Cheapest Paths in Multi-Interface Networks*

Ferruccio Barsi, Alfredo Navarra, and Maria Cristina Pinotti

Dipartimento di Matematica e Informatica, Università degli Studi di Perugia,
Via Vanvitelli 1, 06123 Perugia, Italy.
Emails: {barsi,pinotti}@unipg.it, navarra@dipmat.unipg.it

Abstract. Let $G = (V, E)$ be a graph which models a set of wireless devices (nodes V) that can communicate by means of multiple radio interfaces, according to proximity and common interfaces (edges E). The problem of switching on (activating) the minimum cost set of interfaces at the nodes in order to guarantee the coverage of G was recently studied. A connection is covered (activated) when the endpoints of the corresponding edge share at least one active interface. In general, every node holds a subset of all the possible k interfaces. Such networks are known as the multi-interface networks. In such networks, we study a basic problem called *Cheapest Paths*, corresponding to the well-known Shortest Path problem in graph theory. *Cheapest Paths* turns out to be polynomially solvable in $O(k|V||E|)$ in general, and in $O(k|E|)$ in the uniform cost case, i.e., when the cost of activating an interface is the same for any interface.

Keywords: energy saving, wireless network, multi-interface network

1 Introduction

While technology advances and more powerful devices are released, special effort is required for managing new kinds of communication problems. Nowadays wireless devices hold multiple radio interfaces, allowing switching from one communication network to another according to required connectivity and related quality. The selection of the "best" radio interface for a specific connection might depend on various factors. Namely, availability of specific interfaces in the devices, the required communication bandwidth, the cost (in terms of energy consumption) of maintaining an active interface, the available neighbors and so forth. Different interfaces may have different costs. While managing such connections, a lot of effort must be devoted to energy consumption issues. Devices are, in fact, usually battery powered and the network survivability might depend on their persistence in the network. This introduces challenging and natural optimization problems that must take care of different variables at the same time. Generally speaking, given a graph $G = (V, E)$, where V represents the set of wireless devices, each $v \in V$ is associated with a set of available interfaces $I(v)$, and E is the set of possible connections according to proximity of devices and the available interfaces that they may share, the problem can be stated as follows. Given a source node s and a target node t in G, what is the cheapest way, i.e., which subset of available interfaces in some nodes must be activated in order to guarantee a path between s and t while minimizing the overall cost? Note that a connection is satisfied when the endpoints of the corresponding edge share at least one active interface. Moreover, for each node $v \in V$ there is a set of available

* The research was partially funded by the European project COST Action 293, "Graphs and Algorithms in Communication Networks" (GRAAL).

interfaces, from now on denoted as $I(v)$. $\bigcup_{v \in V} I(v)$ determines the set of all the possible interfaces available in the network, whose cardinality is k.

1.1 Related Work

The problem originated from [1] where a slightly different model is introduced. That model considers the necessity of activating all the connections expressed by G while minimizing the overall cost. We can refer to such a problem as *Coverage* of G instead of *Connectivity*. Different interfaces may have different costs and also mutually exclusive interfaces were considered. These are interfaces that, if activated, preclude the activation of some other interfaces. [1] provides a sketchy proof concerning the *NP*-hardness of the Coverage problem and experimental results were shown. In [3, 2] the Coverage problem was formally defined. It was called Cost Minimization in Multi-Interface Networks (k-CMI for short), where the constant k is the number of available interfaces among all the network G. The algorithmic approach led to interesting hardness and approximation results for various graph classes like complete graphs, trees, planar, bounded degree and general graphs. Moreover, both uniform cost and non-uniform cost interfaces were considered. Indeed, the uniform cost model is equivalent to ask for the minimum total number of activated interfaces inside the network in order to cover all the connections. Results in similar context have been obtained in [4, 2] but for a slightly different scenario where k is not known in advance, i.e., k depends on the input instance (Unbounded case). Such a variation of the problem was called Cost Minimization in Unbounded Multi-Interface Networks (CMI for short).

1.2 Our Results

In this paper, we study the so called *Cheapest Path* problem. Given a graph $G = (V, E)$ and a source node $s \in V$, we are interested in determine all the paths of minimum costs from s to any other node. The cost of a path is given by the cost of the cheapest set of interfaces necessary to cover the edges of the path. The interest in this problem comes from the necessity to study in multi-interface networks a problem corresponding to the well-known shortest path tree determination for standard graphs.

We consider the unbounded case, when the number k is not known a priori. We show that in both uniform and non-uniform cost interface cases, the problem can be solved polynomially, and we provide two algorithms with time complexity $O(k|V||E|)$ and $O(k|E|)$, respectively.

1.3 Outline

The next section provides definitions and notation in order to formally describe and study the *Cheapest Path* problem. Section 3 contains considerations and hardness results along with algorithms for solving the problem. Finally, Section 5 contains conclusive remarks.

2 Definitions and Notation

This section provides the definitions and the notation to formally define and study the *Cheapest Path* problem.

Unless otherwise stated, the network graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ is always assumed to be simple (i.e., without multiple edges), undirected and connected. The set of neighbors of a node $v \in V$ is denoted by $N(v)$, the set of all the available interfaces in v by $I(v)$. Let k be the cardinality of all the available and distinct interfaces in the network, that is, $k = \left|\bigcup_{v \in V} I(v)\right|$. For each node $v \in V$, an appropriate interface assignment function W guarantees that each network connection (i.e., each edge of G) is *covered* by at least one interface.

Definition 1. *A function* $W\colon V \to 2^{\{1,\dots,k\}}$*, with* $k = \left|\bigcup_{v \in V} I(v)\right|$*, is said to* cover *graph* $G = (V, E)$ *if for each* $(u, v) \in E$ *the set* $W(u) \cap W(v) \neq \emptyset$*.*

Moreover, for each node $v \in V$, let $W_A(v)$ be the set of switched on (activated) interfaces. Clearly, $W_A(v) \subseteq W(v) \subseteq I(v)$.

The cost of activating an interface for a node is assumed to be identical for all nodes and given by cost function $c\colon \{1, \dots, k\} \to \mathbb{N}$. The cost of interface i is denoted as c_i. Moreover, in the *uniform* case, all the interfaces have the same cost $c > 0$.

A path P in G from a given source node s to a target node v is denoted by a sequence of couples: for each node $v_j \in P$, besides node v_j itself, the interface i_j used to reach v_j is given. Namely, $i_j \in W_A(v_j)$. For example, the sequence $P = \langle (s \equiv v_0, 0), (v_1, i_1), \dots, (v \equiv v_t, i_t) \rangle$, denotes a path P from s to v that moves on the nodes $s, v_1, \dots, v_{t-1}, v$ and that reaches node v_j via interface i_j, for $1 \leq j \leq t$. Interface 0 is used to denote "no interface" since the source is not reached by any other node in P. Conventionally, $c_0 = +\infty$. However, the source needs to activate interface i_1 in order to reach v_1, hence the cost of activating the edge (s, v_1) is $2c_1$. In general, the cost for activating the path P is

$$d_P(v) = \sum_{j=1}^{t} \mathit{cost}\left((v_{j-1}, i_{j-1}), (v_j, i_j)\right)$$

where

$$\mathit{cost}\left((v_{j-1}, i_{j-1}), (v_j, i_j)\right) = \begin{cases} 2c_{i_j} & \text{if } i_{j-1} \neq i_j \\ c_{i_j} & \text{otherwise} \end{cases}$$

Let $\delta(v)$ be the minimum cost to activate a path from the source node s to node v, that is, $\delta(v) = \min\{d_P(v)\colon P \text{ is any path from } s \text{ to } v\}$. In addition, let the *cheapest path* CP_v from the source s to v be any path P from s to v such that $d_P(v) = \delta(v)$. An i-path P from s to v is a path from s to v that reaches v via interface i. Let $d_P(v, i)$ denote the cost of the i-path P, whereas $\delta(v, i)$ denotes the minimum cost among all the i-paths from s to v. Besides, let the *cheapest i-path* $CP_{v,i}$ from the source node s to node v be any i-path P such that $d_P(v, i) = \delta(v, i)$. Clearly, $\delta(v) = \min\{\delta(v, i)\colon i \in W(v)\}$. Whenever clear by the context we remove P from the notation d_P.

3 Cheapest Paths

In this section we study the usually called Shortest Path problem but in the context of multi-interface networks. Actually, in these networks, dealing with shortest paths is not of both practical and theoretical interest. In fact, as shown in Section 2, the cost of an edge is not set up a priori as in the standard problem, but depends on the activated interfaces at its endpoint. The *Cheapest Path* (*CP* for short) problem can be formulated as follows.

Cheapest Path (CP)

***Input*:**	A graph $G = (V, E)$, an allocation of available interfaces $W\colon V \to 2^{\{1,\dots,k\}}$ covering graph G, an interface cost function $c\colon \{1,\dots,k\} \to \mathbb{N}$ and a source node $s \in V$.
***Solution*:**	A set of $n-1$ paths, one for each node but s. For each node $v \in V\backslash\{s\}$, a path P from s to v must be specified by a sequence of couples of the form (v_j, i_j), with $v_0 = s$, $i_0 = 0$, $v_t = v$ and $v_j \in V$, $i_j \in W(v_j)$ for $1 \leq j \leq t$ with the meaning that node v_j is reached by means of interface i_j.
***Goal*:**	For each node $v \in V$, find $\delta(v)$ along with a cheapest path CP_v.

In order to solve the CP problem, let us start with a basic property that holds for the cheapest path problem in multi-interface network because the edges have non negative cost. Letting a *simple* path be a path that does not pass through a same node more than once, it holds:

Lemma 1. *Given a graph G, the cheapest path between any two nodes of G can be found among the simple paths.*

Proof. Let $\{s, v_1, v_2, \dots, v_{i-1}v_i, \dots, v_j, \dots, v_t\}$ be the sequence of nodes touched by CP_{v_t} in G. By contradiction, let $v_i = v_j$ for some $1 \leq i < j \leq t$. The path $\{s, v_1, v_2, \dots, v_{i-1}, v_j, \dots, v_t\}$ obtained by removing all the subpath from v_i to v_j is still a path from v_1 to v_t and its cost is at most the same as that of the original path. □

From Lemma 1, the following Corollary holds.

Corollary 1. *Given a graph G, the cheapest path between any two nodes of G is composed of at most $n-1$ hops.*

In order to better understand differences with the standard shortest path problem, let us consider the simple network of Figure 1, and assume we want to determine the cheapest path problem from source node a in the graph G, in a setting where the costs of interfaces 1, 2 and 3 are 1.5, 1.5 and 1, respectively. We find that $\delta(d) = 6$ and $CP_d = \langle(a,0),(e,2),(f,2),(d,2)\rangle$, whereas $\delta(g) = \frac{15}{2}$ and $CP_g = \langle(a,0),(b,1),(c,1),(d,3),(g,3)\rangle$. However the supbapth of CP_g from a to d, that is $\langle(a,0),(b,1),(c,1),(d,3)\rangle$, has cost $\frac{13}{2} > \delta(d)$. Therefore, it turns out, that the main property of the standard shortest path problem does not hold. Precisely:

Proposition 1. *A subpath of a cheapest path is not necessarily a cheapest path itself.*

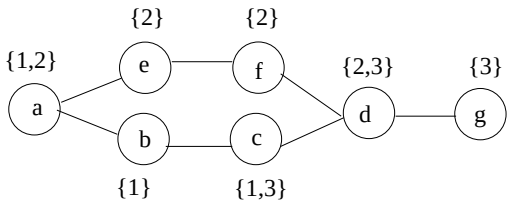

Fig. 1. A sample network. In brackets for each node there are the corresponding available interfaces.

While the sub-optimality property does not hold in general, it holds when we consider a subpath of a cheapest path characterized not only by its final node, but also by the interface used in its last hop.

Lemma 2. *Given a graph $G = (V, E)$ and a source node $s \in V$, let CP_v be a cheapest path from s to v that passes through node u and reaches u via interface i. Then, the i-subpath of CP_v from s to u is a cheapest i-path.*

Proof. Let P denote the i-subpath of CP_v from s to u. By contradiction, suppose that P is not a cheapest i-path, and hence $d_P(u, i) > \delta(u, i)$. Since a cheapest path $CP_{u,i}$ reaches u via interface i, replacing P with $CP_{u,i}$ in CP_v, a new path P' from s to v with cost $d_{P'}(v) = \delta(v) - (d_P(u, i) - \delta(u, i)) < \delta(v)$ is found, contradicting the definition of cheapest path. □

As a consequence, the cost of a cheapest path from s to v can be easily determined when the endpoints of the final edge of such a path and the interface used to reach v are given. In fact:

Lemma 3. *Given a graph $G = (V, E)$ and a source node $s \in V$, the cost $\delta(v, i)$ of the cheapest i-path CP_v from s to v which has node u as the parent of v satisfies:*

$$\delta(v, i) = \min\{\delta(u, i) + c_i, \delta(u) + 2c_i\}.$$

Proof. If j were the interface used to reach node u in the cheapest i-path CP_v, by Lemma 2, the cost of the cheapest i-path CP_v would satisfy $\delta(v, i) = \delta(u, j) + cost((u, j), (v, i))$. Recalling that $1 \leq j \leq k$ and $cost((u, j), (v, i)) = 2c_i$ if $j \neq i$, $\delta(v, i) = \min\{\delta(u, i) + c_i, \min\{\delta(u, j) + 2c_i \colon 1 \leq j \neq i \leq k\}\} = \min\{\delta(u, i) + c_i, \min\{\delta(u, j) + 2c_i \colon 1 \leq j \; lek\}\} = \min\{\delta(u, i) + c_i, \delta(u) + 2c_i\}$. □

The CP problem, for a given graph $G = (V, E)$ and a given source $s \in V$, is solved by Procedure `Parents_Setup` (see Algorithm 2). This determines, for each node $v \in V$, $v \neq s$ and for each interface $i \in W(v)$, the cost $\delta(v, i)$ of the cheapest i-path $CP_{v,i}$, the parent node $\pi(v, i)$ of v on $CP_{v,i}$, and the cost $\delta(v)$ of the cheapest i-path CP_v. In order to make use of Procedure `Parents_Setup`, we initialize with $+\infty$ (see Algorithm 1), for each $v \in V$ and $i \in I(v)$, the variables $d(v)$, $d(v, i)$ which estimate $\delta(v)$ and $\delta(v, i)$, respectively.

Algorithm 1 The `Initialize` procedure

Procedure Initialize(graph: $G = (V, E)$; node: s)

1: $d(s) := 0$
2: **for** each $i \in I(s)$ **do**
3: $\quad d(s, i) := 0$
4: **end for**
5: **for** each $v \in V$, $v \neq s$ **do**
6: $\quad$ **for** each $i \in I(v)$ **do**
7: $\quad\quad d(v, i) := +\infty$
8: $\quad\quad d(v) := +\infty$
9: $\quad$ **end for**
10: **end for**
11: Parents_Setup(G, $\{s\}$, $|V| - 1$)

At any step, the Procedure `Parents_Setup` computes, for each node $v \in V$, $v \neq s$ and for each interface $i \in W(v)$, for the cheapest i-path from s to v visited so far, its cost $d(v, i)$ and the parent node $\pi(v, i)$ of v on such a path. The values $d(v, i)$ and $d(v)$ are updated according to Lemma 3. At the end of the procedure, since all the possible simple paths in G have been visited, it yields $d(v, i) = \delta(v, i)$ and $d(v) = \delta(v)$, for each node $v \in V$, $v \neq s$ and for each interface $i \in W(v)$.

Algorithm 2 The `Parents_Setup` procedure

Procedure Parents_Setup(graph: G; set of nodes: U; int: *count*)

1: $TMP := \emptyset$
2: **for** each $u \in U$ **do**
3: $\quad TMP := TMP \bigcup N(u)$
4: $\quad$ **for** each $v \in N(u)$ **do**
5: $\quad\quad$ **for** each $i \in W(u) \bigcap W(v)$ **do**
6: $\quad\quad\quad$ **if** $d(v, i) > d(u, i) + c_i$ **then**
7: $\quad\quad\quad\quad d(v, i) := d(u, i) + c_i$
8: $\quad\quad\quad\quad \pi(v, i) := u$
9: $\quad\quad\quad$ **end if**
10: $\quad\quad\quad$ **if** $d(v, i) > d(u) + 2c_i$ **then**
11: $\quad\quad\quad\quad d(v, i) := d(u) + 2c_i$
12: $\quad\quad\quad\quad \pi(v, i) := u$
13: $\quad\quad\quad$ **end if**
14: $\quad\quad\quad$ **if** $d(v) > d(v, i)$ **then**
15: $\quad\quad\quad\quad d(v) := d(v, i)$
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end for**
18: $\quad$ **end for**
19: **end for**
20: **if** $--count > 0$ **then**
21: $\quad$ Parents_Setup(G,TMP,*count*)
22: **end if**

As regard to the time complexity of Procedure `Parents_Setup`, it is worth noting that the double *for* at code lines 2 to 4 leads to m executions of the third *for*. In fact, such double

for explores at most all the possible connections among nodes when $U \equiv V$. Since Procedure `Parents_Setup` is called $n-1$ times, and the *for* at code line 5 iterates for at most k rounds, the time complexity of the algorithm is $O(knm)$.

Clearly, when $\ell = |V| - 1$, by Lemma 1, all the simple paths have been generated and for each $v \in V$ and $1 \leq i \leq k$, $d(v,i) = \delta(v,i)$ and $d(v) = \delta(v)$.

Lemma 4 (Correctness of Procedure `Parents_Setup`). *At the end of the ℓ-th call of Procedure `Parents_Setup`, with $0 \leq \ell \leq |V| - 1$:*

1) all the paths of at most ℓ hops from the source s have been considered;

2) for each reached node $v \in U$, $d(v,i)$ is the cost of the cheapest i-path from s to v with at most ℓ hops and $\pi(v,i)$ is the parent node of v in some cheapest i-path CP_v, where $1 \leq i \leq k$.

Proof. 1) Follows directly by noting that at each call of Procedure `Parents_Setup`, all the neighbors, and hence all the outgoing edges, of the input set U are explored, and then such a set is given as input U in the next call. So, starting from the source s, after the first call, U matches $N(s)$. At the second call, all the nodes at two hops from s are then reached since these are all the neighbors of the neighbors of s. The visit of the graph keeps on working in the same way until the ℓ-th level.

Concerning 2), let us prove by induction on ℓ. The result is true for $\ell = 1$, when all the cost of cheapest paths from s and its neighbors $N(s)$ are computed. Clearly, s is the parent node in all such paths.

In any subsequent execution of Procedure `Parents_Setup`, we consider all possible ways to extend by one hop a path, with at most $\ell - 1$ hops, from s to a node given in input in U. Precisely, for each $u \in U$ and for each $v \in N(u)$, it is considered to reach v by extending the cheapest paths found so far from s to u by the couple (v,i) for $i \in W(u) \bigcap W(u)$. The cost of each extended path is determined by applying Lemma 3, and compared with that of the previous known cheapest i-path from s to v, in code lines 5–17. If a cheaper path is found, $d(v,i)$ and $\pi(v,i)$, and consequently $d(v)$, are updated. After the ℓ-th execution of Procedure `Parents_Setup`, all the paths with at most ℓ hops have been considered. □

Once all the parents have been sorted out, for each node v, we can make use of Procedure `Cheapest_Path` (see Algorithm 3) with input parameters (G,s,v,0) in order to obtain the cheapest path CP_v as the sequence of couples (node, interface). Such a procedure takes $O(nk)$ time.

4 Uniform Case

In this section, the CP problem for multi-interface networks with uniform cost is studied. In this case, each interface has the same cost c, and without loss of generality $c = 1$. A path edge costs 1 or 2 depending on whether it uses or not the same interface as the previous edge on the path. Hence, according to Corollary 1, any cheapest path from s to v satisfies $\delta(v) \leq 2(|V| - 1)$.

Algorithm 3 The `Initialize` procedure

Procedure Cheapest_Path(graph: G; node: s, t; interface: j)
1: **if** $s \neq t$ **then**
2: $i := 0$
3: **while** $d(t) \neq d(t,i)$ **do**
4: $i++$
5: **end while**
6: **if** $j! = 0$ **and** $d(t,j) + c_j < d(t,i) + 2c_j$ **then**
7: $i := j$
8: **end if**
9: Cheapest_Path(G,s,$\pi(t,i)$,i)
10: return (t,i)
11: **else**
12: return $(s,0)$
13: **end if**

In this context, and considering again Figure 1, let us assume we want to solve CP with source node a. We obtain $\delta(d) = 4$ with $CP_d = \langle (a,0), (e,2), (f,2), (d,2)\rangle$ and $\delta(g) = 6$ with two different cheapest paths from a to g, namely, $CP_g = \langle (a,0), (e,2), (f,2), (d,2), (g,3)\rangle$ and $CP'_g = \langle (a,0), (b,1), (c,1), (d,3), (g,3)\rangle$. Although the subpath of CP'_g from a to d is not the cheapest path from a to d, the subpath of CP_g from a to d does it. In fact, it can be proved that:

Lemma 5. *Given a graph G and a pair of nodes s and v, in the uniform case, there is at least a cheapest path CP_v from s to v whose subpaths are, at their turn, cheapest paths.*

Proof. By contradiction, let us consider that no cheapest paths from s to v satisfy the above claim. Then, consider a cheapest path $P = \langle (s,0), (v_1,i_1), \ldots, (v_{t-1},i_{t-1}), (v,i_t)\rangle$ with cost $\delta(v)$. Its subpath $P' = \langle (s,0), (v_1,i_1), \ldots, (v_{t-1},i_{t-1})\rangle$. which is not optimal, has cost $d(v_{t-1}) \leq \delta(v) - 1$. Let P^* be a cheapest path from s to v_{t-1} with cost $\delta(v_{t-1})$. Clearly, $\delta(v_{t-1}) \leq d(v_{t-1}) - 1 \leq \delta(v) - 2$. Since the edge $e = (v_{t-1}, v)$ belongs to the graph G, and since by definition $W(v_{t-1}) \cap W(v) \neq \emptyset$, there is a new path $\hat{P}$ obtained by the concatenation of the cheapest path P^* from s to v_{t-1} with the edge $e = (v_{t-1}, v)$ whose cost $d_{P^*}(v) \leq \delta(v_{t-1}) + 2 \leq \delta(v)$. Hence, there exists a cheapest path which satisfies the sub-optimality property, raising a contradiction. □

According to Lemma 5, in the remaining of this section we can restrict our attention to the cheapest paths whose subpaths are, at their turn, cheapest paths.

Let $W_\alpha(v)$ be the set of all the interfaces that allow to reach a node v from the source s with optimal cost $\delta(v)$. Note that $W_A(v) \subseteq W_\alpha(v) \subseteq W(v) \subseteq I(v)$. Besides, to build a cheapest path CP_v, for each node v and each interface $i \in W_\alpha(v)$, a parent node $\pi(v,i)$ of v on a cheapest path that reaches v via interface i has to be maintained. Indeed, again differently from the usual shortest path problem but similarly to the non-uniform case, the set of the cheapest paths from a source node to each node in a graph does not form a tree. The cheapest path from s to v, along with its cost $\delta(v)$, can be easily determined when the parent node u of v and the interfaces used to reach u are known. In fact:

Lemma 6. *Given a graph $G = (V, E)$ and a source node $s \in V$, the cost $\delta(v)$ of the cheapest path CP_v from s to v which has node u as the parent of v is given by:*

$$\delta(v) = \begin{cases} \delta(u) + 1 & \text{if } W_\alpha(u) \bigcap W(v) \neq \emptyset \\ \delta(u) + 2 & \text{otherwise} \end{cases}$$

Moreover, the set of interfaces that allow to reach v from u with cost $\delta(v)$ is given by:

$$W_\alpha(u, v) = \begin{cases} W_\alpha(u) \bigcap W(v) & \text{if } W_\alpha(u) \bigcap W(v) \neq \emptyset \\ W(u) \bigcap W(v) & \text{otherwise} \end{cases}$$

Proof. If there is at least one interface in common between $W_\alpha(u)$ and $W(v)$, it is possible to extend the cheapest path CP_u up to reach v just activating a single interface. Otherwise CP_u can only be extended switching the same interface in both nodes u and v. It follows that the set of interfaces that can be used to reach node v from u is given by $W_\alpha(u, v)$. □

Procedure `Uniform_Parents_Setup` (see Algorithm 5) greedily builds a set S of nodes for which the cost of the cheapest path has been found. For each node $u \in S$, the procedure maintains the cost $\delta(u)$ of CP_u, the set of all the interfaces $W_\alpha(u)$ that allow to reach u with cost $\delta(u)$ and, for each $i \in W_\alpha(u)$ a parent node $\pi(u, i)$ of u on some CP_u that reaches u via interface i.

Algorithm 4 The `Uniform_Initialize` procedure

Procedure Uniform_Initialize (graph: $G = (V, E)$; source node s)
1: $d(s) := 0$
2: $W_\alpha(s) := \emptyset$
3: $\pi(s, 0) := -1$
4: **for** each $v \in V \backslash S$ **do**
5: $\quad d(v) := 2|V|$
6: $\quad W_\alpha(v) := \emptyset$
7: $\quad \pi(v, 0) := -1$
8: **end for**
9: $S := \{s\}$
10: $\delta(s) := 0$
11: Uniform_Parents_Setup$(G, S, s, |V| - 1)$

To start, Procedure `Uniform_Initialize` (see Algorithm 4) sets $S = \{s\}$, $\delta(s) = 0$, $W_\alpha(s) = \emptyset$ and $\pi(s, 0) = -1$ because, obviously, the cheapest path from s to s has cost 0, no interface are used to reach s, and no node is parent of s (conventionally, -1). Moreover, for each vertex $u \in V \backslash S$, the procedure sets the estimated cost $d(v)$ for CP_v to $2|V|$, $W_\alpha(v) = \emptyset$ and $\pi(v, 0) = -1$.

Procedure `Uniform_Parents_Setup` examines the nodes in the neighborhood of the last node inserted into S. For each node $v \in N(u)$ and $v \in V \backslash S$, according to Lemma 6, the procedure computes (code lines 2–15) the cost of the path obtained by extending the cheapest path CP_u up to v and also finds the interfaces to traverse (u, v). If the new path

has cost smaller than or equal to the current estimated cost to reach v, all the information, $d(v)$, $\delta(v)$, $W_\alpha(v)$, and $\pi(v,i)$ for each $i \in W_\alpha(u,v)$, are updated accordingly. Procedure `Uniform_Parents_Setup` is called $|V|-1$ times until $V \backslash S$ becomes $\emptyset$. Its time complexity is $O(km)$, see Theorem 1.

Algorithm 5 The `Uniform_Parents_Setup` procedure

Procedure Uniform_Parents_Setup(graph: G; set: S, node: u, int: *count*)

for each $v \in N(u)$ **and** $v \notin S$ **do**
  if $W_\alpha(u) \cap W(v) \neq \emptyset$ **then**
    $cost := 1$
    $W_\alpha(u,v) := W_\alpha(u) \cap W(v)$
  else
    $cost := 2$
    $W_\alpha(u,v) := W(u) \cap W(v)$
  end if
  if $d_u + cost < d_v$ **then**
    $d(v) := d(u) + cost$
    for $i \in W_\alpha(v)$ **do**
      $\pi(v,i) := -1$
    end for
    $W_\alpha(v) := W_\alpha(u,v)$
    for $i \in W_\alpha(u,v)$ **do**
      $\pi(v,i) := u$
    end for
  end if
  if $d(u) + cost = d(v)$ **then**
    $W_\alpha(v) := W_\alpha(v) \bigcup W_\alpha(u,v)$
    for $i \in W_\alpha(u,v)$ **do**
      $\pi(v,i) := u$
    end for
  end if
end for
$u := v$ such that $v \in V \backslash S$ and $d(u) = \min d(v)$
$\delta(u) := d(u)$;
$S := S \bigcup \{u\}$;
if $--count > 0$ **then**
  Uniform_Parents_Setup(G, S, u, *count*)
end if

Lemma 7 (Correctness of Procedure `Uniform_Parents_Setup`). *At the end of the ℓ-th call of Procedure `Uniform_Parents_Setup`, with $1 \leq \ell \leq |V|-1$:*
1) S has cardinality $\ell + 1$;
2) for each node $v \in S$, $d(v) = \delta(v)$ and for each $i \in W_\alpha(v)$, $\pi(v,i)$ is the parent of v on some cheapest path CP_v that reaches v via interface i. Moreover, $W_\alpha(v)$ is the complete set of interfaces that allow to reach v with cost $\delta(v)$.

Proof. Let us prove by induction on the number of call of the procedure. When $\ell = 1$, the claim easily follows by Lemma 6.

Assume that at the end of the ℓ-th call of the procedure, $\ell \geq 2$, v has been erroneously inserted into S and let v be first node for which this happens. That is, either:
(i) there is a cheapest path CP_v^* from s to v that reaches v via interface $\bar{\imath}$, but $\bar{\imath} \notin W_\alpha(v)$, or
(ii) $d(v) > \delta(v)$.

Note that, in order to reach v, CP_v^* must leave the set S somewhere: let y be the first node on CP_v^* which is not in S and let x be the node just before y. By inductive hypothesis, since $x \in S$, $d(x) = \delta(x)$ and $W_\alpha(x)$ contains all the interfaces that can be used to reach x in any cheapest path. Let x be selected in the t-th call of Procedure `Uniform_Parents_Setup` and let $t < \ell$. Since the neighborhood of x has been examined during the $t+1$-th call, and $t+1 \leq \ell$, and since (x, y) belongs to a cheapest path, by Lemma 6, we know $d(y) = \delta(y)$ and $W_\alpha(x, y) \subseteq W_\alpha(y)$. Since y is a predecessor of v, it must hold $\delta(y) < \delta(v) \leq d(v)$ (recall $c = 1$). Therefore, Procedure `Uniform_Parents_Setup` must have then considered extracting node y instead of node v. Let us now deal with the two contradictions separately.

Considering case (i), since Procedure `Uniform_Parents_Setup` has rejected y in favor of v, it means that y cannot be a predecessor of v. So, either CP_v^* does not exist and hence all the interfaces are already inserted in $W_\alpha(v)$ or $y \equiv v$ (i.e., $t+1 = \ell$) and hence $\bar{\imath} \in W_\alpha(u, v) \subseteq W_\alpha(v)$.

Considering case (ii), since v has been selected instead of y, it means that $\delta(v) \leq d(v) \leq d(y) = \delta(y)$. Clearly this cannot be true if y is a predecessor since $\delta(y) < \delta(v)$, so it can only be $y \equiv v$ and therefore $\delta(y) = \delta(v) = d(v)$. □

Theorem 1. *Given a graph $G = (V, E)$, with $|V| = n$ and $|E| = m$, the problem of finding the cheapest paths from a given source $s \in V$ is solvable in $O(mk)$ time, and $O(nk)$ space.*

Proof. As regard to the time complexity, assuming that both the union and intersection operations require $O(k)$ time each, overall the $n-1$ calls of Procedure `Uniform_Parents_Setup` takes $O(mk)$ time to examine the edges of G. Then, in order to compute for $n-1$ times the minimum among the estimated costs $d(v)\colon v \in V \backslash S$, we can exploit the facts that: (1) $\delta(v) < 2|V|$ for each $v \in V \backslash S$; and (2) the cost of the cheapest path inserted in S are non decreasing. Hence, it is possible to store the set of the estimated costs $d(v)\colon v \in V \backslash S$ in an array Q of size $2|V|$ such that $Q[i] = \{v : d(v) = i, \quad v \in V \backslash S\}$ is a doubly linked list. Using such a structure Q, the $n-1$ extractions of the minimum from Q take overall $O(|Q|) = O(n)$ time.

In order to discuss the space complexity, observe that each node v requires overall $O(k)$ space to save the parent nodes $\pi(v, i)$ on the cheapest paths and the set of interfaces that can be used to reach it.

Hence, to solve the CP problem in the uniform case $O(mk)$ time and $O(kn)$ space are required. □

Finally, for each node v, we can make use of Procedure `Uniform_Cheapest_Path` (similar to Procedure `Cheapest_Path`, see Algorithm 6) with input parameters (G,s,v,0) in order to obtain the cheapest path CP_v as the sequence of couples (node, interface). Such a procedure takes $O(n)$ time.

Algorithm 6 The `Uniform_Cheapest_Path` procedure

Procedure Uniform_Cheapest_Path (graph: G; nodes s, t; interface: j)

if $s \neq t$ **then**
  if $j \notin W_\alpha(t)$ **then**
    select $i \in W_\alpha(t)$
  else
    $i := j$
  end if
  Cheapest_Path$(G, s, \pi(t,i), i)$
  return (t, i)
else
  return $(s, 0)$
end if

5 Conclusion

We have considered a basic problem in the context of multi-interface networks. Namely, the well-known shortest path tree determination has become the *Cheapest Path* problem. We have shown how the complexity of the original problem has increased, and we have provided suitable algorithms. Many other interesting problems remain unexplored in the context of multi-interface networks. The study of standard optimization problem in this context is challenging for future work.

References

1. M. Caporuscio, D. Charlet, V. Issarny, and A. Navarra. Energetic Performance of Service-oriented Multi-radio Networks: Issues and Perspectives. In *Proceedings of the 6th International Workshop on Software and Performance (WOSP)*, pages 42–45. ACM Press, 2007.
2. R. Klasing, A. Kosowski, and A. Navarra. Cost minimisation in wireless networks with bounded and unbounded number of interfaces. *to appear in Networks.*
3. R. Klasing, A. Kosowski, and A. Navarra. Cost minimisation in multi-interface networks. In *Proceedings of the 1st EuroFGI International Conference on Network Control and Optimization (NET-COOP)*, volume 4465 of *Lecture Notes in Computer Science*, pages 276–285. Springer-Verlag, 2007.
4. A. Kosowski and A. Navarra. Cost minimisation in unbounded multi-interface networks. In *Proceedings of the 2nd PPAM Workshop on Scheduling for Parallel Computing (SPC)*, Lecture Notes in Computer Science. Springer-Verlag, 2007.