Edge-swapping algorithms for the minimum fundamental cycle basis problem

Edoardo Amaldi

DEI, Politecnico di Milano, Piazza L. da Vinci 32, 20133 Milano, Italy

Leo Liberti

LIX, École Polytechnique, F-91128 Palaiseau, France

Francesco Maffioli

DEI, Politecnico di Milano, Piazza L. da Vinci 32, 20133 Milano, Italy

Nelson Maculan

COPPE, Universidade Federal do Rio de Janeiro, P.O. Box 68511, 21941-972, Rio de Janeiro, Brazil

Submitted: 30 September 2006

Abstract

We consider the problem of finding a fundamental cycle basis with minimum total cost in an undirected graph. This problem is NP-hard and has several interesting applications. Since fundamental cycle bases correspond to spanning trees, we propose a local search algorithm, a tabu search and variable neighborhood search in which edge swaps are iteratively applied to a current spanning tree. We also present a mixed integer programming formulation of the problem whose linear relaxation yields tighter lower bounds than other known formulations. Computational results obtained with our algorithms are compared with those from the best available constructive heuristic on several types of graphs. [1]

Keywords: cycle basis, fundamental cycle, fundamental cut, edge swap, heuristic, metaheuristic.

[1] This article extends the conference paper [1].
Email addresses: `amaldi@elet.polimi.it` (Edoardo Amaldi), `liberti@lix.polytechnique.fr` (Leo Liberti), `maffioli@elet.polimi.it` (Francesco Maffioli), `maculan@cos.ufrj.br` (Nelson Maculan).

1 Introduction

Let $G = (V, E)$ be a simple undirected graph with n nodes and m edges, weighted by a non-negative cost function $w : E \rightarrow \mathbb{R}^+$; we use the notation $w(F) = \sum_{e \in F} w_e$ for subsets $F \subseteq E$. A *cycle* is a subset γ of E such that every node of V is incident with an even number of edges in γ. 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 $GF(2)$. When the graph G is connected, a set of $\nu = m - n + 1$ cycles is a *cycle basis* if it is a basis in this cycle vector space associated to G. There are special cycle bases that can be derived from the spanning trees of G. Let $T \subseteq E$ denote the edge set of any spanning tree of G; 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 (or, with a slight abuse of notation, the chords of T). Any chord uniquely identifies a cycle consisting of the chord itself and the unique path in T connecting the two nodes of the chord. These ν cycles are called *fundamental cycles* and they form a *Fundamental Cycle Basis* (FCB) of G with respect to T (see below for the definition of weakly FCBs). Since the cycle space of a graph is the direct sum of the cycle spaces of its edge-biconnected components, we assume that G is edge-biconnected, i.e., G contains at least two edge-disjoint paths between any pair of nodes.

Figure 1 shows the difference between a non-fundamental cycle basis and a fundamental one.

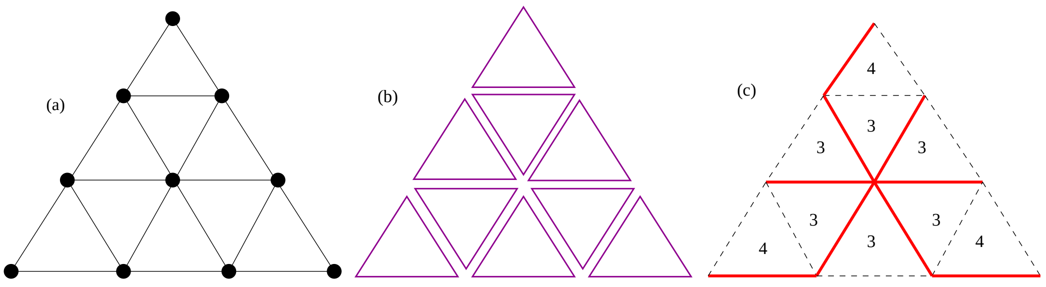

Fig. 1. (a) A triangular grid graph with unit edge costs, (b) a minimum cycle basis with cost 27 and (c) a minimum FCB with cost 30.

In this paper we consider the following problem.

Min FCB: Given an edge-biconnected graph $G = (V, E)$ with non-negative costs assigned to the edges, find a fundamental cycle basis $\{\gamma_1, \ldots, \gamma_\nu\}$ of minimum total cost, i.e., which minimizes $\sum_{i=1}^{\nu} w(\gamma_i)$ where $w(\gamma_i)$ denotes the sum of the costs of all edges in cycle γ_i.

Previous and related work. Cycle bases have been used in the field of electrical networks since the time of Kirchoff [2]. Interest in minimum FCBs arises in a variety of application fields, such as electrical circuit testing [3], generating minimal perfect hash functions (used in compiler design) [4], planning cyclic timetables [5,6], coding of ring compounds [7] and planning complex syntheses in organic chemistry [8]. Many of the early results about the graph-theoretical structure of FCBs are due to a sequence of papers by Sysło [9–14]. In particular, 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 [12]. Moreover, 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. An algorithm for enumerating the FCBs is also presented in [14]. More recently, *weakly fundamental cycle bases*, in which it suffices that there exists an ordering of the cycles $\gamma_1, \dots, \gamma_\nu$ such that $\gamma_j \setminus (\gamma_1 \cup \dots \cup \gamma_{j-1}) \neq \emptyset$ for all j, $2 \leq j \leq \nu$, have also been considered, see for instance [15]. Notice that for a cycle basis to be fundamental it has to satisfy the above condition for all possible orderings of the cycles. In this work we focus on fundamental cycle bases that are fundamental and, for the sake of brevity, do not mention strictness.

Although the problem of finding a general cycle basis of minimum total cost can be solved in polynomial time (see [16] and the recent improvements [17,18]), requiring (strict) fundamentality makes the problem NP-hard [19]. The approximability of MIN FCB is addressed in [20], where the problem is shown to be APX-hard even when restricted to unweighted graphs, a $O(\log^2 n \log\log n)$-approximation algorithm is given for arbitrary graphs, and tighter approximability bounds are obtained for dense graphs, including a polynomial-time approximation scheme for complete graphs. Constructive heuristic methods for solving the MIN FCB problem have been proposed in [4,19,21], and tested on a variety of regular and randomly generated instances.

A well-known problem related to MIN FCB is the minimum routing cost spanning tree problem, see e.g. [22,23]. Given a weighted graph G as in MIN FCB, one looks for a spanning tree T of G which minimizes the sum of the costs of the paths on T between all pairs of nodes. In spite of the similarities, MIN FCB differs substantially from this problem since the sum is taken only over the pairs of nodes corresponding to the chords of T. Note that for weighted graphs, the cost of all chords is also included in the objective function.

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. In Section 3 the same type of edge swaps is adopted within two metaheuristic schemes, namely a variable neighborhood 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

Due to the correspondence between fundamental cycle bases and spanning trees and the fact that the set of all spanning trees can be efficiently explored by iteratively swapping pairs of edges (see e.g. [24,25]), we consider a local search algorithm for MIN FCB based on edge swaps. Starting from the spanning tree associated to an initial FCB, at each iteration we swap a chord e with one of the branches in the fundamental cycle induced by e so that the total FCB cost is decreased, until the cost cannot be further decreased, i.e., a local minimum is found. In practice, it was verified experimentally that allowing swaps that keep the objective function value constant results in a more effective exploration of the solution space.

For each chord $e = \{u, v\}$ of G with respect to the spanning tree T, let $\langle u, v\rangle$ be the unique path connecting u, v in T and let $\gamma_T^e = \langle u, v\rangle \cup \{u, v\}$ the unique fundamental cycle induced by e. For each branch b of T, the removal of b from T induces the partition of the node set V into two subsets S_T^b and $\bar{S}_T^b$. Denote by δ_T^b the *fundamental cut* of G induced by the branch b of T, i.e., $\delta_T^b = \delta(S_T^b) = \{\{u, v\} \in E \mid u \in S_T^b, v \in \bar{S}_T^b\}$. It is easy to verify that each chord e belongs to all fundamental cuts δ_T^b induced by the branches b of the fundamental cycle γ_T^e. Denoting by $\mathcal{C}(T)$ the set of cycles in the FCB associated to T, then $w(\mathcal{C}(T))$ denotes the cost of this FCB.

2.1 Initial solutions

Initial solutions are obtained by applying a very fast "tree-growing" procedure [21], where a spanning tree and its corresponding FCB are generated by adding nodes to the tree according to predefined criteria. The adaptation of Paton's procedure to the MIN FCB problem proceeds as follows. Initially the node set V_T of the tree only contains a root node v_0, and the set W of nodes to be examined is taken as V. Then at each step a node $u \in W \cap V_T$ 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 W. 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 identifying short fundamental cycles early in the process. The performance of this procedure is comparable to other existing tree-growing techniques [19,4]. Initial solutions can also be obtained via the approximation algorithm given in [26] for constructing spanning trees with average stretch $O((\log n)^2 \log\log n)$ in time $O(m \log n + n(\log n)^2)$. The *average stretch* of a tree T is defined as $\text{avestretch}(T) = \frac{1}{|E|} \sum_{\{u,v\}\in E} \frac{w_T(u,v)}{w_{uv}}$, where $w_T(u,v) = \sum_{\{x,y\}\in\langle u,v\rangle} w_{xy}$. The relation between average stretch and FCB cost is given by

$$\begin{aligned}
|E|\text{avestretch}(T) &= \sum_{\{u,v\}\in E} \frac{w_T(u,v)}{w_{uv}} = \sum_{b\in T} 1 + \sum_{\{u,v\}\in E\setminus T} \frac{w_T(u,v)}{w_{uv}} \\
&= |T| + \sum_{e\in E\setminus T} \frac{w(\gamma_T^e) - w_e}{w_e} \\
&= |T| + \sum_{e\in E\setminus T} \frac{w(\gamma_T^e)}{w_e} - (|E| - |T|) \\
&= \sum_{e\in E\setminus T} \frac{w(\gamma_T^e)}{w_e} + 2n - m - 2,
\end{aligned}$$

that is an FCB-like cost where the cost of each cycle is scaled by the weight of the corresponding chord, plus an additive constant. For unweighted graphs where $w_e = 1$ for all $e \in E$, this becomes simply $w(\mathcal{C}(T)) + 2n - m - 2$ and the approximation guarantee for the average stretch then translates directly into an approximation guarantee for the FCB. For the weighted graphs, a related technique yielding an approximation of $O(w(E) \log n \log^2 n)$ is given in [27].

2.2 Selection of a best edge swap

Let $\pi = (e, b)$, where e is a chord and b is a branch of γ_T^e, be the edge swap such that:

$$\pi T = T \cup \{e\} \setminus \{b\}.$$

An edge swap $\pi = (e, b)$, with $b \in \gamma_T^e$, is an improving edge-swap for T if it decreases the total FCB cost, that is if $\Delta_\pi = w(\mathcal{C}(T)) - w(\mathcal{C}(\pi T)) > 0$.

In our edge-swapping local search algorithm, we start from the FCB associated to the spanning tree provided by the constructive procedure of Section 2.1. Then at each iteration we apply to the current spanning tree T the edge swap $\pi = (e, b)$ which yields a maximum decrease in the objective function. The algorithm terminates when no improving edge swap exists for the current spanning tree.

At each iteration of the edge-swapping local search, we consider the spanning tree T of G associated to the current FCB. Two procedures are needed: FCBCost(T), a procedure which computes the cost of the fundamental cycle basis associated with T, and a procedure BestSwap(T), which finds a chord

Algorithm 1 Local search

```
INPUT: an edge-biconnected graph $G = (V, E)$, a spanning tree $T \subset E$ of $G$
OUTPUT: a spanning tree $T'$ of $G$ such that $w(\mathcal{C}(T')) \leq w(\mathcal{C}(T))$
Let $T' = T$
while $w(\mathcal{C}(T')) \leq w(\mathcal{C}(T))$ do
    Let $T = T'$
    Find a best edge swap $\pi$
    Let $T' = \pi T$
end while
```

and a branch of the induced fundamental cycle whose swap yields the largest decrease in the objective function. We show next that an edge swap $\pi = (e, b)$ where e is a chord and b a branch of γ^e_T, acts on a fundamental cycle γ^f_T, where f is another chord of T, by either fixing it (i.e. by leaving it unchanged), or by mapping it to the symmetric difference of γ^f_T and γ^e_T.

The following elementary facts are stated without proof.

(1) For any edge swap $\pi = (e, b)$, where e is a chord and b a branch of γ^e_T, π fixes γ^e_T, and, if $f \neq e$ is another chord of T such that $\gamma^e_T \cap \gamma^f_T = \emptyset$, π also fixes γ^f_T.
(2) For any pair of chords e, f of T, there exists a branch b of T such that $e, f \in \delta^b_T$ if and only if $b \in \gamma^e_T \cap \gamma^f_T$.

By Fact (1), the only case where a swap $\pi = (e, b)$ does *not* fix a fundamental cycle γ^f, for a different chord f of T, is when $\gamma^e_T \cap \gamma^f_T \neq \emptyset$. Figure 2 illustrates the following result.

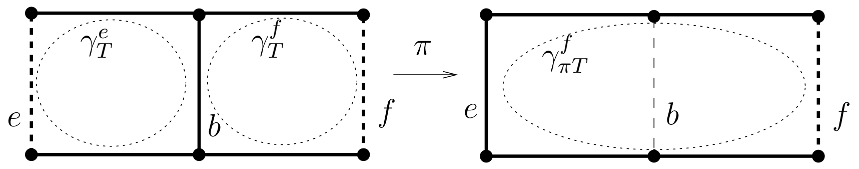

Fig. 2. Illustration of the statement of Theorem 1.

Theorem 1 *Consider an edge swap $\pi = (e, b)$, where e is a chord and b a branch of γ^e_T. For any chord $f \neq e$ of T, if $b \in \gamma^e_T \cap \gamma^f_T$, then once the swap π is performed the fundamental cycle induced by the chord f, denoted by $\gamma^f_{\pi T}$ or $\pi(\gamma^f_T)$, is such that $\gamma^f_{\pi T} = \gamma^f_T \triangle \gamma^e_T$.*

Proof. First, by Fact (2) we have that $e, f \in \delta^b_T$. Then we need the two following claims, which are illustrated in Figure 3.

Claim 1. For all $h \in \delta^b_T$ such that $h \neq b$, $\gamma^h_T \cap \delta^b_T = \{b, h\}$.
Proof. Since γ^h_T is the simple cycle consisting of h and the unique path in T connecting the endpoints of h through b, the only edges that belong both to

the cycle and to the cut of b are b and h.

Claim 2. For all pairs of chords $g, h \in \delta_T^b$ 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. that g_1, h_1, g_2, h_2 are labeled so that $g_1, h_1 \in S_T^b$ and $g_2, h_2 \in \bar{S}_T^b$. 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 $q = q'$, and hence $\gamma' = \gamma$.

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

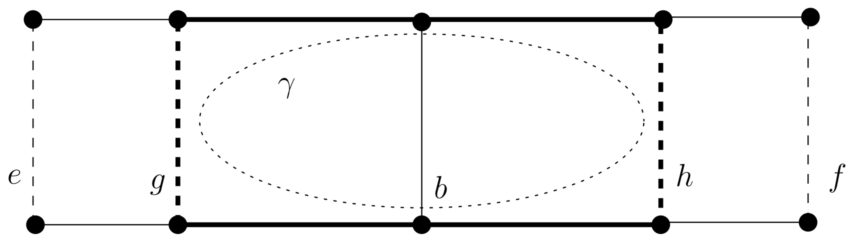

Fig. 3. Illustration of Claims 1 and 2 in the proof of Theorem 1.

From Theorem 1, it follows that the change Δ_π in FCB cost due to an edge swap $\pi = (e, b)$, where e is a chord and b a branch of the current spanning tree T belonging to γ_T^e, only depends on the change in cost of those fundamental cycles γ_T^f (where $f \neq e$ is a chord of T) which contain b.

Corollary 2 *Let $\pi = (e, b)$ be an edge swap with e a chord and b a branch of γ_T^e, and let $F(b) = \{f \in E \setminus T \mid b \in \gamma_T^f\}$. Then*

$$\Delta_\pi = 2 \sum_{f \in F(b)} w(\gamma_T^e \cap \gamma_T^f) - |F(b)| w(\gamma_T^e). \tag{1}$$

Proof. By Theorem 1, the only fundamental cycles that change under the action of π are those that contain b, and can therefore be used to quantify the change Δ_π. We have:

$$\begin{aligned}\Delta_\pi &= \sum_{f\in F(b)} (w(\gamma_T^f) - w(\gamma_{\pi T}^f)) = \\ &= \sum_{f\in F(b)} (w(\gamma_T^f) - w(\gamma_T^e \triangle \gamma_T^f)) = \\ &= \sum_{f\in F(b)} (2w(\gamma_T^e \cap \gamma_T^f) - w(\gamma_T^e))\end{aligned}$$

because $w(\gamma_T^e \triangle \gamma_T^f) = w(\gamma_T^e) + w(\gamma_T^f) - 2w(\gamma_T^e \cap \gamma_T^f)$. The result follows. □

In the rest of this section, we present two different techniques to determine a best edge swap at each iteration. The first one, described in Section 2.3, uses least common ancestors to identify fundamental cycles. The second one, described in Section 2.4, efficiently updates lists of fundamental cuts and cycles by using symmetric differences of edge sets. As we shall see in Section 2.5, although the worst-case complexity estimate favours the first one, the second one turns out to be around 20 times faster in practice.

2.3 Least Common Ancestor-based algorithm

To compute the symmetric difference we need to determine the intersection of two cycles. This can be achieved by computing Least Common Ancestors (LCA, also called Nearest Common Ancestors). Given a spanning tree T of G, a root r, and two vertices $u, v \in V$, the LCA of u, v is the first common vertex on the unique paths on T, $\langle u, r\rangle$ from u to r and $\langle v, r\rangle$ from v to r. There exists a LCA query algorithm which takes amortized $O(\log^2 n)$ time for dynamically updating rooted spanning trees with n vertices [28]. The output of the LCA query algorithm shall be denoted by $\mathrm{LCA}(u, v, T, r)$.

The following elementary lemma is stated without proof.

Lemma 3 *Let $e = \{u, v\}$ be a chord and r any root of T; let p be the unique path in T between u and v. Then p contains a unique least common ancestor x of u, v with respect to the root r of T.*

The procedure FCBCost, which is described in Algorithm 2, cycles over the chords of T and computes the cost of the corresponding fundamental cycles by using LCAs. It relies on the following observation, which is implied by Lemma 3.

Lemma 4 *Let $e = \{u, v\}$ be a chord and r a vertex of T. If $p_1 = \langle u, r\rangle$ and $p_2 = \langle v, r\rangle$ are the unique paths in T between respectively u and r, and v and r, then $\gamma_T^e = p_1 \triangle p_2 \cup \{e\}$.*

Algorithm 2 FCBCost(T). Returns the FCB cost w_T associated to T.

```
Let $w_T = 0$, fix an arbitrary root $r$ of $T$
for $v \in V$ do
    $\mu(v)$ = weight of the unique path in $T$ from $r$ to $v$
end for
for $f = \{u, v\} \in E \backslash T$ do
    $y = \mathrm{LCA}(u, v, T, r)$                                      $O(\log^2 n)$
    $w(\gamma_T^f) = w(f) + (\mu(u) - \mu(y)) + (\mu(v) - \mu(y))$      [by Lemma 4]
    $w_T \leftarrow w_T + w(\gamma_T^f)$
end for
Return $w_T$
```

Due to the loop over f and the LCA computation, the complexity of Algorithm 2 is $O(m \log^2 n)$.

The set $\gamma_T^e \cap \gamma_T^f$ in (1) can be computed by using LCAs.

Proposition 5 *Let $\pi = (e, b)$ be an edge swap with $e = \{u, v\}$ a chord and b a branch of γ_T^e, and let $f = \{t, z\} \neq e$ be another chord of T. Let x, y be the LCAs of t, z with respect to the rootings u, and respectively v, of T. Let p be the unique path in T between x and y. Then $\gamma_T^e \cap \gamma_T^f = p$.*

Proof. Let x' be the LCA of u, v w.r.t. the rooting t, and y' the LCA of u, v w.r.t. the rooting z of T. For v_1, v_2 distinct vertices in V, we indicate by $\langle v_1, v_2 \rangle$ the unique path in T from v_1 to v_2. Then by definition of LCA we have: $\langle x, y \rangle \subseteq T$, $\langle x', y' \rangle \subseteq T$, $\langle x, u \rangle \subseteq T$, $\langle u, x' \rangle \subseteq T$, $\langle y, v \rangle \subseteq T$, $\langle v, y' \rangle \subseteq T$. Suppose $x \neq x'$ or $y \neq y'$. Putting the above paths together, we find that (x, y, v, y', x', u, x) is a sequence of vertices on a path in T. Since x appears as the first and last element, this path is closed, hence T is a spanning tree containing a cycle, which is a contradiction. Hence $x = x'$ and $y = y'$. By Lemma 3, p belongs to both γ_T^e and γ_T^f, as claimed (see Figure 4). $\square$

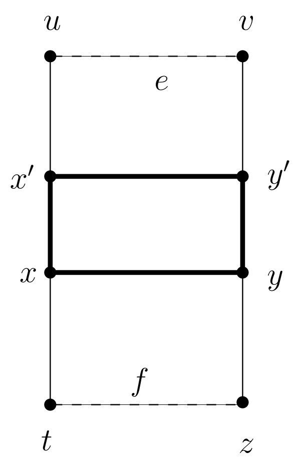

Fig. 4. Illustration of the proof of Proposition 5.

Corollary 2 and Proposition 5 suggest the following method: given an edge swap $\pi = (e, b)$ for a chord e and a branch b of γ_T^e, cycle over all other chords $f \neq e$ such that $b \in \gamma_T^f$ and compute Δ_π. We can then choose the edge swap leading to the best improvement. This is implemented in a slightly more efficient way in Algorithm 3 by assigning a weight ξ to the vertices $u \equiv v_1, \dots, v_k, v_{k+1} \equiv v$ in the unique path in T joining the endpoints of the chord e. For all $i \leq k$, the weight $\sum_{j \leq i} \xi(v_j)$ corresponds to the change in FCB cost when $b = \{v_i, v_{i+1}\}$ is selected for the swapping branch in $\pi = (e, b)$.

The procedure BestSwap is described in Algorithm 3. Its complexity is $O(m^2 n)$, due to the nested loops over e and f and the path computation between x and y.

Algorithm 3 LCA-based implementation of BestSwap(T).

```
Initialize $\Delta^* = 0, \pi^* =$ identity swap
for $e = \{u, v\} \in E \backslash T$ do
  Let $\rho = (v_1, \dots, v_k, v_{k+1})$ be the unique path in $T$ joining $u, v$
  for $v_i \in \rho$ do
    Initialize $\xi(v_i) = 0$
  end for
  for $f = \{t, z\} \in E \backslash (T \cup \{e\})$ do
    $x = \mathrm{LCA}(t, z, T, u)$
    $y = \mathrm{LCA}(t, z, T, v)$
    if $x \neq y$ then
      Let $p$ be the unique path (of weight $w(p)$) in $T$ joining $x, y$      $O(n)$
      $d(e, f) = w(\gamma_T^e) - 2w(p)$                                 [by Corollary 2]
      $\xi(x) \leftarrow \xi(x) + d(e, f)$
      $\xi(y) \leftarrow \xi(y) - d(e, f)$
    end if
  end for
  Let $v_i \in \rho$ be such that $\sum_{j \leq i} \xi(v_j)$ is minimum over $i \leq k$
  Let $b = \{v_i, v_{i+1}\}$, $\pi = (e, b)$, $\Delta = \sum_{j \leq i} \xi(v_j)$
  if $\Delta < \Delta^*$ then
    $\Delta^* = \Delta$
    $\pi^* = \pi$
  end if
end for
Return $\pi^*$ and $\Delta^*$
```

2.4 *Symmetric difference-based algorithm*

This implementation of BestSwap is conceptually simpler than the LCA-based one. We maintain lists of fundamental cuts and cycles associated to the current spanning tree, and use them to compute the cost of each edge swap (see Alg. 4). Since applying an edge swap to a spanning tree may change the fundamental cycle and cut structures considerably, efficient procedures are needed to compute Δ_π and update the fundamental cut and cycle lists. We show that any edge swap $\pi = (e, f)$ applied to a spanning tree T, where $e \in T$ and $f \in \delta^e_T$, changes a cut δ^h_T if and only if f is also in δ^h_T. Furthermore, $\pi(\delta^h_T) = \delta^h_{\pi T}$ is given by the symmetric difference $\delta^h_T \triangle \delta^e_T$. By Theorem 1 a similar statement holds for cycles. This makes it easy to maintain fundamental cut and cycle data structures that can be updated efficiently, by performing symmetric differences of edge sets, when π is applied to T.

Algorithm 4 Symmetric difference-based implementation of BestSwap(T).

```
Initialize $\Delta^* = 0, \pi^* =$ identity swap
For each $e \in T$ compute $\delta^e_T$; for all $f \in E \setminus T$ compute $\gamma^e_T$
**for all** $e \in T, f \in \delta^e_T$ s.t. $f \neq e$ **do**
  Let $\pi = (e, f)$
  Compute $\mathcal{C}(\pi T)$ and hence $\Delta_\pi$
  **if** $\Delta_\pi < \Delta^*$ **then**
    Let $\Delta^* = \Delta_\pi$
    Let $\pi^* = \pi$
  **end if**
**end for**
Return $\pi^*$ and $\Delta^*$
```

For each pair of nodes $u, v \in V$, $\langle u, v\rangle$ denotes 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. 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 $P_T(e, c) = \{p_i(e, c) \mid i = 1, \ldots, 4\}$. 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, or the first one to break ties (see Figure 5). 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.

The following lemma implies that the shortest paths from the endpoints of e to the endpoints of c do not contain e.

Lemma 6 *For any branch $e \in T$ and chord $c \in E \backslash T$, we have $c \in \delta^e_T$ if and only if $\bar{P}_T(e, c) = P^*_T(e, c)$, that is $e \notin P^*_T(e, c)$.*

Proof. First assume that $c \in \delta^e_T$. Denoting by u_c, v_c the endpoints of c and by

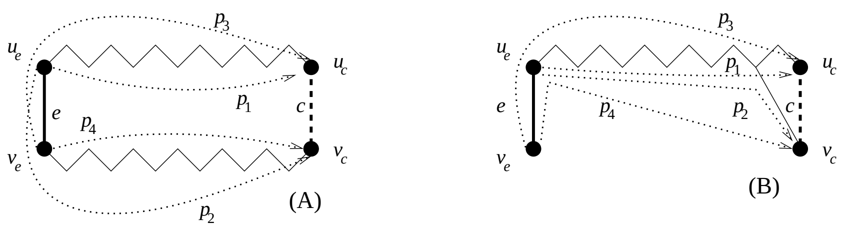

Fig. 5. (A) If c is in the fundamental cut induced by e, $\bar{P}_T(e,c) = P^*_T(e,c) = \{p_1, p_4\}$. Otherwise, up to symmetries, we have the situation depicted in (B) where $\bar{P}_T(e,c) \neq P^*_T(e,c)$.

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 $u_c, u_e \in S^e_T$ and $v_c, v_e \in \bar{S}^e_T$. Since there is a unique path q in T connecting u_c to v_c, then $e \in q$. Thus, there are unique 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 $q_1 \subseteq S^e_T$, $q_2 \subseteq \bar{S}^e_T$ (i.e. neither q_1 nor q_2 contains e). 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 is not in q_1 nor in 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 $\pi = (e, f)$ be an edge swap with $e \in T, f \in \delta^e_T$ and $f \neq e$. Note that $f \in E \setminus T$. First we note that the cut in G induced by e with respect to T is the same as the cut induced by f with respect to πT.

Proposition 7 *For any $\pi = (e, f)$ with $e \in T$ and $f \in \delta^e_T$ such that $f \neq e$, we have $\pi(\delta^e_T) = \delta^f_{\pi T}$.*

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 8 *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 6, 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 $\bar{P}_{\pi T}(f,e) = \bar{P}_T(e,f) = P^*_T(e,f) = P^*_{\pi T}(f,e)$ 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 $e \in p_1^T$, $e \in S_T^h$ and $f \in \bar{S}_T^h$. Since $f \in \delta_T^e$, by Lemma 6 there are shortest paths q_1^T, q_2^T connecting the endpoints of e and f such that $q_1^T \subseteq S_T^e$, $q_2^T \subseteq \bar{S}_T^e$. Since $e \in S_T^h$, $f \in \bar{S}_T^h$ and T is tree, there is an i in $\{1, 2\}$ such that $h \in q_i^T$ (assume w.l.o.g. that $i = 1$), so $q_1^T = r_1^T \cup \{h\} \cup r_2^T$ where $r_1^T \subseteq S_T^h$ connects h and e, and $r_2^T \subseteq \bar{S}_T^h$ connects h and f. If we let $q^T = r_1^T \cup \{e\} \cup q_2^T$, then q^T is the unique path in S_T^h connecting h and f. Since r_2^T connects h and f in $\bar{S}_T^h$, we must conclude that $f \in \delta_T^h$, which is a contradiction. □

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

Theorem 9 *For each $h \in T$ such that $h \neq e$ and $f \in \delta_T^h$, we have $\pi(\delta_T^h) = \delta_T^h \triangle \delta_T^e$.*

The proof is lengthy and hence is given in Appendix A.

2.5 Complexity comparison

It was shown in Section 2.3 that the complexity of the BestSwap procedure based on LCA computation is $O(m^2 n)$.

In the alternative algorithm of Section 2.4, fundamental cuts and cycles are determined and updated by using symmetric differences of edge sets, which require linear time in the size of the sets. Since for any spanning tree T there are $m-n+1$ fundamental cycles with at most n edges, and $n-1$ fundamental cuts with at most m edges, 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 a $O(m^2 n^2)$ worst-case complexity, apparently leading to a worse algorithm.

Running the implementation of both algorithms, however, showed a consistent performance difference in favour of the second algorithm. More precisely, tested on a set of 60 instances of various types (see the table in Appendix B), the LCA-based algorithm was on average around 20 times slower than the symmetric difference-based one (with a standard deviation of around 10); profiling the code suggested that this was essentially due to the algorithm rather than the implementation. An intuitive explanation may be found in the fact that worst-case complexity analysis may not accurately reflect the performance of the second algorithm. Take for instance the update of the fundamental cuts: there is a first loop on the $n-1$ tree branches inducing the cuts, and a second nested loop of $O(m)$ chords in each cut. This eventually leads to the (apparently) very costly $O(m^2n^2)$ term. On average, however, each cut is very far from attaining the worst-case estimation of m edges. Instead, one should consider the total number of edges counted jointly by the two nested loops, which is usually much lower than the worst-case $m(n-1)$.

In support of the above view, the following simple average-case observations can be made for unweighted graphs.

(1) Let $\bar{k}$ be the average cardinality of each cycle in the minimum FCB: then $\bar{k} = \frac{f^*}{m-n+1}$ where f^* is the cost of the minimum FCB.
(2) Each chord belongs on average to the $\bar{k}-1$ cuts induced by its own cycle.
(3) The average length of each cut is therefore $\frac{m(\bar{k}-1)}{n-1}$, and the total number of edges counted jointly by the two nested loops is on average $m(\bar{k}-1)$.
(4) Asymptotically, $m(\bar{k}-1)$ is $O(f^*)$.

It follows that, for unweighted graphs, the cost of the minimum FCB is a reasonable estimate of the complexity cost of each swap. Since the main focus of this work is experimental, in the sequel we only report the computational results for the faster algorithm based on symmetric differences.

Finally, we remark that the computing time taken for a single iteration of the local search heuristic (i.e. an application of the `BestSwap` algorithm) is (asymptotically) comparable with the time taken for constructing the initial solution (see e.g. worst-case complexity bounds in [26]). This suggests that such heuristics are not well-suited for real-time or time-critical applications.

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 neighborhood search (VNS) [29] and tabu search (TS) [30].

3.1 Variable neighborhood search

VNS is a relatively recent metaheuristic which relies on iteratively exploring neighborhoods of growing size to identify better local optima [29]. More precisely, in VNS one attempts to escape from a local minimum x' by choosing another random starting point in increasingly larger neighborhoods of x'. If the cost of the local minimum x'' obtained by applying the local search from x' is smaller than the cost of x', then x'' becomes the new best local minimum and the neighborhood size is reset to its minimal value. This procedure is repeated until a given termination condition is met. VNS was successfully tested on many combinatorial and continuous optimization problems [31].

For the MIN FCB problem, given a locally optimal spanning tree T' provided by the local search algorithm, we consider a neighborhood 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 neighborhood of size p is obtained by generating a sequence of p random edge swaps and applying it to T'. The algorithm is summarized in Figure 6.

For the whole test set, the parameters were set as follows: minimum neighborhood size $k = 2$ ($k = 1$ would simply explore the local search neighborhood again), maximum neighborhood size $K = 5$, number of trials in each neighborhood $F = 1$.

```
(1) Let $T^*$ be a locally minimal spanning tree found with LS, and
    $\varphi^*$ its FCB cost
(2) Initialize minimum and maximum neighborhood sizes $k, K$ and
    number $F$ of local searches in each neighborhood
(3) Set $p = k$ and $T = T^*$
(4) **for** all $i \leq F$ **do**
        **for** all $j \leq p$ **do**
            Choose random edge swap $\pi = (b, c)$
            $T = \pi T$
        Let $T' = \text{LS}(T)$ and $\varphi'$ be the FCB cost of $T'$
        **if** $\varphi' < \varphi^*$ **then** let $T^* = T'$, $\varphi^* = \varphi'$ and go to step (3)
(5) $p \leftarrow p + 1$
(6) **if** $p > K$ **then** terminate **else** go to step (4)
```

Fig. 6. The VNS algorithm for the MIN FCB problem. In step 4, LS(T) denotes the spanning tree provided by the local search algorithm.

3.2 Tabu search

Our implementation of TS [30] includes diversification steps “à la VNS” (vTS). In order to escape from local minima, a best available edge swap is applied to the current solution (even if it worsens the FCB cost) and is inserted in a tabu list. If all possible edge swaps are tabu or a pre-determined number S of successive non-improving moves is exceeded, t random edge swaps are applied to the current spanning tree; this move is called a *random shaker* of size t. The number t increases until a limit K is reached, and is then re-set to a starting size k. The procedure runs until a given termination condition is met. The algorithm is summarized in Figure 7.

For the whole test set, we took: maximum tabu list size $\Lambda = 10$, minimum and maximum random shaker sizes $k = 2, K = 30$, non-improving moves limit $S = 20$.

```
 (1) Let $T^*$ be a locally minimal spanning tree found with LS, and
     $\varphi^*$ its FCB cost
 (2) Initialize tabu list length $\Lambda$, minimum and maximum random
     shaker size $k, K$, non-improving moves limit $S$
 (3) Initialize FIFO tabu list $L = \emptyset$, cost-increasing edge swap
     counter $s = 0$, random shaker size $p = k$
 (4) Choose a cost-increasing edge swap $\pi \neq 1$ s.t. cost increase is
     minimal and $\pi \notin L$; let $s = s + 1$
 (5) **if** such a $\pi$ does not exist, or $s > S$ **then** let $\pi$ be a *random
     shaker* of size $p$; set $p = p + 1$ and $s = 0$
 (6) **if** $p > K$ **then** terminate
 (7) Let $T = \pi T^*$ and $L = L \cup \{\pi^{-1}\}$
 (8) **if** $|L| > \Lambda$ **then** remove oldest edge swap from $L$
 (9) Let $T' = \mathrm{LS}(T)$ and $\varphi'$ be the FCB cost of $T'$
(10) **if** $\varphi' < \varphi^*$ **then** let $\varphi^* = \varphi'$, $T^* = T'$, $s = 0$, $p = k$
(11) Go to step (4)
```

Fig. 7. The TS algorithm for the Min FCB problem. In step 7, π^{-1} indicates the inverse edge swap; in step 4, $\mathrm{LS}(T)$ denotes the spanning tree provided by the local search algorithm.

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. Variable length tabu lists were also tried, but the results were similar to those obtained with fixed-length lists.

4 Lower bounds

Lower bounds on the cost of the optimal solutions are useful to assess the performance of heuristics. The linear relaxations of three different mixed integer programming formulations were discussed in [32]. In this section, we describe an improved formulation that uses non-simultaneous flows on arcs to ensure that the cycle basis is fundamental. Consider an edge-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 for the Min FCB problem provides much tighter bounds than those considered in [32]:

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

$$\text{s.t.} \quad \sum_{j \in \delta(k)} (x_{kj}^{kl} - x_{jk}^{kl}) = 1 \qquad \forall \{k, l\} \in E \tag{3}$$

$$\sum_{j \in \delta(i)} (x_{ij}^{kl} - x_{ji}^{kl}) = 0 \qquad \forall \{k, l\} \in E,\ \forall i \in V \backslash \{k, l\} \tag{4}$$

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

$$x_{ji}^{kl} \leq z_{ij} \qquad \forall \{k, l\} \in E,\ \forall \{i, j\} \in E \tag{6}$$

$$\sum_{\{i,j\} \in E} z_{ij} = n - 1 \tag{7}$$

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

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

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 exits 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 (3) and (4) account for a directed path connecting nodes k and l. Note that the flow balance constraint for node l is implied by constraints (3) and (4). Since constraints (5) and (6) 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 (7) ensures that the connected subgraph defined by the z variables is a spanning tree. The objective function (2) 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 $\{k, l\}$). The main shortcoming of this formulation is that it contains a large number of variables and constraints and hence it is hard to solve to optimality even for medium-sized instances.

The bounds reported in the next section were obtained by letting CPLEX

MIP solver (v. 9.1, [33]) process the root node of the branch-and-bound tree, with automatic cut generation enabled. For small instances, the solver was run to completion.

4.1 Efforts towards tighter lower bounds

A substantial effort was undertaken in order to obtain tighter lower bounds but unfortunately to no avail. A number of different ILP formulations and corresponding Lagrangian relaxations of some of the constraints were considered. Almost invariably, one of the two resulting subproblems was solvable in polynomial time and the other was computationally as difficult as (or more than) solving the MIN FCB problem itself. When two easy subproblems were obtained, the resulting bounds were weaker or the same as those obtained by solving the linear relaxation of the formulation given above.

One of the most (apparently) promising Lagrangian relaxations we considered is the following "partial" ILP formulation:

$$\min z = \sum_{i\in E}\sum_{k=1}^{\nu} w_i x_{ik} \tag{8}$$

$$s.t. \quad \sum_{k=1}^{\nu} x_{ik} \geq 1 \qquad \forall i \in E \tag{9}$$

$$\{i \mid y_i = 1\} \qquad \text{is a spanning tree of G} \tag{10}$$

$$\sum_{k=1}^{\nu} x_{ik} \leq 1 + \nu y_i \, \forall i \in E \tag{11}$$

$$\sum_{i\in\delta_j} x_{ik} = 2 s_{jk} \quad \forall j \in V, k \leq \nu \tag{12}$$

$$\sum_{i\in E} x_{ik} \geq 3 \qquad \forall k \leq \nu. \tag{13}$$

where:

$$x_{ik} = 1 \qquad \text{if edge } i \in \text{cycle } k \qquad \text{and 0 otherwise} \tag{14}$$

$$y_i = 1 \quad \text{if edge } i \in \text{spanning tree } T \quad \text{and 0 otherwise} \tag{15}$$

$$s_{jk} = 1 \qquad \text{if vertex } j \in \text{cycle } k \qquad \text{and 0 otherwise} \tag{16}$$

and δ_j is the star of vertex j (there are $\Theta(mn)$ variables in (8)-(13), the same as in (2)-(7)). The only "complicating constraints" of this formulation, involving both the x and y variables, are those in Equation (11). By relaxing these constraints we can decompose the MIN FCB problem into two separate (simpler) problems. We relax the $|E|$ constraints (11) using Lagrange multipliers

λ_i for all $i \in E$ to obtain:

$$\begin{aligned}
z(x,y,\lambda) &= \sum_{i\in E} w_i \sum_{k=1}^{\nu} x_{ik} + \sum_{i\in E} \lambda_i \left(\sum_{k=1}^{\nu} x_{ik} - 1 - \nu y_i \right) = \\
&= \sum_{i\in E} (w_i + \lambda_i) \sum_{k=1}^{\nu} x_{ik} - \nu \sum_{i\in E} \lambda_i y_i - \sum_{i\in E} \lambda_i = \\
&= z_1 - z_2 - z_3
\end{aligned}$$

where

$$\begin{aligned}
z_1(x,\lambda) &= \sum_{i\in E} (w_i + \lambda_i) \sum_{k=1}^{\nu} x_{ik} \\
z_2(y,\lambda) &= \nu \sum_{i\in E} \lambda_i y_i \\
z_3(\lambda) &= \sum_{i\in E} \lambda_i .
\end{aligned}$$

This separates the problem into two independent subproblems: Max Spanning Tree (y variables, objective function z_2) and a special form of Min Eulerian Cycle (x variables, objective function z_1). The Max Spanning Tree problem is formulated as $\min_y \{z_2(y,\lambda) \mid (10)\}$ and solved combinatorially with a greedy method. The special Min Eulerian Cycle problem is formulated as $\min_{x,s} \{z_1(x,s,\lambda) \mid (9) \wedge (12) \wedge (13)\}$ and also solved combinatorially (we omit the details). The Lagrangian dual problem is therefore

$$\max_{\lambda\geq 0} (\min_{x,s} \{z_1(x,\lambda) \mid (9) \wedge (12) \wedge (13)\} - \min_y \{z_2(y,\lambda) \mid (10)\} - z_3(\lambda)),$$

which we tried to solve using a subgradient method [34], but to no avail, getting a bound that was identical to the relaxation of the Min FCB formulation (2)-(7).

Since the constraints in the Min FCB formulation are the same as in the minimum routing cost tree formulation, we tried to tighten the bound by adding valid cuts used in [23]. In particular, we considered the cardinality cuts stating that each subtree must contain a number of edges equal to the number of nodes minus one. However, the solution of the linear relaxation of the above formulation, albeit fractional, already satisfied all these cuts on all tested instances.

For large instances, even solving the linear relaxation of our formulation is a very challenging task. In those cases, a lower bound was obtained by finding the minimum (non-fundamental) cycle basis of the graph using an improved version of Horton's algorithm [16,18,35]. On average, this is a very weak lower bound, whose difference with respect to the optimal FCB cost grows with the size of the instance. In the case of mesh graphs, we established experimentally that the gap between this lower bound and the heuristic solution increases about linearly with number of vertices in the graph (in the case of unweighted graphs, Thm. 6.6 of [38] gives an asymptotic $\frac{1}{1024}\ln(n)$ lower bound for square grids). Further information about bounds and asymptotic sizes for mesh grids can be found in [36] and the references cited therein.

5 Computational results

Our edge-swapping local search algorithm and metaheuristics were implemented in C++ and tested on various types of unweighted and weighted graphs. CPU times refer to a Pentium 4 2.66 GHz processor with 1 GB RAM running Linux. We considered the following classes of instances:

- star graphs and rectangular mesh graphs with known global optima,
- square mesh graphs with unit costs on the edges,
- Euclidean graphs generated randomly, with weights proportional to the edge lengths,
- two instances taken from a real-life periodic timetabling problem,
- one instance taken from a real-life electrical circuit testing application,
- 2D and 3D toroidal graphs with unit costs on the edges.

Some instances are from the literature, while others were generated on purpose. In the next subsections, we provide the corresponding information and report the results obtained for each one of them. Each instance was tested with the following heuristics: the local search algorithm (LS), the NT-heuristic cited in [4], and the VNS and TS algorithms described in Section 3 (which were run for 10 minutes after finding the first local optimum, unless specified otherwise). For most instances, we also report a lower bound.

5.1 Graphs with known optima

N-Star graphs. These graphs can be represented as regular polygons having N sides. The nodes are those adjacent to the N sides and a node at the center of the polygon; the edges are the N sides and all edges connecting the central node with each of the other nodes. An example with $N = 8$ sides is shown in Figure 8(a). The spanning tree giving rise to the optimal FCB consists of all edges connecting the central node with each of the other nodes. All the cycles in the optimal FCB have 3 edges; each of the N sides of the polygon is a chord. The optimal FCB cost is $3N$. We considered the instances for all N up to 50, and in all cases the optimal solution was reached without any edge swap, i.e., the initial constructive heuristic always found the optimal solution.

Rectangular mesh graphs. These graphs consist of a rectangular array of $4 \times N$ nodes; Figure 8(b) shows an example with $N = 9$, together with the optimal solution. Although it is easy to show that the optimal solutions for such instances are all similar to that given in Figure 8(b), constructive heuristics fail to find it. The optimal FCB has $2(n-3)+6$ square cycles (4 edges each) and $N-3$ rectangular cycles (6 edges each) with an optimal cost of $14N-28$. All instances from $N = 10$ to $N = 100$ were tried, and the local search heuristic always managed to find the optimum, either within 2 or 4 edge swaps.

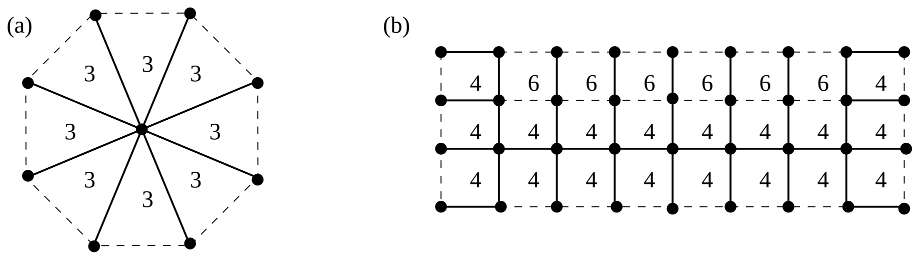

Fig. 8. Instance of 8-star graph with optimal FCB (a). Instance of (4×9)-rectangular mesh with optimal FCB (b).

5.2 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. Due to the large number of symmetries in these graphs, there are many different spanning trees with identical associated FCB costs. Uniform cost square mesh graphs have $n = N^2$ nodes and $m = 2N(N-1)$ edges. Table 1 reports the FCB costs and corresponding CPU times. The lower bounds for this class of graphs correspond to the formula $4(m-n+1)$, namely to the cost of a minimum non-fundamental cycle basis. In the case of mesh graphs, it is also a minimum weakly fundamental cycle basis.

For this class of graphs LS finds much better solutions than NT, but the differences in CPU times are enormous. TS performs very slightly better than VNS on average.

Since these results were first presented in [1], considerable attention has been devoted to the Min FCB problem. In particular, the focus in [36] is on mesh graphs, and the authors improve the FCB costs by around 16% with respect to Table 1 and the lower bounds by around 67%. Our methods, of course, are applicable to general graphs, whereas those developed in [36] are tailored to mesh graphs. In any case, it appears clear that reducing the gap between lower and upper bounds is a definite challenge for future research.

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

Table 1
Computational results (CPU times in format h:mm:ss) for $N \times N$ mesh graphs. Values marked with * denote the best value found for the instance.

5.3 Random Euclidean graphs

We have generated simple random edge-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 $0 < p < 1$ (also called the *edge density*). 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. The computational results are given in Table 2. Lower bounds have been computed by solving a linear relaxation of the formulation in Section 4. The average percentage gap between the LS heuristic and lower bounding FCB costs is 8.19%, the standard deviation is ± 5.15%, and the mode is 6% (these values have been obtained considering a larger instance set than that reported in Table 2, namely 81 instances). 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 [32]. The performance of VNS and TS is very similar on this class of graphs.

5.4 Instances from real-life applications

Periodic Timetabling. An interesting application of Min FCB arises in periodic timetabling for transportation systems. In [5], the timetables of the Berlin underground are designed by considering a mathematical programming model based on the Periodic Event Scheduling Problem (PESP) [37] 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

	$p = 0.2$							
n	LS	Time	NT	Time	VNS	TS	Bound	Time
10	216.698†	0	222.987	0	216.698†	216.698†	216.698†	0
20	1052.38†	0	1178.15	0	1052.38†	1052.38†	1052.38†	0:56
30	3315.89	0	3868.78	0	3111.71*	3311.71*	2750.92	0:28
40	4634.04	0	6154.1	0.01	4504.84*	4505.84*	4065.187	16:58
50	7007.34	0:01	9326.82	0.01	6991.53*	6991.53*	6448.711	2:38:51

	$p = 0.4$							
n	LS	Time	NT	Time	VNS	TS	Bound	Time
10	472.599	0	504.275	0	459.305†	459.305*	459.305†	0:02
20	2021.82	0	2801.28	0	2021.37*	2021.37*	1894.747	0:08
30	4467.13	0	5482.21	0.01	4455.2*	4455.2*	4265.6	22:56
40	7685.97	0:01	9509.93	0.01	7648*	7648*	-	-
50	11096.8	0:05	15667.5	0.01	11022.8*	11022.8*	-	-

	$p = 0.6$							
n	LS	Time	NT	Time	VNS	TS	Bound	Time
10	581.525	0	593.475	0	547.406†	547.406†	547.406†	0:08
20	2776.22	0	3959.41	0	2756.6*	2756.6*	2627.558	0:59
30	7031.2	0	8243.03	0.01	6979.15	6967.98*	6445.83	39:32
40	11686.0	0:02	13469.5	0.01	11513*	11513*	-	-
50	19387.3	0:10	24440.3	0.02	19174.1*	19174.1*	-	-

	$p = 0.8$							
n	LS	Time	NT	Time	VNS	TS	Bound	Time
10	992.866	0	992.866	0	775.838†	775.838†	775.838†	0:26
20	3478.11	0	4019.34	0	3383.45*	3383.45*	3164.9	2:31
30	8971.78	0:01	10847.8	0.01	8384.32*	8384.32*	7823.848	1:43:05
40	14946.4	0:07	20229.2	0.01	14870.7	14792*	-	-
50	25349.9	0:12	28586.8	0.02	25061.2*	25245.5	-	-

Table 2
Computational results (CPU times in seconds or hh:mm:ss) for Euclidean random graphs. Lower bound values marked with † denote an optimal solution (MIP solved to optimality by CPLEX v. 9.1 [33]). FCB costs marked with * denote the best value found for the instance. Missing values are due to excessive CPU timings.

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 finding 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, which makes them difficult. The results reported in Table 3 for instance `timtab2` (with 88 nodes and 316 edges), which is available from MIPLIB (`http://miplib.zib.de`), 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 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 `l-fixed`), this additional condition obviously leads to FCBs with substantially larger costs. Missing values in the table, marked with “-”, are due to a missing implementation of the corresponding algorithm which deals with mandatory spanning tree edges. Lower bounds were obtained by solving the linear relaxation of the formulation in Section 4.

Testing of electrical networks. The MIN FCB problem also arises in the event-driven simulation of electrical circuits [3]. The electrical network is decomposed into current sources (represented by branches in the tree) and voltage sources (represented by chords in the co-tree). Fundamental cycles are then used to compute the voltage across the current sources, and fundamental cuts to determine the current through the voltage sources. When the network evolves in time, it is necessary to perform edge swapping, and the best choice appears to be the one minimizing the length of the fundamental cycles. The instance `gm19` (from A. Brambilla, DEI, Politecnico di Milano) has 414 nodes and 1091 edges. The graph was too large for the lower bound to be computed by solving the linear relaxation of the formulation in Section 4, so we reported the minimum (non-fundamental) cycle basis cost computed by Horton’s algorithm [16].

	Local search		NT [4]	VNS		TS		Bound
Instance	FCB cost	Time	FCB cost	FCB cost	Time	FCB cost	Time	FCB cost
`timtab2`	40520	0.7s	50265	39801*	30s	39841	30s	31220.534
`l-fixed`	46072	0.13s	-	-	-	46002*	30s	39907.96
`gm19`	2592	1.5s	2662	2584*	600s	2584*	600s	$2267^{\dagger}$

Table 3
Computational results for real-life applications. Best values are marked with *. CPU times for the NT heuristic are approximately 0s. The bound marked with † is given the cost of the MCB computed by Horton’s algorithm.

5.5 TUM cycle basis test set

A collection of 2D and 3D toroidal graphs and of hypercubic graphs are available from `http://www-m9.ma.tum.de/dm/cycles/mhuber`. This website at Technische Universität München (TUM) also contains the costs of the minimum (not necessarily fundamental) cycle bases, which provide lower bounds to the minimum FCB costs. Toroidal graphs are D-dimensional grids of n^D nodes each of which is adjacent to each neighboring node, with wrap-around adjacency at the border. Computational results (for a subset of the instances) are reported in Tables 4 and 5. Lower bounds, reported in the last column, refer to the minimal non-fundamental cycle bases.

For hypercubic graphs, the locally optimal FCB found by the initial constructive heuristic was never improved by edge-swapping heuristics. This is strong

	LS		NT [4]		VNS	TS	Bound
n	Cost	Time	Cost	Time	Cost	Cost	Cost
5	140	0	154	0	138*	138*	106
10	770	0.38	834	0	738*	738*	416
15	2004	5.86	2372	0.01	1930	1926*	926
20	3884	21.05	4856	0.03	3822*	3868	1636
25	6650	80.46	8792	0.06	6414*	6650	2546
30	9680	241.73	14076	0.09	9674	9666*	3656
33	11988	442.8	18296	0.14	11976*	11994	4418

Table 4
Computational results for some 2D toroidal graphs from the TUM test set. Values marked with * denote the best value found for the instance.

computational evidence that the initial constructive heuristic actually finds the global optimum.

	LS		NT [4]		VNS	TS	Bound
n	Cost	Time	Cost	Time	Cost	Cost	Cost
5	1609	2.25	1771	0.02	1567*	1567*	1007
6	3062	16.93	3242	0.04	3024*	3028	1738
7	5409	58.87	5895	0.09	5409	5361*	2757
8	8234	107.33	9030	0.14	8210*	8214	4112
9	13105*	662.11	14525	0.61	13105	13105	5851
10	17890*	1037.36	20248	1.33	17890	17890	8022

Table 5
Computational results for the 3D toroidal graphs from the TUM test set. Values marked with * denote the best value found for the instance.

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 TS metaheuristics look very promising, even though computationally more intensive. We established structural results that allow an efficient implementation of the proposed edge swaps. We presented a new MIP formulation whose linear relaxation provides tighter lower bounds than known formulations on several classes of graphs. Finally, we tested LS, NT, VNS and TS heuristics on several classes of graphs. Our VNS and TS implementations are on average roughly equivalent in performance, and they compared favorably with straight local search but have higher computational requirements.

Acknowledgements

We are grateful to two anonymous referees for numerous comments that greatly improved the paper.

References

[1] E. Amaldi, L. Liberti, N. Maculan, F. Maffioli, Efficient edge-swapping heuristics for finding minimum fundamental cycle bases, in: C. Ribeiro, S. Martins (Eds.), Experimental and Efficient Algorithms — WEA2004 Proceedings, Vol. 3059 of LNCS, Springer-Verlag, 2004, pp. 15–29.

[2] G. Kirchhoff, Über die auflösung der gleichungen, auf welche man bei der untersuchungen der linearen verteilung galvanischer ströme geführt wird, Poggendorf Annalen Physik 72 (1847) 497–508.

[3] A. Brambilla, A. Premoli, Rigorous event-driven (red) analysis of large-scale nonlinear rc circuits, IEEE Transactions on Circuits and Systems–I: Fundamental Theory and Applications 48 (8) (2001) 938–946.

[4] N. Deo, N. Kumar, J. Parsons, Minimum-length fundamental-cycle set problem: New heuristics and an empirical investigation, Congressus Numerantium 107 (1995) 141–154.

[5] C. Liebchen, R. Möhring, A case study in periodic timetabling, in: D. Wagner (Ed.), Electronic Notes in Theoretical Computer Science, Vol. 66, Elsevier, 2002.

[6] C. Liebchen, Finding short integral cycle bases for cyclic timetabling, in: Algorithms — ESA2003 Proceedings, Vol. 2832 of LNCS, Springer-Verlag, 2003, pp. 715–726.

[7] E. J. Sussenouth, A graph theoretical algorithm for matching chemical structures, J. Chem. Doc. 5 (1965) 36–43.

[8] P. Vismara, Reconnaissance et représentation d'éléments structuraux pour la description d'objets complexes. application à l'élaboration de stratégies de synthèse en chimie organique, Ph.D. thesis, Université de Montpellier II, France (1995).

[9] E. Hubicka, M. Sysło, Minimal bases of cycles of a graph, in: Recent Advances in Graph Theory, 2nd Czech Symposium in Graph Theory, Academia, Prague, 1975, pp. 283–293.

[10] M. Sysło, An efficient cycle vector space algorithm for listing all cycles of a planar graph, Colloquia Mathematica Societatis János Bolyai (1978) 749–762.

[11] M. Sysło, On cycle bases of a graph, Networks 9 (1979) 123–132.

[12] M. Sysło, On some problems related to fundamental cycle sets of a graph, in: R. Chartrand (Ed.), Theory of Applications of Graphs, Wiley, New York, 1981, pp. 577–588.

[13] M. Sysło, On some problems related to fundamental cycle sets of a graph: Research notes, Discrete Mathematics (Banach Centre Publications, Warsaw) 7 (1982) 145–157.

[14] M. Sysło, On the fundamental cycle set graph, IEEE Transactions on Circuits and Systems 29 (3) (1982) 136–138.

[15] D. Hartvigsen, E. Zemel, Is every cycle basis fundamental?, Journal of Graph Theory 13 (1) (1989) 117–137.

[16] J. Horton, A polynomial-time algorithm to find the shortest cycle basis of a graph, SIAM Journal of Computing 16 (2) (1987) 358–366.

[17] T. Kavitha, K. Mehlhorn, D. Michail, K. Paluch, A faster algorithm for minimum cycle bases of graphs, in: Proceedings of ICALP, Vol. 3124 of LNCS, Springer-Verlag, 2004, pp. 846–857.

[18] E. Amaldi, R. Rizzi, Personal communication .

[19] N. Deo, G. Prabhu, M. Krishnamoorthy, Algorithms for generating fundamental cycles in a graph, ACM Transactions on Mathematical Software 8 (1) (1982) 26–42.

[20] G. Galbiati, R. Rizzi, E. Amaldi, On the approximability of the minimum strictly fundamental cycle bases problem, Technical Report, DEI, Politecnico di Milano, 2007 .

[21] K. Paton, An algorithm for finding a fundamental set of cycles of a graph, Communications of the ACM 12 (9) (1969) 514–518.

[22] B. Y. Wu, G. Lancia, V. Bafna, R. Chao, K.-M. Ravi, C. Tang, A polynomial-time approximation scheme for minimum routing cost spanning trees, SIAM Journal on Computing 29 (1999) 761–778.

[23] M. Fischetti, G. Lancia, P. Serafini, Exact algorithms for minimum routing cost trees, Networks 93 (3) (2002) 161–173.

[24] H. Gabow, E. Myers, Finding all spanning trees of directed and undirected graphs, SIAM Journal of Computing 7 (3) (1978) 280–287.

[25] A. Shioura, A. Tamura, T. Uno, An optimal algorithm for scanning all spanning trees of undirected graphs, SIAM Journal of Computing 26 (3) (1997) 678–692.

[26] M. Elkin, Y. Emek, D. Spielman, S.-H. Teng, Lower-stretch spanning trees, in: STOC '05: Proceedings of the 37th Annual ACM Symposium on Theory of computing, ACM, New York, 2005, pp. 494–503.

[27] M. Elkin, C. Liebchen, R. Rizzi, New length bounds for cycle bases, Information Processing Letters 104 (2007) 186–193.

[28] K. Mehlhorn, S. Näher, Leda, a platform for combinatorial and geometric computing, Communications of the ACM 38 (1) (1995) 96–102.

[29] P. Hansen, N. Mladenović, Variable neighbourhood search, in: P. Pardalos, M. Resende (Eds.), Handbook of Applied Optimization, Oxford University Press, Oxford, 2002.

[30] A. Hertz, E. Taillard, D. de Werra, Tabu search, in: E. Aarts, J. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, Chichester, 1997, pp. 121–136.

[31] P. Hansen, N. Mladenović, Variable neighbourhood search: Principles and applications, European Journal of Operations Research 130 (2001) 449–467.

[32] L. Liberti, E. Amaldi, N. Maculan, F. Maffioli, Mathematical models and a constructive heuristic for finding minimum fundamental cycle bases, Yugoslav Journal of Operations Research 15 (1) (2005) 15–24.

[33] ILOG, ILOG CPLEX 8.0 User's Manual, ILOG S.A., Gentilly, France (2002).

[34] B. Guta, Subgradient optimization methods in integer programming, with an application to a radiation therapy problem, Ph.D. thesis, Kaiserslautern University, Germany (2003).

[35] L. Lissoni, Implementazione di un algoritmo efficiente per determinare una base di cicli minima di un grafo, tesi di Laurea, DEI, Politecnico di Milano, Italy (2003).

[36] C. Liebchen, G. Wünsch, E. Köhler, A. Reich, R. Rizzi, Benchmarks for strictly fundamental cycle bases, in: C. Demetrescu (Ed.), Experimental Algorithms — WEA 2007, Vol. 4525 of LNCS, Springer, New York, 2007, pp. 365–378.

[37] P. Serafini, W. Ukovich, A mathematical model for periodic scheduling problems, SIAM Journal of Discrete Mathematics 2 (4) (1989) 550–581.

[38] N. Alon, R. Karp, D. Peleg, D. West, A graph-theoretic game and its application to the k-server problem, SIAM Journal on Computing (1995).

Appendix A: proof of Theorem 9

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 distinguish four separate cases, illustrated in Figures 9-12.

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

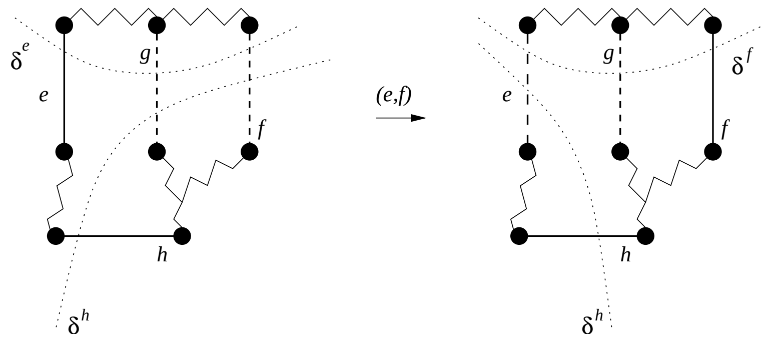

Fig. 9. Claim 1: If $g \in \delta^h \cap \delta^e$ then $g \notin \pi(\delta^h)$.

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 7 $g \in \delta^f_{\pi T}$. Thus in πT there exist paths $r_1^{\pi T}$ in $S^f_{\pi T} = S_T^e$ and $r_2^{\pi T}$ in $\bar{S}^f_{\pi T} = \bar{S}_T^e$ 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 6, 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.

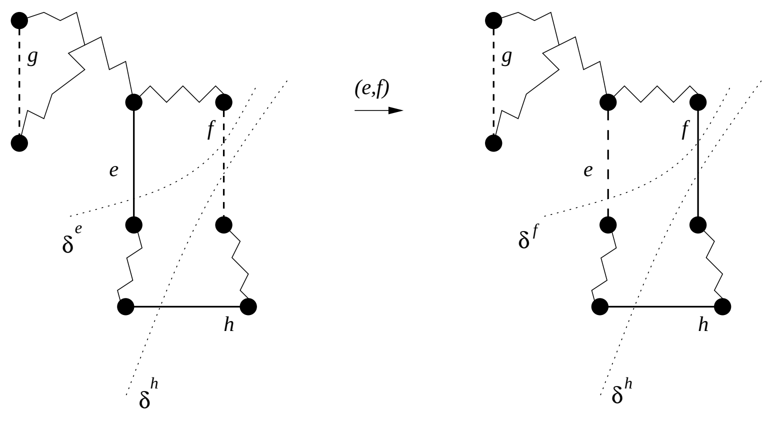

Fig. 10. Claim 2: If $g \notin \delta^h \cup \delta^e$ then $g \notin \pi(\delta^h)$.

Let 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 $g \notin \delta^f_{\pi T} = \delta^e_T$, 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 6, 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

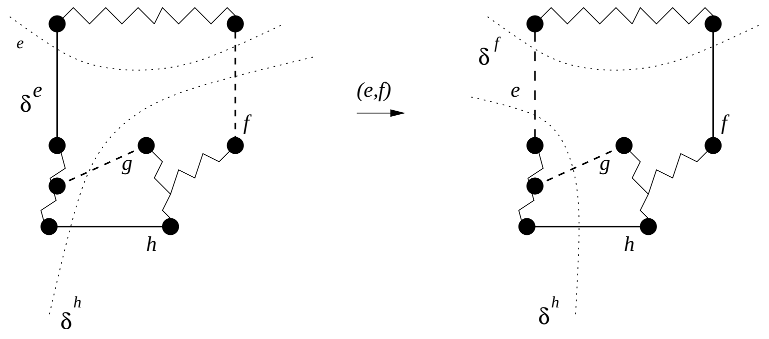

Fig. 11. Claim 3: If $g \in \delta^h$ and $g \notin \delta^e$, then $g \in \pi(\delta^h)$.

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 $r^T \subseteq S_T^h$ 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. $p_1^T = \{h\} \cup r^T \cup \{e\} \cup q_1^T$.

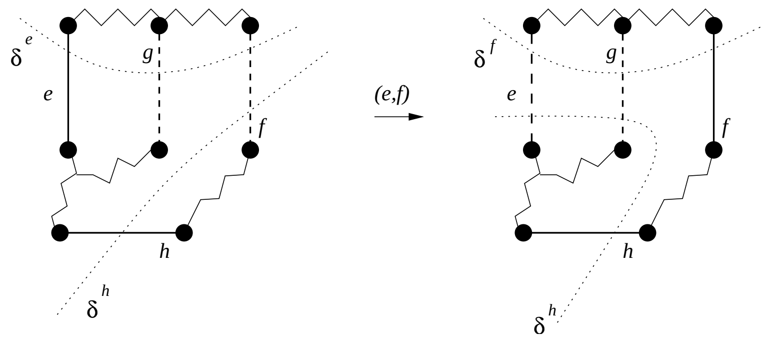

Fig. 12. Claim 4: If $g \notin \delta^h$ and $g \in \delta^e$, then $g \in \pi(\delta^h)$.

Since the path $r^T \cup q_2^T$ does not include e and connects h, g, $e \notin p_2^T$. Thus $p_2^{\pi T} = p_2^T$ and $h \notin p_2^{\pi T}$. On the other hand π sends p_1^T to a unique shortest path $p_1^{\pi T}$ connecting h, g that includes f. Since $f \in \delta_T^h$, there are shortest paths $s_1^T \subseteq S_T^h, s_2^T \subseteq \bar{S}_T^h$ that do not include h. Since $p_2^T \subseteq S_T^h$, s_1^T may only touch the same endpoint of h as p_2^T. Thus the endpoint of h touched by p_1^T also originates s_2^T. Since $s_2^T \subseteq \bar{S}_T^h$, $h \notin s_2^T$. Since $p_1^{\pi T}$ joins h, g, contains f and is shortest, $p_1^{\pi T} = s_1^T \cup \{f\} \cup u$, where $u^{\pi T}$ is a shortest path from f to g (which exists because by Proposition 7 $g \in \delta_{\pi T}^f$), which shows that $h \notin p_1^{\pi T}$. Thus by Lemma 6, $g \in \pi(\delta^h)$. The second possible case is that $e \in S_T^h, g \in \bar{S}_T^h$. Since $g \in \delta_T^e$ there are shortest paths p_1^T, p_2^T connecting e, g such that neither includes e. Assume w.l.o.g. $h \in S_T^e$. Since e, g are partitioned by δ_T^h, exactly one of p_1^T, p_2^T includes h (say $h \in p_1^T$, which implies $p_1^T \subseteq S_T^h$). Let q_1^T be the sub-path of p_1^T joining h and g and not including h, and let r^T be the sub-path of p_1^T joining h and e and not including h. Let $q_2^T = r^T \cup \{e\} \cup p_2^T$. We have that q_2 is a shortest path joining h, g not including h. Thus $\bar{P}_T(h, g) = \{q_1^T, q_2^T\} = P_T^*(h, g)$, and by Lemma 6 $g \in \delta_T^h$, which is a contradiction. □

Appendix B: Comparison of best swap algorithms

Computational comparison of the local search algorithm using LCA-based and symmetric difference-based implementations. Average of user CPU times ratio LCA-based/Symmdiff-based: 21.381733, standard deviation: 10.145212.

Instance name	**FCB Cost**	**LCA-based** (seconds)	**Symmdiff-based** (seconds)
euclid-40_0.8	13698.1	261.345	12.4031
euclid-50_0.3	10085.6	66.1819	3.69544
euclid-75_0.2	16878.5	284.533	10.2924
hypercube-7_0	603.327	146.554	17.8823
hypercube-7_1	512.612	135.625	12.5261
hypercube-7_2	573.665	128.384	10.0195
hypercube-7_3	515.37	194.44	20.1819
hypercube-7_4	583.176	174.868	14.1029
hypercube-7_5	583.601	163.057	18.6732
hypercube-7_6	568.314	150.146	16.8114
hypercube-7_7	577.642	149.693	15.0577
hypercube-7_8	555.623	152.57	18.2222
hypercube-7_9	553.632	140.768	15.2127
hypercube-7	1986	1.75073	8.75267
mesh-15_0	539.587	53.8818	1.57976
mesh-15_1	533.461	58.3101	1.82272
mesh-15_2	513.962	53.0169	1.71174
mesh-15_3	491.852	53.0849	2.05369
mesh-15_4	536.154	59.5309	2.34864
mesh-15_5	496.06	56.0165	2.22666
mesh-15_6	509.269	41.6437	1.67375
mesh-15_7	461.934	48.8766	1.3318
mesh-15_8	530.033	46.3799	1.46478
mesh-15_9	551.304	46.4889	1.74074
mesh-15	1318	22.7785	0.955855
random-40_0.8	1846	3.54346	0.086986
random-50_0.3_0	279.792	91.1901	9.70252
random-50_0.3_1	323.47	101.088	7.18991
random-50_0.3_2	318.589	90.3893	10.5884
random-50_0.3_3	307.708	86.9288	6.72598
random-50_0.3_4	341.978	73.8478	5.75413
random-50_0.3_5	351.295	117.428	10.4234
random-50_0.3_6	338.125	91.2041	4.05738
random-50_0.3_7	327.712	138.806	10.0675
random-50_0.3_8	312.191	98.1721	4.83727
random-50_0.3_9	347.057	146.437	10.0145
random-50_0.3	1410	11.9282	0.652901
random-75_0.2	2239	82.7704	2.36664
torus-15_0	779.19	136.352	12.3541
torus-15_1	684.073	102.291	12.927
torus-15_2	694.008	79.7819	9.8385
torus-15_3	688.822	98.3271	11.7462
torus-15_4	679.94	87.3487	13.6699
torus-15_5	677.581	77.7322	12.3981
torus-15_6	729.846	97.2192	12.0492
torus-15_7	693.137	95.9574	13.191
torus-15_8	785.313	103.293	12.7601
torus-15_9	761.546	84.1552	10.5694
torus-15	1994	46.8199	6.11907
triangle-20_0	630.693	312.561	7.50186
triangle-20_1	676.275	297.625	6.92595
triangle-20_2	644.77	264.222	6.30804
triangle-20_3	637.528	296.559	6.33004
triangle-20_4	655.714	250.214	4.96824
triangle-20_5	623.176	273.085	5.95609
triangle-20_6	633.518	304.851	5.9101
triangle-20_7	553.599	235.029	4.20336
triangle-20_8	617.789	259.001	5.3002
triangle-20_9	655.251	286.719	7.02793
triangle-20	1934	115.115	2.98455