Discrete Optimization 25 (2017) 175–188

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Improved compact formulations for a wide class of graph partitioning problems in sparse graphs

Dang Phuong Nguyen[a,*], Michel Minoux[b], Viet Hung Nguyen[b], Thanh Hai Nguyen[a], Renaud Sirdey[a]

[a] CEA, LIST, Embedded Real Time System Laboratory, Point Courrier 172, 91191 Gif-sur-Yvette, France
[b] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, 4 place Jussieu, Paris, France

ARTICLE INFO

Article history:
Available online 30 May 2016

Keywords:
Graph partitioning
Triangle inequality
SONET/SDH network design

ABSTRACT

Given an undirected connected graph $G = (V, E)$ where $|V| = n$ and $|E| = m$, we consider a wide class of graph partitioning problems, which includes as special cases several versions classically considered in the literature. These problems are to find a partition of the nodes in V into clusters satisfying several generic constraints (of set function type) on the clusters, in order to minimize the number (or the total weight) of the edges whose end-nodes do not belong to the same cluster. Partitions of V are often modeled by using compact integer programming formulations containing $O(n^3)$ triangle inequalities. The latter is the same whatever the sparsity of graph G could be, i.e. its size does not depend on m. In this paper, we show that one can reduce the size of the integer programming formulation to $O(nm)$ triangle inequalities. Moreover, it is shown that, when the additional constraints on the clusters satisfy some monotonicity property, the strength of the linear programming relaxation is preserved by this reduction. We present numerical experiments on two important special cases arising from applications to show the benefit in terms of computational efficiency of using the reduced formulation.

1. Introduction

The graph partitioning problem with set constraints investigated in the present paper can be generically defined as follows. Given an undirected connected graph $G = (V, E)$ where $V = \{1, \ldots, n\}, |E| = m$ and a length $l_e \in \mathbb{Z}_+$ is associated with each edge $e \in E$, find a partition of V into disjoint sets (or clusters) such that:

* Corresponding author.
E-mail addresses: dang-phuong.nguyen@cea.fr (D.P. Nguyen), michel.minoux@lip6.fr (M. Minoux), hung.nguyen@lip6.fr (V.H. Nguyen), thanhhai.nguyen@cea.fr (T.H. Nguyen), renaud.sirdey@cea.fr (R. Sirdey).

http://dx.doi.org/10.1016/j.disopt.2016.05.003
1572-5286/

- every cluster $C \subset V$ satisfies a constraint of the form $\mathcal{G}(y^C) \leq 0$ where y^C is the incidence vector of C in $\{0,1\}^n$ and $\mathcal{G} : \{0,1\}^n \to \mathbb{R}$ is a given monotone nondecreasing pseudoboolean function (Note that $\mathcal{G}$ can also be viewed as a set function: $\mathcal{P}(V) \to \mathbb{R}$ which associates the real value $\mathcal{G}(y^C)$ with each subset $C \subset V$),
- the sum of the lengths of the edges having end-nodes in different clusters is minimized.

The above class of problems (denoted GPP-SC, where SC stands for "set constraints") is fairly general, and includes several special cases of practical interest in particular:

(1) The case when each cluster C is required to satisfy a node weight constraint of the form $\sum_{v \in C} w_v \leq W$ where w_v for all $v \in V$ are given *nonnegative* node weights and W is a given upper limit on the total node weight of the cluster. This corresponds to considering the linear constraint $\mathcal{G}(y^C) \leq 0$ where $\mathcal{G}$ is the nondecreasing pseudoboolean function defined by $\sum_{v \in V} w_v y_v^C - W \leq 0$. Note that this special case of GPP-SC is the classical version of the graph partitioning problem defined in Garey and Johnson's book [1] which is known to be NP-hard [2]. In this paper, we will refer to this special case as *the graph partitioning problem under knapsack constraints* (GPKC).

(2) The case when each cluster C is subject to a constraint of the form $\mathcal{G}(y^C) \leq 0$ where $\mathcal{G}$ is a given nondecreasing quadratic pseudoboolean function $\{0,1\}^n \to \mathbb{R}$ defined as:

$$\mathcal{G}(y^C) = \sum_{(u,v) \in E} t_{uv}(y_u^C + y_v^C - y_u^C y_v^C) - T \leq 0$$

for given $t_{uv} \in \mathbb{R}_+$ for all $(u,v) \in E$ and T a given positive constant. If one considers t_{uv} as the capacity of the edge (u,v), then $\sum_{(u,v) \in E} t_{uv}(y_u^C + y_v^C - y_u^C y_v^C)$ represents the total capacity over the edges having at least one end-node in the cluster C. The constraint limits this capacity to a constant T. This variant of graph partitioning arises as a subproblem of the SONET/SDH network design problem [3] and has been introduced in [4] under the name of *graph partitioning problem under capacity constraints*. In the present paper, this special case of GPP-SC will be denoted GPCC. It has been shown in [4] that GPCC is NP-hard.

Note that in the definition of GPP-SC, the number of clusters is not bounded and it is part of the output of the optimization process. The graph partitioning problem with unbounded number of clusters was considered by several authors such as Sørensen [5], Labbé et al. [6]. Due to the fact that all the lengths $l_e,\ e \in E$ are positive, the objective of minimization of the sum of the lengths of the edges having end-nodes in different clusters tends already to limit the number of clusters in the optimal partition. But in many cases, it is interesting to limit the number of clusters to some given constant $k < n$ [7–9]. In Section 4, we discuss the extension of GPP-SC with an upper bound k on the number of clusters.

Most of the 0/1 programming formulation for representing the node partitions of a graph partitioning problem fall into one of the two following categories: the *Node–Cluster* model [3,4] which is based on the choice of decision variables x_{iq} which are equal to 1 iff node i belongs to cluster q; the *Node–Node* model [10–12,5,6,4] which is based on the choice of decision variables x_{ij} which are equal to 1 iff node i and node j are not in the same cluster. The former is a 0/1 quadratic program [3,4]. The latter is a 0/1 linear program and makes use of $O(n^2)$ variables and $O(n^3)$ triangle inequalities [10–12,5,6,4]. Approaches of mixing these models have been experimented in [7–9] where the authors make use of variables both from the Node–Cluster model and from the Node–Node model. This results in a mixed 0/1 program which has also $O(n^2)$ variables. Note that the latter can be strengthened by triangle inequalities [7]. Computational experiments show that, as a general rule, the continuous relaxation of the Node–Cluster model is weaker than the one of the

Node–Node model [4]. The Node–Cluster model is also highly symmetric making branch-and-bound based algorithms very inefficient. Reducing symmetries in this model has been investigated in [7,9,4].

A thorough and precise comparison between Node–Node and Node–Cluster models is out of the scope of the present paper, and we restrict here to investigating the Node–Node model for GPP-SC with special emphasis on the case of *sparse graphs*. As the number of triangle inequalities in this model is $O(n^3)$, it quickly becomes extremely large as n increases and even the relaxations turn out to be difficult to solve. Some authors have tried to overcome this difficulty by dualizing all or only a subset of the triangle inequalities via a Lagrangian approach [13]. Another possible approach considered in the present paper, is to try and reduce the number of triangle inequalities without weakening the relaxation. We will show that with only $O(nm)$ triangle inequalities, instead of $O(n^3)$, we can obtain an equivalent formulation, not only for the Node–Node model for GPP-SC, but also for its relaxations. Obviously, such a reduction opens the way to considerable improvement in case of sparse graphs where $m \ll \frac{n(n-1)}{2}$. Moreover, we will show that this reduction of the triangle inequalities remains valid for GPP-SC even in the presence of an upper limit constraint on the number of clusters.

The paper is organized as follows. In Section 2 we describe the Node–Node model for GPP-SC. We introduce in Section 3 a reduction of this model preserving the strength of the relaxation and featuring only $O(nm)$ triangle inequalities. In Section 4, this reduction is extended to the case when an upper bound on the number of clusters is imposed. In Section 5, we show how the improved compact formulation can be applied to both special cases GPKC and GPCC and computational experiments are discussed in Section 6 to show the benefit in terms of computational efficiency of using the reduced formulation.

2. Basic 0/1 programming model for GPP-SC

The basic 0/1 programming model for GPP-SC which is considered in the sequel can be described as follows. We introduce $\frac{n(n-1)}{2}$ binary variables x_{uv} for all the pairs of nodes $u, v \in V, u < v$, such that

$$x_{uv} = \begin{cases} 0 & \text{if } u \text{ and } v \text{ belong to the same cluster,} \\ 1 & \text{otherwise.} \end{cases}$$

The triangle inequalities together with the binary constraints are often used to define the partitions of a graph G [11,10]. They will be used here as part of the constraints for formulating GPP-SC. Denoting $\mathcal{T}$ the set of all triples (u, v, w) of nodes in V such that $u < v < w$ and E_n the set of all ordered pairs of node in V (i.e. $|E_n| = \frac{n(n-1)}{2}$), these constraints can be written as:

$$\text{(I)} \begin{cases} \forall (u,v,w) \in \mathcal{T} & \\ x_{uv} + x_{uw} \geq x_{vw} & (1) \\ x_{uv} + x_{vw} \geq x_{uw} & (2) \\ x_{vw} + x_{uw} \geq x_{uv} & (3) \\ x_{uv} \in \{0,1\} \qquad \forall (u,v) \in E_n. & \end{cases}$$

The number of triangle inequalities in system (I) is $3\binom{n}{3}$ whatever the value of m may be. It has been shown by Chopra [12] that for series–parallel graphs, the linear programming relaxation of (I) characterize completely the convex hull of the incidence vectors of all the possible partitions of V, i.e. the vectors that are the solutions of (I).

The Node–Node model for GPP-SC can be described as follows. Denoting $\mathbb{1}$ the n-vector with all components equal to 1, for every node $u \in V$ and for the cluster C that contains u, the incidence vector y^C

is equal to the n-vector $\mathbb{1} - \chi^u$, where χ^u is the n-vector with components defined for all $v \in V$ as:

$$\chi^u_v = x_{|uv|} = \begin{cases} x_{uv} & \text{if } u < v, \\ x_{vu} & \text{if } u > v, \forall v \in V. \\ 0 & \text{if } u = v, \end{cases}$$

Now the constraint $\mathcal{G}(y^C) \leq 0$ to be satisfied by every cluster C in the solution can be equivalently reformulated as the n separate constraints $\mathcal{G}(\mathbb{1} - \chi^u) \leq 0, \quad \text{for all } u \in V$. Denoting $g_u(x) = \mathcal{G}(\mathbb{1} - \chi^u), \quad \text{for all } u \in V$, the Node–Node model for GPP-SC is:

$$(\text{IP}) \begin{cases} \min & \displaystyle\sum_{(u,v)\in E} l_{uv} x_{uv} & & \\ \text{s.t.} & \forall (u,v,w) \in \mathcal{T} & & \\ & x_{uv} + x_{uw} \geq x_{vw} & & (1) \\ & x_{uv} + x_{vw} \geq x_{uw} & & (2) \\ & x_{vw} + x_{uw} \geq x_{uv} & & (3) \\ & g_u(x) \leq 0 & \forall u \in V & (4) \\ & x_{uv} \in \{0,1\} & (u,v) \in E_n. & \end{cases}$$

Observe that assuming $\mathcal{G}$ nondecreasing with respect to y^C implies that each pseudoboolean function $g_u : \{0,1\}^n \to \mathbb{R}_+$ should be *nonincreasing* with respect to x (note that g_u actually only depends on the subset of variables x_{ij} such that either $i = u$ or $j = u$). Let $(\overline{\text{IP}})$ denote the continuous relaxation of (IP). This formulation has $O(n^2)$ variables and $O(n^3)$ constraints.

Variants of the above graph partitioning problem with various additional constraints have already been considered by several authors [11,5,6,4]. These works proposed branch-and-bound or branch-and-cut algorithms which are all based on (IP) enhanced by using various types of valid inequalities. Consequently, these algorithms have to solve repeatedly the continuous relaxation $(\overline{\text{IP}})$ of (IP). As already mentioned above, the (IP) model does not take advantage of the possible sparsity of the graph under consideration: it always requires $O(n^3)$ constraints, even in the case $m \ll \frac{n(n-1)}{2}$. Although the latter is just a standard linear program with a polynomial number of constraints, it was reported in the literature (e.g. [13,4]) that it is rather difficult to solve in the presence of all the $3\binom{n}{3}$ triangle inequalities even for small values of n (i.e. $n \leq 20$). In the next section, we will show that we can obtain an equivalent formulation for (IP) and $(\overline{\text{IP}})$ with only $3m(n-2)$ triangle inequalities, instead of $3\binom{n}{3}$ as classically.

As will become apparent in the sequel, the above formulation (IP) is quite general and encompasses many possible variants of graph partitioning. Depending on the type of problem considered, the functions $g_u(x)$ involved in constraint (4) can be linear or nonlinear. In the latter case, solving (IP) with a MILP solver requires *linearization* of these functions (this will be the case of GPCC for instance, see Section 6.2 in the computational section). However, it should be stressed that all the results in the forthcoming Sections 3 and 4 concerning the reduction of the number of triangle inequalities will be applicable *independently of the linearization technique used.*

3. Improved 0/1 programming model for GPP-SC

We now process to show that, for the case of sparse graphs, the number of triangle inequalities can be significantly reduced while preserving the equivalence both for (IP) and $(\overline{\text{IP}})$. The idea is instead of considering all triples of $\mathcal{T}$, we only consider those such that at least one pair of nodes forms an edge in E. Precisely, let $\mathcal{T}'$ be the family of these triples, i.e. $\mathcal{T}' = \{(u,v,w) : u < v < w \in V$ and at least one of the edges $(u,v), (u,w)$ and $(v,w) \in E\}$. Then the reduced programming model is obtained from (IP) with the

triangle inequalities expressed only for the triples in $\mathcal{T}'$

$$
(\text{RIP}) \begin{cases} \min & \displaystyle\sum_{(u,v)\in E} l_{uv}x_{uv} & & \\ \text{s.t.} & \forall (u,v,w) \in \mathcal{T}' & & \\ & x_{uv} + x_{uw} \geq x_{vw} & & (5) \\ & x_{uv} + x_{vw} \geq x_{uw} & & (6) \\ & x_{vw} + x_{uw} \geq x_{uv} & & (7) \\ & g_u(x) \leq 0 & \forall u \in V & (8) \\ & x_{uv} \in \{0,1\} & (u,v) \in E & \\ & x_{uv} \in [0,1] & (u,v) \in E_n \setminus E. & \end{cases}
$$

It is clear that $|\mathcal{T}'| \leq m(n-2)$ thus the number of triangle inequalities in (RIP) is at most $3m(n-2)$. The continuous relaxation of (RIP) is denoted ($\overline{\text{RIP}}$). Due to this reduction of the triangle inequalities, ($\overline{\text{RIP}}$) will obviously be more interesting than ($\overline{\text{IP}}$) for LP solvers when applied to sparse graphs.

Given a point $x \in \mathbb{R}^{|E_n|}$, we note x_E the restriction of x on $\mathbb{R}^{|E|}$ i.e. the components of x whose index are in E. We will show the following main lemma.

Lemma 3.1. *Given a point $x^r \in [0,1]^{|E_n|}$ satisfying the inequalities (5–8). There always exists a point $x \in [0,1]^{|E_n|}$ satisfying the inequalities (1–4) such that $x_E = x^r_E$.*

To prove Lemma 3.1, let us specify how to construct x from x^r:

> Let us consider the graph $G = (V, E)$ where the edges in E are weighted by x^r_E and let p_{uv} denote the value of the shortest path in G between u and v with respect to weights x^r_E. Then $x \in [0,1]^{|E_n|}$ is defined as: $x_{uv} = \min\{1, p_{uv}\}$ for all $(u,v) \in E_n$.

Before showing that the resulting x is feasible for ($\overline{\text{IP}}$), let us prove the following proposition.

Proposition 3.2. *$x_{uv} \geq x^r_{uv}$ for all $(u,v) \in E_n$.*

Proof. Let us consider any pair (u,v) and suppose that $\{u \equiv u_0, \ldots, u_p \equiv v\}$ is the shortest path in G between (u,v) with respect to weights x^r_E. Let us consider successively the triangles composed of vertex u_0 and an edge of the shortest path as showed in Fig. 1. Since each of these triangles contains at least one edge in E, x^r satisfies the triangle inequalities (5–7) issued from these triangles. We want to show that $x^r_{|u_0u_p|} \leq \sum_{\eta=1}^{p} x^r_{|u_{\eta-1}u_\eta|}$. In fact, by induction:

- If $p = 1$, it is obvious that $x^r_{|u_0u_1|} = x^r_{|u_0u_1|}$.
- If $p = 2$, the inequality $x^r_{|u_0u_2|} \leq x^r_{|u_0u_1|} + x^r_{|u_1u_2|}$ is one of the triangle inequalities (5–7) for the triple (u_0, u_1, u_2) that is verified by x^r.
- Assume that the inequality was satisfied for $p = p_r$ for some $p_r \geq 2$, i.e. that

$$x^r_{|u_0u_p|} \leq \sum_{\eta=1}^{p_r} x^r_{|u_{\eta-1}u_\eta|}.$$

We will show that it is also true for $p = p_r + 1$. In fact, one of the triangle inequalities for the triple $(u_0, u_{p_r}, u_{p_r+1})$ gives

$$x^r_{|u_0u_{p_r+1}|} \leq x^r_{|u_0u_{p_r}|} + x^r_{|u_{p_r}u_{p_r+1}|}.$$

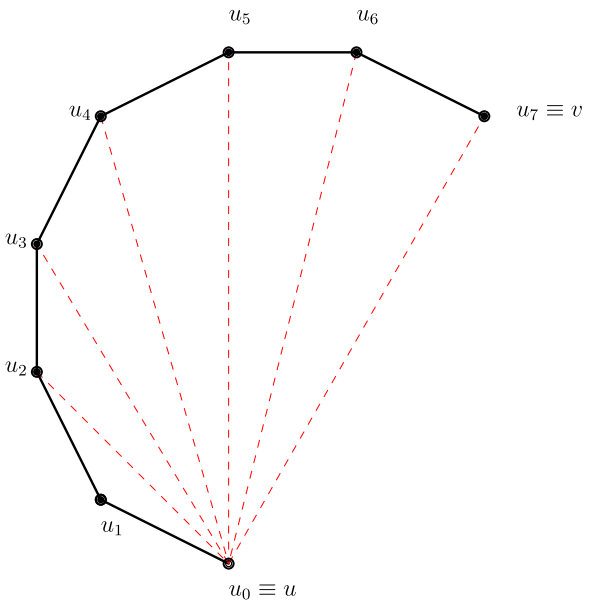

Fig. 1. A example of triangularization for a path uv.

By combining with the induction hypothesis we obtain

$$x^r_{|u_0u_p|} \leq \sum_{\eta=1}^{p_r+1} x^r_{|u_{\eta-1}u_\eta|}.$$

As $u_0 = u, u_p = v$ and $\sum_{\eta=1}^{p} x^r_{|u_{\eta-1}u_\eta|} = p_{uv}$, we deduce that $x^r_{uv} \leq \min\{1, p_{uv}\} = x_{uv}$. □

We are now ready to provide a proof for Lemma 3.1.

Proof of Lemma 3.1. We prove that the point x constructed as above from x^r verifies the inequalities (1–4) and in addition $x_E = x^r_E$.

We prove first that x satisfies the triangle inequalities (1–3). We only need to prove that x satisfies (1), by symmetry, x satisfies also (2) and (3). Let us consider any triple (u, v, w) in $\mathcal{T}$ and suppose that p_{uv}, p_{vw} and p_{uw} are respectively the shortest path lengths in G between $(u, v), (v, w)$ and (u, w) with respect to weights x^r_E. Note that the union of two shortest paths between (u, v) and (u, w) is also a path between (v, w), thus we have $p_{uv} + p_{uw} \geq p_{vw}$. Hence $\min\{1, p_{uv}\} + \min\{1, p_{uw}\} \geq \min\{1, p_{vw}\}$ which implies inequality (1).

We now show that x satisfies the inequalities (4). Indeed, as g is nonincreasing, we have $g_u(x) \leq g_u(x^r) \leq 0$ for all $u \in V$.

Finally, we show that $x_E = x^r_E$, i.e. we show that if (u, v) is an edge in E, the shortest path in G between u and v with respect to x^r_E is x^r_{uv}. This is shown by contradiction. Suppose that (u, v) is an edge in E and suppose that the shortest path in G between u and v with respect to x^r_E is $\{u \equiv u_0, \ldots, u_p \equiv v\} \neq (u, v)$. From the construction of x and Proposition 3.2 we have $p_{uv} = \sum_{\eta=1}^{p} x^r_{|u_{\eta-1}u_\eta|} > x^r_{uv}$. A contradiction. Hence we conclude that $x_E = x^r_E$. □

We now show the main result of this section:

Theorem 3.3. ($\overline{\text{IP}}$) *and* ($\overline{\text{RIP}}$) *have the same optimal value.*

Proof. The result of the theorem readily follows from Lemma 3.1 using the fact that $x_E = x^r_E$. □

A similar result holds for (IP) and (RIP) since in the construction of x from $x^r, x^r_E \in \{0,1\}^{|E|}$ leads $x \in \{0,1\}^{|E_n|}$.

Corollary 3.4. (IP) *and* (RIP) *have the same optimal value.*

4. Extension to the case when the number of clusters is bounded from above

In this section, it will be shown that part of the above results can be extended to the case when the number of clusters is bounded from above by a given constant. Note that the limitation on the number of clusters is somewhat conflicting with the monotone nondecreasing property of the cluster constraints in GPP-SC due to the fact that the number of clusters will increase as the number of nodes in the clusters is decreased. Therefore Theorem 3.3 seems to be non applicable in this case. Nevertheless, we prove in the following that the result of Corollary 3.4 is still applicable. To achieve this and exploiting some analogies with the analysis in [14], we introduce a vector $z \in \{0,1\}^{|V|}$ that is defined as follows: $z_i = 1$ if and only if i is the smallest node index in the cluster that contains i, in this case the node i is called a *representative*. Therefore z can be computed from x as, for all $i \in V$,

$$z_i = \begin{cases} 1 & \text{if } \sum_{j \in V,\ j<i} (1 - x_{ji}) = 0, \\ 0 & \text{otherwise.} \end{cases}$$

For a given positive integer k, the Node–Node model for GPP-SC with at most k clusters is then:

$$\text{(kIP)} \begin{cases} \min & \displaystyle\sum_{(u,v)\in E} l_{uv} x_{uv} & & \\ \text{s.t.} & \forall (u,v,w) \in \mathcal{T} & & \\ & x_{uv} + x_{uw} \geq x_{vw} & & \\ & x_{uv} + x_{vw} \geq x_{uw} & & \\ & x_{vw} + x_{uw} \geq x_{uv} & & \\ & g_u(x) \leq 0 & \forall u \in V & (8) \\ & z_u - x_{vu} \leq 0 & \forall u, v \in V,\ v < u & (9) \\ & z_u + \displaystyle\sum_{v=1}^{u-1} (1 - x_{vu}) \geq 1 & \forall u \in V & (10) \\ & \displaystyle\sum_{u=1}^{n} z_u \leq k & & (11) \\ & x_{uv} \in \{0,1\} & (u,v) \in E_n & \\ & z_u \in [0,1] & u \in V. & \end{cases}$$

Constraints (8) are the same as in (IP) and as in (RIP) where $g_u : \{0,1\}^n \to \mathbb{R}_+$ is a nonincreasing pseudoboolean function. Constraints (9) ensure that every cluster contains no more than one representative (i.e. no more than one node i such that $z_i = 1$). Constraints (10) guarantee that a cluster contains at least one representative. Finally, constraint (11) ensures that the number of clusters is no more than a given positive integer number k. The *reduced programming model* (kRIP) is obtained by replacing the set of triples $\mathcal{T}$ with $\mathcal{T}'$.

Lemma 4.1. *Given* $(x^r, z^r) \in \{0,1\}^{|E_n|} \times \{0,1\}^{|V|}$ *a feasible point for* (kRIP). *It is always possible to find a feasible point* $(x,z) \in \{0,1\}^{|E_n|} \times \{0,1\}^{|V|}$ *for* (kIP) *such that* $x_E = x^r_E$.

The main idea of the proof of this lemma is quite similar to the one given in Section 3, except for the construction of (x,z) from (x^r, z^r) which is now done as follows:

Assume that (x^r, z^r) is a feasible point of (kRIP). We can construct a partition P of V as follows. For node i from 1 to $n = |V|$, if $z_i = 1$ (i.e. i is a representative) then for all node j from $i+1$ to n, j is assigned to the same cluster as i if and only if j has not been assigned before and $x^r_{ij} = 0$. The point $(x, z) \in \{0,1\}^{E_n} \times [0,1]^V$ is then computed from the partition P as:

$$\text{for all } (i,j) \in E_n, \quad x_{ij} = \begin{cases} 0 & \text{if } i \text{ and } j \text{ belong to the same cluster,} \\ 1 & \text{otherwise} \end{cases}$$

and

$$\text{for all } i \in V, \quad z_i = \begin{cases} 1 & \text{if } i \text{ is the smallest node index in the cluster that contains the node } i, \\ 0 & \text{otherwise.} \end{cases}$$

To show that such a point (x, z) is feasible point for (kIP), we need the following proposition:

Proposition 4.2.

(1) *Node $j > i$ is assigned to the same cluster as the representative i if and only if i is the smallest index of a representative such that $x^r_{ij} = 0$.*
(2) *$z = z^r$.*
(3) *For every representative $i \in V$ (i.e. $z_i = 1$), $x_{ji} = x^r_{ji} = 1$, for all $j \in V$, $j < i$ and $x_{ij} \geq x^r_{ij}$, for all $j \in V$, $j > i$.*

Proof. The proof directly follows from the construction of (x, z). □

We are now ready to provide a proof for Lemma 4.1.

Proof of Lemma 4.1. We prove that (x, z) constructed as above from (x^r, z^r) is a feasible point of (kIP) and in addition that $x_E = x^r_E$.

The point (x, z) is deduced from the partition construction and thus x satisfies the triangle inequalities whereas z satisfies the representative constraints (9) and (10). As z^r satisfies the constraint (11) in (kRIP) and $z = z^r$ (Proposition 4.2 item (2)), z also satisfies this constraint. For every representative $i \in V$, $z_i = 1$, Proposition 4.2 item (3) guarantees that constraints (8) are satisfied for i, since $g_i(x^r) \leq 0$ and g_i is a nonincreasing function. For every node u that is not a representative, assume that u belong to the cluster with the representative i, we have $g_u(x) = g_i(x) \leq 0$ as they are defined on the same cluster. Thus all the constraints of (kIP) are satisfied by (x, z) and we can conclude that (x, z) is a feasible point for (kIP).

We now show that $x_E = x^r_E$ (i.e. $x_{uv} = x^r_{uv}$, for all $(u, v) \in E$). Indeed, for all $(u, v) \in E$:

- if $x^r_{uv} = 1, u$ and v are not in the same cluster since otherwise, there would exist a cluster with i as representative such that $x^r_{iu} = x^r_{iv} = 0$, and the triangle inequality $x^r_{iu} + x^r_{iv} \geq x^r_{uv}$ would be violated. Hence $x_{uv} = x^r_{uv} = 1$.
- if $x^r_{uv} = 0, u$ and v will be assigned to the same cluster. Indeed, we remark that for any representative i such that $x^r_{iu} = 0$, the triangle inequality $x^r_{iu} + x^r_{uv} \geq x^r_{iv}$ implies $x^r_{iv} = 0$, and reciprocally for any representative i such that $x^r_{iv} = 0$, the triangle inequality $x^r_{iv} + x^r_{uv} \geq x^r_{iu}$ implies $x^r_{iu} = 0$. Due to this remark, if i is the smallest representative index such that $x^r_{iu} = 0$, then i is also the smallest representative index such that $x^r_{iv} = 0$. By Proposition 4.2 item (1), u and v are then assigned to the same cluster with i as representative and hence $x_{uv} = x^r_{uv} = 0$. □

A similar result as Corollary 3.4 now holds for (kIP) and (kRIP).

Theorem 4.3. (kIP) *and* (kRIP) *have the same optimal value.*

Proof. The proof of the theorem follows trivially from Lemma 4.1 by the fact that $x_E = x^r_E$. □

5. Applications involving special cases of GPP-SC

5.1. Graph partitioning problem under knapsack constraints (GPKC)

The graph partitioning problem under knapsack constraints (GPKC) has been defined in Introduction section as a special case of GPP-SC. The problem contains knapsack constraints of type $\sum_{v\in C} w_v \leq W$ for each cluster C in the partition. In the Node–Node model, knapsack constraints can be written as:

$$\sum_{v\in V}(1 - x_{|uv|})w_v \leq W, \quad \forall u \in V. \ (12)$$

These constraints express that for all $u \in V$, the total weight of the cluster that contains u is bounded by a given constant W. Note that the total weight of each cluster is equal to sum of the weights of nodes contained in it. Note that in our formulation, all pairs in E_n are ordered thus the nodes u and v are in the same cluster or not, determined by $x_{|uv|}$, i.e. x_{vu} if $v < u, x_{uv}$ if $v > u$ and 0 if $u \equiv v$.

Clearly the left-hand side of knapsack constraints (12) (when being expressed with the sign "$\leq$") is a nonincreasing function and therefore the results of Sections 3 and 4 apply.

5.2. Telecommunication network design problem with capacity constraints (GPCC)

For various technological reasons, network operators often want to partition the node set V into clusters on which a certain network topology is imposed. For instance, in SONET/SDH optical networks, a common requirement is that every cluster is connected by a local network forming a cycle. Local networks are then interconnected by a secondary federal network which has one access node in each local network. Access nodes carry all the traffic internal to their local network and all the traffic exiting it but have a limited capacity. If we consider the traffic demand $t_{(u,v)}$ as the capacity of the edge (u, v), then the capacity of a local network (cluster) with node set $U \subset V$ follows our definition of capacity. As the topology and the capacity of local networks are imposed, the cost of these networks is almost fixed (except the cost of physical cables for building them) once the partition of V is determined. Thus, the objective of the problem could be focused on minimizing either the number of local networks (clusters) or the cost of the federal network. For the latter, an objective function often used it to minimize the total length of the edges in the interconnection with lengths given by the product of the traffic and the distance between nodes.

The SONET/SDH network design problem minimizing the number of local networks has been introduced in 2003 by Goldschmidt et al. [3] under the name SRAP problem. Bonami et al. [4] modeled this problem as a variant of the graph partitioning problem that they call graph partitioning under capacity constraints (GPCC) where the constraints on the weights of the clusters are replaced with constraints related to the edges incident to the nodes of each cluster. Suppose that, each edge $e \in E$ is assigned a capacity $t_e \in \mathbb{Z}_+$. For any subset $U \subseteq V$, we define the capacity of U as the sum of the capacities of the edges incident to at least one node of U, i.e. the edges in $E(U) \cup \delta(U)$ where $E(U)$ is the set of the edges with both end nodes in U and $\delta(U)$ is the set of the edges with exactly one end in U. The capacity constraint is to bound the capacity of each cluster by a given constant T. The objective function we consider is to minimize the total length of the edges in the interconnection (with weights given by the lengths l_e) between the clusters.

Let us rewrite the capacity constraints proposed in [3] in the Node–Node model:

$$\sum_{(v,w)\in E} t_{vw} x_{|uv|} x_{|uw|} \geq \sum_{(v,w)\in E} t_{vw} - T, \quad \forall u \in V. \tag{13}$$

It expresses the fact that the complement of the capacity of the cluster containing u should be greater than the total capacity of the graph minus T. This is equivalent to say that the capacity of the cluster containing u should be bounded by T. This constraint is in quadratic form and its left-hand side (when being expressed with the sign "$\leq$") is nonincreasing as the traffic t_{vw} are all positive. This can be thus integrated to the constraints (4) of (IP) and ($\overline{\text{IP}}$) and our reduction of triangle inequalities applies.

6. Computational experiments

In this section, we present computational results obtained with the improved compact formulation (RIP) as compared with the standard compact formulation (IP) for both variants GPKC and GPCC of GPP-SC. To carry out the computational comparisons, a sufficiently diverse set of test instances of relatively small size (typically less than 100 nodes) was needed. However (a) such instances for GPKC are only very scarce in the existing literature (e.g. only four instances with $n < 100$ can be found in the DIMACS data set [15]); (b) there is no data set corresponding to the sparse instances for GPCC. Therefore it was decided to generate the required test set for GPKC and GPCC in the following way:

- For a choice of graph type and for a given number of vertices n and number of edges m, the graph is firstly generated. To verify our results in the present paper, four well known sparse graph types are chosen to be generated that are: series–parallel graph, planar grid graph, toroidal grid graph and random graph. Except the planar grid graphs and the toroidal grid graphs that have fixed structure, the series–parallel graphs are generated using a generator named Task Graphs For Free (TGFF) [16] and the random graphs are generated by picking edges uniformly at random until the number of edges reaches m, and testing connectedness.
- The edge weights $t_{uv},\ (u, v) \in E$ and the node weights $w_u,\ u \in V$ are drawn independently and uniformly from the interval $[1, 1000]$.
- The upper bounds of the knapsack constraints W and of the capacity constraints T are chosen in such a way as to ensure that the generated instances will not be "too easy" to solve. More precisely we used METIS [17] to create an "optimal" partition of the graph with k clusters that we call the initial partition, we then do 1000 random perturbations of this partition. The bounds W and T are then chosen so that only 10% of these partitions correspond to feasible solutions.

All experiments are run on a machine with Intel Core i7-3630QM 2.40 GHz processor and 16 GB of RAM. The solver CPLEX 12.6 is used to solve respectively ($\overline{\text{IP}}$), (IP), ($\overline{\text{RIP}}$) and (RIP). CPLEX pre-solve is switched off as is classically done to avoid possible undesirable side-effects due to the uncontrolled behavior of this "black-box" procedure. All computation times are CPU seconds and the computation times of (IP) and (RIP) are subject to a *time limit of* 12 000 s while the computation times of ($\overline{\text{IP}}$) and ($\overline{\text{RIP}}$) are subject to a *time limit of* 3600 s.

6.1. Experimental results for GPKC

In Table 1 we report the results obtained for (IP) and (RIP) when applying on GPKC instances, denoted (IPKC) and (RIPKC) respectively and for their linear relaxations ($\overline{\text{IPKC}}$) and ($\overline{\text{RIPKC}}$). In our experiments, each instance belongs to one of four graph types: series–parallel graphs, planar grid graphs, toroidal grid graphs and random graphs. Series–parallel graphs are highly sparse while random graphs are denser graphs

Table 1
Computation results of (IPKC) and (RIPKC) for sparse graphs.

Graph types	n, m	(IPKC) CPU	($\overline{\text{IPKC}}$) CPU	(RIPKC) CPU	($\overline{\text{RIPKC}}$) CPU	Cont.Rlx GAP(%)
Series–parallel	22, 38	2.18	0.22	0.65	0.02	6.7
Series–parallel	25, 40	3.43	0.55	0.67	0.04	8.4
Series–parallel	27, 45	6.36	0.63	0.38	0.05	6.1
Series–parallel	30, 50	19.36	1.96	2.12	0.10	5.6
Series–parallel	35, 60	34.25	2.04	3.10	0.13	4.3
Series–parallel	40, 65	114.3	6.40	5.13	0.27	4.5
Series–parallel	45, 75	486.1	11.63	8.28	0.35	7.2
Series–parallel	50, 80	–	23.48	15.32	0.85	4.7
Series–parallel	60, 90	–	186.33	33.61	3.45	4.2
Series–parallel	80, 130	–	–	83.38	10.67	5.8
Series–parallel	100, 170	–	–	95.66	25.12	4.1
Series–parallel	120, 180	–	–	588.58	68.26	5.2
Series–parallel	130, 200	–	–	3276.3	98.27	2.7
Series–parallel	140, 210	–	–	8783.4	155.66	5.9
Planar grid 4 × 10	40, 66	120.3	5.78	5.41	0.36	6.3
Planar grid 5 × 10	50, 85	–	26.96	16.8	0.89	6.3
Planar grid 6 × 10	60, 104	–	174.31	61.4	1.72	5.7
Planar grid 7 × 10	70, 123	–	951.72	123.94	3.26	6.1
Planar grid 8 × 10	80, 142	–	–	427.7	12.19	7.5
Planar grid 9 × 10	90, 161	–	–	1478.2	15.77	8.0
Planar grid 10 × 10	100, 180	–	–	2147.2	26.51	5.4
Planar grid 11 × 10	110, 199	–	–	8044.8	40.78	7.3
Toroidal grid 4 × 10	40, 80	424.4	6.39	18.37	0.44	10.1
Toroidal grid 5 × 10	50, 100	–	36.68	117.29	1.03	10.6
Toroidal grid 6 × 10	60, 120	–	199.63	431.51	2.08	8.2
Toroidal grid 7 × 10	70, 140	–	1233.61	3361.7	3.66	9.7
Toroidal grid 8 × 10	80, 160	–	–	10 934.5	14.23	11.6
Random graph	22, 120	58.8	0.54	16.75	0.15	9.3
Random graph	25, 150	120.2	1.13	34.34	0.7	12.5
Random graph	27, 150	379.5	1.25	139.7	0.49	10.8
Random graph	30, 200	1436.8	2.79	458.5	0.98	8.4
Random graph	35, 250	–	7.97	945.6	2.65	10.5
Random graph	40, 280	–	14.5	3326.9	4.97	10.2
Random graph	45, 200	–	19.0	3884.2	1.47	12.7
Random graph	50, 200	–	35.88	9024.8	2.32	11.4

with $\frac{m}{n} \approx (4\text{–}8)$. As for each value of n, m, we have five instances, the first two columns in this table report the average CPU time to obtain the solution of (IPKC) and its linear relaxation ($\overline{\text{IPKC}}$), the third and fourth columns show the average CPU time to obtain the solution of (RIPKC) and its continuous relaxation ($\overline{\text{RIPKC}}$), and the last column gives the average value of the integrality gap.

As can be seen from Table 1, (RIPKC) and ($\overline{\text{RIPKC}}$) are much better than (IPKC) and ($\overline{\text{IPKC}}$) in terms of computation times. The difference becomes more significant as the number of nodes increases. With series–parallel graphs, the gain of (RIPKC) and ($\overline{\text{RIPKC}}$) is extremely clear as we report a reduction of solution time by a factor 10–50 for ($\overline{\text{RIPKC}}$) as compared with ($\overline{\text{IPKC}}$) and by a factor 3–60 for (RIPKC) as compared with (IPKC). The difference naturally decreases as m increases. For instance with random graphs, we report a reduction of solution time by a factor 3–15 for ($\overline{\text{RIPKC}}$) as compared with ($\overline{\text{IPKC}}$) and by a factor 3 for (RIPKC) as compared with (IPKC). There are also instances for which CPLEX is not even capable to solve the continuous relaxation with the classical formulation within the prescribed time-limits but succeeds at finding an optimal integer solution with our reduced formulation (series–parallel (80, 130) and bigger, planar grid 8 × 10 and larger, toroidal grid 8 × 10). With (RIPKC) we can solve exactly large instances, e.g., ($n = 140$) for series–parallel graphs, ($n = 110$) for planar grid graphs, ($n = 80$) for toroidal grid graphs and ($n = 50$) for random graphs.

Also it can be seen that the continuous relaxation ($\overline{\text{RIPKC}}$) is rather strong for GPKC, especially for instances related to series–parallel and planar grid graphs (6.5% on average). For toroidal grid and random graphs, the gaps is slightly bigger (10.5% on average).

Table 2
Comparison between (RIPKC) and (kRIPKC) for random sparse graphs. NF indicates non-feasible instances.

n, m	Instance number	(RIPKC)		((k* − 1)RIPKC)	((k* − 2)RIPKC)
		CPU	k*	CPU	CPU
30, 200	#1	276	3	NF(0.12)	NF(0.13)
	#2	402	5	NF(0.14)	NF(0.12)
	#3	452	5	1395	NF(0.15)
	#4	507	6	1558	NF(0.17)
	#5	558	8	1681	1847
35, 250	#1	733	4	NF(0.22)	NF(0.17)
	#2	911	5	NF(0.21)	NF(0.21)
	#3	906	5	3286	NF(0.38)
	#4	992	6	NF(0.25)	NF(0.20)
	#5	1071	7	4492	NF(0.28)
40, 280	#1	2954	4	NF(0.30)	NF(0.20)
	#2	3038	5	NF(0.29)	NF(0.31)
	#3	3353	6	9822	NF(0.35)
	#4	3827	7	9376	NF(0.35)
	#5	4158	9	>12 000	>12 000
45, 200	#1	3535	5	NF(0.39)	NF(0.33)
	#2	3642	5	NF(0.39)	NF(0.30)
	#3	3701	7	>12 000	NF(0.43)
	#4	3936	9	>12 000	NF(0.41)
	#5	4053	10	>12 000	>12 000

Compared efficiency with and without upper bound on the number of clusters

Computation experiments are also made to compare the efficiency of solving GPKC with and without upper bound on the number of clusters. To highlight the differences between them, the experiments are done in the following way. For any GPKC instance i, we first solve (RIPKC), the corresponding number of clusters of the optimal solution is denoted k_i^*. We then solve (kRIPKC) (i.e. (kRIP) applying on GPKC instances) for the same instance i in adding the upper bound on the number of clusters k_i. It is obvious that if $k_i \geq k_i^*$, the solutions of (RIPKC) and (kRIPKC) are identical. Computational results are shown in Table 2 for $k_i = k_i^* - 1$ and $k_i = k_i^* - 2$. For each value of n, m, five instances are solved. The third and the fourth columns in the table report the CPU time to obtain the optimal solution using (RIPKC) and the number of clusters in the optimal solution, the fifth and the sixth columns report the CPU time to obtain the optimal solution using ((k$^*-1$)RIPKC) and ((k$^*-2$)RIPKC) respectively. The acronym "NF" indicates a non-feasible instance, and the computation time to prove the infeasibility is shown in parenthesis.

As can be seen from Table 2, while all instances can be solved for (RIPKC), there are only part of instances can be solved for (kRIPKC) within the time limit of 12 000 s. Moreover, for the instances that can be solved for both models, (kRIPKC) is from 3 to 6 times slower than (RIPKC). A possible explanation of this is as follows. From the results in Table 2, it is seen that for all instances considered, the number of clusters k^* in the optimal solution to (RIPKC) is close to its minimum possible value. Indeed, for $k = k^* - 1$, about 50% of the instances become infeasible, and for $k = k^* - 2$, about 90% of the instances turn out to be infeasible. The observed increase in the computation times is thus due at least in part, to the fact that, when decreasing k, the instances become close to the boundary between feasibility and infeasibility.

6.2. Experimental results for GPCC

In this section we present the computation results for (IP) and (RIP) on GPCC instances, denoted (IPCC) and (RIPCC) respectively, and for their continuous relaxation ($\overline{\text{IPCC}}$) and ($\overline{\text{RIPCC}}$). We observe that the capacity constraint (13) is in non-convex quadratic form for which CPLEX cannot be used. We therefore have to transform the capacity constraints to linear form via some linearization techniques. We note that the results of the present paper remain applicable whatever linearization technique used. In the

Table 3
Computation results of (IPCC) and (RIPCC) for random graphs.

n, m	(IPCC) CPU	($\overline{\text{IPCC}}$) CPU	(RIPCC) CPU	($\overline{\text{RIPCC}}$) CPU	Cont.Rlx GAP(%)
22, 120	68.3	0.64	17.8	0.15	15.0
25, 125	156.9	1.1	37.3	0.32	10.4
27, 150	344.5	1.4	107.3	0.55	13.7
30, 200	1944.2	3.64	438.5	1.23	14.8
35, 200	–	9.56	1025.9	3.65	16.2
40, 250	–	20.67	3853.4	6.73	18.1
45, 200	–	28.39	3328.5	1.81	16.7
50, 200	–	45.31	11 038.6	2.88	15.2
60, 150	–	327.63	5291.4	4.36	12.3
70, 140	–	–	10 038.2	5.05	11.8
80, 120	–	–	8615.7	9.49	14.1

computational experiments presented below, we use the classical Fortet linearization [18], which is both simple and known to achieve a good compromise between the number of additional variables needed and the strength of the resulting relaxation. Application of this technique to GPCC leads to introduce new variables y_{uvw} to represent each product $x_{|uv|}x_{|uw|}$ thus the constraint (13) becomes:

$$\begin{cases} \displaystyle\sum_{(v,w)\in E} t_{vw}y_{uvw} \geq \sum_{(v,w)\in E} t_{vw} - C, \quad \forall u \in V \\ \max\left\{0, x_{|uv|} + x_{|uw|} - 1\right\} \leq y_{uvw} \leq \min\left\{x_{|uv|}, x_{|uw|}\right\}. \end{cases}$$

Hence our quadratic 0-1 formulations for GPCC become a mixed integer linear program that can be solved more easily by CPLEX. Note that the constraint (13) is the same for (IPCC), (RIPCC), ($\overline{\text{IPCC}}$) and ($\overline{\text{RIPCC}}$) so that the linearization does not influence the comparison between them.

Since usually the traffic matrix for a SONET/SDH optical networks are quite dense, we generate instances for GPCC with number of edges $\frac{m}{n} \approx (1.5\text{–}8)$, the results are shown in Table 3.

As we can see in Table 3, (RIPCC) and ($\overline{\text{RIPCC}}$) is much better than (IPCC) and ($\overline{\text{IPCC}}$) in terms of computation times. We report a reduction of solution time by a factor 3–18 for ($\overline{\text{RIPCC}}$) as compared with ($\overline{\text{IPCC}}$) and by a factor about 4 for (RIPCC) as compared with (IPCC). The difference is even more clear when increasing the number of nodes. The instances used in this section feature higher density than those in the previous section thus we can only solve those up to $n = 80$.

We finally observe that the quality of the continuous relaxation of the Node–Node model for GPCC is not as good as for the GPKC problem: the value of the gap is observed to be in the range 10.4%–18.1%.

7. Conclusions and perspectives

We have given an improved compact formulation for a wide class of graph partitioning problem using $O(nm)$ triangle inequalities instead of $O(n^3)$ in the classical formulation, while preserving equivalence, both in the integral version and in the relaxed version. Numerical experiments comparing our improved formulation with the classical formulation have been presented for two problems: the graph partitioning problem under knapsack constraints and the graph partitioning problem under capacity constraints. These numerical results have shown that solution times are reduced drastically from 3 to 50 times with our improved formulation. There are instances for which the LP solver is not even capable of solving the continuous relaxation with the classical formulation but succeeds at finding optimal integer solutions with our reduced formulation.

As a possible direction for future investigation, exhibiting further constraint structures, lending themselves to significant reduction of the number of triangle inequalities would be worth considering. Also we mention the search for additional polyhedral results related to the formulations of the present paper, possibly leading

to improved computational efficiency of branch-and-bound based procedures. Finally we note that carrying out an extensive comparison in terms of efficiency of Node–Cluster versus Node–Node models remains a potentially interesting subject for future work in the area.

Acknowledgments

We are grateful to the anonymous referees for their comments which led to an improved presentation of the paper.

References

[1] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman & Co, New York, NY, USA, 1979.
[2] L. Hyafil, R.L. Rivest, Graph partitioning and constructing optimal decision trees are polynomial complete problems. Technical Report Rapport de Recherche no. 33, IRIA—Laboratoire de Recherche en Informatique et Automatique Domaine de Voluceau, Rocquencourt 78150—Le Chesnay, France, 1973.
[3] O. Goldschmidt, A. Laugier, E.V. Olinick, Sonet/sdh ring assignment with capacity constraints, Discrete Appl. Math. 129 (1) (2003) 99–128. Algorithmic Aspects of Communication.
[4] P. Bonami, V.H. Nguyen, M. Klein, M. Minoux, On the solution of a graph partitioning problem under capacity constraints, in: A.R. Mahjoub, V. Markakis, I. Milis, V.T. Paschos (Eds.), ISCO, in: Lecture Notes in Computer Science, vol. 7422, Springer, 2012, pp. 285–296.
[5] M.M. Sørensen, Facet-defining inequalities for the simple graph partitioning polytope, Discrete Optim. 4 (2) (2007) 221–231.
[6] M. Labbé, F.A. Özsoy, Size-constrained graph partitioning polytopes, Discrete Math. 310 (24) (2010) 3473–3493.
[7] S. Holm, M.M. Sørensen, The optimal graph partitioning problem, Oper.-Res.-Spektrum 15 (1) (1993) 1–8.
[8] C. Ferreira, A. Martin, C. de Souza, R. Weismantel, L. Wolsey, The node capacitated graph partitioning problem: A computational study, Math. Program. 81 (2) (1998) 229–256.
[9] V. Kaibel, M. Peinhardt, M.E. Pfetsch, Orbitopal fixing, Discrete Optim. 8 (4) (2011) 595–610.
[10] F. Barahona, A. Mahjoub, On the cut polytope, Math. Program. 36 (2) (1986) 157–173.
[11] M. Grötschel, Y. Wakabayashi, A cutting plane algorithm for a clustering problem, Math. Program. 45 (1) (1989) 59–96.
[12] S. Chopra, The graph partitioning polytope on series–parallel and 4-wheel free graphs, SIAM J. Discrete Math. 7 (1) (1994) 16–31.
[13] A. Frangioni, A. Lodi, G. Rinaldi, New approaches for optimizing over the semimetric polytope, Math. Program. 104 (2–3) (2005) 375–388.
[14] Z. Ales, A. Knippel, A. Pauchet, On the polyhedron of the K-partitioning problem with representative variables. ArXiv e-prints, 2014.
[15] D.A. Bader, H. Meyerhenke, P. Sanders, D. Wagner (Eds.), Graph Partitioning and Graph Clustering. 10th DIMACS Implementation Challenge Workshop. February 13–14, 2012. Georgia Institute of Technology, Atlanta, GA, in: Contemporary Mathematics, vol. 588, American Mathematical Society and Center for Discrete Mathematics and Theoretical Computer Science, 2013.
[16] D. Rhodes, R. Dick, K. Vallerio, Task graphs for free. http://ziyang.eecs.umich.edu/~dickrp/tgff/.
[17] G. Karypis, V. Kumar, 2009. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0.
[18] R. Fortet, L'algebre de boole et ses applications en recherche operationnelle, Trab. de Estadistica 11 (2) (1960) 111–118.