Available at
www.ElsevierMathematics.com
POWERED BY SCIENCE DIRECT®
Discrete Applied Mathematics 137 (2004) 311–341
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Cardinality constrained minimum cut problems: complexity and algorithms

Maurizio Bruglieri[a,1], Francesco Maffioli[a], Matthias Ehrgott[b]

[a] *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy*

[b] *Department of Engineering Science, University of Auckland, Private Bag 92019, Auckland, New Zealand*

Received 27 March 2002; received in revised form 23 May 2003; accepted 26 May 2003

Abstract

In several applications the solutions of combinatorial optimization problems (*COP*) are required to satisfy an additional cardinality constraint, that is to contain a fixed number of elements. So far the family of (*COP*) with cardinality constraints has been little investigated. The present work tackles a new problem of this class: the k-cardinality minimum cut problem (k-card cut). For a number of variants of this problem we show complexity results in the most significant graph classes. Moreover, we develop several heuristic algorithms for the k-card cut problem for complete, complete bipartite, and general graphs. Lower bounds are obtained through an SDP formulation, and used to show the quality of the heuristics. Finally, we present a randomized SDP heuristic and numerical results.

Keywords: Cut problems; k-cardinality minimum cut; Cardinality constrained combinatorial optimization; Computational complexity; Semidefinite programming

1. Introduction

In recent years a number of papers have been published in which classical combinatorial optimization problems have been modified by imposing an additional cardinality

[1] The work of M.B. has been supported by Ph.D. MA.C.R.O. Grant of the Department of Mathematics "F. Enriquez", University of Milan.

E-mail addresses: bruglier@elet.polimi.it (M. Bruglieri), maffioli@elet.polimi.it (F. Maffioli), m.ehrgott@auckland.ac.nz (M. Ehrgott).

0166-218X/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0166-218X(03)00358-5

constraint, i.e. feasible solutions were constrained to contain a given number k of elements. Applications of cardinality constrained tree problems are in oil-field leasing [12] and facilities layout [14]. [8] deals with portfolio optimization, when the portfolio has to contain a fixed number of assets. A number of other problems, e.g. the assignment problem [10], have also been studied under cardinality constraints. A survey on the topic with extensive references is available [13]. A class of combinatorial optimization problems which have applications in a wide variety of areas are cut problems, i.e. the problems to find in a given graph a cut of maximal (or minimal) weight. In physics, for example, the maximum cut problem models the problem of finding a ground state of spin glasses having zero magnetization. In VLSI design, it models the problem of minimizing the number of vias (holes on a printed circuit board, or contacts on a chip), see [3]. In numerical analysis it is helpful in finding the L-U factorization of the matrix of a linear system. The minimum cut problem has applications for example in network reliability theory and in compilers for parallel languages. For some of those, the addition of a cardinality constrained might be desirable. With this motivation we set out to investigate k cardinality cut problems. This paper contains the results of our research which has been also the subject of the first author's Ph.D. thesis (see [5]). We start with some basic definitions. Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E.

Definition 1.

1. A *cut* is a partition of vertex set V in two sets V_1, V_2 called the *shores* of the cut. A *cut edge set* $C := \{\{v_1, v_2\} \in E : v_1 \in V_1, v_2 \in V_2\}$ is associated with every cut.
2. Given $s, t \in V$ an *s-t cut* is a cut (V_1, V_2) such that $s \in V_1$ and $t \in V_2$.

Since from the cut edge set C one can easily reconstruct the shores V_1, V_2, in the sequel we shall indifferently define cuts either through the shores or through the cut edge set. Let us introduce the notation $\delta(A, B)$ and $\delta(A)$ for $A, B \subset V$ as follows:

$$\delta(A, B) := \{\{v_1, v_2\} \in E : v_1 \in A, v_2 \in B\},$$

$$\delta(A) := \delta(A, \bar{A}),$$

where $\bar{A}$ denotes $V \setminus A$. Let $w : E \to \mathbb{N}$ be a non-negative integer function on the edge set of graph G. The minimum cut problem (*min cut*) and the maximum cut problem (*max cut*) are the problems to find a cut such that the sum of the weights of the cut edge set C is minimal and maximal, respectively. It will be convenient, to denote the weight of any subset of edges $F \subset E$ by

$$w(F) := \sum_{e \in F} w(e).$$

We can now introduce cardinality constrained cut problems. Let k be a positive integer.

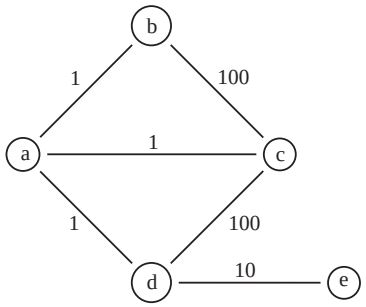

Fig. 1. Illustrating different cardinality constraints.

Definition 2.

1. The k-cardinality minimum cut problem *k-card cut* is the problem to find a cut such that cut edge set C has cardinality k and the sum of the weights of the edges belonging to C is minimal.
2. The k-cardinality minimum *s-t cut* problem *k-card s-t cut* is defined analogously to k-card cut with the additional request that the cut we want to find is an s-t cut.
3. The $\geqslant k$-*card cut* and $\geqslant k$-*card s-t cut* problems are defined analogously to k-card cut and k-card s-t cut only that the cardinality of C is required to be greater than or equal to k.

From the definitions we can make some observations. The simple example given in Fig. 1 shows that k-card cut and $\geqslant k$-card cut can have different optimal solutions. Indeed, $V_1^* = \{b\}$ is an optimal solution for k-card cut with value 101, whereas $V_1' = \{a\}$ is an optimal solution for $\geqslant k$-card cut with value 3, when $k = 2$.

However, we immediately have some relations between k-card cut and $\geqslant k$-card cut. Clearly, the optimal value of $\geqslant k$-card cut is always less than or equal to the optimal value of k-card cut. Furthermore, for every graph class for which k-card cut is easy $\geqslant k$-card cut is easy, too, because it can be solved taking the best solution of p-card cut with $p = k, k+1, \ldots, |E|$, respectively. It is important to note that without cardinality constraints the problems of Definition 2 are all easy to solve, because they become minimal cut problems and several efficient algorithms exist in the literature for solving the latter (see [22,27,31]). The example above shows that the situation is quite different due to the cardinality constraint. In Section 2 below we shall present complexity results for a number of important graph classes. The rest of the paper is organized as follows. In Section 3 we present three heuristics for complete and complete bipartite graphs. A heuristic for general graphs which is not guaranteed to find a cut of the desired cardinality (this problem is shown to be $\mathcal{NP}$-complete in Section 2) is also given. Section 4 presents lower bounds based on LP relaxation and SDP relaxation. In Section 5 a randomized SDP heuristic is given, for which we can prove lower and upper bounds on the deviation of expected cardinality from the desired k and of the weight of the cut from optimality.

2. Complexity of cardinality constrained minimum cut problems

In this section we prove some results on computational complexity of cardinality constrained cut problems. First, we shall prove $\mathcal{NP}$-hardness on general graphs. We then proceed to complete and complete bipartite graphs, for which $\mathcal{NP}$-hardness still holds. Further results are about trees, grid graphs, and planar graphs, for which our problems are polynomially, respectively randomized polynomially solvable.

2.1. General graphs

Our first result shows that even finding a feasible solution is $\mathcal{NP}$-hard.

Theorem 1. *k-card cut and k-card s-t cut are strongly $\mathcal{NP}$-hard even if $w(e)=1$ for all $e \in E$.*

Proof. We prove the result for k-card cut first. It is easy to see that the recognition version of k-card cut belongs to $\mathcal{NP}$. For proving the strong hardness we polynomially reduce *simple max cut* to k-card cut. An instance of simple max cut is an undirected graph $G=(V,E)$ where we look for a cut with the maximum number of edges. We can transform this instance into instances for k-card cut considering the same graph with weight $w(e)=1$ for all $e \in E$ and values of k between 1 and $|E|$. A solution of k-card cut for the maximum feasible value of k is also a solution of simple max cut. The proof follows from strong $\mathcal{NP}$-hardness of simple max cut (see [17, p. 210]). Solving k-card s-t cut for all pairs of vertices s,t and taking the best solution we obtain a solution for k-card cut. Hence k-card s-t cut is also strongly $\mathcal{NP}$-hard. □

The proof of Theorem 1 is not valid for classes of graphs (e.g. planar graphs) for which simple max cut belongs to $\mathcal{P}$ (see [9, p. 247]). The case of planar graphs is discussed in Section 2.6.

Analogously to Theorem 1 we can prove $\mathcal{NP}$-hardness of $\geqslant k$-card cut and $\geqslant k$-card s-t cut.

Proposition 1. *The $\geqslant k$-card cut and $\geqslant k$-card s-t cut problems are strongly $\mathcal{NP}$-hard, even if $w(e)=1$ for all $e \in E$.*

2.2. Complete graphs

For complete graphs all possible cardinalities of cuts can be described.

Lemma 1. *If $G=(V,E)$ with $|V|=n$ is a simple and complete graph k-card cut and k-card s-t cut are feasible if and only if*

$$k = j(n-j) \quad \text{with } j \in \left\{1,\ldots,\left\lfloor \frac{n}{2} \right\rfloor\right\}. \tag{1}$$

Proof. Due to completeness of G, every partition of V into V_1 and V_2 with $|V_1| = j \leqslant \lfloor n/2 \rfloor \leqslant n - j = |V_2|$ defines a cut with $k = j(n-j)$ edges. □

Thus, solving (1) with given k for j, we have that k-card cut and k-card s-t cut are infeasible, if and only if $\Delta := n^2 - 4k < 0$ or if $(n - \sqrt{\Delta})/2$ is not integer. Therefore we have:

Proposition 2. *If G is a simple and complete graph and $w(e) = 1$ for all $e \in E$ then both k-card cut and k-card s-t cut are in $\mathscr{P}$.*

As a consequence of Lemma 1, on complete graphs, the k-card cut problem is equivalent to the *bisection problem*, i.e. to find a minimal cut with fixed cardinality shores. The latter problem is well known in the literature (see [1,9, pp. 256–259]). Although Lemma 1 establishes that for a complete graph the cardinalities of all possible cuts can be determined in polynomial time, we have that the number of all feasible k-cardinality cuts is exponential in the size of the problem for each value of k. Indeed, if j represents the cardinality of the minimal shore as in the proof of Lemma 1, all possible k-cardinality cuts are given by all choices of j elements of V. Thus the number of all possible k-cardinality cuts is $\binom{n}{j} \geqslant 2^j$. We will prove in Proposition 3 that for complete graphs with non-uniform weights on the edges k-card cut is strongly NP-hard. To prove $\mathscr{NP}$-hardness of the weighted problems, we use a reduction of the *equicut* problem. This is to find a cut with shores of size $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ which has minimal weight.

Lemma 2. *If G is a simple complete graph with non-uniform weights on the edges the* equicut *problem is strongly $\mathscr{NP}$-hard.*

Proof. The equicut problem is strongly $\mathscr{NP}$-hard for general graphs as proved in [18]. Now we will reduce the equicut problem for general graphs to the *equicut* problem for complete graphs. Given a graph $G = (V, E)$, $|V| = n$, with edge-weight function $w : E \to \mathbb{N}$ we transform it into a complete graph $G' = (V, E')$ adding the lacking edges. On the new graph G' we consider the edge-weight function $w' : E' \to \mathbb{N}$ defined for all $e \in E'$ in the following way:

$$w'(e) = \begin{cases} w(e) + 1 & \forall e \in E \\ 1 & \forall e \in E' \setminus E. \end{cases}$$

Since the total weight of each equicut in G' differs from the total weight of the corresponding equicut in the original graph by the fixed amount $c = \lfloor n/2 \rfloor \lceil n/2 \rceil$ we have that an optimal equicut in the original general graph corresponds to an optimal equicut in the new (complete) graph. □

Proposition 3. *If G is a simple complete graph with non-uniform weights on the edges then both k-card cut and k-card s-t cut are strongly $\mathscr{NP}$-hard.*

Proof. Reduction from equicut. Given an instance of equicut we can solve it solving the k-card cut problem with $k = \lfloor n/2 \rfloor \lceil n/2 \rceil$. The strong hardness derives from the strong hardness of equicut for complete graphs established by Lemma 2. For proving the strong hardness of k-card s-t cut we can reduce k-card cut to k-card s-t cut like in the proof of Theorem 1. □

We now proceed to show $\mathcal{NP}$-hardness for $\geqslant k$-card cut for which we introduce the following problem. Given an edge-weighted graph $G = (V, E)$ and a positive integer $b \in [|V|/2, |V|]$ the *min cut into bounded sets* problem is the problem of finding a partition of V into disjoint sets V_1 and V_2 such that $|V_1| \leqslant b$, $|V_2| \leqslant b$ and such that the sum of the weights of the edges between V_1 and V_2 is minimal.

Lemma 3. *If $G = (V, E)$ is a complete graph with non-uniform edge weights the* min cut into bounded sets *problem for G is strongly* NP*-hard.*

Proof. The min cut into bounded sets problem is strongly $\mathcal{NP}$-hard for general graphs as proved in [18]. For proving that the problem remains strongly $\mathcal{NP}$-hard for complete graphs, too, we can reduce the general graph case to the complete graph case like in the proof of Lemma 2. □

Proposition 4. *If $G = (V, E)$ is a simple complete graph with non-uniform weights on the edges $\geqslant k$-card cut is strongly $\mathcal{NP}$-hard.*

Proof. Reduction from the min cut into bounded sets problem. Due to completeness of G, a shore S of a $\geqslant k$-card cut with $k = b(n - b)$ is such that

$$|S|(n - |S|) \geqslant b(n - b). \tag{2}$$

Therefore, solving (2) with given b for $|S|$, we have $n - b \leqslant |S| \leqslant b$ and since $|S| = n - |\bar{S}|$ from $n - b \leqslant |S|$ it follows that $|\bar{S}| \leqslant b$, too. Hence the strong hardness derives from the strong hardness of the min cut into bounded sets for complete graphs proved in Lemma 3. □

2.3. Complete bipartite graphs

For a complete bipartite graph $K_{n,m}$ we shall denote the partition of the vertex set $V = L \cup R$ and assume $|L| = n_1$, $|R| = n_2$ so that $|E| = n_1 n_2$. The vertex sets are written as $L = \{l_1, \ldots, l_{n_1}\}$ and $R = \{r_1, \ldots, r_{n_2}\}$. As for complete graphs, we can characterize the cardinalities of possible cuts.

Lemma 4. *Given a complete bipartite graph $K_{n_1,n_2} = (L \cup R, E)$, k-card cut and k-card s-t cut are feasible if and only if*

$$k = jn_1 + in_2 - 2ij \tag{3}$$

with $j \in \{0, 1, \ldots, n_2\}$, $i \in \{0, 1, \ldots, n_1\}$, and $i + j \leqslant \lfloor (n_1 + n_2)/2 \rfloor$.

Proof. Let the vertex set S be a shore of a cut. We can suppose $|S| \leqslant \lfloor (n_1 + n_2)/2 \rfloor$ because otherwise $\bar{S}$ is so and it determines the same cut. Let $i = |S \cap L|$ and $j = |S \cap R|$, thus i is an integer between 0 and $|L| = n_1$, j is an integer between 0 and $|R| = n_2$ and $i + j = |S|$. Therefore

$$\delta(S) = \delta(S \cap L) \cup \delta(S \cap R) \setminus \delta(S \cap L, S \cap R)$$

and

$$|\delta(S)| = |\delta(S \cap L)| + |\delta(S \cap R)| - 2|\delta(S \cap L, S \cap R)| = jn_1 + in_2 - 2ij.$$

In this way the cardinalities of all possible cuts are given by (3). □

Given a k value we can have several pairs of values i, j satisfying (3). For example for the graph $K_{3,2}$ both $i = j = 1$ and $i = 1$, $j = 0$ satisfy (3) for $k = 3$. Moreover, unlike for complete graphs there is no one to one correspondence between the cardinality k of the cut and the cardinality of the minimal shore S of the cut. $S_1 = \{r_2\}$ and $S_2 = \{l_3, r_1\}$ determine both cuts with cardinality $k = 3$. Analogously, it is easy to see that two cuts with different cardinalities can have minimal shores of the same cardinality.

Proposition 5. *Given a complete bipartite graph $K_{n_1,n_2} = (V, E)$ such that $w(e) = 1$ for all $e \in E$, both k-card cut and k-card s-t cut are polynomially solvable.*

Proof. Through formula (3) we can check feasibility in polynomial time. If the problem is feasible, any choice of j vertices of L and of i vertices of R defines a solution for k-card cut. For k-card s-t cut we distinguish two cases:

- Vertices s and t both belong to L (or R) Then if every pair of values i, j satisfying (3) for the given value of k has $j = 0$ or $j = n_1$ then the problem is infeasible. Otherwise suppose i, j satisfy (3) with j between 1 and $n_1 - 1$. In this case a solution S is given by $\{s\}$ union any choice of $j - 1$ vertices of $L \setminus \{s, t\}$ union any choice of i vertices of R.
- Vertex s belongs to L and vertex t belongs to R. Let i, j be a pair of values satisfying (3) for the given value of k. We distinguish three subcases:
  - If $i \leqslant n_2 - 1$ and $j \neq 0$, a solution S is given by $\{s\}$ union any choice of $j - 1$ vertices of $L \setminus \{s\}$ union any choice of i vertices of $R \setminus \{t\}$.
  - If $i \leqslant n_2 - 1$ and $j = 0$, S is given by $\{t\}$ union any choice of $i - 1$ vertices of $R \setminus \{t\}$.
  - Finally, if $i = n_2$, S is given by R union any choice of j vertices of $L \setminus \{s\}$.

This completes the construction of an s-t k-card cut. □

Proposition 6. *Given a complete bipartite graph $K_{n_1,n_2} = (V, E)$ with non-uniform edge-weights, both k-card cut and k-card s-t cut are strongly $\mathcal{NP}$-hard.*

Proof. We reduce the k-card cut problem for complete graphs to the k-card cut problem for complete bipartite graphs. Given the edge-weighted complete graph $K_n = (V, E)$

with $V=\{v_1,\ldots,v_n\}$ and weight function $w: E \to \mathbb{N}$ let us create the edge-weighted complete bipartite graph $K_{n,n}=(L\cup R,E')$ with $L=\{v_1,\ldots,v_n\}$, $R=\{v'_1,\ldots,v'_n\}$ and weight function $w': E' \to \mathbb{N}$ defined for all $e\in E'$ in the following way:

$$w'(e)=\begin{cases} w(\{v_i,v_j\}) & \text{if } e=\{v_i,v'_j\} \text{ with } i\neq j,\\ M & \text{if } e=\{v_i,v'_i\},\end{cases}$$

where $M>2\sum_{e\in E} w(e)$. We observe that if k is the cardinality of a cut for K_n generated by the vertex set S with $|S|=j$, then $2k$ is the cardinality of a cut for $K_{n,n}$. Indeed it is easy to check through formulas (1) and (3) that the cut generated in $K_{n,n}$ by any vertex set S' such that $|S'\cap L|=|S'\cap R|=j$ has cardinality $2k$. Moreover, we observe that the weight of the optimal $2k$-card cut is less than M because $\delta(\{v_1,\ldots,v_j,v'_1,\ldots,v'_j\})$ is a $2k$-card cut having weight less than M. For this reason an optimal $2k$-card cut cannot contain any edge e of the form $e=\{v_i,v'_i\}$. Thus for every shore S' of an optimal $2k$-card cut $v_i\in S'$ if and only if $v'_i\in S'$. Therefore the minimal shore of an optimal $2k$-card cut for $K_{n,n}$ is of the form

$$S'=\{v_{h_1},\ldots,v_{h_j},v'_{h_1},\ldots,v'_{h_j}\}$$

with j satisfying $j(n-j)=k$. Now we want to show that $S=\{v_{h_1},\ldots,v_{h_j}\}$ is an optimal k-card cut for K_n. If S is not optimal then there exists $\tilde{S}=\{v_{p_1},\ldots,v_{p_j}\}$ such that $w(\delta(\tilde{S}))<w(\delta(S))$. But in this way $\tilde{S}'=\tilde{S}\cup\{v'_{p_1},\ldots,v'_{p_j}\}$ defines a $2k$-card cut for $K_{n,n}$ such that

$$\begin{aligned} w'(\delta(\tilde{S}')) &= \sum_{u\in\tilde{S}'\cap L}\ \sum_{v\in R\setminus(\tilde{S}'\cap R)} w'(\{u,v\}) + \sum_{u\in L\setminus(\tilde{S}'\cap L)}\ \sum_{v\in\tilde{S}'\cap R} w'(\{u,v\})\\ &= 2\sum_{u\in\tilde{S}'\cap L}\ \sum_{v\in L\setminus(\tilde{S}'\cap L)} w(\{u,v\})\\ &= 2w(\delta(\tilde{S}))<2w(\delta(S))=w'(\delta(S')) \end{aligned}$$

and this contradicts the optimality of S'. Hence the strong hardness of k-card cut for complete bipartite graphs follows from the strong hardness of k-card cut for complete graphs proved in Proposition 3. For proving the strong hardness of k-card s-t cut we can reduce k-card cut to k-card s-t cut like in the proof of Theorem 1. □

Proposition 7. *Given a complete bipartite graph $K_{n_1,n_2}=(V,E)$ with non-uniform edge-weights, both $\geqslant k$-card cut and $\geqslant k$-card s-t cut are strongly $\mathcal{NP}$-hard.*

Proof. We can reduce the $\geqslant k$-card cut problem for complete graphs to the $\geqslant k$-card cut problem for complete bipartite graphs like in the proof of Proposition 6. In this way the validity of the proposition follows from the strong hardness of $\geqslant k$-card cut for complete graphs proved in Proposition 4. □

2.4. Trees

We shall show that cut problems on trees remain very easy, even with cardinality constraints. The basis of our results is a characterization of cuts proved in [26].

Lemma 5. *For a graph G, $C \subset E$ is the edge set of a cut if and only if C has an even number (possibly zero) of edges in common with any cycle of G.*

Proposition 8. *When G is a tree, k-card cut is in $\mathscr{P}$.*

Proof. Since G is a tree, it has no cycle. So from Lemma 5 any choice of k edges is a k-card cut. Therefore in this case the k edges with smallest weight are a solution of k-card cut. □

To establish the result for s-t cuts we use the following lemma.

Lemma 6. *Let $G=(V,E)$ be a tree and $s,t \in V$ then every s-t cut has an odd number of edges in common with the path between vertex s and vertex t.*

Proof. Let P be the path between vertex s and vertex t (the existence and uniqueness of the path is ensured from G being a tree). Let V_1, V_2 be the shores of an s-t cut and let C be the edge set. It is easy to see that shores V_1 and V_2 define in the graph $G' = (V, E \cup \{s,t\})$ a cut having cut edge set $C' = C \cup \{s,t\}$. Applying Lemma 6 to cycle $P \cup \{s,t\}$ of graph G' it follows that the cardinality of $C \cap P$ must be odd. □

Proposition 9. *When G is a tree, k-card s-t cut belongs to $\mathscr{P}$.*

Proof. Let P be the path between the vertex s and the vertex t. By Lemma 6 every s-t cut has an odd number of edges in common with P. Let F be the set of the k smallest weight edges in G. If $|P \cap F|$ is odd then the edge set $C = F$ is an optimal solution of k-card s-t cut. Else if $|P \cap F|$ is even we obtain C from F modifying F as little as possible to have $|P \cap C|$ odd. There are the following four cases.

1. If $P \setminus F = \emptyset$, let $e^* \in P$ and $\tilde{e} \in \bar{F}$ be such that $w(e^*) = \max_{e \in P} w(e)$, $w(\tilde{e}) = \min_{e \in \bar{F}} w(e)$, respectively. An optimal solution is given by $C = F \setminus \{e^*\} \cup \{\tilde{e}\}$.
2. If $F \setminus P = \emptyset$, let $e^* \in F$ and $\tilde{e} \in \bar{P}$ be such that $w(e^*) = \max_{e \in F} w(e)$, $w(\tilde{e}) = \min_{e \in \bar{P}} w(e)$, respectively. An optimal solution is given by $C = F \setminus \{e^*\} \cup \{\tilde{e}\}$.
3. If $P \cap F = \emptyset$, let $e^* \in F$ and $\tilde{e} \in P$ be such that $w(e^*) = \max_{e \in F} w(e)$, $w(\tilde{e}) = \min_{e \in P} w(e)$, respectively. An optimal solution is given by $C = F \setminus \{e^*\} \cup \{\tilde{e}\}$.
4. In any other case let $e^*, \hat{e}, \tilde{e}$, and $\bar{e}$ be such that $w(e^*) = \min_{e \in P \setminus F} w(e)$, $w(\hat{e}) = \min_{e \in \overline{(P \cup F)}} w(e)$, $w(\tilde{e}) = \max_{e \in F \setminus P} w(e)$, and $w(\bar{e}) = \max_{e \in P \cap F} w(e)$, respectively. If $\overline{(P \cup F)} \neq \emptyset$ then it easy to see that if $w(e^*) - w(\tilde{e}) \leqslant w(\hat{e}) - w(\bar{e})$ an optimal solution is given by $C = F \setminus \{\tilde{e}\} \cup \{e^*\}$, otherwise it is given by $C = F \setminus \{\bar{e}\} \cup \{\hat{e}\}$. If $\overline{(P \cup F)} = \emptyset$ an optimal solution is $C = F \setminus \{\tilde{e}\} \cup \{e^*\}$. □

2.5. Grid graphs

Definition 3. A *simple grid graph* is a graph $G=(V,E)$ with $(h+1)(l+1)$ vertices arranged in $l+1$ horizontal rows and $h+1$ columns, and edges connecting vertices in adjacent rows (columns) vertically (horizontally). The horizontal and vertical lengths of G are h and l, respectively. Let $v_{i,j}$ be the vertex at row i and column j in G for $i=0,1,\ldots,l,\ j=0,1,\ldots,h$.

Our results for grid graphs show that with the exception of $k=1$ and $k=n_2-2$ cuts with cardinality k always exist. The complexity follows from the results we prove about planar graphs in Section 2.6.

Lemma 7. *If $G=(V,E)$ is a simple grid graph it has a cut of cardinality m, where $m=|E|$.*

Proof. Let h and l be the horizontal length and vertical length of G and let $v_{i,j}$ be as defined in Definition 3. It is easy to see that the set T defined as

$$T := \{v_{i,j} \in V : 0 \leqslant i \leqslant l,\ 0 \leqslant j \leqslant h,\ i,j \text{ both even or } i,j \text{ both odd}\}$$

generates a cut with edge set equal to E. □

Lemma 8. *If $G=(V,E)$ is a simple grid graph vertex set T defined in Lemma 7 has $\lceil (h-1)(l-1)/2 \rceil$ vertices of degree 4 and $h+l$ vertices of degree 2 or 3. In particular it has 4 vertices of degree 2 if h and l are both even and it has 2 vertices of degree 2 in any other case.*

The proof of Lemma 8 is omitted since it is trivial.

Proposition 10. *If $G=(V,E)$ with $|E|=m$ is a simple grid graph k-card cut and k-card s-t cut are feasible if and only if*

$$k = 2,3,\ldots,m-2,m. \tag{4}$$

Proof. Let us suppose $h \geqslant 3$ and $l \geqslant 1$ (or vice versa) otherwise the Proposition is trivial. Let T and P be the vertex sets defined in the proofs of Lemma 7 and Lemma 8. Let S denote the smaller shore of a cut. If $k \leqslant 4\lceil (h-1)(l-1)/2 \rceil$ we set S' equal to $p=\lfloor k/4 \rfloor$ vertices of P including vertex $v_{1,1}$ and

$$S := S' \cup \{v_{0,2}\} \quad \text{if } k-4p=3,$$

$$S := S' \cup \{v_{0,0}\} \quad \text{if } k-4p=2,$$

$$S := S' \setminus \{v_{1,1}\} \cup \{v_{0,0}, v_{0,2}\} \quad \text{if } k-4p=1,$$

$$S := S' \quad \text{if } k-4p=0.$$

If $4\lceil (h-1)(l-1)/2 \rceil < k \leqslant m-d$ where

$$d = \begin{cases} 8 & \text{if } h, l \text{ both even,} \\ 4 & \text{otherwise} \end{cases}$$

we set $S' := P$, we add to S' q vertices of the set

$$Q := \{v_{i,j} \in V : i = 0, l,\ 1 \leqslant j \leqslant h-1,\ i, j \text{ both even or } i, j \text{ both odd}\}$$

including vertex $v_{0,2}$, with

$$q := \left\lfloor \frac{k - 4p^*}{3} \right\rfloor, \quad p^* := \left\lceil \frac{(h-1)(l-1)}{2} \right\rceil$$

and we set

$$S := S' \cup \{v_{0,0}\} \quad \text{if } k - 4p^* - 3q = 2,$$

$$S := S' \setminus \{v_{0,2}\} \cup \{v_{0,0}, \tilde{v}\} \quad \text{if } k - 4p^* - 3q = 1,$$

$$S := S' \quad \text{if } k - 4p^* - 3q = 0,$$

where

$$\tilde{v} = \begin{cases} v_{0,h} & \text{if } h \text{ even,} \\ v_{l,0} & \text{if } l \text{ even and } h \text{ odd,} \\ v_{l,h} & \text{if } l \text{ and } h \text{ both odd.} \end{cases}$$

If $k > m - d$ we set

$$S := T \quad \text{if } k = m,$$

$$S := T \setminus \{v_{0,0}\} \quad \text{if } k = m-2,$$

$$S := T \setminus \{v_{0,2}\} \quad \text{if } k = m-3$$

and in addition in the case h and l are both even we set

$$S := T \setminus \{v_{1,1}\} \quad \text{if } k = m-4,$$

$$S := T \setminus \{v_{0,0}, v_{0,2}\} \quad \text{if } k = m-5,$$

$$S := T \setminus \{v_{0,0}, v_{1,1}\} \quad \text{if } k = m-6,$$

$$S := T \setminus \{v_{0,2}, v_{1,1}\} \quad \text{if } k = m-7. \qquad \square$$

2.6. Planar graphs

Considering the relation between the max cut problem and k-card cut established in Theorem 1, and the fact that max cut is polynomial for planar graphs, established by

Theorem 5 of [3] which we report below, it is interesting to consider the polyhedral structure of both problems for planar graphs.

Theorem 2. *Let*

$$P_C(G) := \{x \in \mathbb{R}^{|E|} : 0 \leqslant x_e \leqslant 1;\ x(F) - x(C \setminus F) \leqslant |F| - 1\},$$

where the constraints are for all $e \in E$ *and for all circuits* $C \subset E$ *and all* $F \subset C$, $|F|$ *odd, respectively. Let* $CUT(G)$ *be the cut polytope of* G, *i.e. the convex hull of all incidence vectors of cuts of* G, *then*

$$P_C(G) = CUT(G) \ \Leftrightarrow\ G \text{ is not contractible to } K_5.$$

This theorem shows that the max cut problem is solvable in polynomial time for the class of graphs non-contractible to K_5: The separation problem for all inequalities in the concise description of $CUT(G)$ is solvable in polynomial time since it can be reduced to the computation of n shortest paths as shown in [2]. Since, by Kuratowski's theorem (see Theorem 4.5 of [6]), planar graphs are those graphs which are not contractible to K_5 or $K_{3,3}$, the previous result holds for planar graphs, too. Now let $KCUT(G,k)$ denote the convex hull of all incidence vectors of k-card cut, i.e.

$$KCUT(G,k) := \operatorname{conv}\left\{x \in \{0,1\}^{|E|} : x \text{ is a cut and } \sum_{e\in E} x_e = k\right\}.$$

Therefore

$$KCUT(G,k) \subset CUT(G) \cap \left\{x \in [0,1]^{|E|} : \sum_{e\in E} x_e = k\right\}. \tag{5}$$

If the opposite inclusion held, too, we could conclude

$$KCUT(G,k) = P_C(G) \cap \left\{x \in [0,1]^{|E|} : \sum_{e\in E} x_e = k\right\}$$

and so we also would have a compact description for the k-card cut polytope. But unfortunately the opposite inclusion does not hold in (5) as the example below shows. For the graph G drawn in Fig. 2, $KCUT(G,3)=\emptyset$ because this graph has only cuts with cardinality 2 or 4. But $CUT(G)\cap\{x \in [0,1]^{|E|} : \sum_{e\in E} x_e = k\} \neq \emptyset$ because, for example, $\tilde{x}=(1, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 0, \frac{1}{2})$ belongs to this set. Indeed, $\tilde{x} \in CUT(G)$ because $\tilde{x}=\frac{1}{2}x' + \frac{1}{2}x''$ where $x'=(1,0,1,1,0,1)$ and $x''=(1,1,0,0,0,0)$ are incidence vectors of cuts of G. Moreover, $\sum_{e\in E} \tilde{x}_e = 3$. Examples of grid graphs and triangulations for which the equality does not hold can be easily constructed, too. Are there other graphs for which the two polyhedra coincide? The answer is yes: trees. Due to the proof of Proposition 8 any

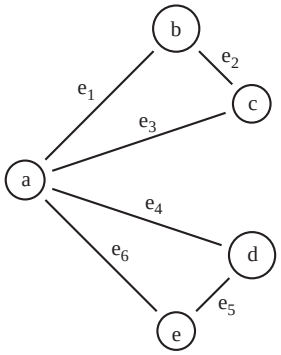

Fig. 2. Planar graph G.

subset of k edges is a k-card cut. Therefore for trees

$$\begin{aligned} KCUT(G,k) &:= \operatorname{conv}\left\{x \in \{0,1\}^{|E|} : \sum_{e\in E} x_e = k\right\} \\ &= \left\{x \in [0,1]^{|E|} : \sum_{e\in E} x_e = k\right\} \\ &= \operatorname{conv}\{x \in \{0,1\}^{|E|}\} \cap \left\{x \in [0,1]^{|E|} : \sum_{e\in E} x_e = k\right\} \\ &=: CUT(G) \cap \left\{x \in [0,1]^{|E|} : \sum_{e\in E} x_e = k\right\}. \end{aligned}$$

Although for planar graphs the polyhedral structure of k-card cut is not useful to understand the complexity of k-card cut we have achieved interesting results reducing the k-card cut problems to *exact perfect matching* problems which can be solved through *random pseudo-polynomial* algorithms.

Definition 4. A *random pseudo-polynomial* algorithm for a decision problem is an algorithm that always answers correctly in the case of a no-instance, whereas for a yes-instance the answer may be wrong, with probability less than a positive constant $\varepsilon < 1$ independent of the input size. Furthermore, it requires a time that is polynomial in the input size when the similarity assumption holds, that is when all numerical data are bounded by a polynomial in the input size.

Let $\mathscr{R}p\mathscr{P}$ denote the class of decision problems that admit a random pseudo-polynomial algorithm.

Definition 5. Given an undirected edge-weighted graph $G=(V,E)$ and an integer W, an *exact perfect matching* with weight W is a subset of the edge set E which covers all vertices in V once and only once, and has the sum of the weights equal to W.

In the following analysis we will also need the definition of a Pfaffian. Given a skew-symmetric matrix M of even order the *Pfaffian* of M, $pf(M)$, is defined as

$$pf(M) := \sqrt{det(M)},$$

where $det(M)$ denotes the determinant of M. Given a graph $G=(V,E)$ with vertex set $V=\{1,\ldots,2n\}$, edge set E, $|E|=m$ and an edge-weight function $w: E \to \mathbb{N}$, let us consider the $2n \times 2n$ skew-symmetric matrix C

$$C_{i,l} = \begin{cases} t_{i,l}\, y^{w(e)} & \text{if } e=\{i,l\} \in E,\ i<l, \\ -t_{i,l}\, y^{w(e)} & \text{if } e=\{i,l\} \in E,\ i>l, \\ 0 & \text{otherwise.} \end{cases}$$

Since the determinant of a skew-symmetric matrix of even order is a perfect square we can express $pf(C)$ as a polynomial in the variables y and $t_{i,l}, \{i,l\} \in E$:

$$pf(C) = \sum_{j=0}^{\bar{W}-1} q_j(t) y^j,$$

where $\bar{W}$ is a strict upper bound for the weight of any perfect matching of G, $q_j(t)$ is a polynomial in t and t is the vector that collects the $t_{i,l}$ for all $\{i,l\} \in E$.

Lemma 9. *There exists an exact perfect matching of weight j in graph G if and only if $q_j(t)$ is not identically zero in $pf(C)$.*

Proof. It has been proved in [25,23] that $q_j(t)$ is the sum of all monomials corresponding to exact perfect matchings of weight j. □

The nice property of $pf(C)$ described in Lemma 9 cannot directly be exploited because it would be a task of exponential complexity to obtain explicitly the monomials of $pf(C)$, since in general the number of perfect matchings in a graph is exponentially large. However, for fixed $t=\bar{t}$ $pf(C(\bar{t}))$ can be evaluated in polynomial time applying Edmond's algorithm for computing the determinant of $C(\bar{t})$ where $C(\bar{t})$ is seen as a matrix with elements in $\mathbb{Z}[y]$, the integrality domain of polynomials in variable y (see [11,16]). So we can understand the shape of $pf(C(t))$ from $pf(C(\bar{t}))$ thanks to Lemma 1 of [30], which we state below.

Lemma 10. *Let $q(t_1,\ldots,t_m)$ be a polynomial of degree n in m variables and $(\bar{t}_1,\ldots,\bar{t}_m)$ be chosen randomly in $\{1,\ldots,N\}^m$. If q is not identically 0, then*

$$Pr\{q(\bar{t}_1,\ldots,\bar{t}_m)=0\} \leqslant \frac{n}{N}.$$

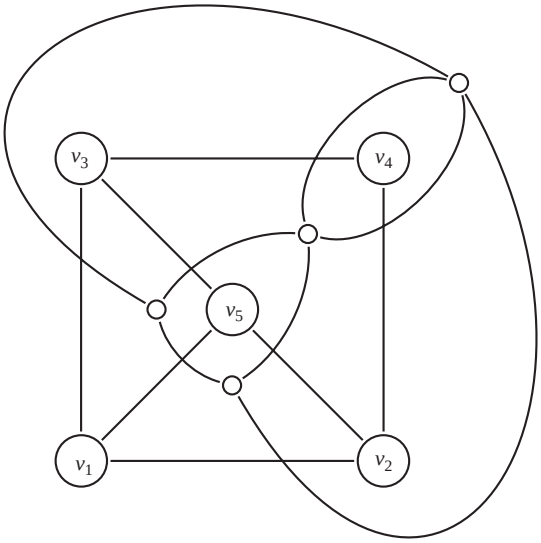

Fig. 3. The planar graph G and its dual G^*.

Using Lemma 9 and Lemma 10 the image of the perfect matching problem, i.e. the weight of all perfect matchings, can be computed through the following random pseudo-polynomial algorithm proposed in [7].

Algorithm 1. *Match*(G, w, ε)

1. Choose randomly $\bar{t}$ in $\{1,\ldots,N\}^m$ with $N = \lceil n/\varepsilon \rceil$;
2. Compute $pf(C)$ for $t = \bar{t}$;
3. For $j = 1,\ldots,\bar{W}$
   If $q_j(\bar{t}) = 0$ then no matching having weight j exists;
   If $q_j(\bar{t}) \neq 0$ then a matching having weight j exists;
   EndFor.

We notice that when the edge-weight function w achieves only $0-1$ values the Algorithm 1 becomes random polynomial ($\mathscr{RP}$) since $\bar{W} = m/2$.

Now we have all elements for proving the following result.

Proposition 11. *When $G = (V, E)$ is a planar graph the k-card cut is in $\mathscr{RP}$ if the edge-weights are uniform and is in $\mathscr{R}p\mathscr{P}$ if the edge-weights are not uniform.*

Let us consider the unweighted case first. Since the graph G is planar it has an associated geometric dual graph G^* (see Fig. 3). As a consequence of Theorems 4 and 5 of [26] every k-card cut of graph G corresponds to a Eulerian subgraph of G^* with k edges. Let $\tilde{G}$ be the planar graph with edge weights 0 and 1 obtained in the following way: We replace each vertex v_i^* of G^* with a cycle having length equal to

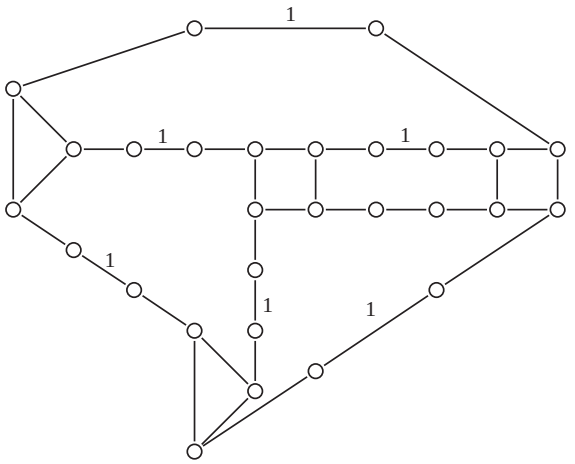

Fig. 4. The "exploded" graph $\tilde{G}$ obtained from G^*.

the degree of v_i^* and we split every edge of G^* in three edges. For every split edge we set the weight of the central edge equal to 1, whereas we set the weights of all other edges equal to 0 (see Fig. 4).

Every solution of the exact perfect matching problem with weight k in $\tilde{G}$ characterizes a Eulerian subgraph with k edges in G^*. The Eulerian subgraph in question is made up by the edges of G^* corresponding to the edges of $\tilde{G}$ with weight 1 in the solution of the exact perfect matching problem. Since for planar graphs with edge weights 0 and 1 the exact perfect matching can be solved through the random polynomial Algorithm 1 we have that for planar graphs with uniform edge-weights the k-card cut problem is in $\mathscr{RP}$.

Consider the weighted case now. From graph G we construct the exploded graph $\tilde{G} = (\tilde{V}, \tilde{E})$ as in the previous case. Now we define two weight functions w_1 and w_2 on the edge set $\tilde{E}$: w_1 is the same 0-1 weight function defined for the previous case and w_2 is defined by

$$w_2(\tilde{e}) = \begin{cases} w(e) & \text{if } w_1(\tilde{e}) = 1, \\ 0 & \text{otherwise} \end{cases}$$

for all $\tilde{e} \in \tilde{E}$ and e is the edge of G from which $\tilde{e}$ has been generated. It is easy to see that an optimal k-card cut in graph G is equivalent to an exact perfect matching in $\tilde{G}$ which has value k with respect to weight function w_1 and which minimizes w_2. This problem can be solved with a random pseudo-polynomial algorithm. Let us consider the $4m \times 4m$ skew-symmetric matrix C where $m = |E|$

$$C_{i,l} := \begin{cases} t_{i,l} x^{w_1(\tilde{e})} y^{w_2(\tilde{e})} & \text{if } \tilde{e} = \{i,l\} \in \tilde{E},\ i < l, \\ -t_{i,l} x^{w_1(\tilde{e})} y^{w_2(\tilde{e})} & \text{if } \tilde{e} = \{i,l\} \in \tilde{E},\ i > l, \\ 0 & \text{otherwise.} \end{cases}$$

Table 1
Complexity results for k-card cut and $\geqslant k$-card cut

Graph class	Unweighted	Weighted	Reference
General	Strongly $\mathcal{NP}$-complete	Strongly $\mathcal{NP}$-hard	Theorem 1
Complete	$\mathcal{P}$	Strongly $\mathcal{NP}$-hard	Propositions 2, 3, 4
Complete bipartite	$\mathcal{P}$	Strongly $\mathcal{NP}$-hard	Propositions 5, 6, 7
Tree	$\mathcal{P}$	$\mathcal{P}$	Proposition 8
Grid	$\mathcal{P}$	$\mathcal{RPP}$	Propositions 10, 11
Planar	$\mathcal{RP}$	$\mathcal{RPP}$	Proposition 11

Then

$$pf(C) = \sum_{j=0}^{m} q_j(t, y)x^j,$$

where $q_j(t, y)$ is the sum of all monomials corresponding to perfect matchings with weight j with respect to w_1. Therefore, for randomly chosen $\bar{t}$ in $\{1, \ldots, N\}^m$ with

$$N = \left\lceil \frac{2m}{\varepsilon} \right\rceil$$

if $q_k(\bar{t}, y) = 0$ then we can establish that no matching having weight k with respect to w_1 exists with a probability of an incorrect answer less than ε. Whereas if $q_k(\bar{t}, y) \neq 0$ then we can state that the optimal value of k-card cut is given by the minimal degree of y in $q_k(\bar{t}, y)$ and this answer is certainly correct. □

2.7. Summarizing table

We summarize the complexity results obtained for k-card cut and $\geqslant k$-card cut in Table 1. We note that the $\mathcal{NP}$-hardness results are also valid for graphs with integer weights unrestricted in sign. k-card cut on trees is also polynomially solvable on trees with unrestricted integer weights, as the proof of Propositions 8 and 9 does not use nonnegativity of weights. The case of planar graphs remains open. More open problems are summarized in Section 2.8 below.

2.8. Open problems

In this subsection we consider various open problems related to the k-card cut problem. As noted above, k-card cut remains open for planar graphs when the weights are unrestricted in sign. Among the graph classes considered in this paper the problem with non-uniform weights in grid graphs also remains open. It is also possible to investigate the $\leqslant k$-card cut problem. On trees this is of course polynomially solvable with the same reasoning as in Section 2.4 above. We are not aware of any research on this problem.

Let $G = (V, E)$ be an undirected graph and let $w : E \to \mathbb{N}$ be a non-negative integral weight function on the edge set. The max k-cut (min k-cut) problem is to find

a partition of the vertex set V into k disjoint subsets $V = (V_1, V_2, \ldots, V_k)$ such that the sum of the weights of the crossing edges $\sum_{1 \leqslant r < s \leqslant k} \sum_{i \in V_r, j \in V_s} w(\{v_i, v_j\})$ is maximized (minimized). Both the max version and the min version of this problem are $\mathcal{NP}$-hard. For the max k-cut [15] develop an approximation algorithm with a factor $1/(1 - 1/k + 2k^{-2} \ln k)$ via a semidefinite programming relaxation that generalizes the approach proposed in [20] for the max cut problem. For the min k-cut problem [29] present an approximation algorithm with a factor $2 - 2/k$. A generalization of this graph partitioning problem is the max (min) k-cut with given size of the parts, that is a max (min) k-cut problem where also the cardinality of each subset V_r of the partition is constrained to be equal to a fixed value t_r for $r = 1, \ldots, k$. [1] present a $1/2$ approximation for the maximization version of this problem. For the minimization version no constant-factor algorithm is known for general graphs; however, when the edge weights satisfy the triangle inequality and k is fixed, [21] present a polynomial-time 3-approximation algorithm.

Another interesting problem is k-card cut on directed graphs. Concerning the computational complexity we notice that the cases which are $\mathcal{NP}$-hard on undirected graphs remain so also on directed graphs since we can polynomially reduce the undirected version of k-card cut to the directed version doubling each edge in two opposite arcs having the same weight as in the original graph. On the other hand, the proofs that k-card cut belongs to $\mathcal{P}$ for undirected trees and to $\mathcal{RPP}$ for undirected planar graphs cannot be immediately adapted to the corresponding directed graphs.

3. Heuristic methods

In this section we develop some heuristic approaches for the k-card cut problem. We distinguish between complete graphs and complete bipartite graphs and general graphs because for the former it is always possible to find a feasible solution in polynomial time (see Propositions 2 and 5), whereas for the latter this is not possible. Therefore the heuristic methods for general graphs not only do not guarantee to find an optimal solution but also do not guarantee to find a feasible solution. We notice that even though all these algorithms are developed for k-card cut they can be easily extended to the k-card s-t cut problem.

3.1. Heuristics for complete graphs

In this subsection we develop heuristic procedures for the special case that G is a complete graph. In this case, as shown by Lemma 1, the cardinality constraint on the cut edge set is equivalent to a cardinality constraint on the shores of the cut. Therefore all heuristic procedures developed here determine a shore S_{heur} satisfying the cardinality constraint and try to minimize the sum of the weights of the edges between S_{heur} and $\bar{S}_{\text{heur}}$. We assume without loss of generality that S_{heur} is the shore of minimal cardinality. In the sequel we use a simple procedure Feasible(K_n, w, k) which either returns the cardinality T of the minimal shore of a k-card cut or establishes that k-card cut is infeasible, in which case it returns 0. To do so, the procedure checks

if $(n - \sqrt{n^2 - 4k})/2$ is a positive integer. We first propose a greedy heuristic. Let us suppose that k-card cut is feasible for graph G and let $T = \text{Feasible}(G, w, k)$. The q-Greedy heuristic generates a set of cuts initializing exhaustively $S_{\text{heur}} = S$ for all subsets $S \subset V$ with $|S| = q \geqslant 0$ and adding to S_{heur} other vertices until the constraint on the cardinality of S_{heur} is satisfied. The vertex $\bar{i}$ added to S at each step is such that the sum of the weights of the edges between $S_{\text{heur}} \cup \{\bar{i}\}$ and its complement set is minimal. Finally, among all the cuts so obtained the algorithm gives cut the cut having the best weight. The computational cost of this heuristic depends strongly on the choice of q. In particular, when $q = T - 1$ it is an enumeration algorithm of all feasible solutions.

Algorithm 2. *q-Greedy*(K_n, w, k)

If Feasible$(K_n, w, k) = 0$ **then Stop**;
else $T :=$ Feasible(K_n, w, k);
$q := \min\{q, T - 1\}$, $S_{\text{heur}} := \emptyset$, $w(\emptyset) := \infty$;
For all $S \subset V$ with $|S| = q$
  For $t = 1, \ldots, T - q$
    $\bar{i} := \arg\min_{i \in V \setminus S} w(\delta(S \cup \{i\}))$;
    $S := S \cup \{\bar{i}\}$;
  EndFor;
  If $w(\delta(S)) < w(\delta(S_{\text{heur}}))$ **then** $S_{\text{heur}} := S$;
EndFor;
Return S_{heur};

The computational cost of this heuristic is $O(\binom{n}{q}(n - \sqrt{n^2 - 4k} - q))$. Since we know the cardinality of the minimal shore of the cut we are looking for, we can start with S_{heur} equal to any subset of V with cardinality T. As long as the total weight of the cut can be decreased swapping a vertex of S_{heur} with a vertex of $\bar{S}_{\text{heur}}$, we do so. Thus we obtain a local improvement heuristic.

Algorithm 3. *Local*(K_n, w, k)

If Feasible$(K_n, w, k) = 0$ **then Stop**;
else $T :=$ Feasible(K_n, w, k);
Let $S_{\text{heur}} \subset V$ be such that $|S_{\text{heur}}| = T$;
Repeat $S := S_{\text{heur}}$;
  For all $v_1 \in S$, $v_2 \in \bar{S}$
    if $w(\delta(S_{\text{heur}} \setminus \{v_1\} \cup \{v_2\})) < w(\delta(S_{\text{heur}}))$
    then $S_{\text{heur}} := S_{\text{heur}} \setminus \{v_1\} \cup \{v_2\}$;
  EndFor;
Until $S = S_{\text{heur}}$;
Return S_{heur};

The computational cost is $O(k^2(\bar{w} - \underline{w}))$ where $\bar{w} := \max_{e \in E} w(e)$ and $\underline{w} := \min_{e \in E} w(e)$. In fact, in the worst case the algorithm starts with a k-cardinality cut having all

edge-weights equal to $\bar{w}$ and it stops with a k-cardinality cut having all edge-weights equal to $\underline{w}$ and at each step the solution only improves by one unit. Therefore the complexity is $(k\bar{w}-k\underline{w})T(n-T)=k^2(\bar{w}-\underline{w})$ since from the definition of T, $T(n-T)=k$.

3.2. Heuristics for complete bipartite graphs

In this subsection we modify the heuristics developed for complete graphs for the case that G is the complete bipartite graph $K_{n_1,n_2}=(V,E)$. We suppose $V=L\cup R$ where L and R are the vertex sets introduced before Lemma 4. Due to Lemma 4 k-card cut is feasible if and only if there is a pair of values i,j such that $j\in\{0,1,\ldots,n_2\}$, $i\in\{0,1,\ldots,n_1\}$ and $jn_1+in_2-2ij=k$. The minimal shore of a cut is made up of i vertices of L and j vertices of R. Therefore to ensure the feasibility of the cut found, all the heuristic procedures developed here determine a shore S_{heur} of this kind and try to minimize the sum of the edge-weights between S_{heur} and $\bar{S}_{\text{heur}}$. We shall again use a procedure Feasible2(K_{n_1,n_2},w,k) to return two values $\tilde{i}$ and $\tilde{j}$ satisfying (3). In case of ties the pair with $i+j$ maximal will be chosen. If the problem is infeasible, the procedure will return the pair $(0,0)$. The q_1,q_2-greedy heuristic generates a set of cuts initializing exhaustively S_{heur} with $S_L\cup S_R$ for all subsets $S_L\subset L$, $S_R\subset R$ such that $|S_L|=q_1$, $|S_R|=q_2$ with $0\leqslant q_1\leqslant\tilde{i}$, $0\leqslant q_2\leqslant\tilde{j}$ and adding vertices first to S_L then to S_R until these sets satisfy the cardinality constraints. The new vertex $\bar{i}$ added in each step is such that the sum of the weights of the edges between $S_{\text{heur}}\cup\{\bar{i}\}$ and its complement set is minimal. Finally, among all the cuts so obtained the algorithm gives out the cut having the best weight. It is easy to see that the number of computations required is

$$O\left(\binom{n_1}{q_1}\binom{n_2}{q_2}((\tilde{i}-q_1)n_1+(\tilde{j}-q_2)n_2)\right).$$

Finally we present the modification of the local improvement heuristic for complete bipartite graphs. We start with a feasible k-card cut, i.e. with $S_{\text{heur}}\subset V$ such that $|S_{\text{heur}}\cap L|=i$, $|S_{\text{heur}}\cap R|=j$ with $jn_1+in_2-2ij=k$. As long as the total weight of the cut can be decreased swapping a vertex of $S_L:=S_{\text{heur}}\cap L$ with a vertex of $L\setminus S_L$ or swapping a vertex of $S_R:=S_{\text{heur}}\cap R$ with a vertex of $R\setminus S_R$, we do so. The complexity of the heuristic is then $O(k(\bar{w}-\underline{w})(n_1^2+n_2^2))$.

3.3. Heuristics for general graphs

Now we consider a general simple graph without loops $G=(V,E)$ with $|V|=n$ and $|E|=m$. Our heuristic generates a set of cuts initializing in turn the shore of the cut with a different vertex of V and adds in each step the vertex that minimizes the difference between the cardinality of the current cut and k until the cardinality can no longer be improved or until further additions would make the cardinality of the shore greater than $\lfloor n/2\rfloor$. If more than one vertex improves the cardinality of the current cut by the same amount we choose the vertex that minimizes the total weight of the cut. Among all the cuts so obtained we choose the cut having best cardinality and in case of cuts with same cardinality we choose one having the best total weight.

The method is given in Algorithm 4, where for all $j \in V$ labels l_j denote the number of edges incident with vertex j which belong to the current cut, labels $\tilde{l}_j$ denote the sum of the weights of those edges, k_T and w_T denote the cardinality and the weight of the current shore T and finally $k_{S_{\text{heur}}}$ and $w_{S_{\text{heur}}}$ denote the cardinality and the weight of the final cut found.

Algorithm 4. *General*(G, w, k)

$S_{\text{heur}} := \emptyset$; $k_{S_{\text{heur}}} := 0$; $w_{S_{\text{heur}}} = 0$;
For all $j \in V$
  $l_j := 0$, $\tilde{l}_j := 0$, $w(\delta(j)) := \sum_{i \in \delta(\{j\})} w_{i,j}$;
  Set $gr(j)$ equal to the degree of vertex j;
EndFor
If $k < \min_{j \in V} gr(j)$ **or** $k > m$ **then return** infeasible;
For all $i \in V$
  $T := \emptyset$; $w_T := 0$; $k_T := 0$; $p := i$; $\delta k^* := |k - gr(p)|$; $\delta w^* := \infty$;
  While $(p \neq -1)$
    $T := T \cup \{p\}$;
    $k_T := k_T + gr(p) - 2l_p$; $w_T := w_T + w(\delta(p)) - 2\tilde{l}_p$;
    If $k_T = k$ **or** $|T| > \lfloor \frac{n}{2} \rfloor$ **then** $p := -1$;
    else begin
      /* Update the labels: */
      $l_p := gr(p) - l_p$, $\tilde{l}_p := w(\delta(p)) - \tilde{l}_p$;
      For all $j \in V$ such that j is adjacent to p

$$l_j := \begin{cases} l_j - 1 & \text{if } j \in T \\ l_j + 1 & \text{if } j \notin T \end{cases}$$

$$\tilde{l}_j := \begin{cases} \tilde{l}_j - w_{j,p} & \text{if } j \in T \\ \tilde{l}_j + w_{j,p} & \text{if } j \notin T \end{cases}$$

      EndFor
      $p := 0$;
      For all $j \notin T$
        $\delta k := |gr(j) - 2l_j - |k - k_T||$;
        $\delta w := w(\delta(j)) - 2\tilde{l}_j$;
        If $\delta k < \delta k^*$ **or** $(\delta k = \delta k^*$ **and** $\delta w < \delta w^*)$
        then $p := j$; $\delta k^* := \delta k$; $\delta w^* := \delta w$;
      EndFor
    End else
    If $p = 0$ **then** $p := -1$;
    End while
  If $|k - k_T| < |k - k_{S_{\text{heur}}}|$ **or** $(k_T = k_{S_{\text{heur}}}$ **and** $w_T < w_{S_{\text{heur}}})$
  then $S_{\text{heur}} := T$; $k_{S_{\text{heur}}} = k_T$; $w_{S_{\text{heur}}} := w_T$;
EndFor
Return S_{heur};

Table 2
Complexity of the heuristics developed for k-card cut

Graph class	Heuristic	Complexity	Reference
Complete	q-Greedy	$O(n^{q+1}(n-\sqrt{n^2-4k}-q))$	Section 3.1
Complete	Local	$O(k^2(\bar{w}-\underline{w}))$	Section 3.1
Complete bipartite	q_1,q_2-Greedy	$O(n_1^{q_1}n_2^{q_2}(n_1^2+n_2^2-n_1q_1-n_2q_2))$	Section 3.2
Complete bipartite	Local	$O(k(\bar{w}-\underline{w})(n_1^2+n_2^2))$	Section 3.2
General	General	$O(n^3)$	Section 3.3

The computational cost of this algorithm is $O\left(n^3\right)$ since it uses three nested "for" cycles on the vertex set. For this heuristic it is possible to find some examples where the heuristic solution does not have cardinality k even if a cut of cardinality k exists. This is to be expected since unless $\mathscr{P}=\mathscr{NP}$ no polynomial algorithm can find a fixed cardinality cut according to Theorem 1. Nevertheless, Algorithm 4 tries to find a cut with cardinality as close as possible to k and among the cuts found with the best cardinality it chooses the cut having the best weight. In Table 2 we summarize the computational complexity of all the heuristics developed for k-card cut in this section.

4. Lower bounds

In this section we consider some methods to generate lower bounds for the k-card cut problem with the aim of estimating the quality of the heuristic solutions developed in the previous section.

4.1. LP relaxation

Let us introduce the characteristic vectors $x=(x_e)\in\{0,1\}^{|E|}$ and $y=(y_i)\in\{0,1\}^{|V|}$ where

$$x_e=\begin{cases}1 & \text{if edge } e \text{ belongs to the solution,}\\ 0 & \text{otherwise,}\end{cases}$$

$$y_i=\begin{cases}0 & \text{if vertex } i \text{ belongs to the first shore of the cut,}\\ 1 & \text{if vertex } i \text{ belongs to the second shore of the cut.}\end{cases}$$

The k-card cut problem can be formulated as the following integer linear program.

$$\min \sum_{\{i,j\}\in E} w_{i,j}x_{i,j} \tag{6}$$

$$\text{subject to } x(E)=k, \tag{7}$$

$$x_{i,j}-y_i-y_j\leqslant 0; \quad \forall\{i,j\}\in E, \tag{8}$$

$$x_{i,j} + y_i + y_j \leqslant 2; \quad \forall\{i,j\} \in E, \tag{9}$$

$$-x_{i,j} + y_i - y_j \leqslant 0; \quad \forall\{i,j\} \in E, \tag{10}$$

$$-x_{i,j} - y_i + y_j \leqslant 0; \quad \forall\{i,j\} \in E, \tag{11}$$

$$y_i \in \{0,1\}; \quad \forall i \in V, \tag{12}$$

where $w_{i,j}$ and $x_{i,j}$ denote respectively $w(\{i,j\})$ and $x_{\{i,j\}}$.

It is easy to see that constraints (8)–(12) force x to be the characteristic vector of a cut, while constraint (7) models the cardinality requirement. We note that variables $x_{i,j}$ are not directly constrained to be either integer or to belong to $[0,1]$ because the set of constraints (8)–(12) imposes them to be so. Therefore this is a linear formulation for k-card cut with n binary variables, m continuous variables and $4m+1$ linear constraints. We can obtain a lower bound for k-card cut considering the LP relaxation of the previous formulation, that is dropping the integrality constraint (12) and requiring

$$y_i \in [0,1]; \quad \forall i \in V. \tag{13}$$

We note that any feasible solution (x,y) of (6)–(11) and(13) has still $x_{i,j} \in [0,1]$ for all $\{i,j\} \in E$ even if these variables are not directly constrained to be so.

4.2. SDP relaxation of k-card cut

First we derive a non-convex formulation of k-card cut. Let $A = (A_{i,j})$ and $B = (B_{i,j})$ be the (symmetric) weighted and the unweighted adjacency matrix of graph G, defined as follows:

$$A_{i,j} := \begin{cases} 0 & \text{if } \{i,j\} \notin E, \\ w_{i,j} & \text{if } \{i,j\} \in E, \end{cases}$$

$$B_{i,j} := \begin{cases} 0 & \text{if } \{i,j\} \notin E, \\ 1 & \text{if } \{i,j\} \in E. \end{cases}$$

Set $L_M := \mathrm{Diag}(Me_n) - M$ for $M = A$ or $M = B$, where, for any vector v, $\mathrm{Diag}(v)$ is the diagonal matrix having vector v on the diagonal and e_n is the vector of all ones in $\mathbb{R}^n$. The matrices L_A and L_B are known as the weighted and the unweighted Laplacian matrix of graph G. For a cut, let $Y = (Y_{i,j}) \in \{-1,1\}^{n^2}$ be a symmetric matrix such that

$$Y_{i,j} = \begin{cases} -1 & \text{if } i \text{ and } j \text{ are in different shores}, \\ 1 & \text{if } i \text{ and } j \text{ are in the same shore}. \end{cases} \tag{14}$$

The following is a non-convex formulation for the k-card cut problem adapted from the formulation of max cut presented in [20].

$$\min L_A \bullet Y \tag{15}$$

$$\text{subject to } L_B \bullet Y = 4k, \tag{16}$$

$$\mathrm{diag}(Y) = e_n, \tag{17}$$

$$\mathrm{rank}(Y) = 1, \tag{18}$$

$$Y \succcurlyeq 0, \tag{19}$$

where "•" denotes the Frobenius product between matrices

$$A \bullet B := \mathrm{trace}(A^{\mathrm{T}}B) = \sum_{i=1}^{n}\sum_{j=1}^{n} A_{i,j}B_{i,j},$$

$\mathrm{diag}(Y)$ is the vector of diagonal entries of Y, $\mathrm{rank}(Y)$ denotes the rank of matrix Y and $Y \succcurlyeq 0$ means that matrix Y must be positive semidefinite (details on semidefiniteness can be found in [32] whereas further details on this formulation can be found in [5]).

If we drop the non-convex rank constraint (18) we obtain a relaxation for k-card cut, which is a semidefinite programming problem, that is a generalization of a linear programming problem consisting in the optimization of a linear function of a matrix variable Y subject to linear equality constraints and to the positive semidefiniteness of Y. We have solved this SDP relaxation of k-card cut through Borchers' C library for SDP called CSDP (see [4]).

To run CSDP we need starting values for the matrix-variables of both primal and dual SDP problems. We remark that Borchers' routine requires these matrices to be positive definite (instead of positive semidefinite) but they are not required to be feasible. Using this code we observed that if the starting values are too far from feasibility then CSDP converges slowly or it cannot converge at all because some parameters used by this routine need to be calibrated for each particular SDP instance. Therefore it is better that those starting points are feasible or at least "almost" feasible. As a starting matrix $\tilde{Y}$ of the primal problem (15)–(17), (19) we have chosen

$$\tilde{Y} := \begin{cases} I & \text{if } k \geqslant \frac{mn}{2(n-1)}, \\ H & \text{otherwise,} \end{cases} \tag{20}$$

where I is the identity matrix and H is a real symmetric $n \times n$-matrix defined by

$$H := \begin{pmatrix} 1 & a & \dots & a \\ a & 1 & \dots & a \\ \dots & \dots & \dots & \dots \\ a & a & \dots & 1 \end{pmatrix}, \tag{21}$$

i.e. H has ones on the diagonal and the constant $a = 1 - 2k/m$ everywhere else. In this way, it is easy to see that matrix $\tilde{Y}$ is positive definite and it is almost feasible since for $k \geqslant mn/2(n-1)$ every constraint of the SDP relaxation of k-card cut is satisfied except $L_B \bullet Y = 4k$. The SDP relaxation of k-card cut has been obtained from formulation (15)–(19) for which equality (14) is valid. Obviously (14) does not hold any more for the SDP relaxation since we have relaxed the rank constraint (18). Therefore we

can obtain an improvement of the SDP relaxation adding inequalities which are valid for (14).

For example we can generate valid inequalities through the following trivial observation. Consider any arbitrary triangle with vertices $p < q < r$ in the graph G. Then any cut has either zero or two edges in common with this triangle. Translated into our model this leads to

$$-Y_{p,q} - Y_{p,r} - Y_{q,r} \leqslant 1,$$

$$-Y_{p,q} + Y_{p,r} + Y_{q,r} \leqslant 1,$$

$$Y_{p,q} - Y_{p,r} + Y_{q,r} \leqslant 1,$$

$$Y_{p,q} + Y_{p,r} - Y_{q,r} \leqslant 1$$

for all pairwise distinct $p,q,r \in \{1,\ldots,n\}$. These inequalities are known as triangle inequalities and were first proposed for strengthening the *SDP* relaxation of the max cut problem in [28]. Since in total we have $4\binom{n}{3} \simeq \frac{2}{3}n^3$ triangle inequalities, we still obtain a tractable relaxation adding all these inequalities to the *SDP* relaxation of k-card cut. It is easy to see that the starting matrix $\tilde{Y}$ defined in (20) satisfies all triangle inequalities. For efficiency reasons we do not add all triangle inequalities at once, but include them successively according to the amount of violation. We have tested different ways to add violated inequalities. We have observed that the bound obtained adding many violated inequalities a few times is worse than adding few violated inequalities many times, but the latter is more expensive than the former. Thus we have attained a good compromise between the quality of the bound and the total computation time adding 20 times the 5 triangle inequalities of each type which are most violated by the current solution. In fact we did not obtain much improvement adding violated inequalities after the 20th iteration.

5. A randomized SDP heuristic

So far we have seen how the SDP relaxation can be used to find good lower bounds for k-card cut. It is also possible to develop approximation algorithms. We recall that given a constant α, an α-approximate algorithm of a minimization (maximization) problem is a polynomial time algorithm which finds a feasible solution with objective value less (greater) than or equal to the optimal value multiplied by α. Since the existence problem of k-card cut is already $\mathcal{NP}$-complete we cannot expect to find an approximate algorithm in this sense. Therefore, for k-card cut we develop an approximation algorithm where the cardinality constraint is satisfied approximately, too. To this end we have adapted Goemans' and Williamson's SDP based randomized approximation algorithm for max cut (see [20,19]) for k-card cut. Let Y be the solution of the strengthened *SDP* relaxation of k-card cut. Since $Y \succcurlyeq 0$, using Cholesky decomposition, we can express $Y_{i,j}$ as

$$Y_{i,j} = v_i^{\mathrm{T}} v_j \quad \text{for all } i,j \in \{1,\ldots,n\} \tag{22}$$

for some vectors $v_i \in \mathbb{R}^n$ with $\|v_i\| = 1$ for $i = 1, \ldots, n$ (see [24]). Let r be a randomly selected vector on the unit sphere $\{x \in \mathbb{R}^n : \|x\| = 1\}$. The hyperplane orthogonal to r separates the vectors v_i into two sets

$$V_1 = \{i \in V : v_i^{\mathrm{T}} r \leqslant 0\} \quad \text{and} \quad V_2 = \bar{V}_1. \tag{23}$$

(V_1, V_2) is chosen as the random cut. This is referred to as the random hyperplane technique. We want to show that this random cut enjoys an approximation property similar to the one described for max cut in [4,20]. The probability that a given edge $\{i, j\}$ is in the random cut is equal to the probability that the random hyperplane separates v_i and v_j. This is equal to the ratio of the angle between vectors v_i and v_j and π. Thus the expected value of the cardinality of the random cut is

$$E[card(cut)] = \sum_{\{i,j\} \in E} \frac{\arccos(v_i^{\mathrm{T}} v_j)}{\pi}. \tag{24}$$

Since

$$0.87856 \frac{1 - x}{2} \leqslant \frac{\arccos(x)}{\pi} \leqslant \frac{1 - x}{2} 1.13822$$

for $-1 \leqslant x \leqslant 1$ we have that with (22)

$$0.87856 \sum_{\{i,j\} \in E} \frac{1 - Y_{i,j}}{2} \leqslant E[card(cut)] \leqslant 1.13822 \sum_{\{i,j\} \in E} \frac{1 - Y_{i,j}}{2}. \tag{25}$$

Since Y is a solution of the SDP relaxation of k-card cut it satisfies the constraint $L_B \bullet Y = 4k$ which is equivalent to

$$\sum_{\{i,j\} \in E} \frac{1 - Y_{i,j}}{2} = k$$

(see [5] for further details). Thus (25) can be rewritten as

$$0.87856k \leqslant E[card(cut)] \leqslant 1.13822k. \tag{26}$$

Hence formula (26) ensures that the expected value on the cardinality of the random cut belongs to a small neighbourhood of k. With regard to the expected value of the weight of the cut we find through similar arguments

$$E[w(cut)] \leqslant 1.13822 \sum_{\{i,j\} \in E} w_{i,j} \frac{1 - Y_{i,j}}{2}. \tag{27}$$

The sum in (27) is equal to the objective function of the strengthened *SDP* relaxation of k-card cut. Since Y is a solution of this relaxation, the sum in (27) is less than the optimal value w^* of k-card cut. Hence we obtain

$$E[w(cut)] \leqslant 1.13822 w^*. \tag{28}$$

The approximation properties (26) and (28) we have shown hold for the expected value of the cardinality and weight of the random cut whereas no approximation properties

are obtained for a single random cut. Therefore for exploiting these approximation properties we have considered several uniform random vectors. For each of them we have generated a random cut through the random hyperplane technique. Among all the random cuts so obtained we have chosen the one with cardinality as close as possible to k and in case of cuts with same cardinality we have chosen the one having the best total weight.

6. Numerical results

6.1. Description of the instances for k-card cut

Since the k-card cut problem has never been considered in the literature no validation instances are publicly available. Therefore we have generated instances of the k-card cut problem for all graph classes examined in this work according to Table 3. General graphs have been generated considering vertices with uniformly distributed random degree in order to have an edge density of 50%. Planar graphs have been generated considering as vertices uniformly distributed random points in a square and linking pairs of them by an edge only if the edge does not cross any edges previously generated and until an edge density equal or less than 50% is achieved. All graphs have random integer weights on the edges uniformly distributed between 1 and 100. For complete and complete bipartite graphs we have generated instances of k-card cut for all feasible values of k, whereas for planar graphs and general graphs we have considered as values of k the cardinalities of the cuts generated by random partitions of the vertex set.

All these instances are publicly available at the web site http://www.elet.polimi.it/upload/bruglier/kcut.html#instances.

6.2. Numerical results

All the heuristics presented were implemented in C and run on a AMD K7 1 GHz computer with 1.2 GB RAM under the Linux 2.2.14 operating system. For the smaller

Table 3
Instances generated for k-card cut

Graph class	No. of graphs	Vertices	Edges	Values of k	No. of instances
Complete	20	25	300	12	240
Complete	10	100	4950	50	500
Complete bipartite	10	$20+10$	200	55	550
Complete bipartite	12	$40+30$	1200	273	3276
Planar graphs	10	30	60–70	30	300
Planar graphs	10	150	370–389	30	300
General graphs	10	30	210–226	30	300
General graphs	10	150	5530–5665	30	300

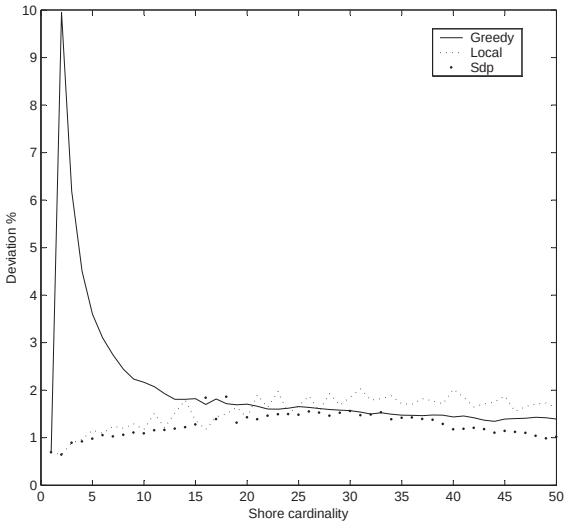

Fig. 5. Results for k-card cut on complete graphs with 100 vertices.

Table 4
Performance of the SDP lower bound

Graph class	Complete	Bipartite	General	Planar
Maximum	7.449	37.173	93.172	97.55
Average	0.621	1.675	3.998	12.454
Median	0.182	0.872	1.640	4.843
Standard deviation	1.023	2.625	7.680	18.942
Skewness	2.902	5.708	6.516	2.383
Fraction above mean	0.413	0.296	0.230	0.263

instances, i.e. the instances corresponding to rows 1, 3, 5, and 7 of Table 3, we have found the optimal solution solving the mixed integer linear formulation for k-card cut given by (6)–(12) by the MIP solver `Cplex 7.0`. This was not possible for the other instances.

Therefore, for the set of small instances we calculated the average relative deviation of the heuristic solution values from the optimal values for each value of k. For all other instances we calculated the average relative deviation of the heuristic solution values from the SDP lower bounds for each value of k. One set of these results is shown graphically in Fig. 5 (we refer to the web site http://www.elet.polimi.it/upload/bruglier/kcut.html#results for further diagrams on the numerical results obtained). CPU times are summarized in Table 5.

The quality of the SDP lower bound has also been tested on the smaller instances and is shown in Table 4. The first three rows show the maximal, average, and median

Table 5
CPU-time (in s) for the heuristics and for the exact formulation

Graph class	Greedy	Local	General	SDP	Exact
Complete, 25	0.01–0.25	0.01–0.15	—	18–70	2–99
Complete, 100	0.02–937	0.02–433	—	537–923	—
Bipartite, $20+10$	0.01–0.62	0.01–0.1	—	0.46–116	0.83–445
Bipartite, $40+30$	0.01–35	0.04–14	—	2.17–449	—
Planar, 30	—	—	0.01–0.04	0.76–91	0.12–0.51
Planar, 150	—	—	0.31–5.18	145–2055	—
General, 30	—	—	0.01–0.06	41–94	1–79
General, 150	—	—	5–144	1275–1956	—

deviation from optimality in %, the others show standard deviation, skewness, and the fraction of instances with deviation above average.

These numbers indicate that the SDP lower bound works best for complete graphs, followed by complete bipartite graphs, and is not as good for general and planar graphs. In all classes there is a relatively small number of instances with large deviations whereas most instances have below average deviations. We notice that the quality of the SDP lower bound improves for the bigger instances.

Looking at the results we can make some observations. Although there is no dominating relation between the heuristics the *SDP* and local search heuristics seem to be more robust than the Greedy heuristic for complete and complete bipartite graphs. Moreover the SDP heuristic reveals to be the most robust heuristic for all other graph classes. However, in several instances of k-card cut, especially for general and planar graphs the SDP heuristic finds a cut with cardinality close to k but not exactly k as required. We note that the infeasible solutions found do not appear in the graphics. In general the results of the SDP heuristic for big instances seem to be better than the results for small instances. The behaviour of the deviations seem not to depend on the value of k: the deviations of instances having big values of k are of the same order of the deviations of instances with small values of k.

In Table 5 we summarize the minimum CPU-times and the maximum CPU-times for running the heuristics and, when possible, the exact formulation. We can see that the running time obtained could be improved because the intention of the paper was not in greatest possible efficiency of the implemented methods, e.g. no special data structures have been used. We notice that although the SDP heuristic is the most expensive (for small instances even more expensive than the exact formulation!), however the increase in CPU time is much less for it than for the other heuristics when the size of the instances increases.

Acknowledgements

The authors thank the anonymous referees for their useful suggestions.

References

[1] A.A. Ageev, M.I. Sviridenko, Approximation algorithms for maximum coverage and max-cut with given sizes of parts, in: Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science, Vol. 1610, Springer, Berlin, 1999, pp. 17–30.

[2] F. Barahona, On some applications of the Chinese postman problem, in: Algorithms and Combinatorics: Paths, Flows, and VLSI-Layout, Springer, Berlin, 1990.

[3] F. Barahona, M. Grötschel, M. Jünger, G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design, Oper. Res. 36 (3) (1988) 493–513.

[4] B. Borchers, CSDP, a C library for semidefinite programming, Optimization Methods and Software 11 (1999) 613–623, http://www.nmt.edu/∼borchers/csdp.html.

[5] M. Bruglieri, K-Cardinality cut problems, Ph.D. Thesis, Dottorato MA.C.R.O., Department of Mathematics 'F. Enriquez', University of Milan, 2000. Available at www.elet.polimi.it/upload/bruglier/kcut/PhD-thesis.zip.

[6] R.G. Busacker, T.L. Saaty, Finite Graphs and Networks: An Introduction with Applications, McGraw-Hill, New York, 1965.

[7] P.M. Camerini, G. Galbiati, F. Maffioli, The image of weighted combinatorial problems, Ann. Oper. Res. 33 (1991) 181–197.

[8] T.J. Chang, N. Meade, J.E. Beasley, Y.M. Sharaiha, Heuristics for cardinality constrained portfolio optimisation, Comput. Operat. Res. 27 (2000) 1271–1302.

[9] M. Dell'Amico, F. Maffioli, S. Martello, Annotated Bibliographies in Combinatorial Optimization, Wiley, Chichester, 1997.

[10] M. Dell'Amico, S. Martello, The k-cardinality assignment problem, Discrete Appl. Math. 76 (1997) 103–121.

[11] J. Edmonds, Systems of distinct representatives and linear algebra, J. Res. Natl. Bur. Standards 71B (1967) 241–245.

[12] M. Ehrgott, J. Freitag, H.W. Hamacher, F. Maffioli, Heuristics for the k-cardinality tree and subgraph problem, Asia Pacific J. Oper. Res. 14 (1) (1997) 87–114.

[13] M. Ehrgott, H.W. Hamacher, F. Maffioli, Fixed cardinality combinatorial optimization problems—a survey, Report in Wirtschaftsmathematik 56 Fachbereich Mathematik, Universität Kaiserslautern, 1999.

[14] L.R. Foulds, H.W. Hamacher, J.M. Wilson, Integer programming approaches to facilities layout models with forbidden areas, Ann. Oper. Res. 81 (1998) 405–417.

[15] A. Frieze, M. Jerrum, Improved approximation algorithms for max k-cut and max bisection, Algorithmica 18 (1997) 67–81.

[16] G. Galbiati, F. Maffioli, On the computation of Pfaffians, Discrete Appl. Math. 51 (1994) 269–275.

[17] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979.

[18] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems, Theoret. Comput. Sci. 1 (1976) 237–267.

[19] M.X. Goemans, Semidefinite programming in combinatorial optimization, Math. Programming 79 (1997) 143–161.

[20] M.X. Goemans, D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, J. ACM 42 (1995) 1115–1145.

[21] N. Guttmann-Beck, R. Hassin, Approximation algorithms for minimum k-cut, Algorithmica 27 (2) (2000) 198–207.

[22] D. Karger, Minimum cuts in near-linear time, http://theory.lcs.mit.edu//∼karger.

[23] R.M. Karp, E. Upfal, A. Widgerson, Constructing a perfect matching is in Random NC, Combinatorica 6 (1) (1986) 35–48.

[24] P. Lancaster, M. Tismenetsky, The Theory of Matrices, Academic Press, Orlando, 1985.

[25] L. Lovasz, On determinants, matchings and random algorithms, Fund. Comput. Theory 79 (1979) 565–574.

[26] G.I. Orlova, Y.G. Dorfman, Finding the maximum cut in a graph, Eng. Cybernet. 10 (1972) 502–506.

[27] M. Padberg, G. Rinaldi, An efficient algorithm for the minimum capacity cut problem, Math. Programming 47 (1990) 19–39.

[28] S. Poljak, F. Rendl, Nonpolyhedral relaxations of graph-bisection problems, SIAM J. Optim. 5 (1995) 467–487.
[29] H. Saran, V. Vazirani, Finding k-cuts within twice the optimal, in: Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science, IEEE Press, Piscataway NJ, 1991, pp. 743–751.
[30] J.T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, J. Assoc. Comput. Machinery 27 (1980) 701–717.
[31] M. Stoer, F. Wagner, A simple min-cut algorithm, J. ACM 44 (1997) 585–591.
[32] H. Wolkowicz, R. Saigal, L. Vandenberghe, Handbook of Semidefinite Programming: Theory, Algorithms, and Applications, Kluwer Academic Publishers, Dordrecht, 2000.