Partial specification of routing configurations

Alexander J. T. Gurney[1] Limin Jia[2] Anduo Wang[1]
Boon Thau Loo[1]
[1]University of Pennsylvania, Philadelphia, USA
[2]Carnegie-Mellon University, Pittsburgh, USA

24 June 2011

Abstract

The formal analysis of routing protocol configurations for safety properties is well established. Methods exist to identify potential protocol oscillations by analysis of the network topology and route preference information. However, if not all of this information is available, then the existing theory does not apply. We present an analysis of *partial specification* of protocol instances and apply it to eBGP and iBGP examples, so that potential oscillations can be detected from the incomplete data. This technique is applicable to the incremental design of network configurations, where some parts of the design have been specified but others are not yet known. We also anticipate that automated tools could be used to 'fill in the blanks' of a partial configuration in some optimal way. To that end, we show how our analysis can be used to derive constraints on an IGP weight matrix, characterizing the set of possible weights that do not lead to BGP oscillation. We propose that these integer constraints could be used as part of a link weight optimization engine, to achieve some traffic engineering goal while not violating global stability.

1 Introduction

The Internet is a network of networks, operated by entities that are often in commercial competition, but who still seek global connectivity. It is therefore essential that the routing system be able to allow these networks to enforce policy that is at odds with what their neighbors might prefer. Unfortunately, this means that there will be some combinations of policy that are wholly incompatible, leading to protocol oscillation and other related problems. Various researchers have identified safety conditions, by which policy is restricted in order to ensure protocol convergence [1, 2, 3, 4]. But the formal treatment of these conditions assumes a model where all route preferences are known, in order for the verification to take place. In reality, many details about connectivity and policy will be unavailable. There is therefore a need for a theory which is able to speak about the correctness of configurations which are only partially known.

There are many aspects of a routing configuration which could be unspecified, including the protocol in use, the network topology, and the local configurations of each router. This paper focuses on the third of these—we assume that we are dealing with some network whose layout is known, and that the choice of routing protocol has been made (it is the Border Gateway Protocol [5]), but that protocol has not yet been fully configured. In particular, the *route preferences* may not be fully specified. We are interested in this scenario because it describes the state that a network operator may see when adjusting (configuring) parts of the router policy configuration, while other parts of the policy are set and the network topology known.

Given the partial specification, we still want to be able to discuss the convergence of the overall system, based on the information that we do know. In this paper, we develop a formalism for modeling partial specifications based on the well-studied *stable paths problem* (SPP) model of interdomain routing [6, 2]. Whereas the conventional SPP definition gives a degree of preference to all routes, our *partial stable paths problem* definition allows some routes to have unknown preference. We are able to show that for any such instance, there are two possible outcomes. One is that the known preferences are already problematic, which means that no matter how the remaining route preferences are resolved, there is a potential oscillation. The other is that there exists at least one way to assign preferences to the remaining routes so that the resulting configuration will be guaranteed to be stable. In other words, if there is no safe way to resolve the unknown preferences, then that fact can be detected purely by examining the known preferences.

One application for analyzing partial problems is the iterative design of network configurations, proceeding from an incomplete view of how the network should behave, and then specifying more and more details, until finally a complete configuration is reached. The partial SPP definition allows us to reason about a set of complete configurations—all those whose preferences are consistent with those already given. Given a partial SPP, we can then determine the entire set of *completions* of that SPP for which convergence is guaranteed. By considering all the ways in which preferences could be assigned to the unranked paths, we can determine a precise characterization of the conditions when extensions to the partial specification are 'safe'. The network operator can use this characterization to guide the incremental configuration.

This paper makes the following contributions.

- We propose a formal model (the partial stable paths problem) for partial specifications, and show how to discover potential oscillations in a sound manner.
- We identify a precise safety condition under which a partial specification has a safe extension.
- We explain how to use our formalism to aid policy configuration.

Roadmap The rest of this paper is organized as follows. In Section 2, we present the formal definition for partial SPP, and explain how to detect possible oscillations based on the partial specification. In Section 3, we use IGP weight assignment in iBGP as an example to motivate using our model to aid incremental policy configuration. We formally define the safety conditions of a partial SPP in Section 4, and continue with our running example, deriving integer constraints on the IGP weight values. Finally we discuss related work (Section 5) and future work (Section 6).

2 A Formal Model of Partial Specifications

In this section, we formally model partially specified policies in terms of a partial variant of the familiar stable paths problem. We also give a safety characterization, in terms of an auxiliary structure called the *paths digraph*.

2.1 The partial stable paths problem

Definition 1. A *partial stable paths problem* (PSPP) S is defined as a tuple $(G, d, \bigcup_{i \in N} P_i, \bigcup_{i \in N} \{(R_i, \leq_i)\})$, where

- $G = (N, E)$ is a graph, and N is the set of nodes and E is the set of directed arcs on N.
- d is a fixed destination node in N.
- P_i is a the set of *permitted paths* from i to d.
- R_i is subset of i's permitted paths P_i, and $\leq_i$ is a total order on R_i.

A PSPP is *well-formed* if P_d consists only of the empty path, from d to itself, and $R_d = P_d$. We only consider well-formed PSPPs. We say a PSPP is *full* if each node's total order is on the entire set of permitted paths (for all i, $R_i = P_i$). A full PSPP is an ordinary stable paths problem.

We can define a partial order among partial SPPs, and a notion of maximal element. For two PSPPs S and T, where

$$\begin{array}{rcl} S & = & (G, d, \{P_i\}_i, \{(R_i, \leq_i)\}_i) \\ T & = & (G', d', \{P'_i\}_i, \{(R'_i, \leq'_i)\}_i) \end{array}$$

we say T extends S (written $S \leq T$) if $G = G'$, $d = d'$, and:

- For all i in N, $P_i = P'_i$.
- For all i in N, $R'_i \supseteq R_i$; and if $p \leq_i q$ for some p and q in R_i, then $p \leq'_i q$ as well.

T is called a *completion* of S, if $S \leq T$ and T is full. Such elements are maximal according to the partial order (there is no element strictly exceeding T).

We write $\uparrow(S)$ to denote the set of all extensions of S, formally: $\uparrow(S) = \{T \mid S \leq T\}$. These are exactly the PSPPs which can be obtained from S by giving preferences to the paths which, in S, did not have specific preferences.

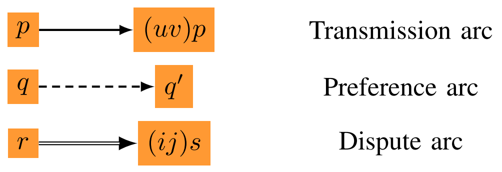


Figure 1: The three kinds of arc that might be present in a digraph.

2.2 Oscillation detection of partial specifications

Given a PSPP, one important question that we would like to answer is if the configuration is safe, as with the SPP theory. For a *partial* configuration to be safe, we mean that it is possible to complete it to a full configuration that has a unique solution (and therefore does not oscillate). If this is not possible then the partial configuration is already unsafe.

Previous work on routing stability in terms of stable paths problems has been founded on a *dispute digraph* structure [6]. Each SPP instance defines a dispute digraph instance. If the SPP instance represents a routing configuration which does not converge, then the accompanying dispute digraph must contain a cycle; so acyclicity of the digraph is a sufficient condition for convergence.

Unfortunately, the dispute digraph definition is not ideal for our purpose of investigating partial specification, especially the safe completions of partial specification. Given PSPP instances S and T, with $S \leq T$, it is not obvious how their dispute digraphs should be related, and so it is difficult to make inferences about the stability of T from the data given in S.

We use a related definition, the *paths digraph*, based on earlier work by Sobrinho [3]. The paths digraph structure *does* have a clear relationship to the partial order on PSPP instances, and so we are able to use it in examining partial specification.

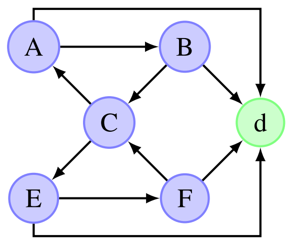
Definition 2. Given a PSPP S, the *paths digraph* is defined as follows. The nodes of the digraph are the permitted paths in S (elements in P_i). There are two kinds of directed arc:

1. An arc from p to q is a *transmission arc*, if q is the immediate extension of p.

2. An arc from p to q is a *preference arc*, if p and q are paths from the same source, and p is preferred to q.

We illustrate these arcs in Figure 1. (The 'dispute arc' is defined and used in the appendix.)

We say that S is *acyclic* if its paths digraph is acyclic. Note that the paths digraph is defined for an SPP, which is a full PSPP. The following theorem shows that acyclicity of the paths digraph is equivalent to acyclicity of the dispute digraph. (The proofs of this and of Theorem 2 are in the appendix.)



Node	Preferences
A	ABd $\prec$ Ad
B	BCEd $\prec$ Bd
C	$\{\text{CAd}, \text{CEd}\}$ unranked
d	d
E	EFd $\prec$ Ed
F	FCAd $\prec$ Fd

Figure 2: A PSPP that does not have a completion.

Theorem 1. *The dispute digraph of a PSPP S contains a cycle if and only if its paths digraph contains a cycle.*

Theorem 1 allows us to use the paths digraph to analyze safety. When we assign preferences to previously unranked paths, the corresponding operation on the paths digraph is simply to add new preference arcs. This means that we can detect cycles in the paths digraph before completing the configuration. The following theorem formalizes this idea, and characterises the instances for which we are able to 'fill in the blanks', and provide a complete, safe routing solution for the given partial configuration.

Theorem 2. *Let S be a PSPP. Then S has an acyclic completion if and only if S is acyclic.*

To give an example that has no safe completion, consider the PSPP in Figure 2. Its paths digraph is shown in Figure 3: it already contains a cycle, which does not go away when new preference arcs are added, and so this PSPP has no acyclic completion.

No matter which preferences are chosen at node C, an oscillatory configuration called a *dispute wheel* [2] is created. If CAd is preferred to CEd (written 'CAd $\prec$ CEd') then nodes A, B and C are in a dispute wheel: node A has ABd $\prec$ Ad, B has BCEd $\prec$ Bd, and C has CAd $\prec$ CEd. But if we instead choose CEd $\prec$ CAd, there is a dispute wheel among C, E and F, since CEd $\prec$ CAd, EFd $\prec$ Ed and FCAd $\prec$ Fd. These options correspond, in the paths digraph, to adding a new preference arc between CAd and CEd. This can be done in either direction, but there is still a cycle.

Since the absence of cycles from the dispute digraph of an SPP is sufficient for convergence [6], we obtain the following theorem.

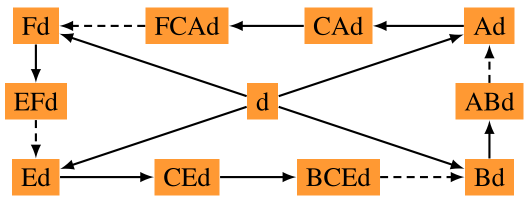


Figure 3: Paths digraph for the PSPP of Figure 2.

Theorem 3. *If a PSPP is acyclic then it has some completion which converges.*

Proof. From Theorem 2, a PSPP which has an acyclic paths digraph must have some completion whose paths digraph is also acyclic; by Theorem 1, its dispute digraph is acyclic, and so convergence is guaranteed. □

3 Guiding Incremental Configuration

The ability to assess the stability of partially specified instances means that stability problems can be detected before the entire network configuration has been given. Our ultimate goal is to aid the process of finding a safe configuration that is a completion of some given partial configuration. The completion process could be incremental, where new preferences are incorporated one at a time, and in such a way as to avoid creating cycles in the paths digraph. Instead of constructing a paths digraph and searching for cycles at each step, we can extract a condition to characterize all the ways that a given PSPP instance could be completed (so that all preferences are specified) without inducing a cycle—that is, given S, the subset of $\uparrow(S)$ containing only acyclic instances.

The advantages of obtaining such a condition include not only that it can be automatically checked, but also that it has the potential to be used by a partially or fully automated process to guide the configuration of the rest of the system.

In this section, we given an example of applying the partial SPP model to extract a safety condition; this is generalized in Section 4.

3.1 Stability of iBGP

Even within an autonomous system (AS), internal BGP (iBGP) presents stability concerns. In particular, interaction with the multi-exit discriminator (MED) attribute is known to be potentially problematic [7, 8, 9]. MED allows a neighboring AS to exercise control over egress point selection, overriding the normal decision procedure based on IGP distance (‘hot-potato’ routing). For particular choices of MED values and IGP weights, oscillations may occur.

In Figure 4, some combinations of IGP weight values will lead to oscillation and others will not. For example, if the weights are $w_{RS} = 1$, $w_{AR} = 5$,

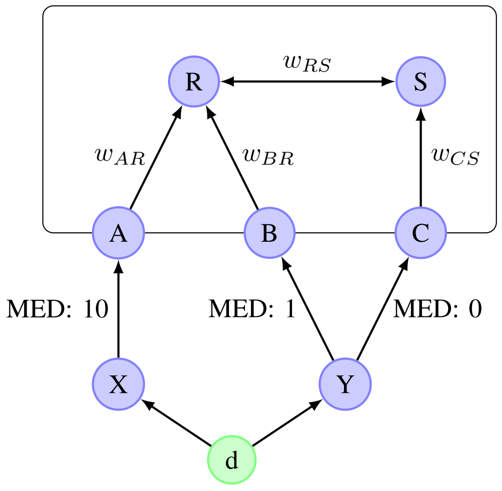


Figure 4: Fill in the blanks: what choices of w_{AR}, w_{BR}, w_{RS} and w_{CS} ensure stability?

$w_{BR} = 4$, and $w_{CS} = 12$ then an oscillation will occur [7]. But this would not be the case if, say, w_{RS} were raised to 20.

3.2 Modeling IGP weight assignment as a PSPP

Given the problemetic interactions between MED and IGP weights, it is desirable to have a process that derives IGP weights that are guaranteed not to cause such problems.

This situation is a good fit for our partial specification analysis. We will treat the IGP-induced preferences as unknown, with all other preferences (including those determined on the basis of MED) as known. Then, we can consider all of the ways in which the unknown preferences can be resolved, without creating a cycle in the paths digraph. Finally, each of these route preferences corresponds to an inequality on IGP weight values: so in the end, we will have a condition composed of linear inequalities (for which see Section 4.2).

In this example, we assume that the IGP weights in the network shown in Figure 4 are unknown. The paths digraph for this partially specified instance is shown in Figure 5. We have drawn additional unoriented preference arcs, between pairs of paths whose preferences are determined by IGP weight. In a completion, there will be *some* preference expressed, but the directions are not yet known.

The number of combinations to check may be reduced, since not all preferences will be capable of being induced by the same IGP weight settings. In this example, we can use the monotonic rule of integer arithmetic ($a < b$ if and only

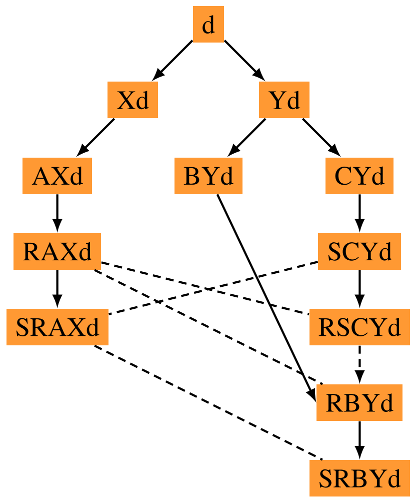


Figure 5: Paths digraph for the example of Figure 4. Dashed lines without arrowheads indicate preference arcs whose direction is yet to be determined.

if $c + a < c + b$) to observe that the preference between RAXd and RBYd must be the same as between SRAXd and SRBYd.

We obtain these three independent choices:

- RAXd versus RBYd,
- RAXd versus RSCYd,
- SRAXd versus SCYd.

There are therefore no more than eight ways of resolving the preferences in this example. By inspection, there are three of these cases where a cycle occurs.

- SRAXd $\prec$ SCYd, and RSCYd $\prec$ RAXd
- RBYd $\prec$ RAXd, and SRAXd $\prec$ SCYd
- RBYd $\prec$ RAXd, and RAXd $\prec$ RSCYd.

To ensure safety, an IGP weight assignment of the network in Figure 4 needs to avoid making any of these preference combinations hold. In other words, such assignment will guarantee the path diagraph to be acyclic, and by Theorem 3, the resulting routing policy converges.

4 Capturing All Safe Completions

The procedure above can be generalized to derive a *cycle formula* for any PSPP. This will capture the combinations of preference that induce cycles in the paths digraph. Its negation therefore captures the safe choices of preference.

In the case of the MED-IGP example, the unknown preferences relate to different choices of IGP weight values, which are integers. We can then translate the cycle formula into a formula on linear integer inequalities. This will describe the set of IGP weight matrices which can be considered safe.

4.1 The cycle formula

Definition 3. A *partially directed acyclic graph* (PDAG) is given by $G = (N, E_d, E_u)$, consisting of a set of nodes N, a set of directed arcs E_d, and a set of undirected arcs E_u, such that the graph (N, E_d) is acyclic [10].

The paths digraph of an acyclic PSPP defines a PDAG, where the undirected arcs are between any pair of nodes (from the same source to the same destination) whose relative preference is unspecified. These will be preference arcs in the completion, pointing one way or another, but for now we only know that there is *some* preference to be defined between those paths. The check for which completions are acyclic amounts to considering all possible orientations of these arcs, and finding which combinations induce cycles in the digraph.

We can trace paths and cycles in a PDAG, following both directed and undirected arcs.

Definition 4. For a path π in a PDAG, define its *enabling formula* to be the conjunction

$$E(\pi) = \bigwedge\{(p \prec q) \mid (p, q) \text{ is an unoriented arc in } \pi\}.$$

The enabling formula is intended to capture the preference decisions that need to be made, in completing a PSPP, for the given path to exist in the paths digraph. Note that $E(\pi)$ is always of finite length, since there are only finitely many unoriented arcs that π can traverse.

The formula includes the symbol '$\prec$' for strict preference. Logically, we will manipulate formulae under the assumption that $\prec$ is the strict part of a preorder relation; in particular, it is irreflexive and transitive. For example, for all p and q, the conjunction $(p \prec q) \wedge (q \prec p)$ is equivalent to falsity.

We can now consider which combinations of preference lead to a cycle in the paths digraph.

Definition 5. For a PDAG G, define the *cycle formula* to be the disjunction

$$C(G) = \bigvee\{E(\pi) \mid \pi \text{ is a cycle in } G\}.$$

Again, this $C(G)$ is always of finite length. It tells us which combinations of preference induce cycles in the paths digraph. Its negation $\neg C(G)$ does the opposite, characterizing the preference decisions that are guaranteed to be safe, formally:

Theorem 4. *Given a PSPP, S, let L be the negation of the cycle formula on its paths digraph. If L is satisfiable, then there is at least one T, with $S \leq T$, such that T satisfies L, and T converges.*

Proof. If L is satisfiable, then there is at least one way of assigning preferences to the unranked paths in S, so that no cycle is created in the paths digraph. Let T be one such resulting PSPP. Then $S \leq T$, and T satisfies L, and T has no cycle in its paths digraph, so T must converge. $\square$

Given a PSPP S, a cycle formula can be automatically constructed using a brute-force algorithm that enumerates all possible combinations of assignments of unknown preferences. The performance of this procedure is not critical for analysis, since the formula is pre-computed. We leave optimization of such an process for future work.

4.2 Transfer to the integer domain

The derivation above gives us a condition in terms of path preferences. For the networking application, we need to translate these preferences back into the language of BGP. We will obtain a version of the cycle formula that makes reference to the attributes of BGP routes, as opposed to abstract relations like '$p \prec q$'. In doing this, we can find that some combinations of preferences that were *apparently* feasible according to the cycle formula, are in fact unobtainable. This can happen if there is no way of configuring the BGP routing policy so that those preferences are the result. This kind of analysis and manipulation could be performed with the aid of an SMT solver capable of handling linear arithmetic, such as Yices [11].

In Figure 5, the unknown path preferences are induced by the IGP weights. Consequently, every preference clause '$p \prec q$' can be translated into an inequality between the weights of paths p and q, (If other attribute values were unknown, this complete rewriting would not necessarily be possible.) A straightforward translation is for the preference 'RAXd $\prec$ RSCYd'. To make this happen, the weight w_{AR} must be strictly less than the sum of weights $w_{RS}+w_{CS}$. So we can translate this preference into the integer inequality $w_{AR} - w_{RS} - w_{CS} < 0$. Each of the other path preferences can be handled in the same way, so we obtain a formula that contains only conjunctions and disjunctions of integer inequalities.

Based on the attributes which follow IGP distance in the BGP decision chain, we can determine whether each inequality should be strict. This yields a more precise characterization of the solution space. For example, when route reflectors are in use, the next attribute after IGP distance is the route reflector originator identifier [12]. If we want path p to be better than path q, and the originator for p has a lower identifier than that for q, then we know that the sum of IGP weights for p must be less than *or equal to* the sum for q. If the weights were equal, then the next tiebreaker would give us the desired preference in any case. But if the originator identifiers were ordered in the opposite sense, then the required IGP distance inequality would be a strict one.

An example of integer-based simplification is as follows. It appears that one possibility for assigning preferences is to have RSCYd better than RAXd, and at the same time SRAXd better than SCYd. But this cannot happen, since the inequalities $w_{RS}+w_{CS} < w_{AR}$ and $w_{RS}+w_{AR} < w_{CS}$ cannot be simultaneously satisfied for natural number weight values.

We finally obtain

$$\begin{aligned}&(w_{RS} + w_{CS} < w_{AR})\\ \vee\ &((w_{AR} < w_{RS} + w_{CS}) \wedge (w_{AR} < w_{BR}))\end{aligned}$$

as the negated formula. This identifies just two ways of avoiding a cycle in the paths digraph.

The mitigation advice in RFC 3345 is to make the link weight w_{RS} large [7], so that each reflector will select a best path from its local cluster, rather than needing to go via another reflector. This formula tells us how large that weight has to be compared to the other link weights, and also what other ways there are to avoid oscillation. For example, if w_{RS} were as small as 1, we would still converge if we set w_{CS} to 1 and w_{AR} to 3. If this could not be done due to w_{AR} also having to be small, then we could ensure convergence by increasing w_{BR}. In each of these cases, we are able to predict the consequent traffic flow.

The consideration of every safe completion has therefore not only revealed additional possibilities for how to make a partial configuration safe: it has also established precise numeric bounds on the integer parameters involved.

5 Related Work

Routing algebra The main alternative formalism to the SPP model is *routing algebra* [13, 4, 14]. There are various different structures used in this theory, but they share the general idea that route preferences can be considered independently from the network topology. This is, in a sense, a form of partial specification: once we have decided on the route *signatures* and on their relative preferences, and verified certain correctness properties, protocol convergence is guaranteed regardless of the layout of the network. Could our notion of partial specification of preference also be treated in an 'algebraic' way? Some steps in this direction have already been taken, since we have convergence results which apply even when some pairs of routes are *incomparable* [14], and a notion of *refinement* whereby those incomparabilities are resolved in one way or another, becoming strict preferences [15].

Indeed, this approach is very close to the 'partial SPPs' we treat in this paper; there are, however, some obstacles to using the algebraic definitions for the partial specification problem. The principal one is that in the algebraic theory, certain properties must hold of path preferences as a matter of definition: for example, the order must be transitive. Therefore, we are forbidden to even speak about certain situations where cyclic preferences occur—they simply cannot be formulated in terms of routing algebra. But we want to be able to say that certain rankings induce cyclic preferences, whereas others do not, and to capture exactly which preferences are involved in either case. We cannot do this, or at least not as straightforwardly, in the algebraic world.

For algebras satisfying the *strict inflationary* property, the set of all paths in the labelled graph can be put in preference order, where a path is always preferred over any extension of that path; and convergence is guaranteed in this case [14]. This viewpoint is compatible with the paths digraph structure, which has transmission arcs (for path extension) and preference arcs, but is required to be acyclic overall.

MED oscillation There is substantial literature on the 'MED problem' in BGP, extending from practical advice [7], to theoretical approaches [8], to proposed protocol changes [9]. Our partial specification analysis is aimed at understanding the shape of the parameter space for avoiding MED-induced oscillation, and assumes the traditional BGP mechanism. Knowing this, more precise advice can be given about which combinations of attribute values are desirable. So far, our approach has not been extended to multipath models, for which the oscillation problem may be alleviated [9].

6 Future Work

We would like to provide an automated tool that could derive the cycle formula from a given partial BGP configuration (or partial SPP instance). It is of concern that derivation of this formula could be very computationally difficult. As presented, we are dealing with a data structure whose components are *all simple paths* in a graph, and considering *all possible* ways of orienting the undirected arcs in a partially directed graph. There are mitigating factors: in the case of the IGP weight example, we could use facts about integer arithmetic to reduce the number of orientations to check, and to simplify the formula.

The next part of the story is to put the formula to good use. We anticipate that our analysis will be a good fit for traffic engineering methods: we provide safety constraints in terms of the integer parameter values, and then some automated process will tune those weight values to achieve some optimal outcome, subject to the constraints. This would mean integrating our generated safety constraint into an existing traffic engineering tool. We could then assess the quality of configurations that were output, as well as the method's performance, for various different scenarios.

Traffic engineering methods can be divided into two categories: there are heuristic techniques, making use of local search, genetic algorithms, and the like; and there are exact techniques, drawn from linear programming and related work in mathematical optimization [16]. The output of our analysis could be used in each of these settings. Firstly, the predicate that we derive can be used to screen candidate configurations in a local search process, such as tabu search [17]. In this way, the search can be restricted to only considering weight assignments that ensure stability. Secondly, the fact that the components of our predicate are linear inequalities suggests that there should be a more intelligent way to combine it with linear programming methods, though the presence of disjunctions is a complication.

Existing techniques for assigning link weights do so from the perspective of optimization, ignoring the fact that these weights are also used in BGP egress point selection. There are some approaches that attempt to capture BGP hot potato logic, by simulating the BGP decision process [18, 19, 20]. This does not directly address the stability issue: it is more concerned with the effect of BGP on the traffic matrix. Additionally, the use of a simulator is somewhat unwieldy, compared to a pure numeric method.

There may be other occasions where the partial specification theory is well matched with an automated configuration process. The theory could also be extended to handle scenarios where other kinds of partial information are involved, such as when the permitted paths are unknown, or when some detail of the protocol rules must be inferred.

Acknowledgement

We are grateful to Jennifer Rexford for discussions on the topic and potential applications of this paper. This research was supported by NSF grants IIS-0812270, CNS-0830949, CNS-0845552, CPS-0932397, TC-0905607, AFOSR Grant No: FA9550-08-1-0352 and Office of Naval Research Grant N00014-10-1-0365.

References

[1] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Transactions on Networking*, pp. 681–692, 2001.

[2] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 232–243, 2002.

[3] J. Sobrinho, "Network routing with path vector protocols: Theory and applications," in *Proc. ACM SIGCOMM*, 2003, pp. 49–60.

[4] T. G. Griffin and J. Sobrinho, "Metarouting," in *Proc. ACM SIGCOMM*, 2005, pp. 1–12.

[5] Y. Rekhter, T. Li, and S. Hares, "RFC 4271: A Border Gateway Protocol 4 (BGP-4)," 2006.

[6] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "Policy disputes in path-vector protocols," in *Proc. IEEE ICNP*, 1999.

[7] D. McPherson, V. Gill, D. Walton, and A. Retana, "RFC 3345: Border Gateway Protocol (BGP) persistent route oscillation condition," 2002.

[8] T. G. Griffin and G. Wilfong, "An analysis of the MED oscillation problem in BGP," in *Proc. IEEE ICNP*, 2002.

[9] A. Flavel and M. Roughan, "Stable and flexible iBGP," in *Proc. ACM SIGCOMM*, 2009.

[10] D. Dor and M. Tarsi, "A simple algorithm to construct a consistent extension of a partially oriented graph," Computer Science Department, University of California, Los Angeles, Tech. Rep. R-185, 1992.

[11] B. Duterte and L. de Moura, "A fast linear-arithmetic solver for DPLL(T)," in *Proc. CAV*, 2006, pp. 81–94.

[12] T. Bates, E. Chen, and R. Chandra, "RFC 4456: BGP route reflection: an alternative to full mesh internal BGP (IBGP)," 2006.

[13] J. Sobrinho, "An algebraic theory of dynamic network routing," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1160–1173, 2005.

[14] A. J. T. Gurney, "Construction and verification of routing algebras," Ph.D. dissertation, University of Cambridge, 2009.

[15] A. J. T. Gurney and T. G. Griffin, "Neighbor-specific BGP: an algebraic exploration," in *Proc. IEEE ICNP*, 2010.

[16] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for Internet traffic engineering," *IEEE Communications Surveys*, vol. 10, no. 1, pp. 36–56, 2008.

[17] B. Fortz and M. Thorup, "Increasing Internet capacity using local search," *Combinatorial Optimization and Applications*, vol. 29, no. 1, pp. 13–48, 2004.

[18] B. Quoitin and S. Uhlig, "Modeling the routing of an autonomous system with C-BGP," *IEEE Network*, vol. 19, no. 6, pp. 12–19, 2005.

[19] S. Cerav-Erbas, O. Delcourt, B. Fortz, and B. Quoitin, "The interaction of IGP weight optimization with BGP," in *Proc. International Conference on Surveillance and Protection*, 2006.

[20] S. Balon and G. Leduc, "BGP-aware IGP link weight optimization in the presence of route reflectors," in *Proc. IEEE INFOCOM*, 2009, pp. 316–324.

A The Paths Digraph and the Dispute Digraph

We previously defined the *paths digraph* and asserted that it was equivalent to the traditional dispute digraph, in the sense that one is acyclic if and only if the other was acyclic. We will now prove this assertion.

Compared to the paths digraph, the dispute digraph of an SPP lacks preference arcs, but instead has *dispute arcs* as well as transmission arcs. See Figure 1 for our notation for these three kinds of arc. A dispute arc exists from a path q

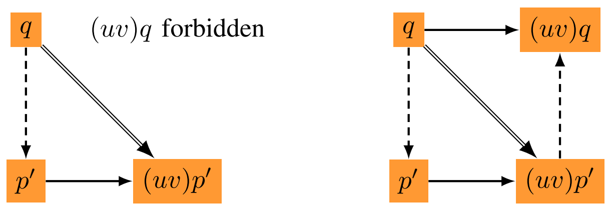


Figure 6: Dispute arcs and their context.

to a path p, where $p = (uv)p'$, if p' and q are paths from j with q preferred to p', and *either* $(uv)q$ is not a permitted path, *or* it is permitted and worse than p. The intention is that this arc encodes the dispute between nodes u and v: node v likes path q, but node u would prefer v to make a different choice.

In the following, we will assume that all three kinds of arc exist in the same digraph. So what we have to prove is that if there is a cycle containing only transmission and dispute arcs, then there is a cycle containing only transmission and preference arcs, and vice versa.

First note that by the definition, there are only two possible configurations in which dispute arcs may occur. These are shown in Figure 6. In particular, if q is preferred to p', and $(uv)q$ and $(uv)p'$ are both permitted paths, then the only way for there *not* to be a dispute arc from q to $(uv)p'$ is if $(uv)q$ is preferred to $(uv)p'$.

Definition 6. A *TP-cycle* is a cycle in the digraph consisting only of transmission and preference arcs.

Definition 7. A *TD-cycle* is a cycle in the digraph consisting only of transmission and dispute arcs.

Lemma 1. *A digraph which contains a TD-cycle also contains a TP-cycle.*

Proof. Let C be a cycle in the given digraph which contains a dispute arc. We show that for any dispute arc in C, we can find a cycle C' which does not contain that arc, replacing it with a transmission and a preference arc. In this way, all dispute arcs can be successively removed, leaving a TP-cycle.

Suppose the chosen dispute arc goes from q to p. Then there must be a path p', with q preferred to p', and $p = (ij)p'$. By definition, there is a preference arc from q to p', and a transmission arc from p' to p. Therefore, the cycle C' goes from q, to p', to p, and then back around C to q. $\square$

For a preference arc from p, define its *negated rank* to be $(n+1) - |p|$, where n is the number of nodes in the network and $|p|$ is the length of p.

Lemma 2. *A digraph which contains a cycle also contains a TD-cycle.*

Proof. The proof is by induction on the sum s of the negated ranks of all preference arcs in a cycle. We show that for all s, if there is a cycle whose sum of negated ranks is s, then there is a TD-cycle.

For the base case, if s is zero, then there are no preference arcs in the cycle: we are done.

For the inductive case, let $s = m + 1$, assuming that the conclusion holds for all $s \leq m$. Choose a preference arc from p to q on the cycle, and consider the next arc, from q to r. There are three cases.

1. **Preference arc.** By transitivity of preference, there is a preference arc from p to r. We have found a new cycle, and the sum of negated ranks is strictly smaller, since the arc from p to r has the same negated rank as the arc from p to q, and the arc from q to r is now omitted.

2. **Transmission arc.** It may be that there is a dispute arc from p to r. If so, then we have found the required cycle, in which this new dispute arc replaces the two previous arcs. A dispute arc is *not* present only when there is another path $(ij)p$, with a transmission arc from p to $(ij)p$, and a preference arc from $(ij)p$ to r. The negated rank of $(ij)p$ is strictly smaller than that for p: in this case also, we have found the required cycle.

3. **Dispute arc.** Whenever there is a dispute arc from q to r, there is some path s such that there is a preference arc from q to s, and a transmission arc from s to r. By transitivity of preference, there is also a preference arc from q to s. We continue as in case (2).

In each case, we are able to deduce the existence of a cycle in which the sum of negated ranks is less than or equal to m.

By induction, in any digraph with a cycle, there is a cycle in which the sum of negated ranks is zero, and which is therefore a TD-cycle. □

Theorem 1. *The dispute digraph of a PSPP S contains a cycle if and only if its paths digraph contains a cycle.*

Proof. Immediate from Lemmas 1 and 2. □

Theorem 2. *Let S be a PSPP. Then S has an acyclic completion if and only if S is acyclic.*

Proof. Suppose that S has a completion T, and that the paths digraph of T is acyclic. The paths digraph for S is a subgraph of the paths digraph for T: the only difference is that T will contain some additional preference arcs. Therefore, the paths digraph for S is acyclic.

For the reverse direction, suppose that S has an acyclic paths digraph. Therefore, it can be topologically sorted. Let $\tau(p)$ denote the index at which path p appears in the sorted order, with $\tau(d)$ being zero. We produce a full SPP by assigning preferences to the paths in S that were permitted, but not yet included in the ranking. For any two such paths p and q, we choose p to be preferred over q if $\tau(p) < \tau(q)$. This corresponds to adding a preference arc to the paths digraph from p to q. When all such preferences have been resolved, we have a paths digraph that is still acyclic. □