Improved compact formulations for graph partitioning 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*

Abstract

Given a graph $G = (V, E)$ where $|V| = n$ and $|E| = m$. Graph partitioning problems on G are to find a partition of the vertices in V into clusters satisfying several additional constraints in order to minimize or maximize the number (or the weight) of the edges whose endnodes do not belong to the same cluster. These problems are usually modeled by a compact integer formulation using $O(n^3)$ triangle inequalities, whatever the value of m is. Thus the sparsity of the graph is not taken into account in the formulation. In this paper, we prove that when the additional constraints satisfying some monotonicity property, one can reduce the number of triangle inequalities to $O(nm)$ without deteriorate the integer formulation and its linear programming relaxation. We then present numerical experiments on two graph partitioning problems with respectively additional knapsack and capacity constraints to show the benefit of using the reduced formulation comparing with the original formulations.

Keywords: Graph Partitioning, Clique Partitioning, Triangle Inequality, SONET/SDH Network Design.

1. Introduction

The graph partitioning problem is a fundamental problem in combinatorial optimization. The common version of the problem as defined in Garey and Johnson [1] is as follows. Given an undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, weights $w_v \in \mathbb{Z}_+$ for each node $v \in V$, weights $l_e \in \mathbb{Z}_+$ for each edge $e \in E$ and a positive integer W, find a partition of V into disjoint sets (or clusters) $V_1, \ldots, V_k$ such that $\sum_{v \in V_j} w_v \leq W$ for $j = 1, \ldots, k$ while minimizing the sum of the lengths of the edges whose endpoints are in different clusters (i.e. the k-Cut defined by the partition). It was shown in [2] that the problem is NP-hard.

A slightly different variant of the above problem is obtained when the number of clusters is not imposed. The common version of this variant that we refer as the *basic graph partitioning problem* (BGPP) is simply to find a partition of V which minimizes or maximizes the sum of the weights of the edges whose end-nodes belonging to two different clusters. This problem with an objective of maximization turns out to be the so-called *clique partitioning problem* that was introduced by

Email addresses: dang-phuong.nguyen@cea.fr (Dang Phuong NGUYEN), michel.minoux@lip6.fr (Michel MINOUX), hung.nguyen@lip6.fr (Viet Hung NGUYEN), thanhhai.nguyen@cea.fr (Thanh Hai NGUYEN), renaud.sirdey@cea.fr (Renaud SIRDEY)

Grötschel and Wakabayashi in [3]. They gave a 0/1 (linear) formulation for the problem and also introduced several facet-defining inequalities for the convex hull of the incidence vectors associated with the 0/1 solutions. Basically, this 0/1 formulation can be described as follows. We have $\frac{n(n-1)}{2}$ binary variables x_{uv} for all the pairs of nodes $u, v \in V$ which say

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

The 0/1 formulation can be then written as follows:

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

where $\mathcal{T}$ is the set of all ordered triplets of V and E_n is the set of all ordered pairs of V (i.e. $|E_n| = \frac{n(n-1)}{2}$). Constraints (1-3) are called *triangle inequalities* and their number is $3\binom{n}{3}$ whatever the value of m may be. Later, Chopra [4] showed that for series-parallel graphs, (IP) coincide with its the linear programming relaxation. Variants of the basic graph partitioning problem with additional constraints have already been considered by several authors [5], [6], [7], [8]. These works proposed branch-and-bound ou branch-and-cut algorithms which are all based on (IP) enhanced by several additional class of valid inequalities. Consequently, these algorithms have to solve repeatedly the linear programming relaxation (LP) of (IP). Handling the $3\binom{n}{3}$ triangle constraints is thus crucial in solutions for (LP). Although the latter is just a standard linear program with a polynomial number of constraints, it was reported in the literature that it is rather difficult in the presence of all the $3\binom{n}{3}$ triangle constraints even for small values of n (i.e. $n \leq 20$). Indeed, the number of triangle constraints becomes quickly large as n increases and (LP) turn out to be difficult to solve even with best linear programming solvers. Several authors have tried to overcome this difficulty by dualizing all or only a subset of the triangle inequalities via a Lagrangian approach [9]. But beyond that, it would be interesting to be able to reduce intrinsically the number of triangle inequalities without weakening (LP), especially for sparse graphs (i.e. when $m = O(n)$). In this paper, we will show that with only $3m(n-2)$ triangle constraints, instead of $3\binom{n}{3}$ as classically, we can obtain an 0/1 formulation equivalent to (IP) , and more importantly an equivalent linear program to (LP). Moreover, we prove that the result holds even for a generic class of the basic graph partitioning problem with additional constraints satisfying some monotonicity property. We next provide several examples of applications showing that this generic class of graph partitioning problem investigated here may encompass many practical applications. Computational results are discussed to show the benefit of using the reduced formulation in term of computational efficiency.

2. Improved compact 0/1 formulation for the basic graph partitioning problems

Let us consider the formulation (IP) for the basic graph partitioning problem. This formulation has $O(n^2)$ variables and $O(n^3)$ constraints which are all linear except the 0/1 constraints. In

the literature, there exists an alternative 0/1 formulation for BGPP which models the belonging relationship of a node to a cluster where the clusters are fixed to the n possibles. This model, called *Node-Cluster* model (as opposed to (IP) which is a Node-Node model) is particularly described in [8] [7] which is quadratic 0/1 program and contain at least $O(n^3)$ variables in their linearized form. Hence, (IP) is the sole 0/1 linear formulation with $O(n^2)$ variables. Moreover, experiments in [7] have shown that the quality of the bound given by (LP) outperforms the one given the linear programming relaxation of the Node-Cluster model.

As we have remarked in the introduction, the model (IP) does not take into account the edges of G, in particular the value of m. There is no difference in solving the BGPP for dense graphs or for sparse graphs, since only the value of n is taken into account and we always need to deal with $O(n^3)$ triangle constraints. In this section, we will show that by taking into account the edges in E to the model, we can reduce the number of triangle constraints to $O(nm)$ (precisely to $3m(n-2)$) constraints while getting equivalences for (IP) and (LP). Moreover, we prove this result for a generic version of BGPP with additional constraints satisfying some monotonicity property. More precisely, let us consider all the vectors $x \in \{0,1\}^{\frac{n(n-1)}{2}}$ satisfying the triangle inequalities (1-3). Hence, these vectors x represent all possible partitions of the nodes in V into clusters. We consider a generic version of BGPP with additional constraints where x also satisfy some q constraints $g_i(x) \leq 0$ with $g_i : \mathbb{R}^{\frac{n(n-1)}{2}} \Rightarrow \mathbb{R}$ for $i = 1, \ldots, q$. The functions g_i for $i = 1, \ldots, q$ should be nonincreasing, i.e. for $x, x' \in \mathbb{R}^{\frac{n(n-1)}{2}}$ with $x \geq x'$, we have $g_i(x) \leq g_i(x')$. As we will see, several additional constraints for the graph partitioning problem considered in the literature satisfy this property.

Thus, we can rewrite the (IP) for the generic BGPP as follows. We want to minimize or maximize the sum of the weights of the edges whose end-nodes belonging to two different clusters :

$$(\text{OBJ}) = \min(\text{or } \max) \sum_{(u,v)\in E} l_{uv} x_{uv}$$

with subject to :

$$(\text{IP}) = \begin{cases} x_{uv} + x_{uw} \geq x_{vw} & (u,v,w) \in \mathcal{T} \quad (1) \\ x_{uv} + x_{vw} \geq x_{uw} & (u,v,w) \in \mathcal{T} \quad (2) \\ x_{vw} + x_{uw} \geq x_{uv} & (u,v,w) \in \mathcal{T} \quad (3) \\ g(x) \leq 0 & \quad (4) \\ x_{uv} \in \{0,1\} & (u,v) \in E_n \end{cases}$$

where $g : \mathbb{R}^{q \times \frac{n(n-1)}{2}} \to \mathbb{R}^q$ and $g(x) = \begin{pmatrix} g_1(x) \\ \ldots \\ g_q(x) \end{pmatrix}$. Let (LP) still denote the linear programming relaxation of this new version of (IP).

We present now our reduction of the triangle constraints for (IP) and (LP). The idea is instead of considering all triplets from V, 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 triplets, i.e. $\mathcal{T}' = \{u,v,w : u \neq v \neq w \in V$ and at least one of the edges uv, uw and $vw \in E\}$. Then reduced formulation is obtained from

(IP) with the triangle constraints expressed only for the triplets in $\mathcal{T}'$.

$$(\text{RIP}) = \begin{cases} x_{uv} + x_{uw} \geq x_{vw} & (u,v,w) \in \mathcal{T}' \quad (5) \\ x_{uv} + x_{vw} \geq x_{uw} & (u,v,w) \in \mathcal{T}' \quad (6) \\ x_{vw} + x_{uw} \geq x_{uv} & (u,v,w) \in \mathcal{T}' \quad (7) \\ g(x) \leq 0 & \quad (8) \\ x_{uv} \in \{0,1\} & (u,v) \in E \\ x_{uv} \in [0,1] & (u,v) \in E_n \backslash E \end{cases}$$

It is clear that $|\mathcal{T}'| \leq m(n-2)$ thus the number of triangle constraints in (RIP) is at most $3m(n-2)$.

Let (RLP) be the linear programming relaxation of (RP). Due to this reduction of the triangle constraints, (RLP) will obviously be more interesting than (LP) for LP solvers.

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 that its index are in E. We will show the following main theorem.

Theorem 2.1. *Given a point $x^r \in [0,1]^{E_n}$ verifying the inequalities (5-8). It always exist a point $x \in [0,1]^{E_n}$ verifying the inequalities (1-4) such that $x_E = x^r_E$.*

To prove Theorem 2.1, 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. Let us compute p_{uv} the value of the shortest path in G between u and v with respect to weights x^r_E. Then $x \in [0,1]^{E_n}$ can be defined as follows: $x_{uv} = \min\{1, p_{uv}\}$ for all $(u,v) \in E_n$.

Before showing that x is feasible for (LP), let us prove the following lemma.

Lemma 2.2. *$x_{uv} \geq x^r_{uv}$ for all $(u,v) \in E_n$.*

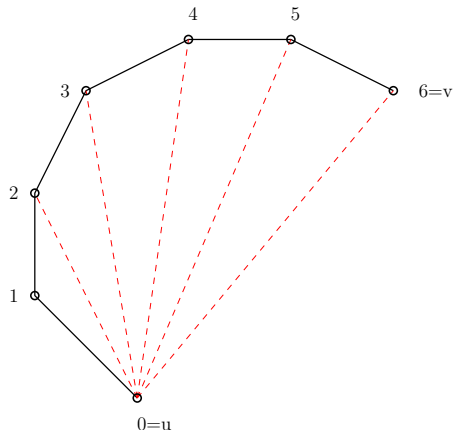

Figure 1: *A example of triangularization for a path uv.*

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

the vertex u_0 and an edge of the shortest path as showed in Figure 1. Since each of these triangles contains at least one edge in E, x^r satisfies the triangle constraints (5-7) issued from these triangles. We want to show that $x^r_{u_0,u_p} \leq \sum_{\eta=1}^{p} x^r_{u_{\eta-1},u_\eta}$. In fact, by recurrence:

- If $p = 1$, it's trivially that $x^r_{u_0,u_1} = x^r_{u_0,u_1}$.
- If $p = 2$, the inequality $x^r_{u_0,u_2} \leq x^r_{u_0,u_1} + x^r_{u_1,u_2}$ is one of the triangle constraints (5-7) for the triplet (u_0, u_1, u_2) that is verified by x^r.
- Assume that the inequality was verified for $p = p_r$ for some $p_r \geq 2$, i.e.

$$x^r_{u_0,u_p} \leq \sum_{\eta=1}^{p_r} x^r_{u_{\eta-1},u_\eta}$$

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

$$x^r_{u_0,u_{p_r+1}} \leq x^r_{u_0,u_{p_r}} + x^r_{u_{p_r},u_{p_r+1}}$$

  By combining with the recurrence hypothesis we obtain

$$x^r_{u_0,u_p} \leq \sum_{\eta=1}^{p_r+1} x^r_{u_{\eta-1},u_\eta}.$$

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 ready right now to provide a proof for Theorem 2.1.

Proof of Theorem 2.1.

We will 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 triplet (u, v, w) in $\mathcal{T}$ and suppose that p_{uv}, p_{vw} and p_{uw} are respectively the shortest path 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 (v, w) is also a path between (u, w), thus we have $p_{uv} + p_{vw} \geq p_{uw}$. Hence $\min\{1, p_{uv}\} + \min\{1, p_{vw}\} \geq \min\{1, p_{uw}\}$ which implies the inequality (1).

We now show that x satisfies the constraint (4). In fact, as g is nonincreasing, we have $g_i(x) \leq g_i(x^r) \leq 0$ for all $i = 1, \ldots, q$. Hence x satisfies the inequalities (8).

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}. We will proof that by absurd. Suppose that (u, v) is an edge in E and suppose the shortest path in G between u and v with respected to x^r_E is $\{u \equiv u_0, \ldots, u_p \equiv v\} \neq (u, v)$. From the construction of x and Lemma 2.2 we have $p_{uv} = \sum_{\eta=1}^{p} x^r_{u_{\eta-1},u_\eta} \geq x^r_{uv}$. Contradiction. Hence we conclude that $x_E = x^r_E$. □

Note that the objective function (OBJ) for (LP), (RLP), (IP) and (RP) depend only on the restriction of x on $\mathbb{R}^E$ (*i.e* x_E) thus the following corollary is direct consequence of Theorem 2.1.

Corollary 2.3. (LP) *and* (RLP) *have the same optimal value.*

A similar result holds for (IP) and (RIP).

Corollary 2.4. (IP) *and* (RIP) *have the same optimal value.*

Proof. We will only prove this corollary for (OBJ) = min $\sum_{(u,v)\in E} l_{uv}x_{uv}$ because the case (OBJ) = max $\sum_{(u,v)\in E} l_{uv}x_{uv}$ is totally similar.

Clearly that (RIP) is a relaxation of (IP) thus we have trivially that Val(IP)$\geq$ Val(RIP). It suffices for us to indicate that Val(IP)$\leq$ Val(RIP) and we have done. In fact, if x^r is a solution of (RIP), by construction we obtain x which verifies all the inequalities (1-4). In addition, $x^r_E \in \{0,1\}^E$ leads $x \in \{0,1\}^{E_n}$. Hence x is a feasible point of (IP) thus we have Val(RIP)$\geq$ Val(IP). □

3. Applications

3.1. Basic graph partitioning problem with knapsack constraints

As we introduced in introduction, the common version of graph partitioning problem (GP) often contains knapsack constraints (e.g. Garey and Johnson [1], Ferreira et al [10]) of type $\sum_{v\in V_j} w_v \leq W$ for each cluster V_j in the partition. In the Node-Node model, knapsack constraints can be written as:

$$\sum_{\substack{v\in V\\ v<u}}(1-x_{vu})w_v + w_u + \sum_{\substack{v\in V\\ v>u}}(1-x_{uv})w_v \leq W, \quad \forall u \in V \qquad (9)$$

These constraints express that $\forall u \in V$, the weight of cluster that contains u is bounded by a given constant W where the weight of each cluster equals to sum of 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_{vu} if $v < u$ and by x_{uv} if $v > u$ ($x_{uu} = 1,\ \forall u \in V$).

Clearly the left handside of knapsack constraints (9) (when being expressed with the sign "$\leq$")is a nonincreasing function and hence can be one of the constraints (4) in the Node-Node model. One can then apply our reduction for (IP) and (LP) with the presence of these constraints.

3.2. Telecommunication network design problem with capacity constraints

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 k-Cut 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. [8] under the name SRAP problem. Bonami et al. [7] 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 C. The objective function we consider is to minimize the k-Cut (with weights given by the lengths l_e) between the clusters.

Let us rewrite the capacity constraints proposed in [8] in the Node-Node model:

$$\sum_{\substack{(v,w)\in\mathcal{P}\\ v\neq u, w\neq u}} t_{vw} x_{uv} x_{uw} \geq \sum_{(v,w)\in\mathcal{P}} t_{vw} - C, \quad \forall u \in V \qquad (10)$$

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 C. This is equivalent to say that the capacity of the cluster containing u should be bounded by C. This constraint is in quadratic form and its left handside (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 (LP) and our reduction of triangle constraints applies.

4. Computational experiments

In this section, we present computational results for the improved compact formulation (RIP) comparing with the complete compact formulations (IP). The results are obtained from the solutions for two problems: the BGPP with knapsack constraints and the GPCC. We also report the gap between the optimal solution of the continuous relaxation and the optimal integer solution in order to prove the good quality of the Node-Node model for graph partitioning problems.

For a given number of vertices n and number of edges m, we generate five instances by picking edges uniformly at random (without recourse) until the number of edges reaches m. The edge weights and the node weights are drawn independently and uniformly from the interval $[1, 1000]$. For each instance, we calibrate the upper bounds of knapsack constraints W and of the capacity constraints C in order to ensure that the generated instances will be not "easy" to solved. To achieve this purpose, we used METIS[11] to estimate the solution with the number of clusters $k = 4$ (or $6,\ 8, 12$) that we call the initial partition, we then do 1000 perturbations of this partition. These bounds W and C then were chosen so that only 10% of these partitions are satisfied.

All experiments are run on a machine with Intel Core i7-3630QM 2.40GHz processors and 16 GiB of RAM. The solver CPLEX 12.6 is used to solve respectively (LP), (IP), (RLP) and (RIP) and to ensure that comparisons will be not biased we switch off CPLEX pre-solve. All computation times are CPU seconds and the computation times of (IP) and (RIP) are subject to a *time limit of 12000 seconds* while the computation times of (LP) and (RLP) are subject to a *time limit of 3600 seconds*.

4.1. Experiment results for BGPP with knapsack constraints

In Table 1 we report the results obtained with (IP), (RIP) and their continuous relaxation. In our experiments, each instance belongs to one of four graph types: series-parallels graph, planar grid graph, toroidal grid graph and random graph. Series-parallels graphs are highly sparse while random graphs are denser graphs with $m \approx (4 \text{ to } 8) \times n$. 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 (IP) and its continuous relaxation (LP), the third and fourth columns show the average CPU time to obtain the solution of (RIP) and its continuous relaxation (RLP), and the last column give the average GAP.

As can be seen from the table (RIP) and (RLP) are much better than (IP) and (LP) in term of computation times. The difference is more accentuated when increases number of nodes. With series-parallels graphs, the gain of (RIP) and (RLP) are extremely clear as we report a reduction of solution time from 10 to 50 times for (RLP) comparing with (LP) and from 3 to 60 times for (RIP) comparing with (IP). The difference naturally decreases as m increases . For instance with radom graphs, we report a reduction of solution time from 3 to 15 times for (RLP) comparing with (LP) and 3 times for (RIP) comparing with (IP). There are also instances for that CPLEX is even not capable to solve the continuous relaxation with the classical formulation but succeed to find optimal integer solution with our reduced formulation. With (RIP) we can solve large instances, e.g., ($n = 140$) for series-parallels graphs, ($n = 110$) for planar grid graphs, ($n = 80$) for toroidal grid graphs and ($n = 50$) for random graphs.

It can be seen also that the quality of the continuous relaxation of the Node-Node model is really good for BGPP with knapsack constraints. As we can see in the table, the gap is from 2.7% to 11.6%.

4.2. Experiment results for GPCC

In this section we present the computation results for the GPCC. We note that the capacity constraint (10) is in quadratic form for which CPLEX has difficulty in solving. Thus we use the classical linearization technique [12] to linearize these constraints. This technique is used in [7] to solve the GPCC problem. The main idea of this technique is to introduce variable y_{uvw} to represent each product $x_{uv}x_{uw}$ thus the constraint (10) becomes:

$$\begin{cases} \sum\limits_{\substack{(v,w)\in\mathcal{P} \\ v\neq u, w\neq u}} t_{vw} y_{uvw} \geq \sum\limits_{(v,w)\in\mathcal{P}} t_{vw} - C, \quad \forall u \in V \\ \max\{0, x_{uv} + x_{uw} - 1\} \leq y_{uvw} \leq \min\{x_{uv}, x_{uw}\} \end{cases}$$

Hence our quadratic 0-1 formulations for GPCC become a mixed integer program that can be solved more easily by CPLEX. Note that the constraint (10) is the same for (QIP), (RQIP), (QP) and (RQP) so that the linearization technique does not affect 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 $m \approx (4 \text{ to } 8) \times n$, the results are showed in the Table 2.

As we can see in Table 2 , (RQIP) and (RQP) is much better than (QIP) and (QP) in term of computation times. We report a reduction of solution time from 3 to 18 times for (RQP) comparing with (QP) and about 4 times for (RQIP) comparing with (QIP). The difference will be more clear when the increasing number of nodes. The instances used in this section are denser than those in the previous section thus we can only solve those of order $n = 50$.

We can see also that the quality of the continuous relaxation of the Node-Node model for GPCC remains good though they are worse than for the BGPP problem. The gap reported is from 10.4% to 18.1%. The reason is that the classical linearization technique that we applied on the capacity constraints (10), deteriorates the qualities of continuous relaxation.

5. Conclusion

We have given an improved compact formulation for a generic version of the basic graph partitioning problem using $O(nm)$ triangle inequalities which is equivalent with the classical one using $O(n^3)$ triangle inequalities. We have presented then numerical experiments comparing our improved formulation with the classical formulation for two problems: the basic graph partitioning problems 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 that CPLEX is even not capable to solve the continous relaxation with the classical formulation but succeed to find optimal integer solution with our reduced formulation.

For future works, it would be interesting to find a reduction of the triangle constraints for graph partitioning problems with all types of additional constraints.

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., Tech. Rep. R 033, IRIA. Rocquencourt (1973).

[3] M. Grötschel, Y. Wakabayashi, A cutting plane algorithm for a clustering problem, Math. Program. 45 (1) (1989) 59–96. doi:10.1007/BF01589097.

[4] S. Chopra, The graph partitioning polytope on series-parallel and4-wheel free graphs, SIAM J. Discret. Math. 7 (1) (1994) 16–31.

[5] M. M. Sørensen, Facet-defining inequalities for the simple graph partitioning polytope, Discrete Optimization 4 (2) (2007) 221 – 231.

[6] M. Labbé, F. A. Özsoy, Size-constrained graph partitioning polytopes, Discrete Mathematics 310 (24) (2010) 3473 – 3493. doi:http://dx.doi.org/10.1016/j.disc.2010.08.009.

[7] 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, Vol. 7422 of Lecture Notes in Computer Science, Springer, 2012, pp. 285–296.

[8] E. V. O. O. Goldschmidt, A. Laugier, Sonet/sdh ring assignment with capacity constraints, Discrete Applied Mathematics 129 (1) (2003) 99 – 128, algorithmic Aspects of Communication. doi:http://dx.doi.org/10.1016/S0166-218X(02)00236-6.

[9] A. Frangioni, A. Lodi, G. Rinaldi, New approaches for optimizing over the semimetric polytope, Mathematical Programming 104 (2-3) (2005) 375–388. doi:10.1007/s10107-005-0620-5.

[10] C. Ferreira, A. Martin, C. de Souza, R. Weismantel, L. Wolsey, The node capacitated graph partitioning problem: A computational study, Mathematical Programming 81 (2) (1998) 229–256. doi:10.1007/BF01581107.

[11] G. Karypis, V. Kumar, MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0 (2009).

[12] R. Fortet, L´algebre de boole et ses applications en recherche operationnelle, Trabajos de Estadistica 11 (2) (1960) 111–118. doi:10.1007/BF03006558.

Table 1: Computation results of (IP) and (RIP) for sparse graphs.

graph types	n,m	(IP)	(LP)	(RIP)	(RLP)	Cont.Rlx
		CPU	CPU	CPU	CPU	GAP(%)
series-parallels	22, 38	2.18	0.22	0.65	0.02	6.7
series-parallels	25, 40	3.43	0.55	0.67	0.04	8.4
series-parallels	27, 45	6.36	0.63	0.38	0.05	6.1
series-parallels	30, 50	19.36	1.96	2.12	0.10	5.6
series-parallels	35, 60	34.25	2.04	3.10	0.13	4.3
series-parallels	40, 65	114.3	6.40	5.13	0.27	4.5
series-parallels	45, 75	486.1	11.63	8.28	0.35	7.2
series-parallels	50, 80	-	23.48	15.32	0.85	4.7
series-parallels	60, 90	-	186.33	33.61	3.45	4.2
series-parallels	80, 130	-	-	83.38	10.67	5.8
series-parallels	100, 170	-	-	95.66	25.12	4.1
series-parallels	120, 180	-	-	588.58	68.26	5.2
series-parallels	130, 200	-	-	3276.3	98.27	2.7
series-parallels	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	-	-	10934.5	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

Table 2: Computation results of (QIP) and (RQIP) for random graphs.

n,m	(QIP)	(QP)	(RQIP)	(RQP)	Cont.Rlx
	CPU	CPU	CPU	CPU	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	11038.6	2.88	15.2