Approximating Minimum Size $\{1,2\}$-Connected Networks

Piotr Krysta

MPI–I–2001–1-001 January 2001

Author's Address

Piotr Krysta
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany
krysta@mpi-sb.mpg.de

Work partially supported by the IST Program of the EU under contract number IST-1999-14186 (ALCOM-FT). The author was supported by Deutsche Forschungsgemeinschaft (DFG) Graduate Scholarship. Most of this work was done while the author was visiting the Combinatorics & Optimization Dept., University of Waterloo, Ontario, Canada, during January-March, 2000, and was partially supported by NSERC grant no. OGP0138432 of Joseph Cheriyan.

Abstract

The problem of finding the minimum size 2-connected subgraph is a classical problem in network design. It is known to be NP-hard even on cubic planar graphs and Max-SNP hard. We study the generalization of this problem, where requirements of 1 or 2 edge or vertex disjoint paths are specified between every pair of vertices, and the aim is to find a minimum subgraph satisfying these requirements. For both problems we give $3/2$-approximation algorithms. This improves on the straightforward 2-approximation algorithms for these problems, and generalizes earlier results for 2-connectivity. We also give analyses of the classical local optimization heuristics for these two network design problems.

1 Introduction

Graph connectivity is an important topic in theory and practice. It finds applications in the design of computer and telecommunication networks, and in the design of transportation systems. Networks with certain level of connectivity, which intuitively means that they provide certain number of connections between sites, are able to maintain reliable communication between sites even when some of the network elements fail. For a survey and further applications, see Grötchel *et al.* [8].

Problem statement. Given a graph with weights on its edges, and an integral *connectivity requirement* function r_{uv} for each pair of vertices u and v, the *vertex connectivity* (*edge connectivity*, respectively) *Survivable Network Design Problem* (SNDP) is to find a minimum weight subgraph containing at least r_{uv} vertex (edge, respectively) disjoint paths between each pair u, v of vertices. If $r_{uv} \in X$ for some set X, for each pair u, v, we denote the problem as X-VC-SNDP (X-EC-SNDP, respectively). The term survivable refers to the fact that the network is tolerant to the failures of sites and links (in case of VC-SNDP) or links (for EC-SNDP). Even the simplest versions of these problems are NP-hard, and so approximation algorithms[1] are of interest.

Previous results for general cases. For the EC-SNDP problem with arbitrary edge weights, Williamson *et al.* [16] have given a $2r_{max}$-approximation algorithm, where r_{max} is the maximum value of the requirement function. This was improved later to a $2\mathcal{H}(r_{max})$-approximation by Goemans *et al.* [6], where $\mathcal{H}(k) = 1 + \frac{1}{2} + \ldots + \frac{1}{k}$ is the kth harmonic number. The best known result is a 2-approximation algorithm, due to Jain [9]. No algorithm with a non-trivial approximation guarantee is known for the general version of the VC-SNDP. For the $\{0, 1, 2\}$-VC-SNDP with arbitrary edge weights, Ravi and Williamson [14] have given a 3-approximation algorithm. For a special case of the VC-SNDP with $r_{max} \leq 3$, Nutov [13] designed a $10/3$-approximation algorithm.

Unweighted low-connectivity problems. The case of low connectivity requirements is of particular importance, as in practice networks have rather small connectivities. There has been intense research in the subarea of network design for low connectivity requirements and assuming that all the edge weights are equal to one (*unweighted problems*) [3, 5, 11, 15]. We focus on the special cases of this problem where each $r_{uv} \in \{1, 2\}$ and G is an unweighted, undirected graph. These are the simplest non-trivial versions of this problem and have been studied for a long time, but tight approximation guarantees and inapproximability results are not fully understood yet.

For the unweighted $\{2\}$-EC-SNDP (or 2-EC) Khuller and Vishkin [11] gave a $\frac{3}{2}$-approximation, which was improved by Cheriyan *et al.* [3] to $\frac{17}{12}$, and recently to $\frac{4}{3}$ by Vempala and Vetta [15]. For the unweighted $\{2\}$-VC-SNDP (or 2-VC) Khuller and Vishkin [11] gave an algorithm with approximation guarantee of $\frac{5}{3}$, which was improved to $\frac{3}{2}$ by Garg *et*

[1]A polynomial time algorithm is called an α-*approximation algorithm*, or is said to achieve an *approximation (or performance) guarantee* of α, if it finds a solution of weight at most α times the weight of an optimal solution. α is also called an *approximation ratio (factor)*.

al. [5], and finally to $\frac{4}{3}$ by Vempala and Vetta [15].

Both unweighted 2-VC and 2-EC problems are NP-hard even on cubic planar graphs. These problems are also MAX SNP-hard [4]. For the $\{1, 2, \ldots, k\}$-VC-SNDP and for $\{1, 2, \ldots, k\}$-EC-SNDP, the results of Nagamochi and Ibaraki [12] imply k-approximation algorithms for the unweighted case – see Proposition 2.

Little is known about the generalizations where arbitrary requirements are allowed, especially for vertex-connectivity, even for unweighted graphs, apart from the results in [9, 6, 16, 14]. The simplest such generalization is one which allows requirements to be either 1 or 2, instead of 2 for every pair. It should be noted that allowing the requirement function r to take values from $\{0, 1, \ldots, k\}$ (i.e. when *zero* is also allowed), for some integer k, makes the unweighted and weighted problems essentially identical. This is because an edge with an integer weight w can always be replaced by a path of Steiner vertices of length w. For instance, unweighted $\{0, 1, 2\}$-VC-SNDP is equivalent to the arbitrary edge weights $\{0, 1, 2\}$-VC-SNDP considered by Ravi and Williamson [14].

Our contributions. For both unweighted $\{1, 2\}$-VC-SNDP and $\{1, 2\}$-EC-SNDP (henceforth denoted by $\{1, 2\}$-VC and $\{1, 2\}$-EC), we give $3/2$-approximation algorithms. This improves on straightforward 2-approximation algorithms for these problems (Proposition 2). Our algorithms are generalizations and extensions of the algorithms of Garg *et al.* [5] and of Khuller and Vishkin [11]. We also present analyses of the classical local optimization heuristics for our problems.

From now on we assume that the edge weights are all equal to one (unweighted problems). Given a requirement function r_{uv} on the pairs u, v of vertices, we define a *requirement $\hat{r}_u$ of a vertex u* as $\hat{r}_u = \max\{r_{uv} : v \in V \setminus u\}$. Garg *et al.* have used $\max(n, 2|I|)$ as a lower bound to the 2-VC problem to get a $3/2$-approximation algorithm, where I is an independent set of vertices and n is the number of vertices in the given graph. Their lower bound does not apply to our problem since some of the vertices in I may have a requirement of one. We generalize this lower bound to take for I' an independent set of vertices that have requirement equal to 2. We use this lower bound and generalize the algorithm of Garg *et al.* and its analysis to prove that the size of an optimal solution to $\{1, 2\}$-VC is at most $\frac{3}{2}\max(n, 2|I'|)$, assuming a vertex of requirement 2 exists.

The lower bound of Khuller and Vishkin for the 2-EC problem also does not apply to the $\{1, 2\}$-EC. We show an appropriate generalization of their idea and a $3/2$-approximation algorithm for the $\{1, 2\}$-EC based on it.

Our performance guarantees for the algorithms in Sections 2.2 and 2.3 are tight with respect to the lower bounds that we use. This follows from the examples given in [5, 11] showing that the obtained approximation guarantees are tight even for 2-VC and 2-EC problems.

Local search or local optimization heuristics are one of the oldest and most widely used methods in solving combinatorial optimization problems, like the traveling salesman or vehicle routing problems [1]. We present analyses of the local optimization heuristics for our problems. Based on ear decompositions of 2-connected graphs, we prove two new lower bounds. This implies algorithms for our problems that are simple and easy to describe. In particular a $7/4$-approximation algorithm for $\{1,2\}$-VC, and a $5/3$-approximation algorithm for $\{1,2\}$-EC.

Organization of the paper. Section 2 contains definitions and preliminary results, section 2.1 describes a decomposition method for our problems, then in section 2.2 we present our algorithm for the $\{1,2\}$-VC problem and we show how to use it to get an algorithm for $\{1,2\}$-EC in section 2.3. Also, a simple and different algorithm for $\{1,2\}$-EC is shown. The next section 2.4 contains the analyses of the local optimization heuristics, and finally section 3 has some concluding remarks.

2 Preliminaries

We consider only undirected, simple graphs. Given a graph $G = (V, E)$, we also write $V(G) = V$ and $E(G) = E$. We use standard terminology from graph theory. We refer to the elements of V as *vertices*, but we also consider *(super) nodes* which will correspond to sets of vertices, or contracted subgraphs of the input graph G. Elements of E, which are undirected pairs of vertices, are called *edges*. A closed path of length l is a cycle, denoted C_l, and an open path means that all the vertices are distinct. Given a cycle C, any edge joining two non-consecutive vertices of C is called a *chord*. A $u - v$ *path* is a path with end vertices u, v. We say that a vertex v is a *cut vertex* if its removal disconnects the graph. If v is a cut vertex of a graph G, and some two vertices x, y are in distinct components of $G \setminus v$, then v *separates* x and y. An edge is a *bridge* if its removal disconnects the graph. $d_G(v)$ denotes the degree of vertex v in graph G.

An *ear decomposition* $\mathcal{E}$ of a graph G is a partition of the edge set into open or closed paths, $\mathcal{E} = \{Q_0, Q_1, \ldots, Q_k\}$, such that Q_0 is the trivial path with one vertex, and each Q_i ($i = 1, \ldots, k$) is a path that has both end vertices in $V_{i-1} = V(Q_0) \cup \cdots \cup V(Q_{i-1})$ but has no internal vertex in V_{i-1}. A (closed or open) *ear* means one of the (closed or open) paths $Q_0, Q_1, \ldots, Q_k$ in $\mathcal{E}$. Given a non-negative integer ℓ, an *ℓ-ear* is an ear with ℓ edges. An ear decomposition $\{Q_0, Q_1, \ldots, Q_k\}$ is *open* if all the ears $Q_2, \ldots, Q_k$ are open. If a graph is 2-vertex(edge)-connected, then we also say that it is 2-VC(EC). The following results are well known.

Proposition 1 *The following are characterizations of 2-EC and 2-VC in a graph.*

1. *A graph is 2-EC if and only if it has no bridge. A graph is 2-VC if and only if it has no cut vertex.*

2. *A graph is 2-EC if and only if it has an ear decomposition. A graph is 2-VC if and only if it has an open ear decomposition. An (open) ear decomposition can be found in polynomial time.*

Lemma 1 ([5]) *Given a 2-EC graph G with a cycle C and edge e being a chord in C, the graph $G - e$ is also 2-EC.*

Let $\mathcal{E}$ be an ear decomposition of a 2-connected graph. We call an ear $S \in \mathcal{E}$ of length ≥ 2 *pendant* if none of the internal vertices of S is an end vertex of another ear $T \in \mathcal{E}$ of length ≥ 2. Let $\mathcal{E}' \subseteq \mathcal{E}$ be a subset of ears of the ear decomposition $\mathcal{E}$. We say that set $\mathcal{E}'$ is *terminal* in $\mathcal{E}$ if: (1) every ear in $\mathcal{E}'$ is a pendant ear of $\mathcal{E}$, (2) for every pair of ears $S, T \in \mathcal{E}'$ there is no edge between an internal vertex of S and an internal vertex of T, and (3) every ear in $\mathcal{E}'$ is open.

Given a rooted tree T, let a "closed interval" $[a, b]$ denote the $a - b$ path in tree T for some two vertices a, b such that b is an ancestor of a, and path $[a, b]$ contains both vertices a, b. Similarly, we define $[a, b)$ as the $a - b$ tree path with b being an ancestor of a and the path contains a and does not b. Also $(a, b]$ is defined by analogy. If a is a (proper) descendant of b in T, then we say that a is *below* or *lower than* b, and b is *above* or *higher than* a. A vertex in a rooted tree is also descendant and ancestor of itself.

We also denote by $opt(G)$ or by just *opt* the value of an optimal solution on G to the problem under consideration.

For a given requirement function $r_{\cdot,\cdot}$ defined on the pairs of vertices in $V \times V$, we define a vertex requirement function $\hat{r}_{\cdot}$ on the vertices in V as follows: for any $u \in V$, $\hat{r}_u = \max\{r_{uv} : v \in V \setminus u\}$. The following observation can easily be deduced from the results of Nagamochi and Ibaraki [12].

Proposition 2 *There is a k-approximation algorithm for the unweighted $\{1, 2, \ldots, k\}$-VC-SNDP problem. Also, there is a k-approximation algorithm for the unweighted $\{1, 2, \ldots, k\}$-EC-SNDP problem.*

Proof: [12] has designed an algorithm that for the unweighted k-VC problem finds a sparse subgraph with at most $k(n-1)$ edges, where n is the number of vertices in the input graph, such that the pairwise (local) vertex-connectivities are preserved in this subgraph up to k. Therefore, this gives a feasible solution to the $\{1, 2, \ldots, k\}$-VC-SNDP problem. Since n is a lower bound for this problem (if there is at least one pair u, v with $r_{uv} \geq 2$), this implies a k-approximation algorithm. The same is true for the $\{1, 2, \ldots, k\}$-EC-SNDP problem. ■

2.1 Decomposing into subproblems

We first outline a way to decompose the problem into subproblems. For that, we will follow a standard decomposition method into 2-vertex connected components. This is done as follows.

Let $G = (V, E)$ be a given instance of the $\{1, 2\}$-VC(EC)-SNDP problem. Specify a *lower bound* $lb(G)$ on the value of an optimal solution $opt(G)$ for the problem on G. Decompose G into (maximal) subgraphs $C_1, \ldots, C_l$ that are 2-vertex connected components

(2-VC blocks) of G. Note, that if $e \in E$ is a bridge in G, then some C_i will contain e as the only edge. Since all the edge sets $E(C_i)$ are edge disjoint, it is clear that $\sum_{i=1}^{l} lb(C_i)$ will be a lower bound on $opt(G)$. Thus to prove the approximation guarantee for the original problem on G, we can argue for a guarantee within each subproblem. The maximum over these approximation guarantees will give the performance guarantee of the overall algorithm.

Let $G_i = (V(C_i), E(C_i))$, and $n_i = |V(C_i)|$ for $i = 1, \ldots, l$.

2.1.1 Decomposing $\{1, 2\}$-VC

For each subproblem G_i its requirement function r^i is defined for a vertex $u \in V(C_i)$ as follows: $r^i_{uv} = 2$ if there is $v \in V(C_i)$ with $r_{uv} = 2$, and $r^i_{uv} = 1$ otherwise. The definition is different for the $\{1, 2\}$-EC problem, see Section 2.1.2. Moreover, let for any $u \in V(C_i)$, $\hat{r}^i_u = \max\{r^i_{uv} : v \in V(C_i) \setminus u\}$. The algorithm will process each G_i separately. The following lemma justifies this approach for the $\{1, 2\}$-VC problem.

Lemma 2 *The problem can be solved for each C_i separately and independently. That is the union of the solutions to each C_i gives a solution to the problem on G, and maximum approximation guarantee among the guarantees for subproblems C_i's is the approximation guarantee for the problem on G.*

Proof: The first part follows from $opt(G) = \sum_i opt(G_i)$. Given solutions H_i for G_i, $i = 1, \ldots, l$, $\bigcup_i H_i$ is a feasible solution to G. This is true, since: (1) if $u \in V(C_i)$, $v \in V(C_j)$ ($i \neq j$), then $r_{uv} = 1$, and (2) each H_i is connected, so their union is also connected. Now let H be a solution to G, and let H_i be the subgraph induced by $E(H)$ on the vertex set $V(C_i)$, for $i = 1, \ldots, l$. We argue that H_i is a feasible solution to G_i. Let $u, v \in V(C_i)$ with $r^i_{uv} = 2$. Then $r_{uv} = 2$, so $E(H)$ contains two vertex-disjoint $u - v$ paths. Since G_i was a 2-VC block of G, $E(H_i)$ must also contain two vertex-disjoint $u - v$ paths.

The part of the lemma about the approximation guarantee follows from the discussion in the beginning of Section 2.1. ■

If for some i, $V(C_i)$ does not contain any vertex pair u, v with $r^i_{uv} = 2$, then the solution to C_i will be any spanning tree of C_i. Observe, that since all the requirements are at least 1, $lb(C_i) = |V(C_i)| - 1$ will be the lower bound for the problem within C_i. Therefore, the performance guarantee for C_i will be one. Thus we can focus in Section 2.2, on C_i such that there exists a vertex pair $u, v \in V(C_i)$ with $r^i_{uv} = 2$.

2.1.2 Decomposing $\{1, 2\}$-EC

Let us focus on some C_i, and let $U_i \subset V(C_i)$ be the set of all cut vertices that separate between C_i and other blocks C_j's. For each vertex $u \in U_i$, let $V^i(u) \subset V$ be a set of vertices $w \in (V \setminus V(C_i)) \cup \{u\}$ such that there is a $u - w$ path in G that does not use any edge in $E(C_i)$. Note that $u \in V^i(u)$. We also define $V^i(u)$ to be just $\{u\}$ in case when $u \in V(C_i) \setminus U_i$. With these definitions, we can now define the requirement function r^i for the $\{1,2\}$-EC on G_i: for a vertex pair $u, v \in V(C_i)$ we have $r^i_{uv} = \max\{r_{xy} : x \in V^i(u), y \in V^i(v)\}$. Let also for any $u \in V(C_i)$, $\hat{r}^i_u = \max\{r^i_{uv} : v \in V(C_i) \setminus u\}$. We have the following lemma, analogous to that in Section 2.1.1.

Lemma 3 *The problem $\{1,2\}$-EC can be solved for each C_i separately and independently. That is the union of the solutions to each C_i gives a solution to the problem on G, and maximum approximation guarantee among the guarantees for subproblems C_i's is the approximation guarantee for the problem on G.*

Proof: The first part follows from $opt(G) = \sum_i opt(G_i)$. Given solutions H_i for G_i, $i = 1, \ldots, l$, $\bigcup_i H_i$ is a feasible solution to G. This follows from: (1) transitivity property for edge-connectivity, that is for any $x, y, z \in V$ if there are k edge disjoint $x - y$ paths and k edge disjoint $y - z$ paths, then their union forms k edge disjoint $x - z$ paths, and (2) each H_i is connected, so their union is also connected. Now let H be a solution to G, and let H_i be the subgraph induced by $E(H)$ on the vertex set $V(C_i)$, for $i = 1, \ldots, l$. We argue that H_i is a feasible solution to G_i. Let $u, v \in V(C_i)$ with $r^i_{uv} = 2$. Then $r_{xy} = 2$ for some $x \in V^i(u)$ and $y \in V^i(v)$. So $E(H)$ contains two edge-disjoint $x - y$ paths. It can easily be seen that, by the definition of r^i, this implies that $E(H_i)$ also contains two edge-disjoint $u - v$ paths.

The part of the lemma about the approximation guarantee follows from the discussion in the beginning of Section 2.1. ■

As in Section 2.1.1, if for some i, $V(C_i)$ does not have any pair u, v with $r^i_{uv} = 2$, then the solution to C_i will be a spanning tree of C_i. And the approximation guarantee for such C_i will be one. Thus, we can focus in Section 2.3, on C_i such that there is a vertex pair $u, v \in V(C_i)$ with $r^i_{uv} = 2$.

2.2 Application to $\{1,2\}$-VC

In this section we give a $3/2$-approximation algorithm for $\{1,2\}$-VC. Our lower bound, algorithm and its analysis are extensions and generalizations of the results of Garg *et al.* [5]. The algorithm has two phases. The first phase is the same as in their algorithm. The second phase is different from the second phase in [5], however, some of the cases considered are analogous to [5].

We focus on a subproblem $G_i = (V(C_i), E(C_i))$ such that G_i is 2-vertex connected and there is a pair $u, v \in V(C_i)$ with $r^i_{uv} = 2$. Below we present our lower bound for $\{1,2\}$-VC.

Lemma 4 *If there is a pair $u, v \in V(C_i)$ with $r^i_{uv} = 2$, then $opt(G_i) \geq n_i$ and $opt(G_i) \geq 2|I_i|$, where $I_i \subseteq V(C_i)$ is an independent set of vertices v such that $\hat{r}^i_v = 2$.*

Proof: The first bound of $opt(G_i) \geq n_i$ is obvious. To show the second bound notice that if v is a cut vertex that is common to some two 2-VC blocks C_i and C_j $(i \neq j)$, then any feasible solution to the problem must contain at least $\hat{r}_v^i$ edges within C_i and at least $\hat{r}_v^j$ edges within C_j. ■

In the next three sections, we show how to use this lower bound to obtain a $3/2$-approximation algorithm for the $\{1, 2\}$-VC problem.

2.2.1 The algorithm – phase 1

It is important that in this phase we pick a set E' of edges in G_i that form a 2-vertex-connected subgraph of G_i, regardless of the fact that some of the requirements r_{uv}^i might be one.

1. Perform a depth-first-search in G_i: label edges as tree edges and back edges (i.e., all the other edges that do not belong to the tree). Let T be the DFS tree.
2. W.l.o.g. we can assume that any leaf of T is a vertex $u \in V(C_i)$ with $\hat{r}_u^i = 2$. If this is not true for some leaf, repeatedly shrink the leaf tree edge, until the resulting leaf has requirement 2.
3. We include all the edges of T into E'. Now the algorithm chooses a subset of some back edges so that the resulting solution is 2-vertex-connected: when the DFS backs up from a vertex u, if the parent of u in T threatens to be a cut vertex in the currently chosen E', add to E' the highest going back edge from the subtree rooted at u. Form a *block* out of all the vertices in the subtree rooted at u that are not included in any other block.
4. At the end of the DFS, form a block out of all the vertices that do not belong to any other block. This block is called the *root* block.

The following lemma has been proved in [5, 11].

Lemma 5 ([5, 11]) *The picked subgraph has the following properties: (1) The set of edges E' constitutes a 2-vertex-connected subgraph of G_i. (2) For each block, the tree edges within this block form a spanning tree on the vertices included in the block. (3) Each leaf of T forms a block by itself, and the root of T is contained in the root block.*

We start with T, and if for each block we shrink its vertices into one vertex, then by part (2) of Lemma 5, the resulting graph will also be a rooted tree, say T'. The vertex corresponding to the root block will be the root of T'. Tree T' defines a natural parent-child relation between the blocks. For each block B distinct from the root block, we define a unique vertex in T called the *parent vertex of* B, as the vertex in the parent block of B that has a tree edge from B.

For proofs of the next two lemmas the reader is referred to [5, 11].

Lemma 6 ([11]) *All edges of G_i run, either: (1) within a block, (2) between a block and its parent block, or (3) between a block and a unique vertex in its grandparent block, where the unique vertex is the parent vertex of the parent block.*

Lemma 7 ([5]) *If a vertex in a block has no edge from any of the child blocks, then it has no edge from any descendant block.*

2.2.2 The algorithm – phase 2

Let C be a simple cycle in G_i, and u, v be two distinct vertices on C. A simple path P joining u and v is called a *chordal path* if for any internal vertex w of P, w has degree exactly two and $\hat{r}^i_w = 1$. We prove the following generalization of a lemma from [5] (see Lemma 1).

Lemma 8 *Let H_i be any feasible spanning subgraph of G_i that fulfills all the requirements r^i, and C be a simple cycle in H_i with a chordal path P. Then $H_i \setminus e_0$ also fulfills all the requirements r^i, where e_0 is any edge of P.*

Proof: Clearly, deleting edge e_0 will not disconnect the graph H_i. Therefore, we need only prove that this will not destroy any of the 2-vertex connections in H_i. Let u, v be the two end vertices of P lying on C. Assume towards a contradiction that there are two distinct vertices $x, y \in V(G_i)$ with $r^i_{xy} = 2$, and such that $H'_i = H_i \setminus e_0$ contains a cut vertex w separating x and y in H'_i. Let H^1 and H^2 be the two connected components of H'_i after the deletion of w, such that $x \in H^1$ and $y \in H^2$. Since H_i has at least two vertex disjoint $x - y$ paths, $H''_i = H_i \setminus w$ has at least one $x - y$ path. Furthermore, since $H''_i \setminus e_0$ has no $x - y$ path, edge e_0 must belong to any $x - y$ path in H''_i. This, together with the fact that P is a path of degree two internal vertices, gives that P must be completely contained in any $x - y$ path in H''_i. This means that u and v belong to different components H^1, H^2. But then any $u - v$ path goes through w, which contradicts definition of the chordal path. ■

In the next lemma we prove a property of the first phase which we will use in the 2nd phase.

Lemma 9 *Let B be a non-leaf and non-root block defined by the first phase of the algorithm and let p be the parent vertex of B. Then there is a back edge going from some child block of B to p. This back edge was picked by the algorithm into E'.*

Proof: Let u be the child vertex of p in the DFS tree T. Notice that u is unique, by part (2) of Lemma 5, and u belongs to B. During the bottom-up traversal when adding the back edges, let us focus on the time when the algorithm inspects p from u. We know that then p threatens to be a cut vertex, and u does not. p is the reason for the algorithm that it adds the farthest going back edge, say e, from the subtree rooted at u. This also means that, since u

already is not a cut vertex, there must be some back edge e' going from below into p. We argue that there must be such e' coming from a child block of B. First, by Lemma 6, we know that e' cannot come from any grandchild blocks of B. Assume that e' comes from B, into p. Whenever the algorithm adds a back edge from a subtree rooted at a vertex v (if v is a current vertex from which the algorithm inspects the parent of v), it creates a new block out of the vertices in the subtree rooted at v not contained in any other block. This means that if such back edge e' is added, then the created block B cannot contain vertex u. Contradiction. We see also that this back edge e' must be the farthest back edge coming from a child block. Therefore, it must have been added to E'. ∎

After the first phase set E' consisting of the edges of DFS tree and of one back edge out of each non-root block has been output. $I = \emptyset$, initially. The second phase will try for each block B to delete a tree edge within B, or if it is impossible to find a vertex s with $\hat{r}_s^i = 2$, add it to I, such that I remains independent. The 2nd phase will also modify the set of back edges.

Like in the second phase of the algorithm of Garg *et al.* [5], we traverse the blocks top-down. At each step we fix a block, say B, and B *decides* on the back edges going out of the child blocks of B. The first step is made with the root block: it chooses the farthest going back edge from each of its child blocks. Now, any child block having its back edge decided, decides on the back edges for its child blocks in a way we will specify. Following these steps we proceed towards the leaves.

Let now B be some non-root and non-leaf block for which the decision about the back edge $e = e(B)$ out of it has been made (by the parent block of B). Let v be an end vertex of e and $v \in B$, and $u = u(B)$ be the other end vertex of e. Let $p = p(B)$ be the parent vertex of B. See cases in Fig. 3 for an illustration.

Block Property. We can assume that: (1) the back edge $e(B)$ goes higher than $p(B)$, and (2) there is an $u(B) - p(B)$ path that goes through the ancestor blocks of B.

The above property is obviously true for child blocks of the root block, by the choice decided by the root block. We will show that this property is maintained during the algorithm, see Lemma 10. We introduce some new notation here. Let $a \overset{T}{-} b$ denote the $a - b$ path in tree T, and $a \overset{A}{-} b$ be an $a - b$ path through the vertices in the ancestor blocks of B.

Let now w' be a vertex in path $[v, p)$ that is the highest vertex with $d_T(w') \geq 3$. If there is no such vertex in $[v, p)$, then let $w' = v$. Let also, w'' be the lowest vertex in $(w', p]$ which has a back edge from some child block of B. It is easy to notice that both w' and w'' exist and are well defined (see Lemma 9). We also note that the path $[w', w'']$ has length (i.e. the number of tree edges) at least one. Let q be the parent vertex of w'. The algorithm considers the following cases.

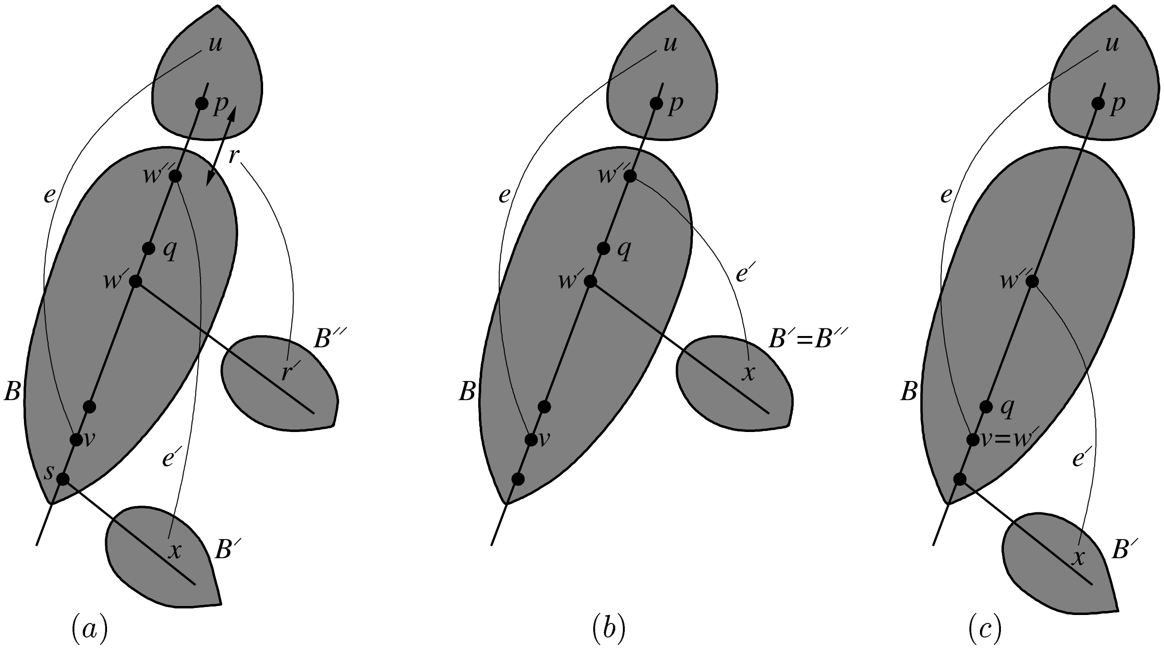

Figure 1: Cases for the analysis of phase-2. Tree edges are solid thick lines, back edges – solid thin lines, and blocks – shaded areas.

1. Assume that $q \neq w''$, and there is a vertex $s \in [q, w'')$ with $\hat{r}_s^i = 2$. In this case we label block B and vertex s with MARKED, and we add s to the lower bound set I. Also B decides to retain the 1st phase choices of the farthest going back edges from all child blocks of B.

2. Assume that either $q \neq w''$, and for any vertex $s \in [q, w'')$, we have $\hat{r}_s^i = 1$, or $q = w''$. In this case the tree edge (q, w') is deleted from the current solution. We will show later that this step preserves the connectivities.

   Let e' be the back edge going from some child block B' of B into vertex w''. Then B decides not to take the farthest going back edge from B', but e' instead. For each other child blocks of B, B decides to retain the choice of the back edges made by the 1st phase.

In case 2 of the above algorithm when we delete a tree edge within block B, we charge that edge paying this way for the back edge going out of B. Therefore, in these cases the back edge is for free, and all such blocks are labeled FREE. Finally, the root block has no back edge going out of it and so is labeled FREE. Each leaf block is itself a vertex of requirement two, and so we label it MARKED, and choose all the leaf vertices into I.

2.2.3 Analysis and approximation guarantee

Lemma 10 *Each case of the* 2*nd phase of the algorithm maintains the Block Property, and preserves the feasibility of the current solution w.r.t.* r^i*.*

Proof: The proof is by induction on the number of steps made by the second phase of the algorithm.
Induction base. The Block Property is obviously true for child blocks of the root block, by the choice decided by the root block. Also, at this point the algorithm has not made any changes, so the solution is 2-vertex connected.
Induction step. Assume that the Block Property holds for the block B that is currently considered by the algorithm. We argue that after each of the cases in the phase 2, the Block Property will be true for any child block of B. Also assume that the current solution is feasible w.r.t. r^i, and we will show that after each of the cases in phase 2, the solution will still be feasible.

Let us consider case 1 of the 2nd phase. In this case B decides to retain the 1st phase choices of the farthest going back edges from all child blocks of B. It is easy to check that the Block Property is maintained for all the child blocks of B. Since the solution is not modified in this case and by the induction assumption, the feasibility is preserved.

Let us now consider case 2 of the 2nd phase. Assume that either $q \neq w''$, and for any vertex $s \in [q, w'')$, $\hat{r}^i_s = 1$, or $q = w''$. B decides to pick a back edge e' from a child block B' into w'', instead of the farthest going back edge, say $e_{B'}$, from B'. We argue that this preserves the feasibility. By Lemma 6 and by the choice of w'', $e_{B'}$ goes into a vertex x in path $(w'', p]$ (where we assume that $(w'', p]$ contains just p, if $p = w''$).

Claim 1. Deleting the farthest going back edge $e_{B'}$ from B' and adding e', preserves the feasibility.
Proof. This change obviously does not disconnect the solution graph. Assume towards a contradiction that after the change some pair of the vertices with requirement 2 is separated by a cut vertex c. Then c must belong to $[w'', x]$. But we know that w'' is above v, and that there is a solution back edge $e = (v, u)$. By the induction assumption the Block Property holds for B, so u is above p, and so also above c. Therefore c cannot be a cut vertex. Contradiction. ■

For all other child blocks distinct from B', block B decides to retain their farthest going back edges picked by the 1st phase. We see that part (1) of the Block Property is maintained: for block B' the back edge e' goes above the parent vertex of B' since it goes above w', and w' is above or equal to the parent vertex of B'. For all other child blocks part (1) of Block Property is obviously maintained. For illustrations, see cases in Figure 1.

We will now show that also removing tree edge (q, w') from the current solution maintains the feasibility and part (2) of the Block Property (it is easy to notice that this will not affect part (1) of the Block Property). We show that there is a simple cycle C in the current solution such that $[w', w'']$ is a chordal path w.r.t. C. Consider the following cases.

1. There is a child block of B attached above v: Let B'' be such block with the highest attachment vertex. Then this attachment vertex is w'. The back edge coming into w'' may come from block B'' or from a different child block. Assume first that it comes from a different child block $B' \neq B''$. So the back edge(s) (r, r') coming from B'' must go higher than w'', but no higher than p, i.e. $r \in (w'', p]$. Assume that the attachment vertex s for block B' is below v (the other case is similar). See (a) in Fig. 1. By Claim 1, edge e' belongs to the solution. It is easy to see that $[w', w'']$ is a chordal path in the cycle $s \overset{T}{-} x - w'' \overset{T}{-} r - r' \overset{T}{-} w' \overset{T}{-} s$. By Lemma 8 we can delete tree edge (q, w'), preserving the feasibility.

   Now we argue that part (2) of the Block Property is maintained: for block B' the path through the ancestor vertices is $w'' \overset{T}{-} p \overset{A}{-} u - v$ and follows tree edges to the parent vertex of B'; for block B'' the path is $r \overset{T}{-} p \overset{A}{-} u - v \overset{T}{-} w'$ and follows tree edges to the parent vertex of B''; for any other child block of B the argument uses the way w', w'' were chosen (this gives that the attachment vertex of any child block is below w' and the back edge goes into $(w'', p]$) and the argument is similar. The path $p \overset{A}{-} u$ exists by the induction assumption on the Block Property for B. And $s \overset{T}{-} x$, $r' \overset{T}{-} w'$ exist, since the child blocks were not touched yet.

   The other case to consider is when the back edge coming into w'' comes from block $B' = B''$. This case is omitted, since it can be done in a similar way. See (b) in Fig. 1.

2. All child blocks of B are attached below or at v: then $v = w'$. Let e' be the back edge coming from a child block B' of B and such that e' goes into the lowest vertex in $(v, p]$. Then the lowest vertex is exactly w''. Let the other end of e' be x. Let also s be the attachment vertex for block B'. By Claim 1 we know that e' is in the current solution graph. See (c) in Fig. 1.

   By the induction assumption on the Block Property for B, path $p \overset{A}{-} u$ exists. And $s \overset{T}{-} x$ exists, since the child blocks were not touched yet. Now $[w', w'']$ is a chordal path in the cycle $v \overset{T}{-} s \overset{T}{-} x - w'' \overset{T}{-} p \overset{A}{-} u - v$. By Lemma 8, tree edge (q, w') can be deleted.

   Part (2) of the Block Property is now true for B', since the path is $w'' \overset{T}{-} p \overset{A}{-} u - v$ and then the path follows tree edges into the parent vertex of B'. For the other child blocks, B decided to retain the choice of the first phase. The argument that the changes maintain the Block Property for these blocks is similar.

This concludes the proof of the induction step as well as the proof of Lemma 10. ■

Lemma 11 *The set I of MARKED vertices is an independent set in G_i. Also all the vertices in I have the requirement of* 2.

Proof: Let us focus on case 1 of the second phase. Only in this case s was chosen into I, and it had $\hat{r}^i_s = 2$. By the definition of w', $d_T(s) = 2$. By the choice of w'', there is no back edge into s from any child block of B. s has no tree edge from any child block, and so has no edge from any child block of B. Therefore, by Lemma 7, s has no edge from any offspring block.

Notice that $s \in I$ must be in the current block B. This is because s cannot be the parent vertex of B, since it has no back edge into it from any child block. By Lemma 9 the parent vertex p has such back edge. So, s must be below p but also above v, and so s is in B. Also, all the blocks are disjoint. Therefore, a block can have at most one vertex in I. We know also that a vertex in I has no edge from any descendant block of the block that contains that vertex. This means that there are no edges between vertices in I. ■

Theorem 1 *The above algorithm is a linear time $3/2$-approximation algorithm for the unweighted $\{1,2\}$-VC problem.*

Proof: Correctness of the algorithm follows from the discussion in Section 2.1, Lemmas 2 and 10. We now prove the approximation guarantee of the phase-1, 2 algorithm in case of a 2-VC subproblem on G_i. Notice that if a block was not labeled MARKED, then some tree edge was deleted from the block, so it was labeled FREE. And so the number of picked edges is at most the number of edges of the DFS tree, plus the number of blocks labeled MARKED, which is $|I|$. Thus the size of the solution is at most $n_i - 1 + |I|$. By Lemma 11, I is an independent set in G_i of requirement two vertices. Thus, by Lemma 4, we have $opt(G_i) \geq 2|I|$ and also $opt(G_i) \geq n_i$. Putting these together gives that the number of edges is bounded by $n_i - 1 + |I| \leq opt(G_i) - 1 + \frac{1}{2}opt(G_i) \leq \frac{3}{2}opt(G_i)$. The algorithm can easily be implemented to run in linear time. ■

2.3 Algorithms for $\{1,2\}$-EC

2.3.1 Application of the previous algorithm

We apply the algorithm of Section 2.2 to obtain a 3/2-approximation algorithm for $\{1,2\}$-EC. By the decomposition in Section 2.1 (and Section 2.1.2), we can focus on a 2-VC subproblem G_i with r^i defined in Section 2.1.2, and assume that there is a vertex pair $u, v \in V(G_i)$ with $r^i_{uv} = 2$. This subproblem is $\{1,2\}$-EC problem on a 2-VC graph G_i.

We run on G_i the phase-$1, 2$ of the previous algorithm w.r.t. $\{1, 2\}$-VC (r^i is treated as VC requirements on G_i). By Theorem 1 the algorithm produces a $3/2$-approximate solution to $\{1, 2\}$-VC on G_i. Since the lower bound in Lemma 4 also applies and the solution is feasible to $\{1, 2\}$-EC, this gives a $3/2$-approximate solution for $\{1, 2\}$-EC.

2.3.2 Simple algorithm

We show now that a simple modification of the algorithm of Khuller and Vishkin [11] leads to a $3/2$-approximation algorithm for $\{1, 2\}$-EC. Let $G = (V, E)$ be a given instance of $\{1, 2\}$-EC with $r_{uv} \in \{1, 2\}$ for any vertex pair $u, v \in V$. We find a DFS spanning tree of G and keep all the tree edges in our solution subgraph H. Whenever the DFS backs-up over a tree edge e, we check whether e is a cut-edge of our current H (i.e. none of the back edges in H covers e). If yes, and if the cut $(S, \bar{S})$ given by e separates some vertex pair x, y with $r_{xy} = 2$, then we add the farthest going back edge that covers e into H. Also, we "mark" the cut $(S, \bar{S})$. Here S means the vertex set of the subtree below e, and $\{x, y\}$ is separated by $(S, \bar{S})$ if S has exactly one of x, y.

To show the approximation guarantee, we use the following simple analysis. The number of tree edges in H is at most $n - 1$ which is at most $opt(G)$. Also, the number of the back edges in H is equal to the number of "marked" cuts $(S, \bar{S})$. Because of the DFS tree property and the way back edges were chosen, any two such cuts are edge disjoint. Therefore the optimal solution to the problem must have at least 2 edges in each of these cuts. This gives that the number of these cuts is at most $\frac{1}{2}opt(G)$. Thus finally the size of the solution is at most $\frac{3}{2}opt(G)$.

2.4 Local optimization heuristics

2.4.1 General Local Optimization Heuristic

Let Π be a minimization problem on $G = (V, E)$, where we want to find a spanning subgraph of G with minimum number of edges and which is feasible for (or w.r.t.) problem Π. Given a positive integer j let us define the *j-opt heuristic* as the algorithm which given any feasible solution $H \subseteq G$ to problem Π, repeats, if possible, the following operation:

- if there are subsets $E_0 \subseteq E \setminus E(H), E_1 \subseteq E(H)$ ($|E_0| \leq j$, $|E_1| > |E_0|$) such that $(H \setminus E_1) \cup E_0$ is feasible w.r.t. Π, then set $H \leftarrow (H \setminus E_1) \cup E_0$.

The algorithm outputs H, if it cannot perform any more of such operations on H. We say that such output solution is *j-opt* w.r.t. Π. Note that if $|E_0| = 0$, then the operation is equivalent to deleting edges E_1 from the current solution, preserving the feasibility w.r.t. Π. If $|E_0| = j$, then we call the operation above a *j-opt exchange*. If j is a fixed constant, the algorithm can be implemented to run in polynomial time.

Let $G = (V, E)$ be a given instance of the $\{1,2\}$-VC(EC) problem. We use the decomposition from section 2.1 for $\{1,2\}$-VC and $\{1,2\}$-EC. By Lemmas 2 and 3 it suffices to consider a 2-VC subproblem G_i of G, where the requirement function r^i was defined for each of the problems in section 2.1, and we assume that there is a vertex pair $u, v \in V(G_i)$ with $r^i_{uv} = 2$.

Let $\mathcal{E}$ be an ear decomposition of a 2-connected graph. Let $\mathcal{E}' \subseteq \mathcal{E}$ be a subset of ears of the ear decomposition $\mathcal{E}$. We say that set $\mathcal{E}'$ is *terminal* in $\mathcal{E}$ if: (1) every ear in $\mathcal{E}'$ is a pendant ear of $\mathcal{E}$, (2) for every pair of ears $S, T \in \mathcal{E}'$ there is no edge between an internal vertex of S and an internal vertex of T, and (3) every ear in $\mathcal{E}'$ is open.

Let $G = (V, E)$ be a given instance of the $\{1,2\}$-VC(EC)-SNDP problem. We use the decomposition from Section 2.1 for $\{1,2\}$-VC and $\{1,2\}$-EC. By Lemmas 2 and 3 it suffices to consider a 2-VC subproblem G_i of G, where the requirement function r^i was defined for each of the problems in Section 2.1, and we assume that there is a vertex pair $u, v \in V(G_i)$ with $r^i_{uv} = 2$. For simplicity, in what follows we drop the subscript and superscript i from G_i, n_i and from $r^i, \hat{r}^i$.

2.4.2 Local optimization heuristic for $\{1,2\}$-VC

The local optimization algorithm for $\{1,2\}$-VC is as follows. Let H be any 2-VC spanning subgraph of G (e.g. $H = G$). First we run the 1-opt heuristic on H w.r.t. 2-vertex connectivity. Let H' be the output 2-vertex connected spanning subgraph of G, i.e. H' is 1-opt w.r.t. 2-VC. Now, compute an open ear decomposition $\mathcal{E}$ of H' (see Proposition 1). Notice, that $\mathcal{E}$ does not contain 1-ears, since they are redundant w.r.t. 2-VC and therefore were removed by the 1-opt heuristic. Let $\mathcal{E}_2$ be the set of all 2-ears in $\mathcal{E}$. Let also $\mathcal{E}_2 = \mathcal{P}_1 \cup \mathcal{P}_2$, where a 2-ear $S \in \mathcal{P}_1$ iff for the internal vertex s of S we have $\hat{r}_s = 1$, and $\mathcal{P}_2 = \mathcal{E}_2 \setminus \mathcal{P}_1$. The second step of the algorithm is: for any 2-ear $S \in \mathcal{P}_1$, remove one of the (two) edges of S, from the current solution H'. Output the final H' as the solution. This finishes the description of the algorithm.

The main idea to prove the lower bound below is to show that $\mathcal{E}_2$ is terminal in $\mathcal{E}$.

Lemma 12 *The optimal solution to the* $\{1,2\}$*-VC problem on* G *fulfills the following* $opt(G) \geq |\mathcal{P}_1| + 2|\mathcal{P}_2|$.

Proof: Note, that H' is 1-opt w.r.t. 2-VC. We first prove the following claim.
Claim. $\mathcal{E}_2$ is terminal in $\mathcal{E}$.
Proof. Assume that $\mathcal{E}_2$ is not terminal. Since every ear is open, either (1) or (2) in the definition of a terminal set is not true.
Assume that: (1) is not true. Then there is a non pendant 2-ear $S \in \mathcal{E}_2$ with the internal vertex s and edges $e_1 = (s, s_1), e_2 = (s, s_2)$, and there must be an ear $T \in \mathcal{E}$ such that s is one of the end vertices of T. But then it is easy to see that one of e_1, e_2 is redundant depending on what is the second end vertex of T: e.g. if s, s_1 are end vertices of T, then

deleting e_1 from H' and choosing T together with e_2 as a new ear gives a new open ear decomposition (0-opt exchange). So $H' \setminus e_1$ is 2-VC by Proposition 1 – contradiction with 1-optimality of H'. The argument is similar if s, s_2 are end vertices of T or if none of s_1, s_2 is an end vertex of T.
Assume that: (2) is not true. Then there are two 2-ears $S, T \in \mathcal{E}_2$ with the internal vertices s, t and edges $e_1 = (s, s_1), e_2 = (s, s_2)$, and $\tilde{e}_1 = (t, t_1), \tilde{e}_2 = (t, t_2)$, respectively. And there is an edge (s, t). It can easily be seen that it is always possible to delete some edge in S and some in T, and add (s, t) to get a new open 3-ear from S, T. We omit here a simple case analysis to see that. This gives a new open ear decomposition, and so a new 2-VC solution (Proposition 1). This is a contradiction, since no 1-opt exchange is possible in H'. ■

By Claim above, we see that set of all internal vertices in 2-ears forms an independent set in G, so by a similar argument as in the proof of Lemma 4, it is obvious that $|\mathcal{P}_1| + 2|\mathcal{P}_2|$ is a lower bound. ■

Theorem 2 *The described algorithm is a local search based* $7/4$*-approximation algorithm for the unweighted* $\{1, 2\}$*-VC problem.*

Proof: *Feasibility.* Let P_1 be the set of all internal vertices in the 2-ears in $\mathcal{P}_1$. After removing some of the edges from the 2-ears in $\mathcal{P}_1$, the original ear decomposition $\mathcal{E}$ implies an open ear decomposition of the graph $G \setminus P_1$. Thus $G \setminus P_1$ is 2-VC by Proposition 1. Also, the vertices in P_1 are not disconnected from $G \setminus P_1$.
Approximation guarantee. Notice that $|E(H')| \leq |\mathcal{P}_1| + 2|\mathcal{P}_2| + \frac{3}{2}(n - |\mathcal{P}_1| - |\mathcal{P}_2|)$, where $|\mathcal{P}_1| + 2|\mathcal{P}_2|$ is the number of edges of the output solution that are in 2-ears (recall that for any 2-ear in $\mathcal{P}_1$ we have picked just one of its edges), and $n - |\mathcal{P}_1| - |\mathcal{P}_2|$ is the number of all other internal vertices lying in the ears of length ≥ 3. The factor $\frac{3}{2}$ is the worst case ratio of the number of edges to the number of internal vertices in an ℓ-ear in $\mathcal{E} \setminus \mathcal{E}_2$ ($\ell \geq 3$). By Lemma 12, $opt(G) \geq |\mathcal{P}_1| + 2|\mathcal{P}_2|$. And using Lemma 4, $opt(G) \geq n$. Therefore, we have $|E(H')| \leq |\mathcal{P}_1| + 2|\mathcal{P}_2| + \frac{3}{2}(n - |\mathcal{P}_1| - |\mathcal{P}_2|) = \frac{3}{2}n - \frac{1}{2}|\mathcal{P}_1| + \frac{1}{2}|\mathcal{P}_2| \leq \frac{3}{2}opt(G) + \frac{1}{4}opt(G)$ $= \frac{7}{4}opt(G)$. ■

Remark. Theorem 2 generalizes a result for the 2-VC problem communicated by J. Cheriyan.

2.4.3 Local optimization heuristic for $\{1, 2\}$-EC

Using the algorithm from Section 6.1 and similar ideas as those in Section 2.3, we can give a local optimization heuristic for the $\{1, 2\}$-EC with an approximation guarantee of $7/4$.

We present here a special version of the local search heuristic based on ideas of Cheriyan *et al.* [3]. We modify their approach to give a local search based $5/3$-approximation algorithm for $\{1, 2\}$-EC.

For a given ear decomposition $\mathcal{E}$, let $\mathcal{E}_\ell$ denote the set of all ℓ-ears in $\mathcal{E}$, and $V(\mathcal{E}_\ell)$ be the set of all internal vertices of the ears in $\mathcal{E}_\ell$.

Let H be a 2-VC spanning subgraph of G. Compute an open ear decomposition $\mathcal{F}$ of H. We show how to transform $\mathcal{F}$ into a new ear decomposition $\mathcal{E}$ (not necessarily open) such that the set $\mathcal{E}_2 \cup \mathcal{E}_3$ is terminal in $\mathcal{E}$. $\mathcal{E}$ will constitute a spanning subgraph of G. We will use a special version of 0-opt and 1-opt exchanges w.r.t. 2-EC. One can assume that $\mathcal{F}$ does not contain 1-ears.

The proof is by induction on the number k of ears in $\mathcal{F} = \{Q_0, Q_1, \ldots, Q_k\}$. If $k = 1$, the claim is clear. Assume that the result holds for $i-1$ ears $\mathcal{F}' = \{Q_0, Q_1, \ldots, Q_{i-1}\}$, and let $\mathcal{E}' = \{Q'_0, Q'_1, \ldots, Q'_{j-1}\}$ be the corresponding ear decomposition with $\mathcal{E}'_2 \cup \mathcal{E}'_3$ terminal in $\mathcal{E}'$.

Step 1. Consider the next (open) ℓ'-ear Q_i from $\mathcal{F}$. Let $Q_i = v_1, v_2, v_3, \ldots, v_{\ell'+1}$. We want to "add" Q_i to $\mathcal{E}'$. If the end vertices $v_1, v_{\ell'+1}$, of Q_i are such that $v_1, v_{\ell'+1} \notin V(\mathcal{E}'_2) \cup V(\mathcal{E}'_3)$, then we set $Q'_j = Q_i$. Otherwise, assume that $v_1, v_{\ell'+1}$ are the internal vertices of some ears in $\mathcal{E}'_2 \cup \mathcal{E}'_3$.

We consider first a case when v_1 is an end vertex of a 2-ear $S \in \mathcal{E}'_2$, and $v_{\ell'+1} \notin V(\mathcal{E}'_2) \cup V(\mathcal{E}'_3)$. Assume that $S = w_1, v_1, w_3$. See (a) in Figure 2. Define a new $(\ell'+1)$-ear $Q'_j = Q_i + e$, where e is one of the edges of S. The other edge of S is deleted. e can be chosen so that Q'_j is open: if $w_3 \neq v_{\ell'+1}$, then $e = (w_3, v_1)$ and $Q'_j = w_3, v_1, v_2, \ldots, v_{\ell'+1}$; otherwise, if $w_3 = v_{\ell'+1}$, then $e = (w_1, v_1)$ and $Q'_j = w_1, v_1, v_2, \ldots, v_{\ell'+1}$. This is a 0-opt exchange.

Assume now that v_1 is an end vertex of a 3-ear $S \in \mathcal{E}'_3$, and $v_{\ell'+1} \notin V(\mathcal{E}'_2) \cup V(\mathcal{E}'_3)$. Let $S = w_1, v_1, w_3, w_4$. See (b) in Figure 2. In this case we define a new $(\ell'+2)$-ear $Q'_j = Q_i + (v_1, w_3) + (w_3, w_4) = w_4, w_3, v_1, v_2, \ldots, v_{\ell'+1}$, and edge (w_1, v_1) is deleted. Notice that if $w_4 = v_{\ell'+1}$, then the new ear Q'_j is not open. This does not violate condition (1) in the definition of the terminal set, since Q'_j has length at least 4. This is a 0-opt exchange.

Set $\mathcal{E}'$ to be $\mathcal{E}' \cup \{Q'_j\}$. This step forces condition (1) in the definition of the terminal set.

Step 2. In this step we fulfill condition (2) for the terminal set. Consider the ear decomposition $\mathcal{E}' = \{Q'_0, Q'_1, \ldots, Q'_j\}$ defined by Step 1. If there is an edge $e = (v, w) \in E(G) \setminus E(\mathcal{F})$ such that $v, w \in V(\mathcal{E}'_2) \cup V(\mathcal{E}'_3)$, then we show that we can perform a 1-opt exchange to avoid such edge e.

Assume first that v and w are internal vertices of some open 2-ears $S, T \in \mathcal{E}'_2$, then we can delete an edge of S and an edge of T and add e to get a new open 3-ear T' in $\mathcal{E}'$: $\mathcal{E}'$ is set to $(\mathcal{E}' \setminus \{S, T\}) \cup \{T'\}$. In detail, we proceed as follows. Let $S = v_1, v, v_3$, $T = w_1, w, w_3$. W.l.o.g. we can assume that $v_1 \neq w_3$. Then we delete edges (v, v_3) of S and (w, w_1) of T. The new 3-ear $T' = v_1, v, w, w_3$ is then open. Finally, $\mathcal{E}'$ is set to $(\mathcal{E}' \setminus \{S, T\}) \cup \{T'\}$. See (a) and (b) in Figure 3 for illustrations.

Assume now that e joins some two $2, 3$-ears at least one of whom is a 3-ear. We show

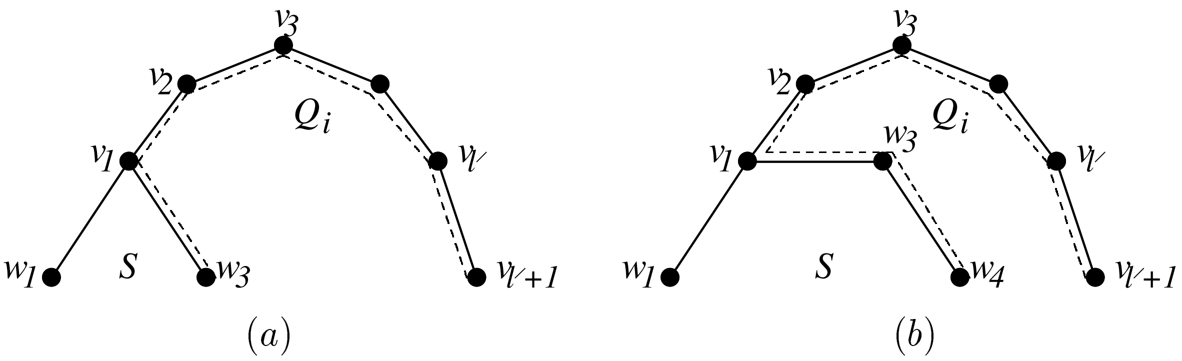

Figure 2: Cases for Step 1 of the local optimization based algorithm. Solid thick lines represent the ears, dashed thin lines show a new ear Q'_j in each of the two cases. In case (a) if $w_3 = v_{\ell'+1}$, then the new ear will be $w_1, v_1, v_2, \ldots, v_{\ell'+1}$ instead of $w_3, v_1, v_2, \ldots, v_{\ell'+1}$.

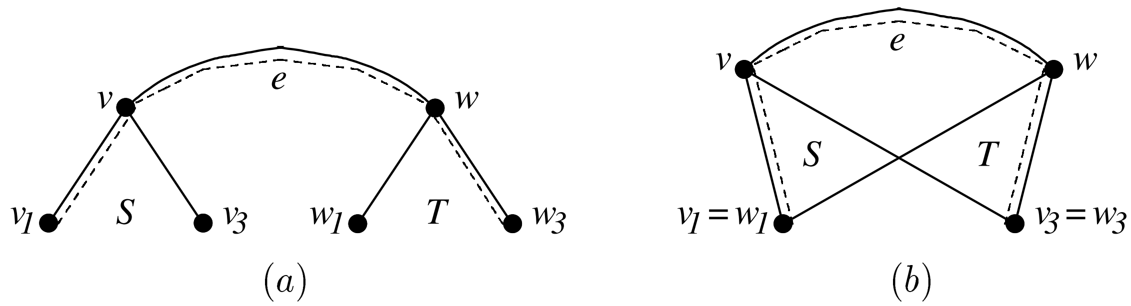

Figure 3: Some cases for Step 2 of the local optimization based algorithm. Solid thick lines represent the ears and edge e, dashed thin lines show a new open 3-ear T' in each of the two cases. In each case we delete edges (v, v_3) and (w, w_1).

one of the cases here when $S \in \mathcal{E}'_2$ and $T \in \mathcal{E}'_3$, and $S = v_1, v, v_3, T = w_1, w, w_3, w_4$. In this case delete edge (v_1, v) (or (v_3, v)) and (w_1, w). The new ear T' is $T' = v_3, v, w, w_3, w_4$ (or $T' = v_1, v, w, w_3, w_4$). Observe that T' might not be open, but its length is at least 4. Thus the conditions for the terminal set are maintained. We then set $\mathcal{E}'$ to be $(\mathcal{E}' \setminus \{S, T\}) \cup \{T'\}$. Other cases, when e joins two 2-ears or two 3-ears are quite similar, and we omit them. Condition (3) of the terminal set is also maintained. This finishes the induction step.

Let $\mathcal{E}$ be the final ear decomposition obtained from $\mathcal{F}$ by the above algorithm, and H' be the graph corresponding to $\mathcal{E}$. Let $\mathcal{P}_1, \mathcal{P}_2$ be as defined in Section 6.1. Also $\mathcal{E}_3 = \mathcal{Q}_1 \cup \mathcal{Q}_2$, where a 3-ear $S \in \mathcal{Q}_1$ iff for the internal vertices s_1, s_2 of S we have $\hat{r}_{s_1} = \hat{r}_{s_2} = 1$, and $\mathcal{Q}_2 = \mathcal{E}_3 \setminus \mathcal{Q}_1$.

For any 2-ear $S \in \mathcal{P}_1$, remove one of the edges of S, from the current solution H'. For any 3-ear $S \in \mathcal{Q}_1$, remove one of the (three) edges of S, from the current solution H'. Output the final H' as the solution. This concludes the description of the algorithm.

To prove the following lower bound we observe that $\mathcal{E}_2 \cup \mathcal{E}_3$ is terminal in $\mathcal{E}$.

Lemma 13 *For the* $\{1,2\}$*-EC problem on* G *we have* $opt(G) \geq |\mathcal{P}_1| + 2|\mathcal{P}_2| + 2|\mathcal{Q}_1| + 3|\mathcal{Q}_2|$.

Proof: It obviously follows from Step 1 and 2 of the algorithm that for the final ear decomposition $\mathcal{E}$, $\mathcal{E}_2 \cup \mathcal{E}_3$ will be terminal in $\mathcal{E}$. (This was the way we were performing these steps).

The $2|\mathcal{P}_2| + 3|\mathcal{Q}_2|$ part of the lower bound follows from "terminality", and from the fact that each internal vertex of any ear in $\mathcal{P}_2 \cup \mathcal{Q}_2$ has a requirement of 2, and so its degree in any feasible solution must be at least 2. Similar argument applies to the part $|\mathcal{P}_1| + 2|\mathcal{Q}_1|$ of the lower bound, but now we argue about internal vertices of requirement 1. ■

Based on Lemma 13, we prove now the approximation guarantee of the given algorithm.

Theorem 3 *The described algorithm is a local search based* $5/3$*-approximation algorithm for the unweighted* $\{1,2\}$*-EC problem.*

Proof: We have now the following estimate $|E(H')| \leq |\mathcal{P}_1| + 2|\mathcal{P}_2| + 2|\mathcal{Q}_1| + 3|\mathcal{Q}_2| + \frac{4}{3}(n - |\mathcal{P}_1| - |\mathcal{P}_2| - 2|\mathcal{Q}_1| - 2|\mathcal{Q}_2|)$. Here, $|\mathcal{P}_1| + 2|\mathcal{P}_2| + 2|\mathcal{Q}_1| + 3|\mathcal{Q}_2|$ is the number of edges of the output solution that are in 2-ears and in 3-ears, and $n - |\mathcal{P}_1| - |\mathcal{P}_2| - 2|\mathcal{Q}_1| - 2|\mathcal{Q}_2|$ is the number of all internal vertices lying in the ears of length ≥ 4.

By Lemma 13, $opt(G) \geq |\mathcal{P}_1| + 2|\mathcal{P}_2| + 2|\mathcal{Q}_1| + 3|\mathcal{Q}_2|$. We also have $opt(G) \geq n$. Thus, $|E(H')| \leq \frac{4}{3}n + \frac{2}{3}|\mathcal{P}_2| + \frac{1}{3}|\mathcal{Q}_2| = \frac{4}{3}n + \frac{1}{3}(2|\mathcal{P}_2| + |\mathcal{Q}_2|) \leq \frac{4}{3}opt(G) + \frac{1}{3}opt(G) = \frac{5}{3}opt(G)$. ■

3 Conclusions

We have presented approximation algorithms for the unweighted $\{1,2\}$-VC and $\{1,2\}$-EC problems, based on depth-first-search methods and on local search heuristics. The depth-first-search based algorithm for $\{1,2\}$-VC gives a $3/2$-approximation, but the algorithm and its analysis is somehow complicated. The main feature of our local search algorithm for $\{1,2\}$-VC is its simlicity, but it gives a worse approximation guarantee. For $\{1,2\}$-EC we have given a very simple depth-first-search algorithm with a simple analysis. The performance guarantees of the depth-first-search algorithms are tight with respect to the lower bounds we use.

Acknowledgments. The author is very grateful to Joseph Cheriyan who has invited him to Waterloo. He would like to thank Joseph for many fruitful discussions, help and for his support. Also, many thanks to the Department of Combinatorics and Optimization at the University of Waterloo, for its hospitality.

References

[1] E. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Chichester, 1997.

[2] J. Cheriyan and R. Thurimella. Approximating Minimum-Size k-Connected Spanning Subgraphs via Matching. In the *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS'96)*, pp. 292–301, 1996.

[3] J. Cheriyan, A. Sebő and Z. Szigeti. An Improved Approximation Algorithm for Minimum Size 2-Edge Connected Spanning Subgraphs. In the *Proc. 6th International Integer Programming and Combinatorial Optimization Conference (IPCO'98)*, LNCS **1412**, pp. 126–136, 1998.

[4] C.G. Fernandes. A better approximation ratio for the minimum size k-edge-connected spanning subgraph problem. *Journal of Algorithms*, **28**, pp. 105–124, 1998. Preliminary version in *Proc. 8th Annual ACM-SIAM SODA*, 1997.

[5] N. Garg, V. Santosh and A. Singla. Improved Approximation Algorithms for Biconnected Subgraphs via Better Lower Bounding Techniques. In the *Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'93)*, pp. 103–111, 1993.

[6] M.X. Goemans, A. Goldberg, S. Plotkin, D.B. Shmoys, È. Tardos and D.P. Williamson. Improved Approximation Algorithms for Network Design Problems. In the *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94)*, pp. 223–232, 1994.

[7] M.X. Goemans and D.P. Williamson. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, **24**, pp. 296–317, April 1995.

[8] M. Grötschel, C. Monma and M. Stoer. Design of survivable networks. In *Handbook in Operations Research and Management Science*, Volume on Networks. North-Holland, Amsterdam, 1995.

[9] K. Jain. A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem. In the *Proc. of the IEEE FOCS*, 1998.

[10] S. Khuller. Approximation algorithms for finding highly connected spanning subgraphs. In *Approximation algorithms for NP-hard problems*, Ed. D.S. Hochbaum. PWS publishing Co., Boston, 1996.

[11] S. Khuller and U. Vishkin. Biconnectivity Approximations and Graph Carvings. *Journal of the ACM*, **41**(2), pp. 214–235, 1994. Preliminary version in *Proc. 24th Annual ACM STOC*, 1992.

[12] H. Nagamochi and T. Ibaraki. A linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. *Algorithmica*, **7**, pp. 583–596, 1992.

[13] Z. Nutov. Approximating multiroot 3-outconnected subgraphs. In the *Proc. 10th ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, pp. 951-952, 1999.

[14] R. Ravi and D.P. Williamson. An Approximation Algorithm for Minimum-Cost Vertex-Connectivity Problems. *Algorithmica*, **18**(1), pp. 21–43, 1997.

[15] S. Vempala and A. Vetta. Factor 4/3 Approximations for Minimum 2-Connected Subgraphs. In the *Proc. 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2000).*

[16] D.P. Williamson, M.X. Goemans, M. Mihail and V.V. Vazirani. A Primal-Dual Approximation Algorithm for Generalized Steiner Network Problems. *Combinatorica*, **15**, pp. 435–454, 1995. Preliminary version in *Proc. 25th Annual ACM STOC*, 1993.

I N F O R M A T I K

Below you find a list of the most recent technical reports of the Max-Planck-Institut für Informatik. They are available by anonymous ftp from `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Anja Becker
Stuhlsatzenhausweg 85
66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-2000-4-003	S.W. Choi, H. Seidel	Hyperbolic Hausdorff Distance for Medial Axis Transform
MPI-I-2000-4-002	L.P. Kobbelt, S. Bischoff, K. Kähler, R. Schneider, M. Botsch, C. Rössl, J. Vorsatz	Geometric Modeling Based on Polygonal Meshes
MPI-I-2000-4-001	J. Kautz, W. Heidrich, K. Daubert	Bump Map Shadows for OpenGL Rendering
MPI-I-2000-2-001	F. Eisenbrand	Short Vectors of Planar Lattices Via Continued Fractions
MPI-I-2000-1-005	M. Seel, K. Mehlhorn	Infimaximal Frames A Technique for Making Lines Look Like Segments
MPI-I-2000-1-004	K. Mehlhorn, S. Schirra	Generalized and improved constructive separation bound for real algebraic expressions
MPI-I-2000-1-003	P. Fatourou	Low-Contention Depth-First Scheduling of Parallel Computations with Synchronization Variables
MPI-I-2000-1-002	R. Beier, J. Sibeyn	A Powerful Heuristic for Telephone Gossiping
MPI-I-2000-1-001	E. Althaus, O. Kohlbacher, H. Lenhof, P. Müller	A branch and cut algorithm for the optimal solution of the side-chain placement problem
MPI-I-1999-4-001	J. Haber, H. Seidel	A Framework for Evaluating the Quality of Lossy Image Compression
MPI-I-1999-3-005	T.A. Henzinger, J. Raskin, P. Schobbens	Axioms for Real-Time Logics
MPI-I-1999-3-004	J. Raskin, P. Schobbens	Proving a conjecture of Andreka on temporal logic
MPI-I-1999-3-003	T.A. Henzinger, J. Raskin, P. Schobbens	Fully Decidable Logics, Automata and Classical Theories for Defining Regular Real-Time Languages
MPI-I-1999-3-002	J. Raskin, P. Schobbens	The Logic of Event Clocks
MPI-I-1999-3-001	S. Vorobyov	New Lower Bounds for the Expressiveness and the Higher-Order Matching Problem in the Simply Typed Lambda Calculus
MPI-I-1999-2-008	A. Bockmayr, F. Eisenbrand	Cutting Planes and the Elementary Closure in Fixed Dimension
MPI-I-1999-2-007	G. Delzanno, J. Raskin	Symbolic Representation of Upward-closed Sets
MPI-I-1999-2-006	A. Nonnengart	A Deductive Model Checking Approach for Hybrid Systems
MPI-I-1999-2-005	J. Wu	Symmetries in Logic Programs
MPI-I-1999-2-004	V. Cortier, H. Ganzinger, F. Jacquemard, M. Veanes	Decidable fragments of simultaneous rigid reachability
MPI-I-1999-2-003	U. Waldmann	Cancellative Superposition Decides the Theory of Divisible Torsion-Free Abelian Groups
MPI-I-1999-2-001	W. Charatonik	Automata on DAG Representations of Finite Trees
MPI-I-1999-1-007	C. Burnikel, K. Mehlhorn, M. Seel	A simple way to recognize a correct Voronoi diagram of line segments
MPI-I-1999-1-006	M. Nissen	Integration of Graph Iterators into LEDA

MPI-I-1999-1-005	J.F. Sibeyn	Ultimate Parallel List Ranking ?
MPI-I-1999-1-004	M. Nissen, K. Weihe	How generic language extensions enable “open-world” desing in Java
MPI-I-1999-1-003	P. Sanders, S. Egner, J. Korst	Fast Concurrent Access to Parallel Disks
MPI-I-1999-1-002	N.P. Boghossian, O. Kohlbacher, H.-. Lenhof	BALL: Biochemical Algorithms Library
MPI-I-1999-1-001	A. Crauser, P. Ferragina	A Theoretical and Experimental Study on the Construction of Suffix Arrays in External Memory
MPI-I-98-2-018	F. Eisenbrand	A Note on the Membership Problem for the First Elementary Closure of a Polyhedron
MPI-I-98-2-017	M. Tzakova, P. Blackburn	Hybridizing Concept Languages
MPI-I-98-2-014	Y. Gurevich, M. Veanes	Partisan Corroboration, and Shifted Pairing
MPI-I-98-2-013	H. Ganzinger, F. Jacquemard, M. Veanes	Rigid Reachability
MPI-I-98-2-012	G. Delzanno, A. Podelski	Model Checking Infinite-state Systems in CLP
MPI-I-98-2-011	A. Degtyarev, A. Voronkov	Equality Reasoning in Sequent-Based Calculi
MPI-I-98-2-010	S. Ramangalahy	Strategies for Conformance Testing
MPI-I-98-2-009	S. Vorobyov	The Undecidability of the First-Order Theories of One Step Rewriting in Linear Canonical Systems
MPI-I-98-2-008	S. Vorobyov	AE-Equational theory of context unification is Co-RE-Hard
MPI-I-98-2-007	S. Vorobyov	The Most Nonelementary Theory (A Direct Lower Bound Proof)
MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid *E*-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-98-2-003	R.A. Schmidt	E-Unification for Subsystems of S4
MPI-I-98-2-002	F. Jacquemard, C. Meyer, C. Weidenbach	Unification in Extensions of Shallow Equational Theories
MPI-I-98-1-031	G.W. Klau, P. Mutzel	Optimal Compaction of Orthogonal Grid Drawings
MPI-I-98-1-030	H. Brönniman, L. Kettner, S. Schirra, R. Veltkamp	Applications of the Generic Programming Paradigm in the Design of CGAL
MPI-I-98-1-029	P. Mutzel, R. Weiskircher	Optimizing Over All Combinatorial Embeddings of a Planar Graph
MPI-I-98-1-028	A. Crauser, K. Mehlhorn, E. Althaus, K. Brengel, T. Buchheit, J. Keller, H. Krone, O. Lambert, R. Schulte, S. Thiel, M. Westphal, R. Wirth	On the performance of LEDA-SM
MPI-I-98-1-027	C. Burnikel	Delaunay Graphs by Divide and Conquer
MPI-I-98-1-026	K. Jansen, L. Porkolab	Improved Approximation Schemes for Scheduling Unrelated Parallel Machines
MPI-I-98-1-025	K. Jansen, L. Porkolab	Linear-time Approximation Schemes for Scheduling Malleable Parallel Tasks
MPI-I-98-1-024	S. Burkhardt, A. Crauser, P. Ferragina, H. Lenhof, E. Rivals, M. Vingron	q-gram Based Database Searching Using a Suffix Array (QUASAR)
MPI-I-98-1-023	C. Burnikel	Rational Points on Circles
MPI-I-98-1-022	C. Burnikel, J. Ziegler	Fast Recursive Division
MPI-I-98-1-021	S. Albers, G. Schmidt	Scheduling with Unexpected Machine Breakdowns