (except that the aggregate weight of a path is used as its rank). The definition and interpretation of a stable configuration remain the same, as well: in a stable configuration, every node prefers its path over all others given that the choices of neighbors are fixed. It is clear that any instance of the $\overline{f}$-SPP can be translated into an instance of SPP having the same topology and set of stable configurations (but not necessarily vice versa).

As an example, we can consider an instance of $\langle +, +, + \rangle$-SPP. In such an instance, π assigns to each node one of three policies, each of which uses addition as the aggregation function (but on the appropriate dimension of edge weights). Notice that if $|\overline{f}| = 1$, then the $\overline{f}$-SPP problem is simply the problem of finding the collection of minimum paths from all nodes to d in a weighted graph in which aggregate path weights are computed using the one function in $\overline{f} = \langle f_1 \rangle$.

3.2 Examples

We present some examples that are useful in understanding some of the complications encountered. In order to succinctly describe instances of $\overline{f}$-SPP, we use diagrams of graphs where the shapes of the nodes represent the policy used at that node. A square node follows the first policy, a circle node follows the second, and a double circle follows the third. This is denoted within the caption of each figure using the notation $\langle$ □,◯,◎$\rangle$ or $\langle$ □,◯$\rangle$. Each edge in a diagram is labelled with a vector of weights indicating the weight of that edge under each policy.

Example 3.3. Figures 3 and 4 show instances of $\langle +, +, + \rangle$-SPP. Figure 3 shows an instance that is equivalent to the SPP instance shown in Figure 1, and the thicker arrows highlight the same stable configuration. Figure 4 shows an instance of of $\langle +, +, + \rangle$-SPP that is unsolvable, and is equivalent to the SPP instance shown in Figure 2.

Example 3.4. Figure 5 shows an instance of $\langle +, + \rangle$-SPP with a stable configuration (highlighted by thicker arrows), and Figure 6 shows an instance of $\langle +, + \rangle$-SPP with no stable configurations. Notice that the only difference between these two instances is the second policy weight of the edges from node 4 to node d: in Figure 5 the weights are $\langle 1, 0 \rangle$ while in Figure 6 they are $\langle 1, 3 \rangle$. This small difference plays a role in the proof of the NP-completeness of $\langle +, + \rangle$-SPP, found in Appendix A.

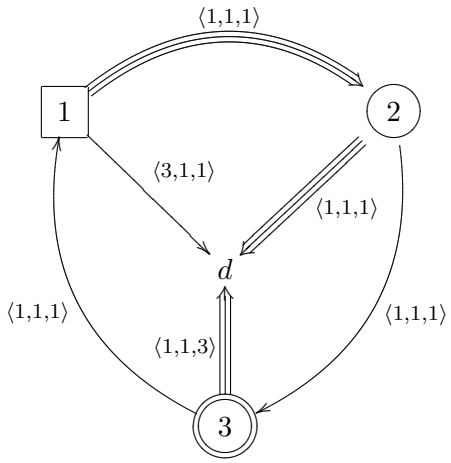


Figure 3: Solvable instance of $\langle +, +, + \rangle$-SPP with policies $\langle \square, \bigcirc, \circledcirc \rangle$.

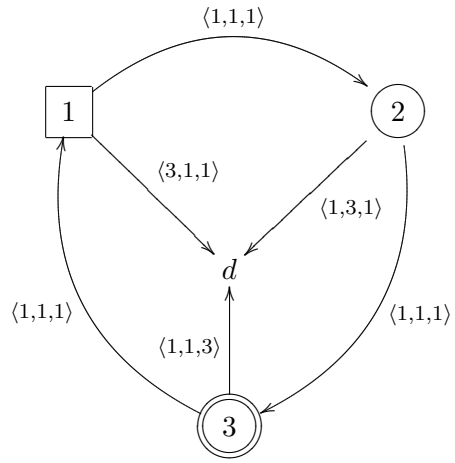


Figure 4: Unsolvable instance of $\langle +, +, + \rangle$-SPP with policies $\langle \square, \bigcirc, \circledcirc \rangle$.

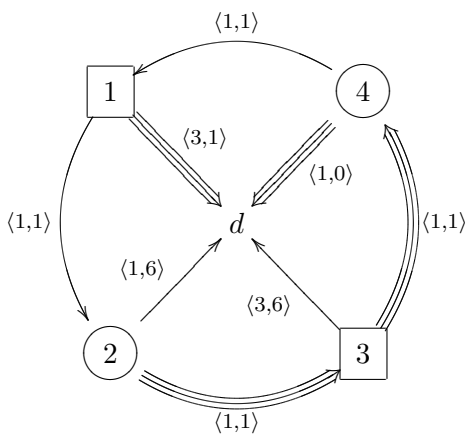


Figure 5: Solvable instance of $\langle +, + \rangle$-SPP with policies $\langle \square, \bigcirc \rangle$.

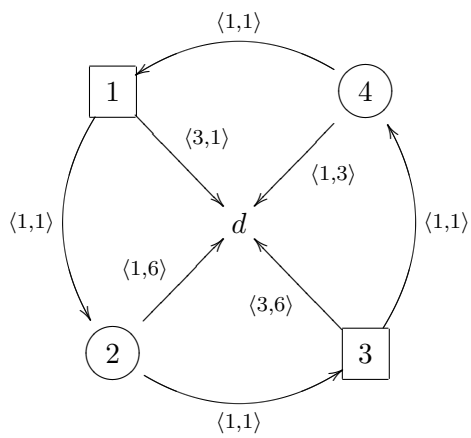


Figure 6: Unsolvable instance of $\langle +, + \rangle$-SPP with policies $\langle \square, \bigcirc \rangle$.

Example 3.5. Figure 7 illustrates an unsolvable instance of $\langle \max, \max \rangle$-SPP. Variants of this instance are used in the proof of Theorem 4.8, found in Appendix A.

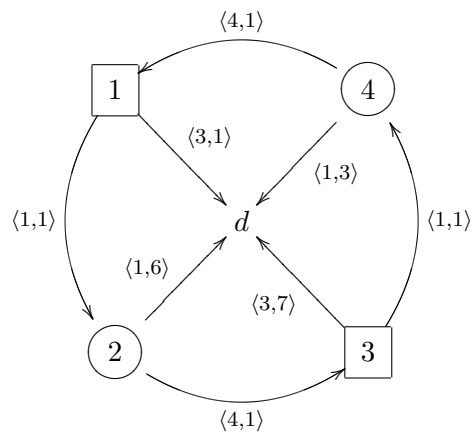


Figure 7: Unsolvable instance of $\langle \max, \max \rangle$-SPP with policies $\langle \Box, \bigcirc \rangle$.

3.3 Overview of Complexity Results

Table 3.3 summarizes the negative and positive results, presented in detail in Sections 4 and 5, about the complexities of various restrictions of SPP. All of the results in Table 3.3 are implied by the approach used in this work, and all the negative results listed as NP-complete in the table are implied by the result in the first row for $\langle f, g \rangle$-SPP with monotonic f, g. We also note that different approaches were used in earlier work to prove the negative results for $\langle +, +, + \rangle$-SPP and $\langle \max, \max, \max \rangle$-SPP [6], and also for $\langle +, + \rangle$-SPP and $\langle +, \max \rangle$-SPP [1].

In Table 3.3, instance of "..." can be replaced with any vector of zero or more aggregation functions. The NP-completeness of $\langle f, g \rangle$-SPP implies the NP-completeness of any variant $\langle f, g, \ldots \rangle$-SPP where $\overline{f}$ contains some finite number of functions in addition to f and g. This is because any instance of $\langle f, g \rangle$-SPP can be reduced to an instance of $\langle f, g, \ldots \rangle$-SPP.

4 Negative Results

The results in this section are "negative" in that they show some computational problems in the $\overline{f}$-SPP family to be NP-complete, and therefore (most probably) beyond resolution by any efficient algorithm, current or future.

4.1 Consequences for Other Abstract Models of BGP

Note that because $\overline{f}$-SPP is a restriction of SPP, the results in this section imply the NP-completeness of the general SPP problem.

complexity	SPP restriction	unsolvable graphs exist	notes
NP-complete	$\overline{f} = \langle f, g, \ldots \rangle$, f, g monotonic	yes	Section 4
	$\overline{f} = \langle \max, \max, \ldots \rangle$	yes	Section 4
	$\overline{f} = \langle +, +, \ldots \rangle$	yes	Section 4 and earlier work [1]
	$\overline{f} = \langle +, \max, \ldots \rangle$	yes	earlier work [1]
	$\overline{f} = \langle +, +, +, \ldots \rangle$	yes	earlier work [6]
	$\overline{f} = \langle \max, \max, \max, \ldots \rangle$	yes	earlier work [6]
P	$\overline{f} = \langle \text{first-hop}, f \rangle$	no	Dijkstra's algorithm
	$\overline{f} = \langle f \rangle$	no	Dijkstra's algorithm
	weights respect each other	no	Section 5
	weights respect topological distance	no	Section 5
	graph has a DAG topology	no	Section 5

Table 1: Negative and positive results for $\overline{f}$-SPP.

Theorem 4.1. *As long as all weights are non-zero, it is the case that for any $\overline{f}$, $\overline{f}$-*SPP *can be reduced to* SPP*.*

Corollary 4.2. *If for any $n = \left|\overline{f}\right|$ we have that $\overline{f}$-*SPP *is NP-complete, then* SPP *is NP-hard.*

Theorem 4.3. *For any given graph, there exists $\overline{f}$ of some dimension n such that* SPP *can be reduced to $\overline{f}$-*SPP*.*

4.2 $\langle +, + \rangle$-SPP and $\langle \max, \max \rangle$-SPP

In proving the difficulty of $\langle f, g \rangle$-SPP for appropriate functions f, g, it is possible to take advantage of the fact that exponentially many instances of a partially-specified $\langle f, g \rangle$-SPP problem have exponentially many valid solutions. This is based on a notion of an *abundant* problem [7] that is complementary to the notion of an *unambiguous* problem [9].

The proofs of the theorems in this section are constructed according to the following sketch. An existing NP-complete problem is chosen such that the space of possible solutions to this problem can be represented using the set of all paths through a particular subgraph of an $\langle f, g \rangle$-SPP instance. This graph is then extended to act like a "switch": if a path through the subgraph represents a solution to the NP-complete problem, then the configuration that creates a route consistent with this path is stable. A stable configuration in the graph *does not* exist if and only if there is no solution to the NP-complete problem.

Definition 4.4. Define the *subset sum* problem SubsetSum as follows. Given a collection $c_1, \ldots, c_n \in \mathbb{Q}$ of rational numbers, is there a subset of this collection that adds up to exactly 1?

Theorem 4.5. *There exists a polynomial-time reduction from* SubsetSum *to* $\langle +, + \rangle$-SPP*.*

The proof is presented in Appendix A.

Corollary 4.6. *Because* SubsetSum *is* NP*-complete,* $\langle +, + \rangle$-SPP *is* NP*-hard. Since* $\langle +, + \rangle$-SPP *is in* P*, it is* NP*-complete.*

A similar approach can be taken to prove the same result for $\langle \max, \max \rangle$-SPP.

Definition 4.7. Define the *Hamiltonian circuit* problem HamCircuit as follows. Given a directed graph (V, E), is there a cycle of length $|V|$ in the graph in which each node appears exactly once?

Theorem 4.8. *There exists a polynomial-time reduction from* HamCircuit *to* $\langle \max, \max \rangle$-SPP*.*

The proof is presented in Appendix A.

4.3 $\langle +, f \rangle$-SPP and Hamiltonian Path

An alternative proof of this result presented in previous work [1] involves building an $\langle +, + \rangle$-SPP problem instance using a directed graph G, where the in-degree of every vertex is at most 3, such that G has a Hamiltonian path iff the $\langle +, + \rangle$-SPP problem instance has a stable configuration.

Lemma 4.9. *Let G be a directed graph in which the in-degree of every vertex is at most* 3*. Determining whether G has a Hamiltonian circuit is an* NP*-complete problem.*

Theorem 4.10. $\langle +, + \rangle$-SPP *is* NP*-complete.*

Theorem 4.11. $\langle +, \max \rangle$-SPP *is* NP*-complete.*

The proofs are presented in earlier work [1].

4.4 $\langle f, g \rangle$-SPP

Thee proofs in Section 4.2 can be generalized. In particular, it is sufficient that the functions f and g satisfy certain properties that make them monotonic. Examination of the instance of $\langle f, g \rangle$-SPP in Figure 8 naturally implies the following definition.

Definition 4.12. An aggregation function $\oplus$ is *conducive to instability* if it satisfies the following equations:[2]

$$\begin{aligned}
1 &> 0 \oplus 0 \\
1 &< 0 \oplus 2 \oplus 1 \\
2 &< 0 \oplus 3 \\
1 &< 0 \oplus 0 \oplus 2 \\
2 &> 0 \oplus 0 \oplus 0 \\
1 &< 0 \oplus 2 \oplus 0 \oplus 0.
\end{aligned}$$

While the above definition is less restrictive than (but implies) the definition for monotonicity we present next, the notion of monotonicity is much more natural.

[2]These equation naturally induce a cycle of best responses in the graph in Figure 8: 1 in $\Rightarrow$ 4 out, 4 out $\Rightarrow$ 3 in, 3 in $\Rightarrow$ 2 in, 2 in $\Rightarrow$ 1 out, 1 out $\Rightarrow$ 4 in, 4 in $\Rightarrow$ 3 out, 3 out $\Rightarrow$ 2 out, 2 out $\Rightarrow$ 1 in.

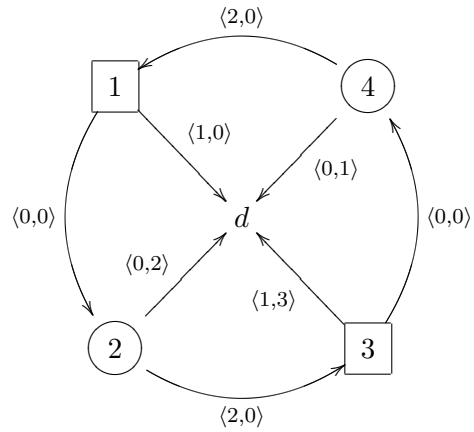


Figure 8: Unsolvable instance of $\langle f, g\rangle$-SPP if f, g are conducive to instability (with policies $\langle \Box, \bigcirc \rangle$).

Definition 4.13. An aggregation function $\oplus$ is *monotonic* iff it satisfies the following equations for all $x, y \in \mathbb{R}$:

$$0 \oplus x = x, \quad x \oplus 0 = x, \quad x \oplus y \geq x, \quad x \oplus y \geq y.$$

Proposition 4.14. *If an aggregation function f is* monotonic *then it is* conducive to instability*.*

Theorem 4.15. *If f and g are monotonic, there exists a polynomial-time reduction from* HamCircuit *to* $\langle f, g\rangle$-SPP*.*

The proof is presented in Appendix A.

5 Positive Results

The results in this section describe restrictions of SPP and $\overline{f}$-SPP problems that can be solved efficiently.

5.1 DAG topology

In the case of SPP or $\overline{f}$-SPP over a graph with a DAG topology, a stable solution always exists. The solution can be found using the following algorithm.

Definition 5.1. The "omnipotent algorithm" is defined in the following way. Given an instance with graph (V, E, d), it performs the following two steps.

1. Use a topological sorting algorithm to list the nodes in V such that no node has an edge directed towards a node later in the list. This can be done in time $O(|V| + |E|)$.
2. Starting from the front of the list, have each node select the best path currently available. Because each outgoing edge is considered only once, this can be done in time $O(|E|)$.

Theorem 5.2. *After one pass through the list, we have reached a stable configuration.*

Proof. We argue by induction that an invariant holds over the entire topologically sorted list of nodes: after a node chooses a path to the destination d, no better path will ever become available.

When we consider the paths of a node, the first hop of each path must be an edge directed towards a node earlier in the list, so the induction hypothesis applies to it. By induction, each of these next-hop nodes has already chosen the best path that will ever be available to it, so it will never change its chosen path. Therefore the best path currently available to the current node is the bast path the will ever be available to the current node.

At the end of one pass, each node has chosen the best path available to it, so the resulting configuration is stable. □

The algorithm described in Definition 5.1 above requires complete knowledge about the topology and policies of the graph, so it is not practical for use in a real network. In real networks, the BGP protocol chooses paths by repeatedly choosing the best available path given knowledge of the paths chosen by upstream neighbors, and then telling its downstream neighbors what its chosen path is. This protocol is also guaranteed to converge on a stable configuration in a DAG topology. Furthermore, if we assume the algorithm operates in synchronous rounds in which all nodes simultaneous pick their best available path and then inform neighbors of available paths, we can guarantee convergence after $O(|V|)$ iterations.

Theorem 5.3. *For a DAG topology and assuming synchronous operation at each node, the BGP protocol will converge to a stable configuration after at most $|V|$ iterations.*

Proof. We will show by induction on i that after the ith iteration, no node whose longest path to d has length i will ever change the path to d that it has selected. Assume we are on the ith iteration. There is some set of nodes whose longest path has length i. If the set is empty, we are done. Otherwise, for each such node, consider the set of nodes that it has edges directed towards. The longest path for each of these nodes has length strictly less than i. By induction, none of these nodes will change its path in this round or any subsequent round. Because the choice of paths in BGP does not change unless the paths advertised by neighbors change, the choice for this node will not change after this round. The longest path from any node to d is bounded by $|V|$, so we must have a stable configuration after $|V|$ rounds. □

5.2 More General Positive Results

It is possible to construct variants of the positive result in Theorem 5.3 by restricting the weights of a graph rather than its topology.

Definition 5.4. We say that the weights of a weighted graph *respect topological distance* if for every weight dimension i, for any two weighted edges e, e' and corresponding aggregate function $\otimes = f_i$, for every edge e'', $w_i(e) \otimes w_i(e') > w_i(e'')$.

Theorem 5.5. *For any graph with weights that respect topological distance, $\overline{f}$-SPP can be solved in polynomial time.*

Proof. The proof is a variant of the proof for Theorem 5.2, except that nodes are first sorted by topological distance from d. Once a node has chosen its best path, considering nodes that are more topologically distant from d can never produce a better path. □

Theorems 5.3 and 5.5 employ similar strategies. In both cases, a constraint on the graph ensures that there exists a partial order on nodes that respects SPP solutions and can be computed in polynomial time. In fact, it is sufficient that the dimensions associated with each edge correspond to each other.

Definition 5.6. We say that the weights in each dimension of a multi-dimensionally weighted graph *respect each other* if for every two weight dimensions i, j, for every node v, for every pair of paths $\overline{u}, \overline{u}'$ from v to d, $f_i(\overline{u}) \geq f_i(\overline{u}')$ iff $f_j(\overline{u}) \geq f_j(\overline{u}')$.

Notice that Definition 5.4 implies Definition 5.6, but not vice versa.

Theorem 5.7. *For any graph with weights that respect each other,* $\overline{f}$-SPP *can be solved in polynomial time.*

Proof. The proof is another variant of the proof for Theorem 5.2, except that nodes are first sorted by the weights along one of the dimensions rather than by distance from d. Once a node has chosen its best path, considering nodes that are more more distant from d along one dimension can never produce a better path along any other dimension. □

6 Related Work

There have been many approaches toward solving the BGP convergence problem, which fall into two broad categories: static and dynamic.

Dynamic approaches address the convergence problem by modifying the BGP protocol. BGP route flap damping [10] is a dynamic approach that attempts to detect routing oscillations and slow them down in order to reduce their negative effects on the network. However, this does not technically qualify as a solution to the convergence problem because it does not eliminate divergence if it exists. Furthermore, flap damping can also have the negative effect of slowing down progress toward convergence.

The safe path vector protocol [4] forces convergence by augmenting path update messages with a list of path update events that caused it. This path update history is then examined for cycles, which occur when an update by one AS indirectly *causes* an update in the same AS. Paths that cause such update cycles are removed from the set of permitted paths for the AS when the update cycle is detected. This approach provably leads to BGP convergence. However, it increases communication costs because each path update message much also include an update event history. Another disadvantage of this approach is that path update cycles are a necessary but not sufficient condition for divergence, so ASs are forced to change their policies even when convergence may have been possible under the original policies.

"Adaptive Policy Management" [11] is another dynamic solution to BGP convergence in which each AS keeps a local history of how many times it selects and then gives up a route. Using this count, ASs adapt their policies so as to attempt to cause convergence, while occasionally trying to revert to their more preferred paths, which may have become stable. This approach shares a mixtures of the benefits and drawbacks of the previous two approaches.

Existing static solutions achieve BGP convergence by strongly restricting the policies that are allowed. The hierarchical routing solution described in related work [2] requires all ASs to consistently categorize links as customer, peer or provider and to strictly prefer customer routes over peer or

provider route and peer route over provider route and, furthermore, to only route through a peer if the next hop after the peer is to a customer of that peer. This induces a DAG topology in the graph of possible routes which guarantees convergence.

The dispute wheel analysis [3] provides static method for proving the solvability of an SPP instance. However, this analysis is sufficient but not necessary for convergence. In fact, the criterion is extremely conservative and cannot prove solvability of any SPP instance that has multiple possible solutions. This means that it probably cannot be applied directly to complex, large-scale networks.

One might reasonably hope to develop further static solutions to the convergence problem by restricting the types of policies or network topologies that are allowed. In this work we have ruled out some such static solutions to the BGP convergence problem by identifying a broad and natural class of routing policies for which it is NP-complete to determine whether convergence is possible. In future work we hope to discover other combinations of policies and network topologies for which convergence is guaranteed or easy to decide.

References

[1] Kevin Donnelly and Assaf Kfoury. On the Stable Paths Problem and a Restricted Variant. Technical Report BUCS-TR-2008-001, CS Dept., Boston University, January 10 2008.

[2] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Netw.*, 9(6):681–692, 2001.

[3] Timothy Griffin, F. Bruce Shepherd, and Gordon T. Wilfong. Policy disputes in path-vector protocols. In *Proceedings of the 7th Annual International Conference on Network Protocols*, pages 21–30, Toronto, Canada, November 1999.

[4] Timothy Griffin and Gordon T. Wilfong. A safe path vector protocol. In *INFOCOM*, pages 490–499, 2000.

[5] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Netw.*, 10(2):232–243, 2002.

[6] Assaf Kfoury and Andrei Lapets. The NP-completeness of the Restricted Stable Paths Problem with Three Aggregating Functions. Technical Report BUCS-TR-2010-004, CS Dept., Boston University, March 2010.

[7] Andrei Lapets. The complexity of natural extensions of efficiently solvable problems. Technical Report BUCS-TR-2010-005, CS Dept., Boston University, March 2010.

[8] Michael Schapira. *The Economics of Internet Protocols.* PhD thesis, Hebrew University of Jerusalem, 2008.

[9] L. Valiant. The relative complexity of checking and evaluating. In *Inf. Process. Lett.*, 5:20–23, 1976.

[10] C. Villamizar, R. Chandra, and R. Govindan. BGP route flap damping, 1998. RFC 2439.

[11] Selma Yilmaz and Ibrahim Matta. An Adaptive Policy Management Approach to BGP Convergence. Technical Report BUCS-TR-2005-028, CS Department, Boston University, July 7 2005.

A Proofs

A.1 Proof of Theorem 4.5

Proof. We want to show that there exists a polynomial-time reduction from SubsetSum to $\langle +, + \rangle$-SPP. Given a collection of numbers $C = \{c_1, \ldots, c_n\}$ where $c_1, \ldots, c_n \in \mathbb{Q}$, and given a value ε such that $0 < \varepsilon \ll |c_i - c_j|$ for $i \neq j$, we can construct the graph presented in Figure 9.

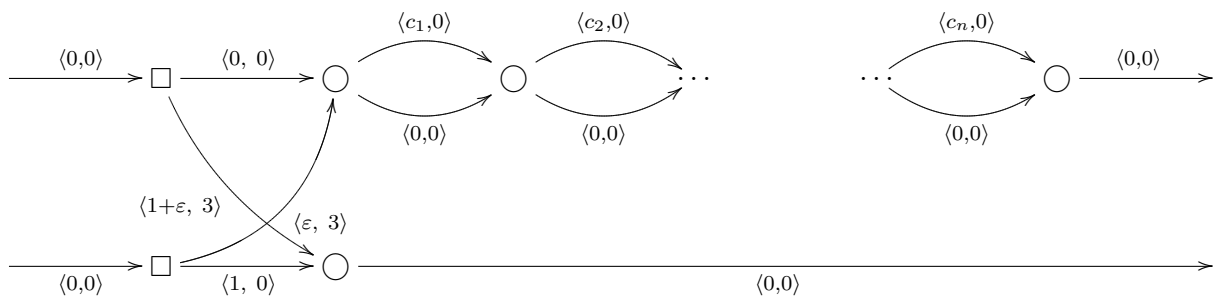


Figure 9: Routes through this graph correspond to solutions to the subset sum problem (with policies $\langle \square, \bigcirc \rangle$).

Notice that if there is a subset sum equaling exactly 1, this switch can route the top-left input along the top path of weight $\langle 1, 0 \rangle$ and the bottom-left input along the bottom path of weight $\langle 1, 0 \rangle$. If only paths of length not equal to 1 exist, it is always the case that either the bottom-left input will be routed up through a path with weight $\langle \varepsilon, 3 \rangle$ (if the sum is less than 1) or the top-left input will be routed down through a path of weight $\langle 1 + \varepsilon, 3 \rangle$ (if the sum is greater than 1).

If we take the unsolvable instance of $\langle +, + \rangle$-SPP in Figure 6 and replace the weights of the edge from node 4 to node d with $\langle \varepsilon, 3 \rangle$, or with $\langle 1 + \varepsilon, 3 \rangle$, the instance remains unsolvable. However, Figure 6 illustrates that if these weights are replaced with $\langle 1, 0 \rangle$, the instance becomes solvable. Thus, it is sufficient to create *two* copies of this instance that share the destination node d, and to replace the two directed edges $(4, d)$ and $(4', d)$. As illustrated in Figure 10, these edges are replaced with an instance of the switch graph in Figure 9 such that the top and bottom paths correspond to the two edges $(4, d)$ and $(4', d)$.

In order for this instance to be solvable, it is necessary for both copies to have a stable configuration. This can only occur if there is a configuration that contains the top and bottom paths in the switch graph, which can only occur if there is a subset of C with a sum of exactly 1. $\square$

A.2 Proof of Theorem 4.8

Proof. Given a graph $G = (V, E)$ for the Hamiltonian path problem, we create $n = |V|$ copies of the set of nodes and arrange them in an $G' = n \times n$ square with each copy of V arranged as a column and each row consisting of n copies of the same node (illustrated in Figure 11).[3] We use superscripts

[3]The particular edges between columns are included for purposes of illustration. Different instances of HamCircuit will produce different collections of edges between columns. Note that there are *no edges* between individual nodes within a column.

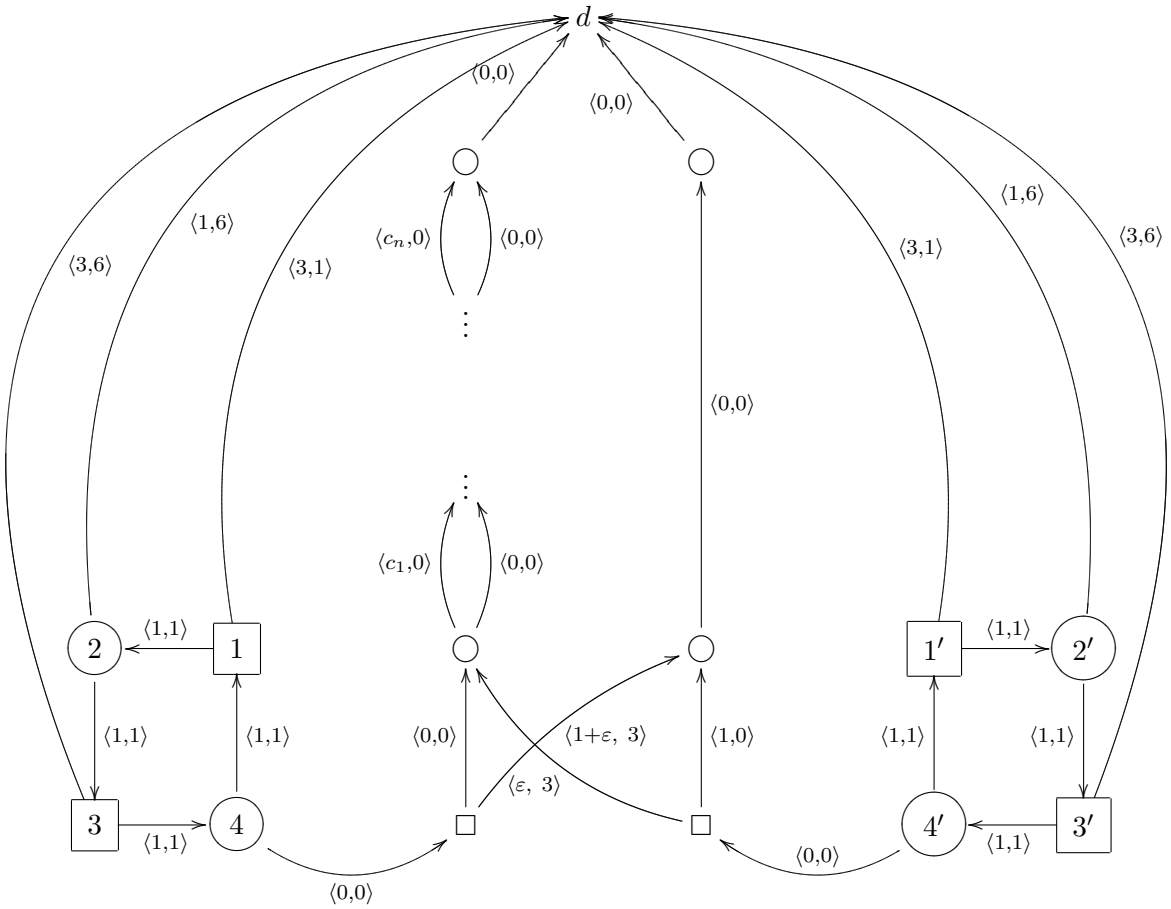


Figure 10: Reduction instance of $\langle +, + \rangle$-SPP with policies $\langle \square, \bigcirc \rangle$.

to refer to the column from which a copy of a node $v_\ell \in V$ is drawn (for example, $v_\ell^{(i)}$ is the copy of $v_\ell \in V$ from the ith column). If there exists an edge between $v_\ell, v_{\ell'} \in V$, then we create an edge $(v_\ell^{(i)}, v_{\ell'}^{(i+1)})$ for all $i \in \{1, \ldots, n-1\}$ in the graph. All these edges have weights $\langle 0, 0 \rangle$. We also add weights to the exit links on the right-hand side of the graph of the form $(3 \cdot 10^\ell, 0)$ where ℓ is the row of the exit link.

Suppose we now treat this new graph G' as an instance of $\langle \max, \max \rangle$-SPP. One of two cases must hold.

- If there exists a Hamiltonian path in G, then there exists a configuration over G' consisting of n separate, entirely disjoint routes starting at each of the nodes in the left-most column and ending at at the nodes in the right-most column.
- If there is no Hamiltonian path in G, then all configurations in G' are such that at least two copies $v_\ell^{(i)}$, $v_\ell^{(j)}$ (where $i \neq j$) of some node $v_\ell \in G$ have overlapping routes to d. If they have overlapping routes, then they must exit to d along the same exit link at row ℓ with some

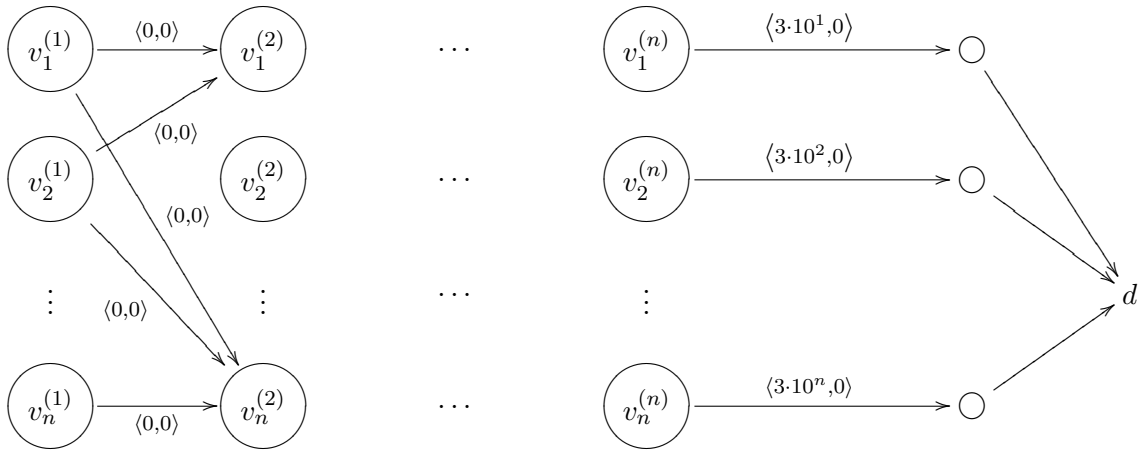


Figure 11: Instance of $\langle \max, \max \rangle$-SPP produced from HamCircuit instance with policies $\langle \square, \bigcirc \rangle$.

weights $\langle 3 \cdot 10^\ell, 0 \rangle$.

We can take advantage of the fact that one of the above two cases must hold. We create for every node $v_\ell \in V$, for every pair $v_\ell^{(i)}, v_\ell^{(j)}$ where $i \neq j$, n copies of a modified unsolvable $\langle \max, \max \rangle$-SPP instance (based on the one in Figure 7) as illustrated in Figure 12. Notice that there are about n^3 such modified copies: for each of the n^2 pairs of node copies where both copies are in the same row, for each possible index $\ell \in \{1, \ldots, n\}$, there is one such instance.

Given the index ℓ of a node in G, let $k = 10^\ell$. Notice that a copy of the instance becomes unsolvable exactly when the cost along the first dimension of both paths going through $v_\ell^{(i)}$, $v_\ell^{(j)}$ is $3 \cdot k$. For any $k' \geq 10 \cdot k$ or $k' \leq k/10$, the instance becomes solvable. This ensures that for any configuration in which some two copies $v_\ell^{(i)}, v_\ell^{(j)}$ share a path to d through some output node, there will exist at least one subgraph that has no stable configuration. The only stable configuration for G' will thus be one in which n disjoint copies of the Hamiltonian path through G are specified by the routes. $\square$

A.3 Proof of Theorem 4.15

Proof. The proof is identical to the proof of Theorem 4.8. It suffices to construct an appropriate unsolvable instance of $\langle f, g \rangle$-SPP that becomes solvable if one of the weights is increased or decreased by a power of 10, which is possible because f and g are monotonic. The rest of the construction is unchanged. $\square$

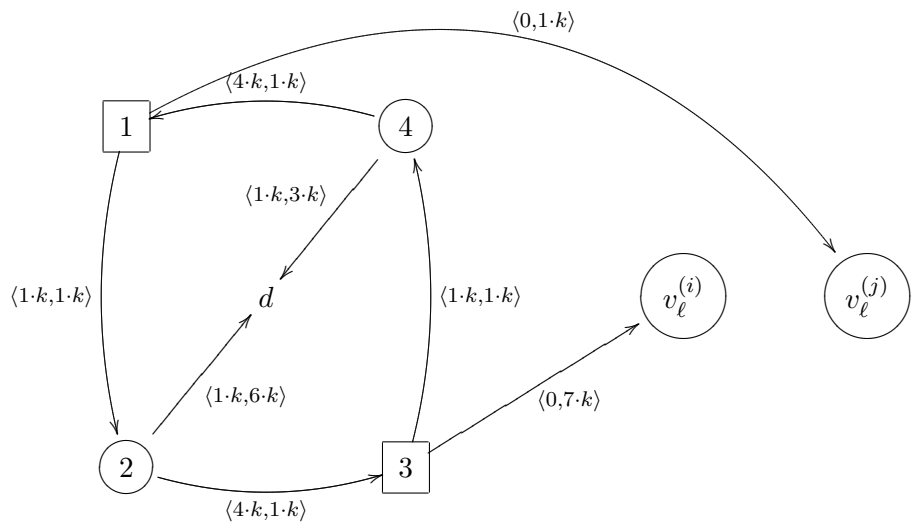


Figure 12: Modified version of unsolvable instance of $\langle \max, \max \rangle$-SPP with policies $\langle \Box, \bigcirc \rangle$ (one such instance exists for every $i, j, \ell \in \{1, \ldots, n\}$ where $k = 10^\ell$ and $i \neq j$).