The NP-completeness of the Restricted Stable Paths Problem with Three Aggregating Functions

Assaf Kfoury Andrei Lapets

March 14, 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 finding a stable solution to an SPP instance.

In this report we consider a particular restricted variant of SPP, which we call $\langle f, g, h\rangle$-SPP, in which there exist three or more different node policies based on aggregation of edge weights. We show that this variant is NP-complete.

1 An Abstract Graph Model of Networks

Consider a particular abstract model of a communication network $\mathcal{N}$, with a single distinguished node d (the "destination node") and three disjoint groups of nodes – the set $V_\square$ of *square* nodes, the set $V_\circ$ of *round* nodes, and the set $V_\diamond$ of *diamond* nodes – corresponding to three different routing policies:

$\mathcal{N} = \langle V, E, \mathcal{W}\rangle$	where
$V = \{d\} \cup V_\square \cup V_\circ \cup V_\diamond$	the set of nodes
$E \subseteq V \times V$	the set of edges
$\mathcal{W} = \{w_\square, w_\circ, w_\diamond\}$	the set of weight (or cost) functions

For every node $X \in V$ we assume there is a directed path from X to d. If $\mathcal{N}$ contains cycles, there may exist infinitely many paths from X to d.

The symbol # ranges over $\{\square, \circ, \diamond\}$. Each of the weight functions $w_\#$ is a total function from E to positive integers. Every node $X \in V_\#$ aims at opening a minimum-weight path to d according to its routing policy, i.e. the routing policy associated with $V_\#$. Thus, every edge (X, Y) from node X to node Y in the network is labelled with three positive integers: $w_\square(X, Y)$, $w_\circ(X, Y)$ and $w_\diamond(X, Y)$,

where $w_{\#}(X,Y)$ is the *weight* (or the *cost* or some other measure to be minimized) of sending a message along (X,Y) for $\#$-nodes, with $\# \in \{\Box, \circ, \diamond\}$.

Remark 1.1. The preceding formulation, using only 3 routing policies, is a simplification that makes it easier to focus on the essential part of a proof, or on what appears the main obstacle in resolving an open problem – in addition to making graph figures friendlier to read, with nodes easily recognized by their shapes (square, round, and diamond). Additional comments on our formulation:

1. All results can be extended to networks $\mathcal{N}$ with an arbitrary number $n \geqslant 3$ of routing policies, expressed by partitioning the nodes of $\mathcal{N}$ into n disjoint groups and labelling each edge of $\mathcal{N}$ with n weights.

2. The case of a network $\mathcal{N}$ with $n \leqslant 2$ routing policies needs to be analyzed differently. It appears to raise problems, or allows for simplifications, not encountered in the case when $n \geqslant 3$.

3. Most of our results and examples use at most 3 routing policies. On occasion we need to consider a 4-th routing policy, as in the proof of Theorem 5.2. But this is primarily for the purpose of not burdening an already complicated construction.

4. A network $\mathcal{N}$ with $n \leqslant 2$ routing policies is one that mentions only 1 or 2 of the 3 kinds of nodes (square, round and diamond). If $n = 1$, we use only square nodes. If $n = 2$, we use only square and round nodes.

5. We choose to have the distinguished node d, the "destination node", outside the set $V_{\Box} \cup V_{\circ} \cup V_{\diamond}$, i.e. d is not associated with any of the routing policies, and this is why we do not place d in a square, round or diamond box in figures. This is just a convenience, on which none of the results (or conjectures or open problems) depends.

Notation 1.2. We use the letter "A" (resp. "B", resp. "C"), appropriately decorated, to name nodes in the set $V_{\Box}$ (resp. $V_{\circ}$, resp. $V_{\diamond}$). We use the letters "X", "Y" and "Z", appropriately decorated, as variables ranging over the set of all nodes $V = \{d\} \cup V_{\Box} \cup V_{\circ} \cup V_{\diamond}$.

Definition 1.3. A *configuration* $\mathbb{C}$ of a network $\mathcal{N}$ is a spanning tree of $\mathcal{N}$ rooted at d, with all paths directed from the leaf nodes of $\mathbb{C}$ to d. For every node $X \in V$, the path from X to d according to $\mathbb{C}$ is denoted $\mathbb{C}(X)$, written as a sequence of nodes:

$$\mathbb{C}(X) = X_0\ X_1\ \cdots\ X_k$$

for some $k \geqslant 0$, where $X = X_0$, $d = X_k$ and $(X_i, X_{i+1}) \in E$ for every $0 \leqslant i < k$. Three weights are associated with the path $\mathbb{C}(X)$, one for every $\# \in \{\Box, \circ, \diamond\}$:

$$\mathsf{weight}_{\#}\left(\mathbb{C}(X)\right) = w_{\#}(X_0, X_1) + w_{\#}(X_1, X_2) + \cdots + w_{\#}(X_{k-1}, X_k)$$

If $X = X_0 = X_k = d$, we pose $\mathsf{weight}_{\#}\left(\mathbb{C}(X)\right) = 0$.

A configuration $\mathbb{C}$ of network $\mathcal{N}$ is *stable* just in case it satisfies the following condition for every $\# \in \{\Box, \circ, \diamond\}$ and every $X \in V_{\#}$:

$$\mathsf{weight}_{\#}\left(\mathbb{C}(X)\right) \leqslant \min\{\ w_{\#}(X,Y) + \mathsf{weight}_{\#}\left(\mathbb{C}(Y)\right)\ \ |\, (X,Y) \in E\}$$

2 Problems

We pose several problems, each a restriction of the same generic problem, namely, the *examination of conditions under which networks have stable configurations.* Throughout, an "efficient" algorithm means one that runs in time which is a low-degree polynomial (in the input size). Specific instances of this problem are:

1. Let $\mathcal{N}$ be an arbitrary network, as defined in Section 1.
   (a) Determine efficiently whether $\mathcal{N}$ has a stable configuration.
   (b) If $\mathcal{N}$ has a stable configuration, define protocols (i.e. algorithms) to construct efficiently such a configuration.
2. Restrict the problem in part 1 above to a network $\mathcal{N}$ with 1 or 2 routing policies.
3. Restrict the problem in part 1, or part 2 above, to the case when we are given 3 upper bounds $K_\square$, $K_\circ$ and $K_\diamond$ on the weights of paths to d. Specifically, we want to determine a stable configuration $\mathbb{C}$ with the additional requirement that for every node $X \in V_\#$ where $\# \in \{\square, \circ, \diamond\}$, we have $\mathsf{weight}_\# (\mathbb{C}(X)) \leqslant K_\#$.
4. Consider a network $\mathcal{N}$ with two disjoint sets of nodes – the set $V_\square$ of *square* nodes and the set $V_\circ$ of *round* nodes. Assume $w_\square$ fixed and we want to determine $w_\circ$ so that: (1) $w_\circ$ satisfies some yet-to-be-specified constraint, and (2) $\mathcal{N}$ has a stable configuration. The yet-to-be-specified constraint may depend on $w_\square$; for example, it can be the constraint modeled by an (efficiently computed) function f which, for every positive integer i, returns a finite set (or interval) of positive integers, with the requirement that for every $(X, Y) \in E$:

   $$w_\circ(X, Y) \in f(w_\square(X, Y))$$

   Can we define $w_\circ$ subject to this constraint so that $\mathcal{N}$ has a stable configuration?

3 Examples

Several examples that are useful in understanding some of the complications encountered.

Example 3.1. Figure 1 is a network with 2 routing policies, which does not stabilize if, at each time unit, A and B *simultaneously* offer to each other their respective current path to destination d. This is what happens if we use the BGP protocol algorithm, adapted to our abstract graph model. (The actual BGP protocol depends on several specific parameters that are not directly represented in our model.) Nevertheless, it is easy to see the network does have 2 stable configurations, namely: (1) the configuration that makes A go to d directly and B to d via A, and (2) the configuration that makes B go to d directly and A to d via B. Neither of these two configurations is constructed by the BGP protocol algorithm.

Example 3.2. Figure 2 is a network with 2 routing policies which has no stable configurations. This fact can be verified by exhaustively checking all configurations of the network – a total of 15 in this case. However, we conjecture that for networks with 2 routing policies, it is possible to *efficiently* determine whether or not they have stable configurations. (For networks with 3 or more routing policies, such a determination is an NP-complete problem; this is stated in Theorem 5.2.)

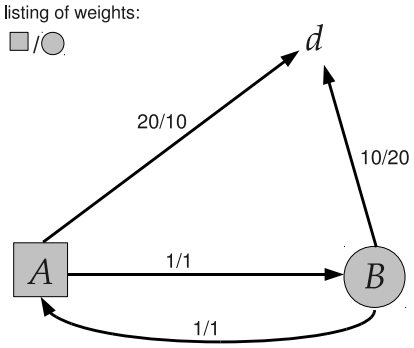

Figure 1: Network with exactly two stable configurations: (1) A goes to d directly and B goes to d via A, (2) B goes to d directly and A goes to d via B. Neither of these two is constructed by the BGP protocol algorithm.

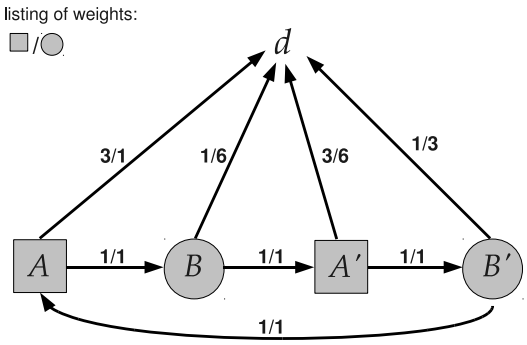

Figure 2: Network with 2 routing policies and no stable configurations.

Example 3.3. Figure 3 and Figure 4 are two networks with 3 routing policies, which have no stable configurations. This fact can be verified by exhaustively checking all configurations of these network – 7 of them for each of the two. These two networks play a crucial role in the proof of Theorem 5.2; the second is obtained from the first by dividing by 2 all weights greater than 1. For later reference, we call them $\mathcal{N}_{\text{unstable}}$ and $\mathcal{N}'_{\text{unstable}}$, respectively.

Example 3.4. Figure 5, Figure 6 and Figure 7, are three networks, each with exactly one stable configuration, obtained by mixing the weights in the two networks in Example 3.3, in one of three possible ways. That the exhibited configuration in each of the three figures is stable is easy to check. Understanding these is helpful in following the proof of Theorem 5.2.

4 Positive Results

Theorem 4.1. *Let $\mathcal{N}$ be a network with exactly one routing policy, i.e. $\mathcal{N}$ contains only square nodes (or, equivalently, the 3 weight functions in $\{w_\square, w_\bigcirc, w_\diamond\}$ are equal). Then $\mathcal{N}$ always has a stable configuration which, moreover, can be constructed efficiently, i.e. in low-degree polynomial time.*

Theorem 4.2. *Let $\mathcal{N} = \langle V, E, \mathcal{W}\rangle$ be a network whose underlying graph is acyclic, i.e. the graph specified by the pair $\langle V, E\rangle$ is a dag. Then $\mathcal{N}$ has a stable configuration.*

The proof of the preceding theorem is an existential proof, it does not define a procedure to find a stable configuration. Nevertheless, by invoking Theorem 4.2

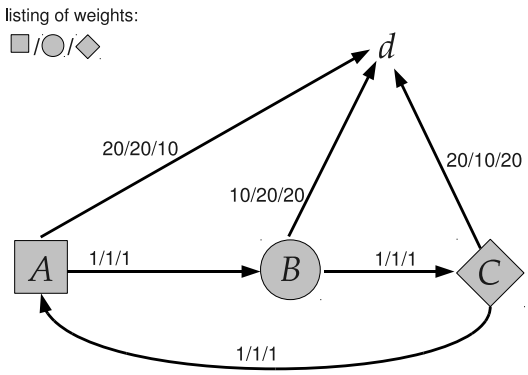

Figure 3: Network with 3 routing policies and no stable configurations, called $\mathcal{N}_{\text{unstable}}$ for later reference.

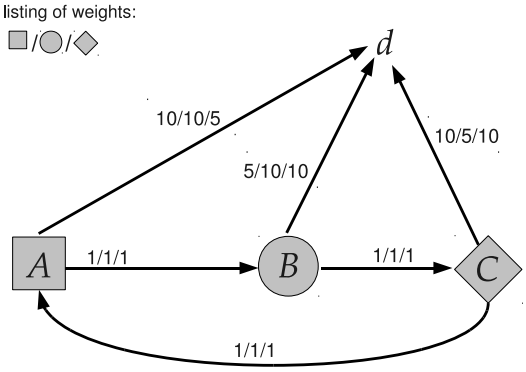

Figure 4: Network with 3 routing policies and no stable configurations, called $\mathcal{N}'_{\text{unstable}}$ for later reference and obtained from $\mathcal{N}_{\text{unstable}}$ in Figure 3 by dividing by 2 all weights greater than 1 (shown in enclosing ellipses).

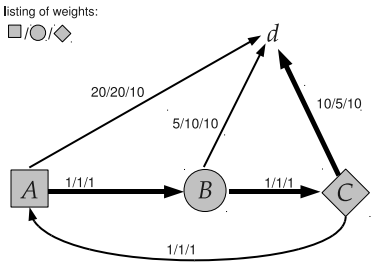

Figure 5: Network with 3 routing policies and exactly one stable configuration (in boldface), called $\mathcal{N}_{\text{stable}}$ for later reference, obtained by mixing the weights of $\mathcal{N}_{\text{unstable}}$ in Figure 3 and $\mathcal{N}'_{\text{unstable}}$ in Figure 4.

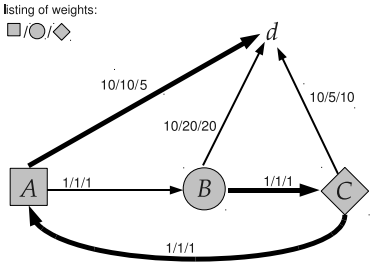

Figure 6: Network with 3 routing policies and exactly one stable configuration (in boldface), called $\mathcal{N}'_{\text{stable}}$ for later reference, obtained by mixing the weights of $\mathcal{N}_{\text{unstable}}$ in Figure 3 and $\mathcal{N}'_{\text{unstable}}$ in Figure 4.

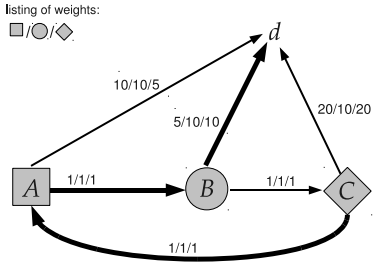

Figure 7: Network with 3 routing policies and exactly one stable configuration (in boldface), called $\mathcal{N}''_{\text{stable}}$ for later reference, obtained by mixing the weights of $\mathcal{N}_{\text{unstable}}$ in Figure 3 and $\mathcal{N}'_{\text{unstable}}$ in Figure 4.

Algorithm 4.3. *Let $\mathcal{N} = \langle V, E, \mathcal{W} \rangle$ be a network whose underlying graph is acyclic, i.e. the graph specified by the pair $\langle V, E \rangle$ is a dag. We can construct a stable configuration in $\mathcal{N}$ in time $\mathcal{O}((m+n) \cdot n)$ where $n = |V|$ and $m = |E|$.*

5 Negative Results

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

The next lemma is used in the proof of Theorem 5.2 together with the networks shown in Figures 5, 6, 7 and 10. The proof consists in building a network $\mathcal{N}(G)$ from a directed graph G, where the in-degree of every vertex is at most 3, such that G is Hamiltonian iff $\mathcal{N}(G)$ has a stable configuration.

Lemma 5.1. *Let G be a directed graph, where the in-degree of every vertex $\leqslant 3$. Whether G has a Hamiltonian circuit is an NP-complete problem.*

Theorem 5.2. *Let $\mathcal{N}$ be a network with (at least) 3 routing policies. Whether $\mathcal{N}$ has a stable configuration is an NP-complete problem.*

Remark 5.3. We first present a preliminary proof of Theorem 5.2; it is preliminary in that it is for a weaker version of the result, namely, for networks with at least 4 routing policies. This preliminary proof is already quite involved, but is necessary in order to understand the proof of the theorem as stated, which we present right after the first.

References

[1] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

A Proofs

A.1 Proof of Lemma 5.1

Let G range over the set of undirected graphs where every node has degree at most 3. Whether such a graph G has a Hamiltonian circuit is an NP-complete problem [1]. This implies whether an arbitrary directed graph where the in-degree and the out-degree of every node $\leqslant 3$ has a Hamiltonian circuit is an NP-complete problem. This implies our lemma.

A.2 Preliminary Proof of Theorem 5.2

This is a "preliminary" proof in that it establishes the desired NP-completeness for networks with at least 4 (rather than 3) routing policies. Though it involves a somewhat complicated construction, it is more transparent than the actual proof of Theorem 5.2, presented in Section A.3 below, which refines it.

Let G be a directed graph where every node has in-degree $\leqslant 3$. We shall construct a network $\mathcal{N}(G)$ from G efficiently, i.e. in polynomial time (in fact in logarithmic space), such that G has a Hamiltonian circuit iff $\mathcal{N}(G)$ has a stable configuration. This will imply the theorem. The construction of $\mathcal{N}(G)$ is in 3 stages: (1) We first construct an intermediary graph G' from G, (2) we then construct what we call the "node substitutes", i.e. the subgraphs that will be substituted for the nodes in G', and (3) we finally construct $\mathcal{N}(G)$ by assembling and connecting all the node substitutes. We can assume that every node in G is accessible from every other node in G, otherwise we can immediately determine that G does not have a Hamiltonian circuit; this implies, in particular, that every node has at least one incoming edge and at least one outgoing edge.

Construction of G'

For convenience, let $G = \langle \{1, 2, \ldots, n\}, E \rangle$, i.e. G has $n \geqslant 1$ nodes denoted by the natural numbers from 1 to n and $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$. We first define a new graph G' from G by splitting node 1 into two nodes, say 1 and $1'$, adding a new edge $(1, 1')$ from 1 to $1'$, and leaving the rest of G untouched. Specifically, let:

$$
\begin{aligned}
&G' = \langle \{1, 1', 2, \ldots, n\}, E' \rangle \qquad \text{where} \\
&E' = \{(1, 1')\} \cup \\
&\qquad \{(j, 1) \,|\, (j, 1) \in E\} \;\cup\; \{(1', k) \,|\, (1, k) \in E\} \cup \\
&\qquad E \cap \{2, \ldots, n\} \times \{2, \ldots, n\}
\end{aligned}
$$

It is easy to see that G has a Hamiltonian circuit iff G' has a Hamiltonian path that starts at node $1'$ and ends at node 1. By hypothesis, every node in G has at most 3 incoming edges, and this property is inherited by G'.

Construction of The Node Substitutes

The desired network $\mathcal{N}(G)$ is obtained by replacing every node $i \in \{1, 2, \ldots, n\}$ – but not the new node $1'$ – in G' by a 10-node graph of the form shown in Figure 10 and denoted $\mathsf{NodeSubstitute}(i)$. The only node omitted from this substitution is node $1'$, which simply becomes node $d(1')$ in

$\mathcal{N}(G)$. Figure 10 corresponds to the case when node i has exactly three incoming edges – namely, $\{(p,i),(q,i),(r,i)\}$ – and one or more outgoing edges – namely, $\{(i,j),\ldots,(i,k)\}$.

If the node $i \in \{1,2,\ldots,n\}$ has only one or two incoming edges in G' – say, $\{(p,i)\}$ or $\{(p,i),(q,i)\}$, resp. – we simply omit the other subgraphs in the construction – namely, the subgraph consisting of the nodes $\{d(q), B'(i), C''(i), d(r), C'(i), A''(i)\}$ or $\{d(r), C'(i), A''(i)\}$, resp., together with all the edges they touch. These cases are not shown in Figure 10.

Consider the subgraph NodeSubstitute(i) of $\mathcal{N}(G)$, as shown in Figure 10. It shows the case when node i has exactly three incoming edges, with the following correspondence:

- Edge (p,i) in G' is mapped to edge $(d(p),\ B''(i))$ in $\mathcal{N}(G)$.
- Edge (q,i) in G' is mapped to edge $(d(q),\ C''(i))$ in $\mathcal{N}(G)$.
- Edge (r,i) in G' is mapped to edge $(d(r),\ A''(i))$ in $\mathcal{N}(G)$.

Each of the edges outgoing from node i, say (i,j), in G' is mapped to an edge "from $d(i)$ to NodeSubstitute(j)" in $\mathcal{N}(G)$, as shown in Figure 10. This means (i,j) is mapped to an edge "from $d(i)$ to one of the 3 nodes in $\{B''(j),\ C''(j),\ A''(j)\}$" – it does not matter which of the 3 – which are the 3 entry points of NodeSubstitute(j). A few additional clarifications about the construction of NodeSubstitute(i):

1. For every $i \in \{2,3,\ldots,n\}$, node $d(i)$ has exactly 4 (or 5 or 6, resp.) incoming edges, if node i in G has 1 (or 2 or 3, resp.) incoming edges; and $d(i)$ has exactly as many outgoing edges as i has outgoing edges in G. In order to exit NodeSubstitute(i), a path must use one of the following:

   - One of the outgoing edges of $d(i)$. Call these the *forward* exits of NodeSubstitute(i).
   - One of the 3 edges directly linked to destination $d(1)$. Call these the *upward* exits of NodeSubstitute(i), namely, $(B''(i),\ d(1))$ or $(C''(i),\ d(1))$ or $(A''(i),\ d(1))$.
   - One of the 3 edges directly linked to $d(1')$. Call these the *backward* exits of NodeSubstitute(i), namely, $(A'(i),\ d(1'))$ or $(B'(i),\ d(1'))$ or $(C'(i),\ d(1'))$.

   A path that enters and exits NodeSubstitute(i), *and* touches $d(i)$, must contain one of the following 3 paths as a subpath:

   - $d(p)\ B''(i)\ A'(i)\ d(i)$
   - $d(q)\ C''(i)\ B'(i)\ d(i)$
   - $d(r)\ A''(i)\ C'(i)\ d(i)$

   Observe that the total weight of these 3 paths is the same, namely, 6/6/6/3 – that is, 6 according to the three first metrics ("square", "round", "diamond") and 3 according to a new fourth metric (called "bullet").

2. Node $d(1)$ has between 4 and 6 incoming edges, part of the definition of NodeSubstitute(1), in addition to as many incoming edges as there are upward exits in

   NodeSubstitute(1) , ... , NodeSubstitute(n),

in addition to the single backward edge $(d(1'),\ d(1))$. Although G' contains the edge $(1, 1')$, we omit the corresponding edge $(d(1),\ d(1'))$ in $\mathcal{N}(G)$, as it plays no role in the reduction. There is *no* outgoing edges from $d(1)$.

3. Node $d(1')$ has as many incoming edges as there are backward exits in

   $$\mathsf{NodeSubstitute}(1)\ ,\ \ldots\ ,\ \mathsf{NodeSubstitute}(n).$$

   And $d(1')$ has as many outgoing edges as node $1'$ has in G' (equivalently, as node 1 has in G) plus the single edge $(d(1'),\ d(1))$.

Every $\mathsf{NodeSubstitute}(i)$ includes as a subgraph a copy of $\mathcal{N}_{\text{unstable}}$, the inherently unstable network of Figure 3, as shown in Figure 10. Moreover, if some or all of the 3 edges

$$\{(A'(i),\ d(i)),\ (B'(i),\ d(i)),\ (C'(i),\ d(i))\}$$

are included in a configuration $\mathbb{C}$, then $\mathbb{C}$ will cover the nodes of a subgraph of $\mathsf{NodeSubstitute}(i)$ which is equivalent (in its stability properties) to:

- $\mathcal{N}_{\text{stable}}$ of Figure 5, if edge $(A'(i),\ d(i))$ are included *but not* edges $(B'(i),\ d(i))$ and $(C'(i),\ d(i))$.
- $\mathcal{N}'_{\text{stable}}$ of Figure 6, if edge $(B'(i),\ d(i))$ are included *but not* edges $(A'(i),\ d(i))$ and $(C'(i),\ d(i))$.
- $\mathcal{N}''_{\text{stable}}$ of Figure 7, if edge $(C'(i),\ d(i))$ are included *but not* edges $(A'(i),\ d(i))$ and $(B'(i),\ d(i))$.
- $\mathcal{N}'_{\text{unstable}}$ of Figure 4, if any 2 or 3 edges in $\{(A'(i),\ d(i)),\ (B'(i),\ d(i)),\ (C'(i),\ d(i))\}$ are included.

In the first of the four cases above, this is so because the total weights along the path $P =$ "$B(i)\ A'(i)\ d(i)$" are:

$$\mathsf{weight}_{\square}\,(P) = 5, \qquad \mathsf{weight}_{\circ}\,(P) = 10, \qquad \mathsf{weight}_{\diamond}\,(P) = 10,$$

and the total weights along the path $Q =$ "$C(i)\ A'(i)\ d(i)$" are:

$$\mathsf{weight}_{\square}\,(Q) = 10, \qquad \mathsf{weight}_{\circ}\,(Q) = 5, \qquad \mathsf{weight}_{\diamond}\,(Q) = 10,$$

corresponding to the weights of the edges (B, d) and (C, d) in network $\mathcal{N}_{\text{stable}}$ which has exactly one stable configuration, namely, the path "$A\ B\ C\ d$" – which becomes the path "$A(i)\ B(i)\ C(i)\ A'(i)\ d(i)$" in $\mathsf{NodeSubstitute}(i)$. A similar explanation applies to each of the the second, third, and fourth case above.

Construction of $\mathcal{N}(G)$

$\mathcal{N}(G)$ is constructed by assembling together

$$\mathsf{NodeSubstitute}(1), \mathsf{NodeSubstitute}(2), \ldots, \mathsf{NodeSubstitute}(n),$$

adding the node $d(1')$, and adding all the edges as specified in points 1, 2 and 3 above.

We need to designate the kind of all the nodes in $\{d(1), d(1'), d(2), \ldots, d(n)\}$, which we take to be a new 4-th kind, called "bullet", distinct from the three earlier kinds ("square", "round" and "diamond").

We also need to assign weights according to the 4 metrics – square, round, diamond, bullet – to all the edges. These weight assignments are shown in Figure 10, except for one weight assignment still to be determined, namely, that of edge $(d(1'),\ d(1))$ which we set to:

$$w_{\square}(d(1'),\ d(1)) = 1, \quad w_{\bigcirc}(d(1'),\ d(1)) = 1, \quad w_{\diamond}(d(1'),\ d(1)) = 1, \quad w_{\bullet}(d(1'),\ d(1)) = 1 + 3 \cdot n$$

which is written "$1\,/\,1\,/\,1\,/\,1 + 3 \cdot n$" in our shorthand notation. Thus the weight assignment of $(d(1'),\ d(1))$ – specifically, its $\bullet$-weight – is the only one that depends on the size n of the graph G. Informally, the purpose of this weight assignment is this: If node $d(1')$ reaches destination $d(1)$ directly via the backward edge $(d(1'),\ d(1))$ rather than through a sequence of forward edges and/or upward edges, then in a stable configuration, all the nodes $A'(i)$, $B'(i)$ and $C'(i)$ will want to reach $d(1)$ via the backward edges and through node $d(1')$.

This completes the construction of the network $\mathcal{N}(G)$. It remains to show that: *G has a Hamiltonian cycle iff $\mathcal{N}(G)$ has a stable configuration.*

Proof of: If G has a Hamiltonian cycle then $\mathcal{N}(G)$ has a stable configuration

Suppose G has a Hamiltonian cycle. Then, by the construction of G', we also have that G' has a Hamiltonian path P from node $1'$ to node 1, from which we show how to construct a stable configuration $\mathbb{C}$ of $\mathcal{N}(G)$. Path P can be specified by:

$$\begin{array}{l} P \ = \ \ell_1\, \ell_2\, \ldots\, \ell_n\, \ell_{n+1} \qquad \text{where} \\ \ell_1 \ = 1',\ \ell_{n+1} \ = \ 1,\ \{\ell_2, \ldots, \ell_n\} \ = \ \{2, 3, \ldots, n\} \quad \text{and} \\ (\ell_{m-1}, \ell_m) \in E' \quad \text{for every } 1 < m \leqslant n+1. \end{array}$$

Consider an arbitrary edge $(\ell, \ell') \in \{(\ell_1, \ell_2),\ (\ell_2, \ell_3), \ldots,\ (\ell_n, \ell_{n+1})\}$. The desired configuration $\mathbb{C}$ will include a suitably defined spanning forest, call it $\mathbb{C}_{\ell'}$, that covers the nodes of $\mathsf{NodeSubstitute}(\ell')$. Since (ℓ, ℓ') is an edge of G', there must be an edge from $d(\ell)$ to $\mathsf{NodeSubstitute}(\ell')$. With no loss of generality suppose this edge enters $\mathsf{NodeSubstitute}(\ell')$ at node $B''(\ell')$. (The case when the edge from $d(\ell)$ enters $\mathsf{NodeSubstitute}(\ell')$ at node $C''(\ell')$ or node $A''(\ell')$ is treated similarly.) The case when $(\ell, \ell') = (p, i)$ is illustrated in Figure 11. The spanning forest $\mathbb{C}_i$, with $i \in \{\ell_2, \ldots, \ell_{n+1}\}$, consists of 4 paths which covers all the nodes of $\mathsf{NodeSubstitute}(i)$:

$d(p)\ B''(i)\ A'(i)\ d(i)$	corresponds to edge (p, i) of P,
$A(i)\ B(i)\ C(i)\ A'(i)\ d(i)$	covers the nodes $A(i)$, $B(i)$ and $C(i)$,
$B'(i)\ C''(i)\ d(1)$	connects "unused" nodes $B'(i)$ and $C''(i)$ upward to $d(1)$,
$C'(i)\ A''(i)\ d(1)$	connects "unused" nodes $C'(i)$ and $A''(i)$ upward to $d(1)$.

$\mathbb{C}_i$ is shown in Figure 11. The desired configuration $\mathbb{C}$ is obtained by collecting together $\mathbb{C}_{\ell_2}$, $\mathbb{C}_{\ell_3}$, …, $\mathbb{C}_{\ell_{n+1}}$. It is straightforward to check that $\mathbb{C}$ is stable, using the weights shown in Figure 10.

Proof of: If $\mathcal{N}(G)$ has a stable configuration then G has a Hamiltonian cycle

Suppose $\mathcal{N}(G)$ has a stable configuration $\mathbb{C}$, from which we shall construct a Hamiltonian path P from $1'$ to 1 in G'. The latter will imply that G has a Hamiltonian cycle, the desired conclusion. Because $\mathbb{C}$ is a configuration, i.e. a spanning tree with all paths directed from the leaf nodes to the root node $d(1)$, we must have:

1. For every $i \in \{1', 2, 3, \ldots, n\}$, $\mathbb{C}$ contains exactly one of the edges outgoing from $d(i)$, because in a spanning tree, every node other than the root has exactly one outgoing edge. This edge connects $d(i)$ to one of the 3 entry nodes of $\mathsf{NodeSubstitute}(j)$ for some $j \in \{1, 2, \ldots, n\}$.

2. For every $i \in \{1, 2, \ldots, n\}$, because $\mathbb{C}$ is also stable, $\mathbb{C}$ must contain exactly one of the 3 following edges: $(A'(i),\ d(i))$, $(B'(i),\ d(i))$, or $(C'(i),\ d(i))$ – if it contains 0, 2 or all 3, of these edges, $\mathbb{C}$ is unstable, from the observations at the end of the construction of $\mathsf{NodeSubstitute}(i)$.

3. Because $\mathbb{C}$ is stable, $\mathbb{C}$ cannot include the edge $(d(1'),\ d(1))$ with weight assignment "1/1/1/1+ $3 \cdot n$". Indeed, if $\mathbb{C}$ did include this edge, then every node in the set:

   $$\{A'(i) \,|\, 1 \leqslant i \leqslant n\} \ \cup\ \{B'(i) \,|\, 1 \leqslant i \leqslant n\} \ \cup\ \{C'(i) \,|\, 1 \leqslant i \leqslant n\}$$

   would have to reach destination node $d(1)$ via the backward edges and through node $d(1')$. But this would make every $\mathsf{NodeSubstitute}(i)$ unstable, as implied by the preceding point. Hence, $\mathbb{C}$ cannot include the edge $(d(1'),\ d(1))$ and $d(1')$ must reach $d(1)$ via a sequence of forward and/or upward edges.

4. Because $\mathbb{C}$ is stable, the path $\mathbb{C}(d(1'))$ cannot in fact include upward edges, because the $\bullet$-weight of the latter is "\$", a large number strictly larger than $w_\bullet(d(1'), d(1)) = 1 + 3 \cdot n$. Hence, $\mathbb{C}(d(1'))$ must be a sequence of forward edges only, and thus of the form:

   $$\mathbb{C}(d(1')) = d(\ell_1)\ X_1\ Y_1\ d(\ell_2)\ X_2\ Y_2\ d(\ell_3)\ \cdots\ X_m\ Y_m\ d(\ell_{m+1})$$

   where $\ell_1 = 1'$, $\ell_{m+1} = 1$, $m \leqslant n$ and every consecutive pair "$X_i\ Y_i$" must be a member of the set:

   $$\{\text{“}B''(j)\ A'(j)\text{”} \,|\, 1 \leqslant j \leqslant n\} \ \cup\ \{\text{“}C''(j)\ B'(j)\text{”} \,|\, 1 \leqslant j \leqslant n\} \ \cup\ \{\text{“}A''(j)\ C'(j)\text{”} \,|\, 1 \leqslant j \leqslant n\}$$

   for every $1 \leqslant i \leqslant m$. The $\bullet$-weight of every forward edge being 1, we have $\mathsf{weight}_\bullet\,(\mathbb{C}(d(1'))) = 3 \cdot m$, a number strictly less than $w_\bullet(d(1'), d(1)) = 1 + 3 \cdot n$ (which is as it should be, since $\mathbb{C}$ is stable).

We claim that, in fact, $m = n$, i.e. every node in $\{d(1'), d(1), d(2), \ldots, d(n)\}$ occurs in the path $\mathbb{C}(d(1'))$. Suppose the contrary, and we will get a contradiction. Consider therefore some node $d(k)$ not occurring in the path $\mathbb{C}(d(1'))$. Node $d(k)$ is the forward exit of $\mathsf{NodeSubstitute}(k)$. Because $\mathbb{C}$ is stable, it must be that exactly one of the following 3 edges of $\mathsf{NodeSubstitute}(k)$ is part of $\mathbb{C}$:

$$(A'(k),\ d(k)),\ (B'(k),\ d(k)),\ (C'(k),\ d(k))$$

With no loss of generality, assume it is the first of these 3 edges, i.e. $(A'(k),\ d(k))$ is in $\mathbb{C}$ while both $(B'(k),\ d(k))$ and $(C'(k),\ d(k))$ are not. This in turn implies that the edge $(B''(k),\ A'(k))$ is in $\mathbb{C}$

too, otherwise $(B''(k),\ d(1))$ would instead be in $\mathbb{C}$ and $A'(k)$ would want to reach $d(1)$ via the path "$A'(k)\ B''(k)\ d(1)$", contrary to assumption. Thus $\mathbb{C}(B''(k))$ is of the form:

$$\mathbb{C}(B''(k)) = B''(k)\ A'(k)\ d(k)\ P$$

for a path P connecting $d(k)$ to $d(1)$. Consider 3 possible cases for P, each contradicting the stability of $\mathbb{C}$:

(a) P consists of forward edges only. In this case P and $\mathbb{C}(d(1'))$ must have a suffix in common. Specifically, P must be of the form "$Q\ \mathbb{C}(d(\ell))$" for some path Q and some $\ell \in \{\ell_2, \ell_3, \ldots, \ell_{m+1}\}$, i.e. the forward exit node $d(\ell)$ of $\mathsf{NodeSubstitute}(\ell)$ is visited by (at least) two distinct paths of $\mathbb{C}$, namely, $\mathbb{C}(d(1'))$ and $\mathbb{C}(B''(k))$. This implies that, in $\mathsf{NodeSubstitute}(\ell)$, (at least) two of the three edges in

$$\{(B''(\ell),\ A'(\ell)),\ (C''(\ell),\ B'(\ell)),\ (A''(\ell),\ C'(\ell))\}$$

are used by $\mathbb{C}$. This contradicts the stability of $\mathbb{C}$, since exactly one of these 3 edges can be used in a stable configuration.

(b) P includes a backward edge. Every backward edge enters $d(1')$, which implies $P = Q\ \mathbb{C}(d(1'))$ for some path Q. In this case,

$$\begin{aligned}
\mathsf{weight}_\square\left(\mathbb{C}(A'(k))\right) &= 4 + \mathsf{weight}_\square\left(\mathbb{C}(d(k))\right) \\
&= 4 + \mathsf{weight}_\square\left(Q\right) + \mathsf{weight}_\square\left(\mathbb{C}(d(1'))\right) \\
&> 1 + \mathsf{weight}_\square\left(\mathbb{C}(d(1'))\right) \\
&= w_\square(A'(k), d(1')) + \mathsf{weight}_\square\left(\mathbb{C}(d(1'))\right)
\end{aligned}$$

Contradicting the stability of $\mathbb{C}$.

(c) P includes an upward edge. Every upward edge enter $d(1)$, which implies $P = Q\ X\ d(1)$ for some path Q (a possibly empty sequence of nodes) and node X in the set:

$$\{A''(i) \,|\, 1 \leqslant i \leqslant n\}\ \cup\ \{B''(i) \,|\, 1 \leqslant i \leqslant n\}\ \cup\ \{C''(i) \,|\, 1 \leqslant i \leqslant n\}$$

If X is a square node (round node, diamond node, resp.), consider the path $\mathbb{C}(B''(k))$ (the path $\mathbb{C}(B''(k))$, the path $\mathbb{C}(A'(k))$, resp.) in order to get a contradiction. With no loss of generality, suppose X is a square node, and consider the path $\mathbb{C}(B''(k)) = B''(k)\ A'(k)\ d(k)\ P = B''(k)\ A'(k)\ d(k)\ Q\ X\ d(1)$ — the two other cases are treated similarly. In this case,

$$\begin{aligned}
\mathsf{weight}_\circ\left(\mathbb{C}(B''(k))\right) &= 1 + 4 + \mathsf{weight}_\circ\left(d(k)\ Q\ X\right) + \$ \\
&> \$ \\
&= w_\circ(B''(k), d(1))
\end{aligned}$$

Contradicting the stability of $\mathbb{C}$.

Hence $\mathbb{C}(d(1'))$ mentions every node in $\{d(1'), d(1), d(2), \ldots, d(n)\}$. It is straightforward to check that the path $\mathbb{C}(d(1'))$ maps back to a Hamiltonian path $\ell_1\, \ell_2\, \ldots\, \ell_n\, \ell_{n+1}$ from $1'$ to 1 in G'. This concludes the proof.

A.3 Final Proof of Theorem 5.2

This is an adaptation of the construction and argument presented in Section A.2 above. We highlight the differences in the construction of $\mathcal{N}(G)$, then focus more closely on the argument, now a bit more involved, that G has a Hamiltonian circuit iff $\mathcal{N}(G)$ has a stable configuration.

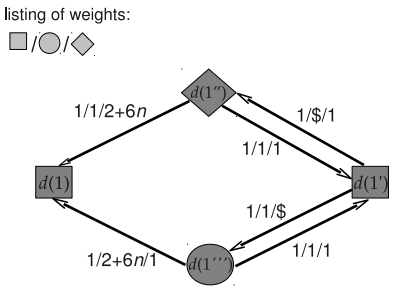

Figure 8: Nodes 1 and $1'$ of G' are mapped to nodes $d(1)$ and $d(1')$ of $\mathcal{N}(G)$, respectively. We add two additional nodes, $d(1'')$ and $d(1''')$, one diamond and one round, in $\mathcal{N}(G)$. This figure shows the subgraph of $\mathcal{N}(G)$ consisting of all the edges between the 4 nodes $d(1), d(1'), d(1'')$ and $d(1''')$, as well as their weight assignments; all edges between these 4 nodes and other nodes of $\mathcal{N}(G)$ are omitted in the figure, namely, all edges from other nodes of $\mathcal{N}(G)$ to $d(1)$, $d(1'')$ and $d(1''')$, and all edges from $d(1')$ to other nodes of $\mathcal{N}(G)$. In a stable configuration $\mathbb{C}$ of $\mathcal{N}(G)$ rooted at $d(1)$, i.e. with all paths directed towards the destination node $d(1)$, *either* $\mathbb{C}$ includes both edge $(d(1''), d(1))$ and edge $(d(1'''), d(1))$, *or* $\mathbb{C}$ includes both edge $(d(1''), d(1'))$ and edge $(d(1'''), d(1'))$. In the first case, $d(1')$ reaches $d(1)$ via $d(1'')$ or $d(1''')$, in the second case $d(1')$ reaches $d(1)$ via other nodes of $\mathcal{N}(G)$.

As in Section A.2, let $G = \langle\{1, 2, \ldots, n\}, E\rangle$. The construction of the intermediary graph G' is identical to that in Section A.2, and the differences in the construction of $\mathcal{N}(G)$ from G' are summarized in Figure 8 and Figure 12. All the earlier comments on the construction of $\mathcal{N}(G)$ in Section A.2 apply again here, with the following differences:

- Nodes 1 and $1'$ of G' are mapped to nodes $d(1)$ and $d(1')$ of $\mathcal{N}(G)$, respectively, as in Section A.2. But now we introduce two new nodes, one diamond $d(1'')$ and one round $d(1''')$, which are connected to $d(1)$ and $d(1')$ as shown in Figure 8.
- In the earlier construction of $\mathcal{N}(G)$ in Section A.2, there was a single backward edge from $d(1')$ to $d(1)$ with weight assignment "$1/1/1/1 + 3 \cdot n$". Now there are two backward paths from $d(1')$ to $d(1)$, namely, $P1$ = "$d(1')\ d(1'')\ d(1)$" and $P2$ = "$d(1')\ d(1''')\ d(1)$", with the following weights:

$$\begin{array}{lll} \mathsf{weight}_{\square}(P1) = 2 & \mathsf{weight}_{\bigcirc}(P1) = 1 + \$ & \mathsf{weight}_{\diamond}(P1) = 3 + 6 \cdot n \\ \mathsf{weight}_{\square}(P2) = 2 & \mathsf{weight}_{\bigcirc}(P2) = 3 + 6 \cdot n & \mathsf{weight}_{\diamond}(P2) = 1 + \$ \end{array}$$

- Every earlier mention of the nodes $A''(i)$ or $A''(j)$ in Section A.2 is now changed to $B'''(i)$ or $B'''(j)$, respectively.
- Every earlier mention of the weight assignment "6/6/6/3" is now changed to "6/6/6".

- As in Section A.2, $\mathcal{N}(G)$ is obtained by assembling $\mathsf{NodeSubstitute}(1), \ldots, \mathsf{NodeSubstitute}(n)$ together. In particular, there is an edge from $d(1')$ to one of the three entry nodes (it does not matter which) of $\mathsf{NodeSubstitute}(k)$ iff there is an edge from node $1'$ to node k in G'.

Let $\mathbb{C}$ be a stable configuration of $\mathcal{N}(G)$. An inspection of the weight assignments in Figure 8 readily shows the following:

($\dagger$) $\mathbb{C}$ includes edge $(d(1''), d(1))$ iff $\mathbb{C}$ includes edge $(d(1'''), d(1))$.

($\ddagger$) $\mathbb{C}$ includes edge $(d(1''), d(1'))$ iff $\mathbb{C}$ includes edge $(d(1'''), d(1'))$.

($\dagger'$) If $\mathbb{C}$ includes both edge $(d(1''), d(1))$ and edge $(d(1'''), d(1))$, then $\mathbb{C}$ includes edge $(d(1'), d(1''))$ or edge $(d(1'), d(1'''))$ – but not both.

($\ddagger'$) If $\mathbb{C}$ excludes both edge $(d(1''), d(1))$ and edge $(d(1'''), d(1))$, then $\mathbb{C}$ includes both edge $(d(1''), d(1'))$ and edge $(d(1'''), d(1'))$.

To prove the left-to-right implication of ($\dagger$), suppose $\mathbb{C}$ includes $(d(1''), d(1))$. Then, given the weights shown in Figure 8, $\mathbb{C}$ must include $(d(1'), d(1''))$, i.e., the path $\mathbb{C}(d(1'))$ from $d(1')$ to $d(1)$ does not use any forward or upward edges of $\mathcal{N}(G)$. Given the weights of Figure 8 again, it follows that $\mathbb{C}(d(1''')) = (d(1'''), d(1))$, i.e., $\mathbb{C}$ includes edge $(d(1'''), d(1))$. A similar argument proves the right-to-left implication of ($\dagger$). Assertion ($\ddagger$) is an immediate consequence of ($\dagger$). We omit the straightforward proof of assertions ($\dagger'$) and ($\ddagger'$).

Proof of: If G has a Hamiltonian cycle then $\mathcal{N}(G)$ has a stable configuration

The argument here follows the earlier argument in Section A.2. We construct the desired configuration $\mathbb{C}$ as in Section A.2, but we need here to cover the two additional nodes $d(1'')$ and $d(1''')$. For this, we simply add the two edges $(d(1''), d(1'))$ and $(d(1'''), d(1'))$ to $\mathbb{C}$. With the weight assignments shown in Figure 8 and Figure 12, it is easy to see that:

$$\mathsf{weight}_{\diamond}\left(\mathbb{C}(d(1''))\right) \;=\; \mathsf{weight}_{\circ}\left(\mathbb{C}(d(1'''))\right) \;=\; 1 + 6 \cdot n$$

Since $w_{\diamond}(d(1''), d(1)) = w_{\circ}(d(1'''), d(1)) = 2 + 6 \cdot n$, the configuration $\mathbb{C}$ is stable. We omit all remaining straightforward details, all adapted from the earlier version.

Proof of: If $\mathcal{N}(G)$ has a stable configuration then G has a Hamiltonian cycle

Suppose $\mathcal{N}(G)$ has a stable configuration $\mathbb{C}$, from which we shall construct a Hamiltonian path P from $1'$ to 1 in G'. By fact ($\dagger$), we have one of two possible cases:

- either $\mathbb{C}$ includes both edge $(d(1''), d(1))$ and edge $(d(1'''), d(1))$,
- or $\mathbb{C}$ excludes both edge $(d(1''), d(1))$ and edge $(d(1'''), d(1))$.

It cannot be the first case because, otherwise, every node in:

$$\{A'(i) \,|\, 1 \leqslant i \leqslant n\} \;\cup\; \{B'(i) \,|\, 1 \leqslant i \leqslant n\}$$

would have to reach destination $d(1)$ via the backward edges and through $d(1'')$, and every node in:

$$\{C'(i) \,|\, 1 \leqslant i \leqslant n\}$$

would have to reach destination $d(1)$ via the backward edges and through $d(1''')$. But this would make every $\mathsf{NodeSubstitute}(i)$ unstable – for the same reason given in Section A.2.

Hence, $\mathbb{C}$ must exclude both edge $(d(1''), d(1))$ and edge $(d(1'''), d(1))$, which implies that $d(1')$ must reach $d(1)$ via a sequence of forward and/or upward edges. And $\mathbb{C}$ must also include the edge $(d(1''), d(1'))$ and edge $(d(1'''), d(1'))$, by fact $(\ddagger')$.

Note that $d(1'), d(1'')$ and $d(1''')$ are a square node, a diamond node and a round node, respectively. Hence, the path $\mathbb{C}(d(1'))$ cannot contain an upward edge, because two of the three weights of the latter ($\square$-weight, $\circ$-weight and $\diamond$-weight) are equal to "\$", which is larger than at least one of the corresponding weights of the paths $P1$ and $P2$. Hence, $\mathbb{C}(d(1'))$ must be a sequence of forward edges only, and thus of the form:

$$\mathbb{C}(d(1')) = d(\ell_1)\ X_1\ Y_1\ d(\ell_2)\ X_2\ Y_2\ d(\ell_3)\ \cdots\ X_m\ Y_m\ d(\ell_{m+1})$$

where $\ell_1 = 1'$, $\ell_{m+1} = 1$, $m \leqslant n$ and every consecutive pair "$X_i\ Y_i$" must be a member of the set:

$$\{\text{“}B''(j)\ A'(j)\text{”} \,|\, 1 \leqslant j \leqslant n\} \ \cup\ \{\text{“}C''(j)\ B'(j)\text{”} \,|\, 1 \leqslant j \leqslant n\} \ \cup\ \{\text{“}B'''(j)\ C'(j)\text{”} \,|\, 1 \leqslant j \leqslant n\}$$

for every $1 \leqslant i \leqslant m$. Inspecting the weights shown in Figure 12, it is easy to see that:

$$\mathsf{weight}_{\square}\left(\mathbb{C}(d(1'))\right) \ =\ \mathsf{weight}_{\circ}\left(\mathbb{C}(d(1'))\right) \ =\ \mathsf{weight}_{\diamond}\left(\mathbb{C}(d(1'))\right) \ = 6 \cdot n$$

We claim that, in fact, $m = n$, i.e. every node in $\{d(1'), d(1), d(2), \ldots, d(n)\}$ occurs in the path $\mathbb{C}(d(1'))$. The proof of the claim is by contradiction, and is a straightforward adaptation of the earlier proof in Section A.2. All remaining details are left to the reader. This concludes the final proof of Theorem 5.2.

A.4 Proof of Theorem 4.2

Let $\mathcal{N} = \langle V, E, \mathcal{W}\rangle$ where $\langle V, E\rangle$ is a dag. Because there is a path from every node X to d, by the definition of what a network is, this dag must be rooted at d.

We assume there is a total ordering $<$ on V, which can be extended to directed paths in $\mathcal{N}$, i.e. finite sequences over V, in the obvious lexicographic ordering. It follows that, given two arbitrary directed paths p and q in $\mathcal{N}$, it is the case that either $p < q$ or $q < p$ or $p = q$.

Let X be a node in $\mathcal{N}$ other than d, with $X \in V_{\#}$ where $\# \in \{\square, \circ, \diamond\}$. Because $\langle V, E\rangle$ is a dag, there are finitely many paths from X to d, say, $p_1, \ldots, p_j$. Define $\mathsf{bestPath}(X)$ as the unique $p \in \{p_1, \ldots, p_j\}$ such that:

1. $\mathsf{weight}_{\#}(p) \leqslant \mathsf{weight}_{\#}(q)$ for every $q \in \{p_1, \ldots, p_j\}$,
2. $p < q$ for every $q \in \{p_1, \ldots, p_j\}$ such that $\mathsf{weight}_{\#}(p) = \mathsf{weight}_{\#}(q)$.

Thus, every node has a best path. We next define a partial order on the set of all best paths. Let X and X be arbitrary nodes other than d, with:

$$\mathsf{bestPath}(X) = Y_0\ Y_1\ \cdots\ Y_k\ d \qquad \text{and} \qquad \mathsf{bestPath}(X') = Y'_0\ Y'_1\ \cdots\ Y'_\ell\ d$$

where $X = Y_0$, $X' = Y'_0$, $k \geqslant 0$ and $\ell \geqslant 0$. We define:

$$\mathsf{bestPath}(X) \succcurlyeq \mathsf{bestPath}(X') \quad \text{iff} \quad Y_k = Y'_0$$

In the special case when $k = 0$, i.e. when $\mathsf{bestPath}(X)$ is of length 1, it must also be the case that $X = Y_0 = Y'_0 = Y$ and $\ell = 0$, i.e. $\mathsf{bestPath}(X) = \mathsf{bestPath}(X')$. We write $\mathsf{bestPath}(X) \succ \mathsf{bestPath}(X')$ if $\mathsf{bestPath}(X) \succcurlyeq \mathsf{bestPath}(X')$ and $\mathsf{bestPath}(X) \neq \mathsf{bestPath}(X')$.

If p is a path of the form $p = p'\ p''$, we write p/p'' to denote p', i.e. what is left of p after chopping off the suffix p''. Consider now a strictly descending chain of best paths, say:

$$\mathsf{bestPath}(X_0) \ \succ\ \mathsf{bestPath}(X_1) \ \succ\ \mathsf{bestPath}(X_2) \ \succ\ \cdots$$

where for every $i \geqslant 0$ we have:

$$\mathsf{bestPath}(X_i) = Y_{i,0}\ Y_{i,1}\ \cdots\ Y_{i,k_i}\ d$$

for some $k_i \geqslant 0$ and where $X_i = Y_{i,0}$. By the observation in the preceding paragraph, if $k_i = 0$, then $\mathsf{bestPath}(X_i)$ is of length 1 and it must be the last entry in the descending chain. Moreover, this chain can be extended as long as the last entry is path of length 2 or more. Consider now the following sequence:

$$q \ = \ \mathsf{bestPath}(X_0)/(Y_{0,k_0}\ d)\ \ \mathsf{bestPath}(X_1)/(Y_{1,k_1}\ d)\ \ \mathsf{bestPath}(X_2)/(Y_{2,k_2}\ d)\ \ \cdots$$

which is clearly a directed path through the network. Because $\langle V, E\rangle$ is a dag, q is acyclic. Hence, the chain of best paths under consideration must terminate, and it must terminate with a best path of length 1.

Hence, in every $\mathcal{N} = \langle V, E, \mathcal{W}\rangle$ where $\langle V, E\rangle$ is a dag, there is always a node X such that $\mathsf{bestPath}(X)$ is of length 1. (This is not the case if $\langle V, E\rangle$ is not a dag, as illustrated by any of the networks depicted in Figure 1, or 2, or 3.)

The rest of the proof is an induction on the size n, i.e. the number of nodes in the network $\mathcal{N} = \langle V, E, \mathcal{W}\rangle$. If $n = 1$ or $n = 2$, the result is immediate: $\mathcal{N}$ has a stable configuration.

For the induction step, suppose the result is true for every network with $n \geqslant 3$ nodes. Consider an arbitrary network $\mathcal{N} = \langle V, E, \mathcal{W}\rangle$ with $n + 1$ nodes. We need to prove that $\mathcal{N}$ has a stable configuration. Consider an arbitrary node X in $\mathcal{N}$ such that the length of $\mathsf{bestPath}(X)$ is 1, i.e. $\mathsf{bestPath}(X) = X\ d$. Such a node X exists, as shown above. We construct another network $\mathcal{N}'$ from $\mathcal{N}$ by deleting node X in a particular way. How X is deleted is shown in Figure 9 informally.

The construction shown in Figure 9 is not yet completed, because it may introduce multiple edges in $\mathcal{N}'$ between the Y-nodes and d. We explain how to eliminate multiple edges in the case of node Y, and the same can be done again for the other Y-nodes in the figure. Suppose that in the original network $\mathcal{N}$, there is already an edge from Y to d, which we call e. Suppose also that $Y \in V_{\#}$ where $\# \in \{\square, \circ, \diamond\}$. In the new network $\mathcal{N}'$, we define the weights of edge (Y, d) as follows:

$$w(Y,d) = \ (w_\square(Y,d)\ /\ w_\circ(Y,d)\ /\ w_\diamond(Y,d))$$
$$= \begin{cases} (w_\square(e)\ /\ w_\circ(e)\ /\ w_\diamond(e)) & \text{if } w_\#(e) < a + m \text{ and } \# = \square, \\ (w_\square(e)\ /\ w_\circ(e)\ /\ w_\diamond(e)) & \text{if } w_\#(e) < b + n \text{ and } \# = \circ, \\ (w_\square(e)\ /\ w_\circ(e)\ /\ w_\diamond(e)) & \text{if } w_\#(e) < c + p \text{ and } \# = \diamond, \\ (a + m\ /\ b + n\ /\ c + p) & \text{otherwise.} \end{cases}$$

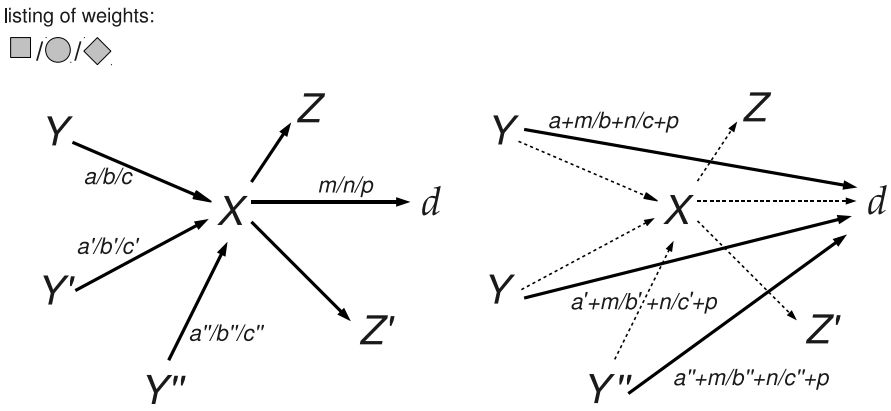

Figure 9: At the top, network $\mathcal{N}$ before node X is deleted. At the bottom, network $\mathcal{N}'$ after node X is deleted. For purposes of illustration, we show the case when there are 3 edges incoming to X and 3 edges outgoing from X, including the edge (X, d). All the edges from the Y-nodes to X, all the edges from X to the Z-nodes, and edge (X, d), are deleted. New edges from the Y-nodes to d are added, with the weights as shown.

The underlying graph of the new network $\mathcal{N}'$ is clearly a dag. By the induction hypothesis, $\mathcal{N}'$ has a stable configuration. It is now an easy argument (omitted here) to show that $\mathcal{N}$ also has a stable configuration, obtained by adding the edge (X, d) to the stable configuration in $\mathcal{N}'$.

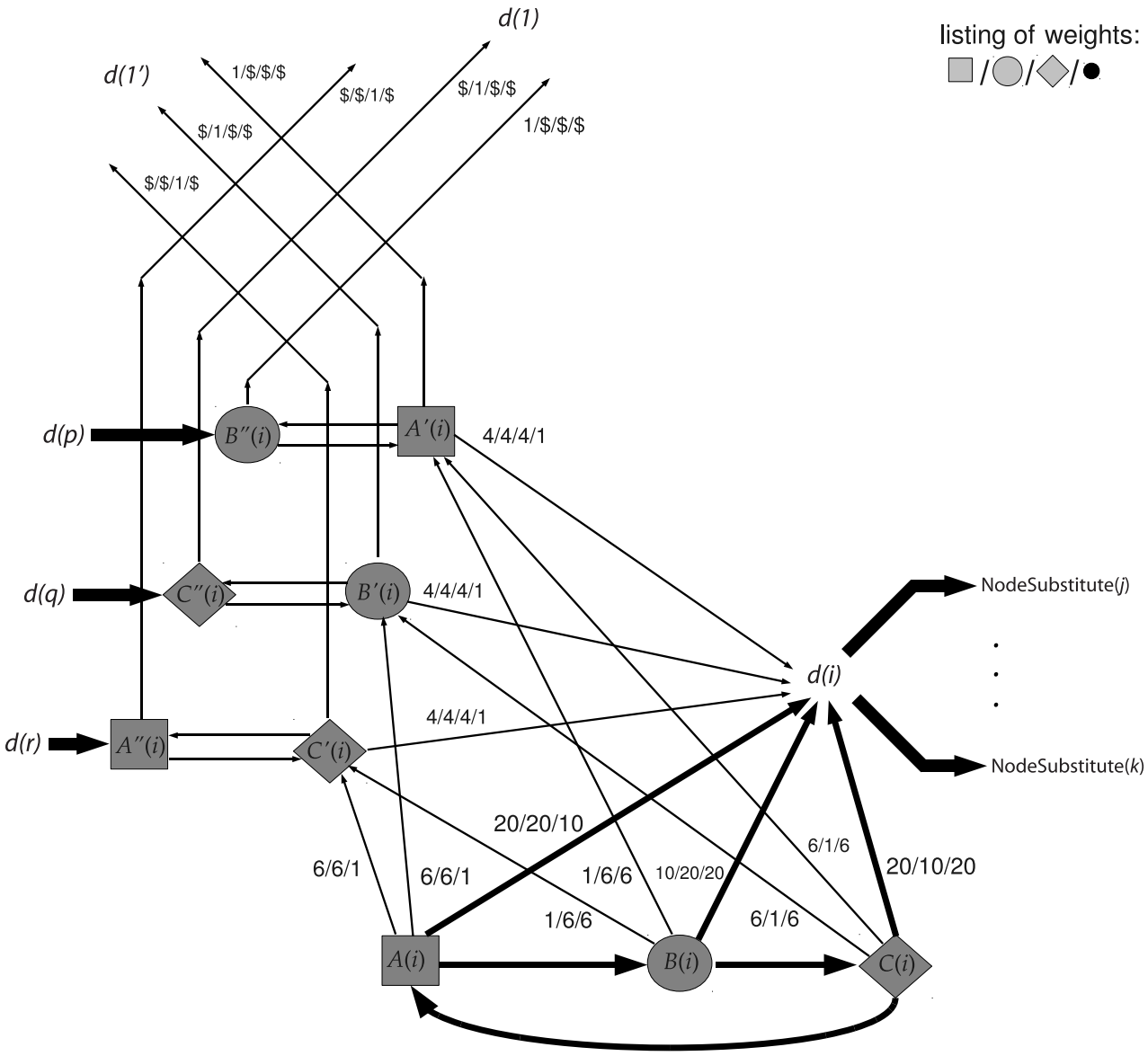

Figure 10: NodeSubstitute(i) replaces node i in the construction of network $\mathcal{N}(G)$ from graph G. We use several conventions to help understand the functioning of NodeSubstitute(i): • Edges inherited from G are in heavy boldface. • The subgraph with edges in medium boldface is a copy of $\mathcal{N}_{\text{unstable}}$, the inherently unstable network in Figure 3. • The weight denoted "$" is a "very large number", e.g. an integer strictly larger than the sum of all the other weights in $\mathcal{N}(G)$ other than "$" itself. • Nodes $d(i), d(p), d(q)$ and $d(r)$ are all of the same kind, which is a new 4-th kind (called "bullet") different from the three other kinds ("square", "round" and "diamond"). • The 4-th weight on edges not accessible from $d(p)$, $d(q)$, and $d(r)$, plays no role and is therefore omitted – these are all the edges outgoing from nodes $A(i)$, $B(i)$ and $C(i)$. • For simplicity we omit weight labels of edges whose weight assignment is "1/1/1" or "1/1/1/1".

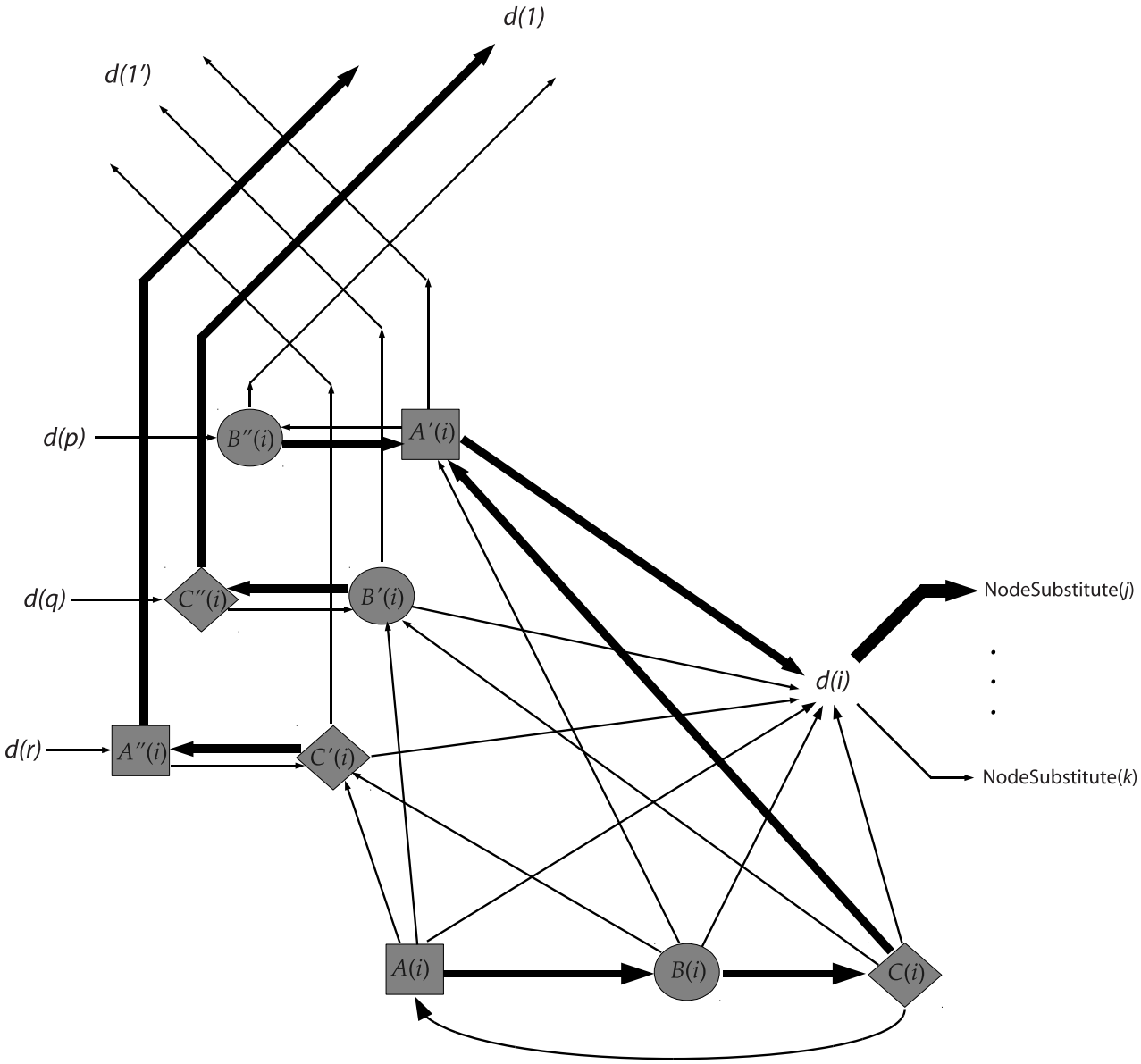

Figure 11: $\mathbb{C}_i$ is a spanning forest (the boldface edges) covering all the nodes of NodeSubstitute(i). $\mathbb{C}_i$ is part of the stable configuration $\mathbb{C}$ of network $\mathcal{N}(G)$ induced by a Hamiltonian path P from $1'$ to 1 in graph G'. The figure illustrates the case when P contains the edge (p, i) from node p to node i, and the edge (i, j) from node i to node j, in G'. Simular figures, here omitted, illustrate the other cases, i.e. when P contains one of the other edges incoming to i or outgoing from i.

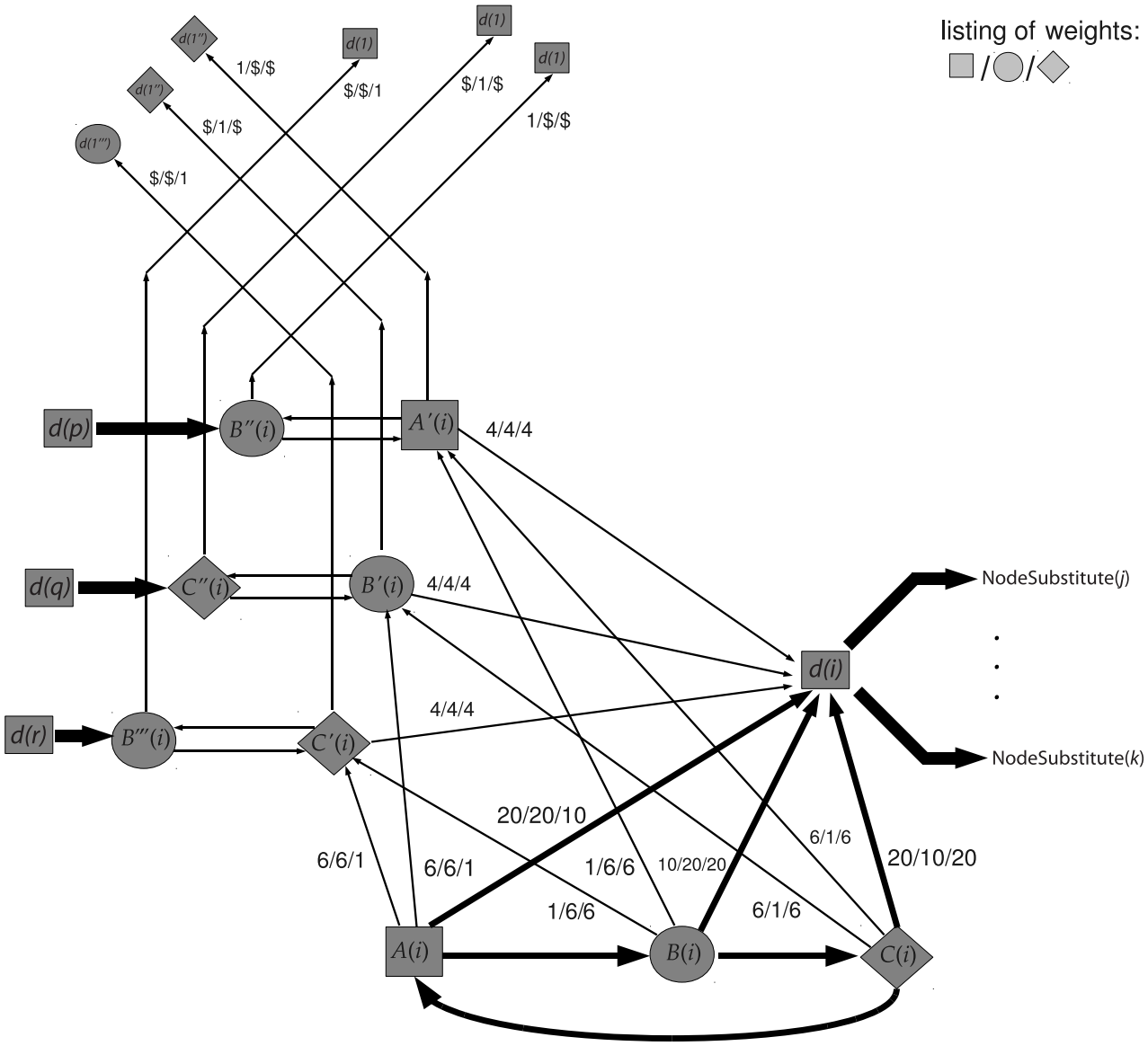

Figure 12: NodeSubstitute(i) restricted to 3 routing policies. We follow the same graphical conventions as in Figure 10, but note the differences: • Nodes $d(i), d(p), d(q)$ and $d(r)$ are now all "square" nodes. • There is no longer a 4-th weight on edges, and all omitted weight labels are "1/1/1". • Square node $A''(i)$ in Figure 10 is now replaced by round node $B'''(i)$.