Efficient edge-swapping heuristics for finding minimum fundamental cycle bases

Edoardo Amaldi, Leo Liberti, Nelson Maculan*, and Francesco Maffioli

DEI, Politecnico di Milano, Piazza L. da Vinci 32, 20133 Milano, Italy
{liberti,amaldi,maculan,maffioli}@elet.polimi.it

Abstract. The problem of finding a fundamental cycle basis with minimum total cost in a graph is NP-hard. Since fundamental cycle bases correspond to spanning trees, we propose new heuristics (local search and metaheuristics) in which edge swaps are iteratively applied to a current spanning tree. Structural properties that make the heuristics efficient are established. We also present a mixed integer programming formulation of the problem whose linear relaxation yields tighter lower bounds than known formulations. Computational results obtained with our algorithms are compared with those from existing constructive heuristics on several types of graphs.

1 Introduction

Let $G = (V, E)$ be a simple, undirected graph with n nodes and m edges, weighted by a non-negative cost function $w : E \to \mathbb{R}^+$. A *cycle* is a subset C of E such that every node of V is incident with an even number of edges in C. Since an elementary cycle is a connected cycle such that at most two edges are incident to any node, cycles can be viewed as the (possibly empty) union of edge-disjoint elementary cycles. If cycles are considered as edge-incidence binary vectors in $\{0,1\}^{|E|}$, it is well-known that the cycles of a graph form a vector space over $GF(2)$. A set of cycles is a *cycle basis* if it is a basis in this cycle vector space associated to G. The cost of a cycle is the sum of the costs of all edges contained in the cycle. The cost of a set of cycles is the sum of the costs of all cycles in the set. Given any spanning tree of G characterized by an edge set $T \subseteq E$, the edges in T are called *branches* of the tree, and those in $E \backslash T$ (the co-tree) are called the *chords* of G with respect to T. Any chord uniquely identifies a cycle consisting of the chord itself and the unique path in T connecting the two nodes incident on the chord. These $m - n + 1$ cycles are called *fundamental cycles* and they form a *Fundamental Cycle Basis* (FCB) of G with respect to T. It turns out [1] that a cycle basis is fundamental if and only if each cycle in the basis contains at least one edge which is not contained in any other cycle in the basis. In this paper we consider the problem of finding Minimum Fundamental Cycle Bases (Min FCB) in graphs, that is FCBs with minimum

* On academic leave from COPPE, Universidade Federal do Rio de Janeiro, Brasil; e-mail: maculan@cos.ufrj.br.

total cost. Since the cycle space of a graph is the direct sum of the cycle spaces of its biconnected components, we assume that G is biconnected, i.e., G contains at least two edge-disjoint paths between any pair of nodes.

Cycle bases have been used in the field of electrical networks since the time of Kirchoff [2]. Fundamental cycle bases can be uniquely identified by their corresponding spanning trees, and can therefore be represented in a highly compact manner. Besides the above-mentioned characterization, Sysło established several structural results concerning FCBs [3, 1, 4]. For example, two spanning trees whose symmetric difference is a collection of 2-paths (paths where each node, excluding the endpoints, has degree 2) give rise to the same FCB [1]. Although the problem of finding a minimum cycle basis can be solved in polynomial time (see [5] and the recent improvement [6]), requiring fundamentality makes the problem NP-hard [7]. In fact, it does not admit a polynomial-time approximation scheme (PTAS) unless P=NP; that is, under the same assumption there exists no polynomial-time algorithm that guarantees a solution within a factor of $1 + \varepsilon$ for every instance and for any $\varepsilon > 0$ [8]. In the same work, a $4 + \varepsilon$ approximation algorithm is presented for complete graphs, and a $2^{O(\sqrt{\log n \log\log n})}$ approximation algorithm for arbitrary graphs.

Interest in minimum FCBs arises in a variety of application fields, such as electrical circuit testing [9], periodic timetable planning [16] and generating minimal perfect hash functions [10].

The paper is organized as follows. In Section 2 we describe a local search algorithm in which the spanning tree associated to the current FCB is iteratively modified by performing edge swaps, and we establish structural results that make its implementation efficient. In Section 3 the same type of edge swaps is adopted within two metaheuristic schemes, namely a variable neighbourhood search and a tabu search. To provide lower bounds on the cost of optimal solutions, a new mixed integer programming (MIP) formulation of the problem is presented in Section 4. Computational results are reported and discussed in Section 5.

2 Edge-swapping local search

In our local search algorithm for the MIN FCB problem, we start from the spanning tree associated to an initial FCB. At each iteration we swap a branch of the current spanning tree with one of its chords until the cost cannot be further decreased, i.e., a local minimum is found.

2.1 Initial feasible solutions

Initial solutions are obtained by applying a very fast "tree-growing" procedure [11], where a spanning tree and its corresponding FCB are grown by adding nodes to the tree according to predefined criteria. The adaptation of Paton's procedure to the MIN FCB problem proceeds as follows. The node set of the initial tree V_T only contains a root node v_0, and the set X of nodes to be examined is initialized at V. At each step a node $u \in X \cap V_T$ (not yet examined) is selected

according to a predefined ordering. For all nodes z adjacent to u, if $z \notin V_T$, the edge $\{z, u\}$ is included in T (the edge is selected), the node z is added to V_T and the node u is removed from X. Nodes to be examined are selected according to non-increasing degree and, to break ties, to increasing edge star costs. The resulting order tends to maximize the chances of finding very short fundamental cycles early in the process. The performance of this tree-growing procedure is comparable to other existing tree-growing techniques [7, 10].

2.2 Edge swap

Using edge swaps to search the solution space of the MIN FCB problem is a good strategy, since all spanning trees of a graph can be obtained from any initial spanning tree by the repeated application of edge swaps [12]. Consider any given spanning tree T of G. For each branch e of T, the removal of e from T induces the partition of the node set V into two subsets S_T^e and $\bar{S}_T^e$. Denote by δ_T^e the *fundamental cut* of G induced by the branch e of T, i.e., $\delta_T^e = \delta(S_T^e) = \{\{u, v\} \in E \mid u \in S_T^e, v \in \bar{S}_T^e\}$. For any chord $f \in \delta_e^T$, let $\pi = (e, f)$ be the edge swap which consists in removing the branch e from T while adding f to T. Denote by πT the resulting spanning tree.

```
Let T be the initial spanning tree constructed as in Section 2.1;
loop
    Δ_opt := 0;
    initialize π_opt to the identity;
    for all e ∈ T
        for all f ∈ δ_T^e with f ≠ e
            π := (e, f);
            if Δ_π ≥ Δ_opt then
                π_opt = π;
                Δ_opt = Δ_π;
            end if
        end for
    end for
    if π_opt is not the identity then
        T := π_opt T;
    end if
until π_opt is the identity
```

Fig. 1. Local search algorithm for the MIN FCB problem.

For any spanning tree T, let $\mathcal{C}(T)$ be the set of cycles in the FCB associated to T, and let $w(\mathcal{C}(T))$ denote the FCB total cost (the function w is extended to sets of edges in the obvious way: $w(F) = \sum_{f \in F} w(f)$ for any $F \subseteq E$). We

are interested in finding edge swaps $\pi = (e, f)$, where e is a branch of the current tree T and f is a chord in the fundamental cut δ_T^e induced by e, such that $w(\mathcal{C}(\pi T)) < w(\mathcal{C}(T))$. For each branch e and chord f, the cost difference $\Delta_\pi = w(\mathcal{C}(T)) - w(\mathcal{C}(\pi T))$ must be computed. Let Δ_{opt} be the largest such Δ_π, and π_{opt} be the corresponding edge swap. If $\Delta_{\text{opt}} \leq 0$ we let π_{opt} be the identity permutation.

The local search algorithm based on this edge-swapping operation is summarized in Fig. 1.

2.3 Efficient implementation

Given the high worst-case computational complexity of each iteration of the basic local search procedure (see Section 2.4), an efficient implementation is of foremost importance. Since applying an edge swap to a spanning tree may change the fundamental cycles and cut structure considerably, efficient procedures are needed to determine the cuts $\delta_{\pi T}^e$ for all $e \in \pi T$, and to compute Δ_π from the data at the previous iteration, namely from T, π and the cuts δ_T^e, for $e \in T$.

Edge swap effect on the cuts In this subsection we prove that any edge swap $\pi = (e, f)$ applied to a spanning tree T, where $e \in T$ and $f \in \delta_T^e$, changes a cut δ_T^h if and only if f is also in δ_T^h. Furthermore, $\pi(\delta_T^h) = \delta_{\pi T}^h$ is the symmetric difference $\delta_T^h \triangle \delta_T^e$. This makes it easy to maintain data structures relative to the cuts that can be updated efficiently when π is applied to T.

For each pair of nodes $u, v \in V$ let $\langle u, v\rangle$ be the unique path in T from u to v. Let $e = \{u_e, v_e\} \in T$ be an edge of the spanning tree and $c = \{u_c, v_c\} \notin T$ be a chord, where the respective endpoints u_e, v_e, u_c, v_c are nodes in V. Let $p_1(e, c) = \langle u_e, u_c\rangle$, $p_2(e, c) = \langle u_e, v_c\rangle$, $p_3(e, c) = \langle v_e, u_c\rangle$, $p_4(e, c) = \langle v_e, v_c\rangle$ and $P_T(e, c) = \{p_i(e, c) \subseteq T \mid i = 1, \ldots, 4\}$. Note that exactly two paths in $P_T(e, c)$ do not contain e. Let $\bar{P}_T(e, c)$ denote the subset of $P_T(e, c)$ composed of those two paths not containing e. Let $P_T^*(e, c)$ be whichever of the sets $\{p_1(e, c), p_4(e, c)\}$, $\{p_2(e, c), p_3(e, c)\}$ has shortest total path length in T (see Fig. 2). In the sequel, with a slight abuse of notation, we shall sometimes say that an edge belongs to a set of nodes, meaning that its endpoints belong to that set of nodes. For a path p and a node set $N \subseteq V(G)$ we say $p \subseteq N$ if the edges of p are in the edge set $E(G_N)$ (i.e., the edges of the subgraph of G induced by N). Furthermore, we shall say that a path connects two edges e, f if it connects an endpoint of e to an endpoint of f.

Lemma 2.1. *For any branch $e \in T$ and chord $c \in E \backslash T$, we have $c \in \delta_T^e$ if and only if $\bar{P}_T(e, c) = P_T^*(e, c)$.*

Proof. First assume that $c \in \delta_T^e$. Denoting by u_c, v_c the endpoints of c and by u_e, v_e those of e, we can assume w.l.o.g. that u_c, v_c, u_e, v_e are labeled so that $u_c, u_e \in S_T^e$ and $v_c, v_e \in \bar{S}_T^e$. Since there is a unique shortest path q in T connecting u_c to v_c with $u_c \in S_T^e$, $v_c \in \bar{S}_T^e$, then $e \in p$. Thus, there are unique shortest sub-paths q_1, q_2 of q such that $q_1 = \langle u_e, u_c\rangle$, $q_2 = \langle v_e, v_c\rangle$ and $q_1 \subseteq$

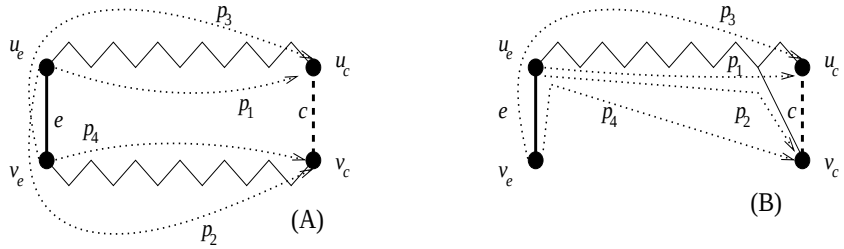

Fig. 2. (A) If c is in the fundamental cut induced by e, $\bar{P}_T(e,c) = P^*_T(e,c) = \{p_1, p_4\}$. Otherwise, up to symmetries, we have the situation depicted in (B) where $\bar{P}_T(e,c) \neq P^*_T(e,c)$.

S^e_T, $q_2 \subseteq \bar{S}^e_T$. Hence $P^*_T(e,c) = \{q_1, q_2\} = \bar{P}_T(e,c)$. Conversely, let $P^*_T(e,c) = \{q_1, q_2\}$, and assume that $e \notin q_1, q_2$. Since either $q_1 \subseteq S^e_T$ and $q_2 \subseteq \bar{S}^e_T$ or vice versa, the endpoints of c are separated by the cut δ^e_T, i.e., $c \in \delta^e_T$. □

Let T be a spanning tree for $G = (V, E)$ and $\pi = (e, f)$ an edge swap with $e \in T, f \in \delta^e_T$ and $f \neq e$. First we note that the cut in G induced by e of T is the same as the cut induced by f of πT.

Proposition 2.2. $\pi(\delta^e_T) = \delta^f_{\pi T}$.

Proof. Since $f \in \delta^e_T$, swapping e with f does not modify the partitions that induce the cuts, i.e., $S^e_T = S^f_{\pi T}$. □

Second, we show that the cuts that do not contain f are not affected by π.

Proposition 2.3. *For each $h \in T$ such that $h \neq e$, and $f \notin \delta^h_T$, we have $\pi(\delta^h_T) = \delta^h_T$.*

Proof. Let $g \in \delta^h_T$. By Lemma 2.1, the shortest paths p^T_1, p^T_2 from the endpoints of h to the endpoints of g do not contain h. We shall consider three possibilities. (1) In the case where e and f do not belong either to p^T_1 or p^T_2 we obtain trivially that $\bar{P}_{\pi T}(e,c) = \bar{P}_T(e,c) = P^*_T(e,c) = P^*_{\pi T}(e,c)$ and hence the result. (2) Assume now that $e \in p^T_1$, and that both e, f are in S^h_T. The permutation π changes p^T_1 so that $f \in p^{\pi T}_1$, whilst $p^{\pi T}_2 = p^T_2$. Now $p^{\pi T}_1$ is shortest because it is the unique path in πT connecting the endpoints of p^T_1, and since $h \notin p^{\pi T}_1, p^{\pi T}_2$ because π does not affect h, we obtain $\bar{P}_{\pi T}(e,c) = P^*_{\pi T}(e,c)$. (3) Suppose that $e \in p^T_1$, $e \in S^h_T$ and $f \in \bar{S}^h_T$. Since $f \in \delta^e_T$, by Lemma 2.1 there are shortest paths q^T_1, q^T_2 connecting the endpoints of e and f such that $q^T_1 \subseteq S^e_T$, $q^T_2 \subseteq \bar{S}^e_T$. Since $e \in S^h_T$, $f \in \bar{S}^h_T$ and T is tree, there is an i in $\{1, 2\}$ such that $h \in q^T_i$ (say, w.l.o.g. that $i = 1$); let $q^T_1 = r^T_1 \cup \{h\} \cup r^T_2$, where $r^T_1 \subseteq S^h_T$ connects h and e, and $r^T_2 \subseteq \bar{S}^h_T$ connects h and f. Let $q^T = r^T_1 \cup \{e\} \cup q^T_2$, then q^T is the unique path in S^h_T connecting h and f. Since r^T_2 connects h and f in $\bar{S}^h_T$, we must conclude that $f \in \delta^h_T$, which is a contradiction. □

Third, we prove that any cut containing f is mapped by the edge swap $\pi = (e, f)$ to its symmetric difference with the cut induced by e of T.

Theorem 2.4. *For each $h \in T$ such that $h \neq e$ and $f \in \delta_T^h$, we have $\pi(\delta_T^h) = \delta_T^h \triangle \delta_T^e$.*

Due to its length, the proof is given in the Appendix.

Edge swap effect on the cycles In order to compute Δ_π efficiently, we have to determine how an edge swap $\pi = (e, f)$ affects the FCB corresponding to the tree T. For each chord h of G with respect to T, let γ_T^h be the unique fundamental cycle induced by h.

Fact *If $h \notin \delta_T^e$, then γ_T^h is unchanged by π.*

The next result characterizes the way π acts on the cycles that are changed by the edge swap π.

Theorem 2.5. *If $h \in \delta_T^e$, then $\gamma_{\pi T}^h = \pi(\gamma_T^h) = \gamma_T^h \triangle \gamma_T^f$, where γ_T^f is the fundamental cycle in T corresponding to the chord f.*

Proof. We need the two following claims.

Claim 1. For all $h \in \delta_T^e$ such that $h \neq e$, $\gamma_T^h \cap \delta_T^e = \{e, h\}$.
Proof. Since γ_T^h is the simple cycle consisting of h and the unique path in T connecting the endpoints of h through e, the only edges both in the cycle and in the cut of e are e and h.

Claim 2. For all pairs of chords $g, h \in \delta_T^e$ such that $g \neq h$ there exists a unique simple cycle $\gamma \subseteq G$ such that $g \in \gamma$, $h \in \gamma$, and $\gamma \backslash \{g, h\} \subseteq T$.
Proof. Let $g = (g_1, g_2)$, $h = (h_1, h_2)$ and assume w.l.o.g. g_1, h_1, g_2, h_2 are labeled so that $g_1, h_1 \in S_T^e$ and $g_2, h_2 \in \bar{S}_T^e$. Since there exist unique paths $p \subseteq T$ connecting g_1, h_1 and $q \subseteq T$ connecting g_2, h_2, the edge subset $\gamma = \{g, h\} \cup p \cup q$ is a cycle with the required properties. Assume now that there is another cycle γ' with the required properties. Then γ' defines paths p', q' connecting respectively g_1, h_1 and g_2, h_2 in T. Since T is a spanning tree, $p = p'$ and $q = q'$; thus $\gamma' = \gamma$.

Consider the cycle $\gamma = \gamma_T^h \triangle \gamma_T^f$. By definition, $e \in \gamma_T^h$, $e \in \gamma_T^f$, $h \in \gamma_T^h$, $f \in \gamma_T^f$. Since $h, f \in \delta_T^e$, by Claim 1 $h \notin \gamma_T^f$ and $f \notin \gamma_T^h$. Thus $h \in \gamma$, $f \in \gamma$, $e \notin \gamma$. Consider now $\pi(\gamma_T^h)$. Since $e \in \gamma_T^h$ and $\pi = (e, f)$, $f \in \pi(\gamma_T^h)$. Furthermore, since π fixes h, $h \in \pi(\gamma_T^h)$. Hence, by Claim 2, we have that $\pi(\gamma_T^h) = \gamma = \gamma_T^h \triangle \gamma_T^f$. □

2.4 Computational complexity

We first evaluate the complexity of applying an edge swap to a given spanning tree and of computing the fundamental cut and cycle structures in a basic implementation. Computing the cost of a FCB given the associated spanning tree T is $O(mn)$, since there are $m - n + 1$ chords of G relative to T and each one

of the corresponding fundamental cycles contains at most n edges. To select the best edge swap available at any given iteration, one has to evaluate the FCB cost for all the swaps involving one of the $n-1$ branches $e \in T$ and one of the (at most $m-n+1$) chords $f \in \delta_T^e$. Since computing a fundamental cut requires $O(m)$, the total complexity for a single edge swap is $O(m^3n^2)$.

In the efficient implementation described in Section 2.3, fundamental cuts and cycles are computed by using symmetric differences of edge sets, which require linear time in the size of the sets. Since there are m fundamental cycles of size at most n, and n fundamental cuts of size at most m, updating the fundamental cut and cycle structures after the application of an edge swap (e, f) requires $O(mn)$. Doing this for each branch of the tree and for each chord in the fundamental cut induced by the branch, leads to an $O(m^2n^2)$ total complexity.

It is worth pointing out that computational experiments show larger speed-ups in the average running times (with respect to the basic implementation) than those suggested by the worst-case analysis.

2.5 Edge sampling

The efficient implementation of the local search algorithm described in Fig. 1 is still computationally intensive, since at each iteration all pairs of tree branches e and chords $f \in \delta_T^e$ must be considered to select the best available edge swap. Ideally, we would like to test the edge swap only for a small subset of pairs e, f while minimizing the chances of missing pairs which yield large cost decreases.

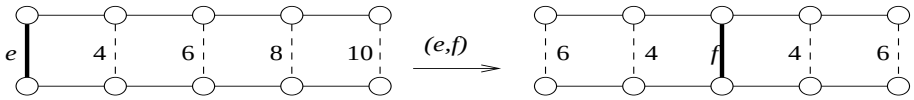

Fig. 3. All edge weights are equal to 1 and the numbers indicated on the chords correspond to the costs of the corresponding fundamental cycles. The cut on the left has a difference between the cheapest and the most expensive cycles of $10-4=6$; after the edge swap the difference amounts to $6-4=2$.

A good strategy is to focus on branches inducing fundamental cuts whose edges define fundamental cycles with "unbalanced" costs, i.e., with a large difference between the cheapest and the most expensive of those fundamental cycles. See Fig. 3 for a simple example. This is formalized in terms of an order $<_b$ on the tree branches. For branches $e_1, e_2 \in T$, we have $e_1 <_b e_2$ if the difference between the maximum and minimum fundamental cycle costs deriving from edges in $\delta_T^{e_1}$ is smaller than that deriving from edges in $\delta_T^{e_2}$. Computational experience suggests that branches that appear to be larger according to the above order tend to be involved in edge swaps leading to largest decreases in the FCB cost.

This strategy can be easily adapted to sampling by ordering the branches of the current spanning tree as above and by testing the candidate edge swaps only

for the first σ fraction of the branches, where $0 < \sigma \leq 1$ is an arbitrary sampling constant.

3 Metaheuristics

To go beyond the scope of local search and try to escape from local minima, we have implemented and tested two well-known metaheuristics: variable neighbourhood search (VNS) [13] and tabu search (TS) [14].

3.1 Variable neighbourhood search

In VNS one attempts to escape from a local minimum $\boldsymbol{x}'$ by choosing another random starting point in increasingly larger neighbourhoods of $\boldsymbol{x}'$. If the cost of the local minimum $\boldsymbol{x}''$ obtained by applying the local search from $\boldsymbol{x}'$ is smaller than the cost of $\boldsymbol{x}'$, then $\boldsymbol{x}''$ becomes the new best local minimum and the neighbourhood size is reset to its minimal value. This procedure is repeated until a given termination condition is met.

For the MIN FCB problem, given a locally optimal spanning tree T' (obtained by applying the local search of Fig. 1), we consider a neighbourhood of size p consisting of all those spanning trees T that can be reached from T' by applying p consecutive edge swaps. A random solution in a neighbourhood of size p is then obtained by generating a sequence of p random edge swaps and applying it to T'.

3.2 Tabu search

Our implementation of tabu search includes diversification steps à la VNS (vTS). In order to escape from local minima, an edge swap that worsens the FCB cost is applied to the current solution and inserted in a tabu list. If all possible edge swaps are tabu or a pre-determined number of successive non-improving moves is exceeded, t random edge swaps are applied to the current spanning tree. The number t increases until a pre-determined limit is reached, and is then re-set to 1. The procedure runs until a given termination condition is met.

Other TS variants were tested. In particular, we implemented a "pure" TS (pTS) with no diversification, and a fine-grained TS (fTS) where, instead of forbidding moves (edge swaps), feasible solutions are forbidden by exploiting the fact that spanning trees can be stored in a very compact form. We also implemented a TS variant with the above-mentioned diversification steps where pTS tabu moves and fTS tabu solutions are alternatively considered. Although the results are comparable on most test instances, vTS performs best on average. Computational experiments indicate that diversification is more important than intensification when searching the MIN FCB solution space with our type of edge swaps.

4 Lower bounds

A standard way to derive a lower bound on the cost of the optimal solutions of a combinatorial optimization problem (and thus to estimate heuristics performance) is to solve a linear relaxation of a (mixed) integer programming formulation. Three different integer programming formulations were discussed in [15].

We now describe an improved formulation that uses non-simultaneous flows on arcs to ensure that the cycle basis is fundamental. Consider a biconnected graph $G = (V, E)$ with a non-negative cost w_{ij} assigned to each edge $\{i, j\} \in E$. For each node $v \in V$, $\delta(v)$ denotes the node star of v, i.e., the set of all edges incident to v. Let $G_0 = (V, A)$ be the directed graph associated with G, namely $A = \{(i, j), (j, i) | \{i, j\} \in E\}$. We use two sets of decision variables. For each edge $\{k, l\} \in E$, the variable $x_{ij}^{kl} \geq 0$ represents the flow through arc $(i, j) \in A$ from k to l. Moreover, for each edge $\{i, j\} \in E$, the variable z_{ij} is equal to 1 if edge $\{i, j\}$ is in the spanning tree of G, and equal to 0 otherwise. For each pair of arcs $(i, j) \in A$ and $(j, i) \in A$, we define $w_{ji} = w_{ij}$.

The following MIP formulation of the MIN FCB problem provides much tighter bounds than those considered in [15]:

$$\min \sum_{\{k,l\} \in E} \sum_{(i,j) \in A} w_{ij} x_{ij}^{kl} + \sum_{\{i,j\} \in E} (1 - 2z_{ij}) w_{ij} \tag{1}$$

$$\sum_{j \in \delta(k)} (x_{kj}^{kl} - x_{jk}^{kl}) = 1 \qquad \forall \{k, l\} \in E \tag{2}$$

$$\sum_{j \in \delta(i)} (x_{ij}^{kl} - x_{ji}^{kl}) = 0 \qquad \forall \{k, l\} \in E,\ \forall i \in V \backslash \{k, l\} \tag{3}$$

$$x_{ij}^{kl} \leq z_{ij} \qquad \forall \{k, l\} \in E,\ \forall \{i, j\} \in E \tag{4}$$

$$x_{ji}^{kl} \leq z_{ij} \qquad \forall \{k, l\} \in E,\ \forall \{i, j\} \in E \tag{5}$$

$$\sum_{\{i,j\} \in E} z_{ij} = n - 1 \tag{6}$$

$$x_{ij}^{kl} \geq 0 \qquad \forall \{k, l\} \in E,\ \forall (i, j) \in A$$

$$z_{ij} \in \{0, 1\} \qquad \forall \{i, j\} \in E.$$

For each edge $\{k, l\} \in E$, a path p from k to l is represented by a unit of flow through each arc (i, j) in p. In other words, a unit of flow exists node k and enters node l after going through all other (possible) nodes in p. For each edge $\{k, l\} \in E$, the flow balance constraints (2) and (3) account for a directed path connecting nodes k and l. Note that the flow balance constraint for node l is implied by constraints (2) and (3). Since constraints (4) and (5) require that $z_{ij} = 1$ for every edge $\{i, j\}$ contained in some path (namely with a strictly positive flow), the z variables define a connected subgraph of G. Finally, constraint (6) ensures that the connected subgraph defined by the z variables is a spanning tree. The objective function (1) adds the cost of the path associated to every edge $\{k, l\} \in E$ and the cost of all tree chords, and subtracts from it

the cost of the tree branches (which are counted when considering the path for every edge $\{k, l\}$).

Besides the quality of the linear relaxation bounds, the main shortcoming of this formulation is that it contains a large number of variables and constraints and hence its solution becomes cumbersome for the largest instances.

5 Some computational results

Our edge-swapping local search algorithm and metaheuristics have been implemented in C++ and tested on three types of unweighted and weighted graphs. CPU times refer to a Pentium 4 2.66 GHz processor with 1 GB RAM running Linux.

5.1 Unweighted mesh graphs

One of the most challenging testbeds for the MIN FCB problem is given by the square $n \times n$ mesh graphs with unit costs on the edges. This is due to the large number of symmetries in these graphs, which bring about many different spanning trees with identical associated FCB costs. Uniform cost square mesh graphs have n^2 nodes and $2n(n-1)$ edges. Table 1 reports the FCB costs and corresponding CPU times of the solutions found with: the local search algorithm (LS) of Fig. 1, the variant with edge sampling (Section 2.5), the NT-heuristic cited in [10], the VNS and tabu search versions described in Section 3. For LS with edge sampling, computational experiments indicate that a sampling constant of 0.1 leads to a good trade-off between solution quality and CPU time for this type of graphs. The lower bounds in the last column correspond to the cost of a non-fundamental minimal cycle basis, that is to four times the number of cycles in a basis: $4(m-n+1)$. For this particular type of graphs, the linear relaxation of the MIP formulation provides exactly the same lower bounds.

5.2 Random Euclidean graphs

To asses the performance of our edge-swapping heuristics on weighted graphs, we have generated simple random biconnected graphs. The nodes are positioned uniformly at random on a 20×20 square centered at the origin. Between each pair of nodes an edge is generated with probability p, with $0 < p < 1$. The cost of an edge is equal to the Euclidean distance between its adjacent nodes. For each n in $\{10, 20, 30, 40, 50\}$ and p in $\{0.2, 0.4, 0.6, 0.8\}$, we have generated a random graph of size n with edge probability p.

Table 2 reports the results obtained with the edge-swapping heuristics (pure local search and metaheuristics) on these random graphs. The first two columns indicate the performance of LS in terms of FCB cost and CPU time. The next two columns correspond to the lower bounds obtained by partially solving the MIP formulation of Section 4. The third and fourth two-column groups indicate the performances of VNS and TS. There was enough available data to ascribe

	LS		LS with edge sampling		NT [10]		VNS	TS	Bound
n	Cost	Time	Cost	Time	Cost	Time	Cost	Cost	Cost
5	72	0:00:00	74	0:00:00	78	0:00:00	72	72	64
10	474	0:00:00	524	0:00:00	518	0:00:00	466*	466*	324
15	1318	0:00:00	1430	0:00:00	1588	0:00:00	1280*	1276*	784
20	2608	0:00:03	3186	0:00:00	3636	0:00:00	2572*	2590*	1444
25	4592	0:00:16	5152	0:00:02	6452	0:00:00	4464*	4430*	2304
30	6956	0:00:47	8488	0:00:03	11638	0:00:00	6900*	6882*	3364
35	10012	0:02:19	11662	0:00:08	16776	0:00:00	9982*	9964*	4624
40	13548	0:06:34	15924	0:00:26	28100	0:00:01	13524*	13534*	6084
45	18100	0:14:22	22602	0:01:00	35744	0:00:01	18100	18100	7744
50	23026	0:31:04	33274	0:01:10	48254	0:00:03	23026	23552	9604

Table 1. Computational results (FCB cost and CPU times (h:mm:ss)) for $n \times n$ mesh graphs having unit edge costs, obtained with different heuristics. The VNS and TS metaheuristics were run for 10 minutes (after finding the first local optimum). Values marked with * denote an improved value with respect to LS. Lower bounds on the optimal value are reported in the last column.

some statistical significance to the average percentage gap between heuristic and lower bounding values (8.19%), and its reassuringly low standard deviation ($\pm$ 5.15%). The maximum frequency value is also a rather low value (6%). It is worth pointing out that the lower bounds obtained by solving the linear relaxation of the formulation presented in Section 4 are generally much tighter than those derived from the formulations considered in [15].

5.3 Weighted graphs from periodic timetabling

An interesting application of MIN FCB arises in periodic timetabling for transportation systems. To design the timetables of the Berlin underground, Liebchen and Möhring [16] consider the mathematical programming model based on the Periodic Event Scheduling Problem (PESP) [17] and the associated graph G in which nodes correspond to events. Since the number of integer variables in the model can be minimized by identifying an FCB of G and the number of discrete values that each integer variable can take is proportional to the total FCB cost, good models for the PESP problem can be obtained by looking for minimum fundamental cycle bases of the corresponding graph G.

Due to the way the edge costs are determined, the MIN FCB instances arising from this application have a high degree of symmetry. Such instances are difficult because, at any given heuristic iteration, a very large number of edge swaps may lead to FCBs with the same cost. Notice that this is generally not the case for weighted graphs with uniformly distributed edge costs. The results reported in Table 3 for instance `timtab2`, which is available from MIPLIB (`http://miplib.zib.de`) and contains 88 nodes and 316 edges, are promising. According to practical modeling requirements, certain edges are mandatory and must belong to the spanning tree associated to the MIN FCB solution. The

	$p = 0.2$							
n	LS	CPU time	Bound	CPU time	VNS	CPU time	TS	CPU time
10	216.698†	0	216.698†	0	216.698†	0	216.698†	0
20	1052.38†	0	1052.38†	0:56	1052.38†	0	1052.38†	0
30	3315.89	0	2750.92	0:28	3111.71*	0:14	3315.89	0
40	4634.04	0	4065.187	16:58	4504.84*	0:22	4633.45*	0
50	7007.34	0:01	6448.711	2:38:51	6991.53*	1:11	7007.34	0:02
	$p = 0.4$							
n	LS	CPU time	Bound	CPU time	VNS	CPU time	TS	CPU time
10	472.599	0	459.305†	0:02	459.305†	0	472.599	0
20	2021.82	0	1894.747	0:08	2021.37*	0:04	2021.37*	0
30	4467.13	0	4265.6	22:56	4455.2*	0:29	4455.2*	0:01
40	7685.97	0:01	-	-	7648*	1:46	7684.53*	0:02
50	11096.8	0:05	-	-	11022.8*	9:32	11073.4*	0:12
	$p = 0.6$							
n	LS	CPU time	Bound	CPU time	VNS	CPU time	TS	CPU time
10	581.525	0	547.406†	0:08	547.406†	0	547.406†	0
20	2776.22	0	2627.558	0:59	2756.6*	0:08	2756.6*	0
30	7031.2	0	6445.83	39:32	6979.15*	1:13	7031.2	0:03
40	11686.0	0:02	-	-	11513*	6:40	11683.4*	0:04
50	19387.3	0:10	-	-	19174.1*	7:06	19174.1*	1:06
	$p = 0.8$							
n	LS	CPU time	Bound	CPU time	VNS	CPU time	TS	CPU time
10	992.866	0	775.838†	0:26	775.838†	0	775.838†	0
20	3478.11	0	3164.9	2:31	3383.45*	0:13	3383.45*	0:02
30	8971.78	0:01	7823.848	1:43:05	8384.32*	2:42	8930.17*	0:02
40	14946.4	0:07	-	-	14870.7*	5:30	14902.2*	0:16
50	25349.9	0:12	-	-	25061.2*	31:55	25245.5*	0:53

Table 2. Computational results (FCB costs, CPU times (mm:ss), lower bounds) for Euclidean random graphs. Lower bound values marked with † denote an optimal solution (MIP solved to optimality). FCB costs are marked with † when the metaheuristic improved on the value found by LS. Missing values are due to excessive CPU timings.

above-mentioned instance contains 80 mandatory edges out of 87 tree branches, and most of the these 80 fixed edges have very high costs. As shown in Table 3 (instance `liebchen-fixed`), this additional condition obviously leads to FCBs with substantially larger costs.

6 Concluding remarks

We described and investigated new heuristics, based on edge swaps, for tackling the Min FCB problem. Compared to existing tree-growing procedures, our local search algorithm and simple implementation of the VNS and Tabu search metaheuristics look very promising, even though computationally more intensive. We established structural results that allow an efficient implementation of

	Local search		NT [10]	VNS		TS		Lower bound
Instance	FCB cost	Time	FCB cost	FCB cost	Time	FCB cost	Time	FCB cost
`liebchen`	40520	0.7s	50265	39801*	30s	39841*	30s	31220.534
`liebchen-fixed`	46072	0.13s	-	-	-	46002*	30s	39907.96

Table 3. Computational results for Liebchen's instance. Missing values are due to a missing implementation of the corresponding algorithm which deals with mandatory spanning tree edges. Values marked with * correspond to an improvement with respect to the LS solution.

the proposed edge swaps. We also presented a new MIP formulation whose linear relaxation provides tighter lower bounds than known formulations on several classes of graphs.

References

1. Sysło, M.: On some problems related to fundamental cycle sets of a graph. In Chartrand, R., ed.: Theory of Applications of Graphs, New York, Wiley (1981) 577–588
2. Kirchhoff, G.: Über die auflösung der gleichungen, auf welche man bei der untersuchungen der linearen verteilung galvanisher ströme geführt wird. Poggendorf Annalen Physik **72** (1847) 497–508
3. Sysło, M.: On cycle bases of a graph. Networks **9** (1979) 123–132
4. Sysło, M.: On the fundamental cycle set graph. IEEE Transactions on Circuits and Systems **29** (1982) 136–138
5. Horton, J.: A polynomial-time algorithm to find the shortest cycle basis of a graph. SIAM Journal of Computing **16** (1987) 358–366
6. Amaldi, E., Rizzi, R.: Personal communication. (2003)
7. Deo, N., Prabhu, G., Krishnamoorthy, M.: Algorithms for generating fundamental cycles in a graph. ACM Transactions on Mathematical Software **8** (1982) 26–42
8. Galbiati, G., Amaldi, E.: On the approximability of the minimum fundamental cycle basis problem. Workshop on Approximation and Online Algorithms (ALGOWAOA03), Budapest (in press in the Lecture Notes in Computer Science series) (2003)
9. Brambilla, A., Premoli, A.: Rigorous event-driven (red) analysis of large-scale nonlinear rc circuits. IEEE Transactions on Circuits and Systems–I: Fundamental Theory and Applications **48** (2001) 938–946
10. Deo, N., Kumar, N., Parsons, J.: Minimum-length fundamental-cycle set problem: New heuristics and an empirical investigation. Congressus Numerantium **107** (1995) 141–154
11. Paton, K.: An algorithm for finding a fundamental set of cycles of a graph. Communications of the ACM **12** (1969) 514–518
12. Shioura, A., Tamura, A., Uno, T.: An optimal algorithm for scanning all spanning trees of undirected graphs. SIAM Journal of Computing **26** (1997) 678–692
13. Hansen, P., Mladenović, N.: Variable neighbourhood search. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics, Dordrecht, Kluwer (2003)
14. Hertz, A., Taillard, E., de Werra, D.: Tabu search. In Aarts, E., Lenstra, J., eds.: Local Search in Combinatorial Optimization, Chichester, Wiley (1997) 121–136

15. Liberti, L., Amaldi, E., Maculan, N., Maffioli, F.: Mathematical models and a constructive heuristic for finding minimum fundamental cycle bases. Submitted to Yugoslav Journal of Operations Research (2003)
16. Liebchen, C., Möhring, R.H.: A case study in periodic timetabling. In Wagner, D., ed.: Electronic Notes in Theoretical Computer Science. Volume 66., Elsevier (2002)
17. Serafini, P., Ukovich, W.: A mathematical model for periodic scheduling problems. SIAM Journal of Discrete Mathematics **2** (1989) 550–581

Appendix: proof of Theorem 2.4

To establish that, for each $h \in T$ such that $h \neq e$ and $f \in \delta_T^h$, we have $\pi(\delta_T^h) = \delta_T^h \triangle \delta_T^e$, we proceed in four steps.

We prove that: (1) $g \in \delta_T^h \cap \delta_T^e \Rightarrow g \notin \pi(\delta_T^h)$, (2) $g \notin \delta_T^h \cup \delta_T^e \Rightarrow g \notin \pi(\delta_T^h)$, (3) $g \in \delta_T^h \backslash \delta^e \Rightarrow g \in \pi(\delta_T^h)$, and (4) $g \in \delta^e \backslash \delta_T^h \Rightarrow g \in \pi(\delta_T^h)$.

When there is no ambiguity, δ_T^e is written δ^e.

Claim 1: $g \in \delta^h \cap \delta^e \Rightarrow g \notin \pi(\delta^h)$.

Proof. Since $g \in \delta^h$ there are shortest paths p_1^T, p_2^T connecting g, h and not

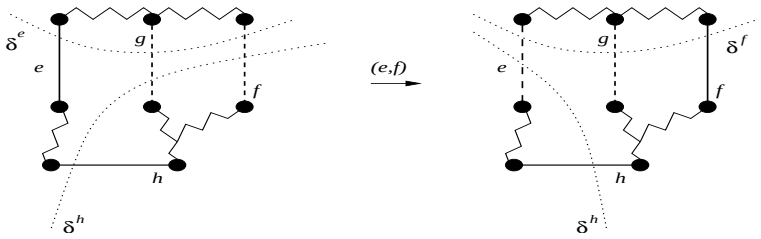

Fig. 4. Claim 1: If $g \in \delta^h \cap \delta^e$ then $g \notin \pi(\delta^h)$.

containing h, such that $p_1^T \subseteq S_T^h$ and $p_2^T \subseteq \bar{S}_T^h$. Since $g \in \delta^e$, either p_1^T or p_2^T must contain e, but not both. Assume w.l.o.g. that $e \in p_1$, $e \notin p_2$ (i.e., $e \in S_T^h$). Thus π sends p_1^T to a path $p_1^{\pi T}$ containing f, whereas $p_2^{\pi T} = p_2^T$. Thus $P_{\pi T}^*(h,g) = \{p_1^{\pi T}, p_2^{\pi T}\}$. Since $f \in \delta_T^h$, there exist shortest paths $q_1^T \subseteq S_T^h$ and $q_2^T \subseteq \bar{S}_T^h$ connecting f, h and not containing h. Because $q_2^T \subseteq \bar{S}_T^h$, $e \notin q_2^T$. In πT, q_2^T can be extended to a path $q^{\pi T} = q_2^T \cup \{f\}$. By Proposition 2.2 $g \in \delta_{\pi T}^f$. Thus in πT there exist paths $r_1^{\pi T}$ in $S_{\pi T}^f = S_T^e$ and $r_2^{\pi T}$ in $\bar{S}_{\pi T}^f = \bar{S}_T^e$ connecting f, g and not containing f. Notice that the path $q^{\pi T} \cup r_1^{\pi T}$ connects the endpoint of h in $\bar{S}_T^h$ and g, and $p_2^{\pi T}$ connects the same endpoint of h with the opposite endpoint of g. Thus $p_1^{\pi T} = \{h\} \cup q^{\pi T} \cup r_1^{\pi T}$, which means that $h \in p_1^{\pi T}$, i.e., $P_{\pi T}^*(h,g) \neq \bar{P}_{\pi T}(h,g)$. By Lemma 2.1, the claim is proved.

Claim 2: $g \notin \delta^h \cup \delta^e \Rightarrow g \notin \pi(\delta^h)$.
Proof. By hypothesis, $g \in S_T^e \cap S_T^h$ or $g \in \bar{S}_T^e \cap \bar{S}_T^h$. Assume the former w.l.o.g. Let

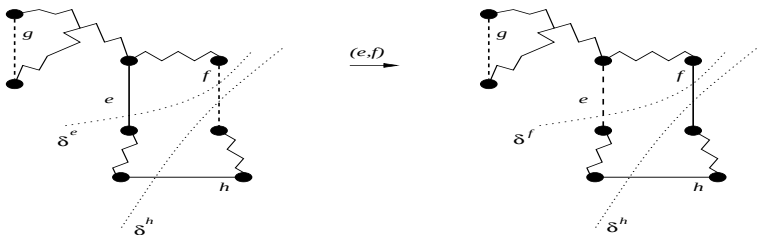

Fig. 5. Claim 2: If $g \notin \delta^h \cup \delta^e$ then $g \notin \pi(\delta^h)$.

p_1^T, p_2^T be unique shortest paths from h to g, and assume $h \in p_1^T$. Thus $p_2^T \subseteq S_T^h$. Since $g \notin \delta^e$, there are unique shortest paths q_1^T, q_2^T from e to g such that e is in one of them, assume $e \in q_1^T$. Since both $h, e \in T$, there is a shortest path r^T between one of the endpoints of h and one of the endpoints of e, while the opposite endpoints are linked by the path $\{h\} \cup r^T \cup \{e\}$. Suppose $e \in p_1^T$. Then since both endpoints of g are reachable from e via q_1^T, q_2^T, and e is reachable from h through r^T, it means that $e \in p_2^T$. Conversely, if $e \notin p_1^T$, then $e \notin p_2^T$. Thus, we consider two cases. If e is not in the paths from h to g, then π fixes those paths, i.e., $h \in p_1^{\pi T}$ and $h \notin p_2^{\pi T}$, that is $g \notin \delta^h$. If e is in the paths from h to g, then both the unique shortest paths $p_1^{\pi T}$ and $p_2^{\pi T}$ connecting h and g in πT contain f. Since $g \notin \delta_{\pi T}^f = \delta_T^e$, there are shortest paths $s_1^{\pi T}, s_2^{\pi T}$ connecting f to g one of which, say $s_1^{\pi T}$, contains f. Moreover, since both $h, f \in T$, there is a shortest path $u^{\pi T}$ connecting one of the endpoints of h to one of the endpoints of f, the other shortest path between the opposite endpoints being $\{h\} \cup u^{\pi T} \cup \{f\}$. Thus, either $p_1^{\pi T} = u^{\pi T} \cup s_1^{\pi T}$ and $p_2^{\pi T} = \{h\} \cup u^{\pi T} \cup \{f\} \cup s_2^{\pi T}$, or $p_1^{\pi T} = u^{\pi T} \cup \{f\} \cup s_2^{\pi T}$ and $p_2^{\pi T} = \{h\} \cup u^{\pi T} \cup s_1^{\pi T}$. Either way, one of the paths contains h. By Lemma 2.1, the claim is proved.

Claim 3: $g \in \delta^h \backslash \delta^e \Rightarrow g \in \pi(\delta^h)$.
Proof. Since $g \in \delta^h$, there are shortest paths $p_1^T \subseteq S_T^h, p_2^T \subseteq \bar{S}_T^h$ connecting h and g, none of which contains h. Assume w.l.o.g. $e \in S_T^h$. Suppose $e \in p_1^T$, say $p_1^T = q^T \cup \{e\} \cup r^T$. Consider $s^T = q^T \cup \{h\} \cup p_2^T$ and r^T. These are a pair of shortest paths connecting e and g such that e does not belong to either; i.e., $g \in \delta_T^e$, which contradicts the hypothesis. Thus $e \notin p_1^T$, i.e., π fixes paths π_1^T, π_2^T; thus $\bar{P}_{\pi T}(h, g) = \bar{P}_T(h, g) = P_T^*(h, g) = P_{\pi T}^*(h, g)$, which proves the claim.

Claim 4: $g \in \delta^e \backslash \delta^h \Rightarrow g \in \pi(\delta^h)$.
Proof. First consider the case where $e, g \in S_T^h$. Since $g \notin \delta_T^h$, the shortest paths p_1^T, p_2^T connecting h, g are such that one of them contains h, say $h \in p_1^T$, whilst $p_2^T \subseteq S_T^h$. Since $g \in \delta^e$, there are shortest paths q_1^T, q_2^T entirely in S_T^h, connecting e, g such that neither contains e. Since both $e, h \in T$ there is a shortest path

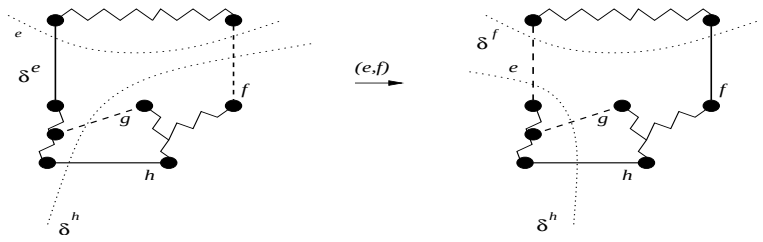

Fig. 6. Claim 3: If $g \in \delta^h$ and $g \notin \delta^e$, then $g \in \pi(\delta^h)$.

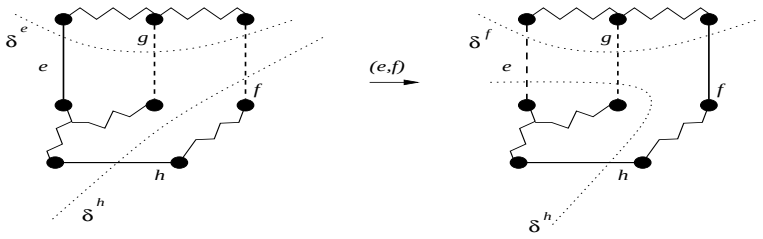

Fig. 7. Claim 4: If $g \notin \delta^h$ and $g \in \delta^e$, then $g \in \pi(\delta^h)$.

$r^T \subseteq S^h_T$ connecting an endpoint of h to an endpoint of e, the opposite endpoints being joined by $\{h\} \cup r^T \cup \{e\}$. Thus w.l.o.g. $p^T_1 = \{h\} \cup r^T \cup \{e\} \cup q^T_1$. Since the path $r^T \cup q^T_2$ does not include e and connects h, g, $e \notin p^T_2$. Thus $p^{\pi T}_2 = p^T_2$ and $h \notin p^{\pi T}_2$. On the other hand π sends p^T_1 to a unique shortest path $p^{\pi T}_1$ connecting h, g that includes f. Since $f \in \delta^h_T$, there are shortest paths $s^T_1 \subseteq S^h_T, s^T_2 \subseteq \bar{S}^h_T$ that do not include h. Since $p^T_2 \subseteq S^h_T$, s^T_1 may only touch the same endpoint of h as p^T_2. Thus the endpoint of h touched by p^T_1 also originates s^T_2. Since $s^T_2 \subseteq \bar{S}^h_T$, $h \notin s^T_2$. Since $p^{\pi T}_1$ joins h, g, contains f and is shortest, $p^{\pi T}_1 = s^T_1 \cup \{f\} \cup u$, where $u^{\pi T}$ is a shortest path from f to g (which exists because by Proposition 2.2 $g \in \delta^f_{\pi T}$), which shows that $h \notin p^{\pi T}_1$. Thus by Lemma 2.1, $g \in \pi(\delta^h)$. The second possible case is that $e \in S^h_T, g \in \bar{S}^h_T$. Since $g \in \delta^e_T$ there are shortest paths p^T_1, p^T_2 connecting e, g such that neither includes e. Assume w.l.o.g. $h \in S^e_T$. Since e, g are partitioned by δ^h_T, exactly one of p^T_1, p^T_2 includes h (say $h \in p^T_1$, which implies $p^T_1 \subseteq S^h_T$). Let q^T_1 be the sub-path of p^T_1 joining h and g and not including h, and let r^T be the sub-path of p^T_1 joining h and e and not including h. Let $q^T_2 = r^T \cup \{e\} \cup p^T_2$. We have that q_2 is a shortest path joining h, g not including h. Thus $\bar{P}_T(h, g) = \{q^T_1, q^T_2\} = P^*_T(h, g)$, and by Lemma 2.1 $g \in \delta^h_T$, which is a contradiction. □