HAL
open science

Improved linearized models for Graph Partitioning Problem under Capacity Constraints

Viet Hung Nguyen, Michel Minoux

To cite this version:

Viet Hung Nguyen, Michel Minoux. Improved linearized models for Graph Partitioning Problem under Capacity Constraints. Optimization Methods and Software, 2017, 32 (4), pp.892-903. 10.1080/10556788.2016.1230209 . hal-01369135

HAL Id: hal-01369135

https://hal.sorbonne-universite.fr/hal-01369135v1

Submitted on 30 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

To appear in *Optimization Methods & Software*
Vol. 00, No. 00, Month 20XX, 1–13

Improved linearized models for Graph Partitioning Problem under Capacity Constraints

Viet Hung Nguyen[a*] and Michel Minoux[a]

[a]*Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6*
4 place Jussieu, Paris, France

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

We investigate a variant of the Graph Partitioning Problem with capacity constraints imposed on the clusters, giving rise to quadratic constraints in 0-1 formulations. Several compact linearized models of the problem are proposed and analyzed: a) a model featuring $O(n^3)$ binary variables which results from the application of the standard Fortet linearization technique; b) a more compact model featuring only $O(n^2)$ binary variables, obtained by linearization after reformulation of the quadratic constraints as bilinear constraints; c) a strengthened version of the latter model, still featuring $O(n^2)$ variables.
Computational experiments comparing the relative strength and efficiency of the various models on a series of test instances involving complete graphs with up to 50 nodes are reported and discussed.

Keywords: graph partitioning; capacity constraint; 0/1 quadratically constrained programming; linearization techniques; branch-and-bound.

AMS Subject Classification: 90C10;90C57;90C27;90C11

1. Introduction

The graph partitioning problem is a fundamental problem in combinatorial optimization. The basic version of the problem as defined in Garey and Johnson [6] (problem ND14) is as follows. Given an undirected graph $G = (V, E)$ with node set $V = \{1, \ldots, n\}$, weights $w_v \in \mathbb{Z}_+$ for each node $v \in V$, lengths $t_e \in \mathbb{Z}_+$ for each edge $e \in E$ and a positive integer K, find a partition of V into disjoint sets (or clusters) $V_1, \ldots, V_k$ such that $\sum_{v \in V_j} w_v \leq K$ for $j = 1, \ldots, k$ 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 [8] that the problem is NP-hard.
In this paper, we consider a variant of the graph partitioning problem that we 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. The lengths t_e for all $e \in E$ will be called *the link capacities* in our problem. For any node subset $U \subseteq V$, we define the capacity of U as the sum of the link 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. In our problem, the *capacity constraint* is to bound the capacity

*Corresponding author. Email: Hung.Nguyen@lip6.fr

of each cluster by a given constant C. The objective function considered is the same as in the definition in Garey and Johnson [6], i.e. to minimize the total link capacity of the k-Cut between the clusters. Note that as well as in the definition in Garey and Johnson, the number of the clusters k is not an input of our problem (it is part of the solution to the problem).

The GPCC problem has applications in the field of telecommunication network optimization, in particular it turns out to be a relevant model for optimum design of optical networks (see e.g. references [1, 7, 10]). In this application, the node set V corresponds to geographical sites and $t_{(u,v)}$ to the traffic demands between locations u and v. 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 traffics 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 $U \subset V$ agrees with 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 link capacity of the k-Cut.

The purpose of the present paper is to investigate and compare several 0-1 integer linear programming models for GPCC which can be qualified as *compact*, i.e. featuring a polynomial number of variables and constraints (by contrast, the model underlying the column generation approach in [10] which is a large scale set partitioning model with exponentially many columns, is essentialy noncompact). Note that two main compact 0-1 models for graph partitioning, namely Node-Cluster models and Node-Node models, have been investigated in the litterature where binary variables represent respectively relations of membership between nodes and clusters (case of the Node-Cluster model) and relations between nodes belonging to a same cluster (case of the Node-Node model). Existing works in the litterature make use of these models in various ways depending on which specific constraints are considered. This is the case of [9],[2], [3] and [4] which address variants of graph partitioning different from GPCC. In [9], the authors discuss the use of Node-Node model for balanced graph partionning and compare it with the SDP approach. In [2], the author considers several Node-Cluster models for balanced graph partitionning problems where the number of clusters and their size are constrained. In [3], [4], the Node-Cluster model also has been investigated, in particular, a comparision of the quadratic and linearized forms of Node-Cluster model has been discussed. Concerning GPCC, we can mention [7], [1] and [11]. In [1], the authors have compared the performance of Node-Node models and Node-Cluster models applied to GPCC. It is concluded that for dense graphs, the Node-Cluster model outperforms the Node-Node model in branch-and-bound algorithms in spite of a worse quality of continuous relaxation. However, for sparse graphs, we have shown in [11] that an improvement of the Node-Node model can help to outperform the Node-Cluster model when applied to GPCC. In the present paper, we aim at improving the best solution for GPCC presented in [1] and thus, we restrict to complete graphs and to models of Node-Cluster type. Section 2 discusses two basic Node-Cluster models for GPCC, namely:

a) a basic node-cluster model denoted (NC), which is a quadratic 0-1 program involving $O(n^2)$ variables, $O(n)$ quadratic constraints and $O(n^2)$ linear constraints aimed at breaking symmetry (a necessary ingredient in view of improving efficiency of Branch-and-Bound procedures);
b) a compact linear 0-1 model denoted (L-NC) deduced from (NC) by applying the well-known standard linearization [5], and featuring $O(n^3)$ 0-1 variables and $O(n^2)$ constraints. This is the Node-Cluster model for GPCC used in [1].

Clearly, in view of the large number of variables, the latter model, though linear, cannot be expected to be practically useful to handle instances of GPCC with significantly more than, say, 50-60 nodes. As an attempt at overcoming this limitation, we investigate in section 3 alternative compact linear models featuring $O(n^2)$ variables only; these are deduced by applying linearization after reformulating (NC) as a bilinear 0-1 programming problem, a technique close in spirit to the one proposed by Sherali and Smith in [13] which, as far as we know, has never been applied before to the GPCC problem. Our contributions in the context of the GPCC are twofold:

a) it is shown how to exploit some special structures present in the GPCC problem to derive more compact models; this gives rise to a first linear $O(n^2)$ model denoted (BL-NC);
b) we show how relaxations stronger than those which can be obtained by applying the standard approaches in [13] can be obtained, by proposing the efficient computation of improved bounds on the additional variables involved in the linearization; this gives rise to a strengthened version of the latter model, denoted (S-BL-NC).

Finally the various compact linear 0-1 formulations are compared computationally in section 4 on a series of test problems involving instances of complete graphs with up to 50 nodes. The comparison of (L-NC), (BL-NC) and (S-BL-NC) shows that the latter clearly outperforms the other two models both in terms of strength of the relaxations and in terms of computation time.

2. Integer Programming Models

Further on, we consider the GPCC when $G = K_n$ the complete graph of n nodes. Hence, for every ordrered pair (u, v) of nodes, there is an edge (u, v) and a capacity $t_{(u,v)}$. Note that this is not restrictive since the models can be applied to arbitrary graph $G = (V, E)$ by fixing $t_{(u,v)} = 0$ for every ordrered pair (u, v) of nodes such that $(u, v) \notin E$.

2.1 *Node-Cluster Model [7]*

We first present the model for GPCC given by Goldschmidt et al. in [7]. Note that the model was originally designed for the so called k-SRAP problem where the number of clusters in the partition is at most k but we adapt it here to the case when the number of clusters is not constrained (we later show how to modify it back to the k-SRAP problem). Also the model presented in [7] was a linearization of the quadratic model we present here using a standard technique that we recall in Section 2.3. Let $x_{ui} = 1$ if the node u is assigned to cluster i and $x_{ui} = 0$ otherwise. Define $T_u = \sum_{v \neq u} t_{(u,v)}$ as the total capacities of the edges incident to node u. The total capacities outside the clusters is

then equal to the total capacities minus the capacities inside the clusters, i.e.

$$\frac{1}{2}\sum_{u=1}^{n} T_u - \sum_{i=1}^{n}\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} t_{(u,v)} x_{ui} x_{vi}$$

The model can be written as follows:

$$\begin{aligned}
\min\ & \frac{1}{2}\sum_{u=1}^{n} T_u - \sum_{i=1}^{n}\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} t_{(u,v)} x_{ui} x_{vi} \\
\text{s. t.:}\ & \sum_{u=1}^{n} x_{ui} T_u - \sum_{u=1}^{n-1}\sum_{v=u+1}^{n} x_{ui} x_{vi} t_{(u,v)} \leq C && i = 1, \ldots, n && (1) \\
& \sum_{i=1}^{n} x_{ui} = 1 && u = 1, \ldots, n && (2) \\
& x_{ui} \in \{0, 1\}
\end{aligned}$$

The first constraints are the capacity constraints for each cluster i for all $i = 1, \ldots, n$. The second constraint imposes that each node is assigned to exactly one cluster. The objective function is to minimize the total capacity between the clusters.

2.2 *The Node-Cluster Model with symmetry breaking*

As noted by [10] the Node-Cluster model is highly symmetric (it is easy to see that the same partition has many representations in the model) and gives poor results in practice. Some constraints were proposed in [10] to remove some of the symmetry of the model. In [1], the authors have proposed two families of constraints that remove all the symmetry related to having several different representations for the same partition. They impose that if the cluster indexed by i is not empty then the node of index i should be the smallest index of a node contained in it by adding the constraints:

$$\begin{aligned}
x_{ui} &\leq x_{ii} && i = 1, \ldots, n-1 \text{ and } u = i+1, \ldots, n \\
x_{ui} &= 0 && i = 2, \ldots, n \text{ and } u = 1, \ldots, i-1
\end{aligned}$$

From now on, we shall omit the variables x_{ui} for $1 \leq u < i \leq n$ in our models as they are equal to 0. Moreover, note that we can now model the k-SRAP problem by simply bounding the number of non-empty clusters (i.e. $\sum_{i=1}^{n} x_{ii} \leq k$). In summary, the final Node-Cluster model with symmetry breaking is as follows:

$$\begin{aligned}
\min\ & \frac{1}{2}\sum_{u=1}^{n} T_u - \sum_{i=1}^{n}\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} t_{(u,v)} x_{ui} x_{vi} \\
\text{s. t.:}\ & \text{constraints } (1), (2) \\
& x_{ui} \leq x_{ii} && 1 \leq i < u \leq n && (3) \\
& x_{ui} \in \{0, 1\} && 1 \leq i \leq u \leq n
\end{aligned}$$

2.3 *A first compact linearized model with $O(n^3)$ variables*

2.3.0.1 . This is obtained by applying the classical linearization technique introduced by Fortet [5]. In this linearization, each product $x_{ui}x_{vi}$ for all $(u,v) \in E$ and $i = 1,\dots,\min(u,v)$ is replaced with a variable y_{uvi} and the following constraints are added in (NC): for all $(u,v) \in E, i = 1,\dots,\min(u,v)$,

$$y_{uvi} \leq x_{ui} \tag{4}$$
$$y_{uvi} \leq x_{vi} \tag{5}$$
$$y_{uvi} \geq x_{ui} + x_{vi} - 1 \tag{6}$$
$$y_{uvi} \geq 0 \tag{7}$$

The objective function and the capacity constraints can then be rewritten as

$$\min \frac{1}{2}\sum_{u=1}^{n} T_u - \sum_{i=1}^{n-1}\sum_{u=i}^{n-1}\sum_{v=u+1}^{n} t_{(u,v)}y_{uvi},$$
$$\sum_{u=i}^{n} x_{ui}T_u - \sum_{u=i}^{n-1}\sum_{v=u+1}^{n} y_{uvi}t_{(u,v)} \leq C \qquad \text{for all } i = 1,\dots,n \tag{8}$$

As $t_{uv} \geq 0$ for each $(u,v) \in E$ and as the constant C in the capacity constraints is positive, it is clear that solutions that are optimal for the objective function and comply with the capacity constraints will also maximize the value of y_{uvi} for all $(u,v) \in E$ and $i = 1,\dots,\min(u,v)$. Hence, in the linearized model of (NC), the constraints (6) and (7) which bound y_{uvi} from bottom can be omitted, thus leading to the follwing linearized model for (NC):

$$(\mathtt{L-NC})\begin{cases} \min & \frac{1}{2}\sum_{u=1}^{n} T_u - \sum_{u=i}^{n-1}\sum_{v=u+1}^{n} t_{(u,v)}y_{uvi} \\ & \text{s.t.: constraints } (8),(2),(3) \\ & y_{uvi} \leq x_{ui} \qquad 1 \leq i \leq u < v \leq n \\ & y_{uvi} \leq x_{vi} \qquad 1 \leq i \leq u < v \leq n \\ & x_{ui} \in \{0,1\} \qquad 1 \leq i \leq u \leq n \end{cases}$$

3. Towards more compact linearized models with $O(n^2)$ variables

3.1 *General principles*

We can see that in (NC) the quadratic terms $\sum_{u=i}^{n-1}\sum_{v=u+1}^{n} t_{(u,v)}x_{ui}x_{vi}$ for $i = 1,\dots,n$ are the same in both the objective function and in the capacity constraints. They can be rewritten as $\sum_{u=i}^{n-1} x_{ui}(\sum_{v=u+1}^{n} t_{(u,v)}x_{vi})$ for $i = 1,\dots,n$. The method introduced by Sherali and Smith in [13] applied to linearized these quadratic terms consists in two phases.

- In the first phase, the quadratic term $\sum_{u=i}^{n-1} x_{ui}(\sum_{v=u+1}^{n} t_{(u,v)}x_{vi})$ is restated as a

bilinear term $\sum_{u=i}^{n-1} x_{ui}\lambda_{ui}$ with

$$\sum_{v=u+1}^{n} t_{(u,v)}x_{vi} = \lambda_{ui}, \tag{9}$$

- In the second phase, the latter is linearized by introducing $z_{ui} = x_{ui}\lambda_{ui}$ and setting

$$\lambda_{\min}^{ui} x_{ui} \leq z_{ui} \leq \lambda_{\max}^{ui} x_{ui} \tag{10}$$
$$\lambda_{\min}^{ui}(1 - x_{ui}) \leq \lambda_{ui} - z_{ui} \leq \lambda_{\max}^{ui}(1 - x_{ui}) \tag{11}$$

where $\lambda_{\min\,/\,\max}^{ui} = \min\,/\,\max\{\sum_{v=u+1}^{n} t_{(u,v)}x_{vi} : x \in \bar{X}\}$ where $\bar{X}$ is a suitable relaxation of the subprogram of (NC) which does not involve the quadratic constraints.

PROPOSITION 3.1 *In the application of the Sherali-Smith approach to (NC),*

(i) for all $i = 1, \ldots, n$ and $u = i + 1, \ldots, n$, $\lambda_{\min}^{ui} = 0$ holds.
(ii) for a given $u = 1, \ldots, n$, we have $\lambda_{\max}^{ui} = \lambda_{\max}^{ui'}$ for all $i, i' <= u$.

Proof. (i) It is easy to see that $\lambda_{\min}^{ui} = 0$ for any $\bar{X}$ as we can always have a solution in $\bar{X}$ such that $x_{vi} = 0$ for all $v = u + 1, \ldots n$.

(ii) Given $1 < u \leq n$, let $\bar{X}$ be any relaxation of (NC) which may include the capacity constraints and let $1 \leq i < i' \leq n$, we have $\lambda_{\max}^{ui} = \max\{\sum_{v=u+1}^{n} t_{(u,v)}x_{vi} : x \in \bar{X}\}$ and $\lambda_{\max}^{ui'} = \max\{\sum_{v=u+1}^{n} t_{(u,v)}x_{vi'} : x \in \bar{X}\}$ as we can see that the contraints and the objective are totally separable and interchangeable for i and i'. Thus, $\lambda_{\max}^{ui} = \lambda_{\max}^{ui'} = \lambda_{\max}^{u}$ for any relaxation $\bar{X}$.

■

Hence, as $\lambda_{\max}^{ui} = \lambda_{\max}^{ui'}$ for all $i, i' <= u$ (Proposition 3.1(ii)), let $\lambda_{\max}^{u}$ denote this common value.

Remark 1 $\lambda_{\max}^{n} = 0$.

Proof. The remark obviously follows from the formula $\lambda_{\max}^{ni} = \max\{\sum_{v=n+1}^{n} t_{(n,v)}x_{vi} : x \in \bar{X}\}$. ■

Taking into account Proposition 3.1 and setting $h_{ui} = \lambda_{ui} - z_{ui}$, we can rewrite the linearization constraints (9), (10) et (11) as follows:

$$\sum_{v=u+1}^{n} t_{(u,v)}x_{vi} = h_{ui} + z_{ui}, \tag{12}$$
$$0 \leq z_{ui} \leq \lambda_{\max}^{u} x_{ui} \tag{13}$$
$$0 \leq h_{ui} \leq \lambda_{\max}^{u}(1 - x_{ui}) \tag{14}$$

We can then state the compact linearized (NC) model (called p-BL-NC for *preliminary bilinear NC*) as follows.

$$\text{(p-BL-NC)}\ \min \frac{1}{2}\sum_{u=1}^{n} T_u - \sum_{i=1}^{n}\sum_{u=i}^{n} z_{ui} \tag{15}$$

$$\text{s.t.:}\ \sum_{v=u+1}^{n} t_{(u,v)} x_{vi} = h_{ui} + z_{ui} \qquad i = 1..n, \quad u = i..n \tag{16}$$

$$\sum_{u=i}^{n} x_{ui} T_u - \sum_{u=i}^{n-1} z_{ui} \leq C \qquad i = 1..n \tag{17}$$

$$\text{constraints } (2), (3)$$

$$0 \leq z_{ui} \leq \lambda^u_{\max} x_{ui} \qquad i = 1..n, \quad u = i..n \tag{18}$$

$$0 \leq h_{ui} \leq \lambda^u_{\max}(1 - x_{ui}) \qquad i = 1..n, \quad u = i..n \tag{19}$$

$$x_{ui} \in \{0, 1\} \qquad i = 1..n, \quad u = i..n$$

Since optimal solutions of (p-BL-NC) should maximize as much as possible the variables z_{ui} for $i = 1, \ldots, n$ and for $u = i, \ldots, n$, we note that the variables h_{ui} play the role of slack variables in (16). The latter is the only constraint involving h_{ui} except the bound constraint (19). Hence, without loss of generality, we can eliminate the variables h_{ui} from the model, leading to the equivalent formulation.

$$\text{(g-BL-NC)}\ \min \frac{1}{2}\sum_{u=1}^{n} T_u - \sum_{i=1}^{n}\sum_{u=i}^{n} z_{ui}$$

$$\text{s.t.:}\ \sum_{v=u+1}^{n} t_{(u,v)} x_{vi} \geq z_{ui} \qquad i = 1..n, \quad u = i..n \tag{20}$$

$$\text{constraints } (17), (2), (3)$$

$$0 \leq z_{ui} \leq \lambda^u_{\max} x_{ui} \qquad i = 1..n, \quad u = i..n$$

$$x_{ui} \in \{0, 1\} \qquad i = 1..n, \quad u = i..n$$

where (g-BL-NC) stands for *generic bilinear NC*.
Note that in [13], the general forms of (p-BL-NC) and (g-BL-NC) have also been presented. The latter is only a relaxation of the former in the general case. In spite of this, the authors in [13] show that the general form of (g-BL-NC) outperforms the one of (p-BL-NC) in their numerical experiments. By above arguments, in the case of (NC), we can deduce the following stronger theoretical result.

PROPOSITION 3.2 *The optimal values of the linear programming relaxation of (p-BL-NC) and (g-BL-NC) coincide.*

3.2 *First estimates of the upper-bound parameters and the model (BL-NC)*

In this section, we discuss how to estimate the parameters $\lambda^u_{\max}$ for $u = 1, \ldots, n$ in (g-LB-NC). Recall that in the original method suggested in [13], $\lambda^{ui}_{\max} = \max\{\sum_{v=u+1}^{n} t_{(u,v)} x_{vi} | x \in \bar{X}\}$ where $\bar{X}$ is suitable relaxation of the subprogram of (NC) which does not contain the quadratic constraints. A first way of estimating $\lambda^{ui}_{\max}$ is to

simply pick $\bar{X} = \{0,1\}^n$ which implies $\lambda_{\max}^{ui} = \sum_{v=u+1}^{n} t_{(u,v)}$ for all $i = 1, \ldots, u$. We obtain then the following model.

$$
(\texttt{BL}-\texttt{NC}) \begin{cases}
\min \dfrac{1}{2}\displaystyle\sum_{u=1}^{n} T_u - \sum_{i=1}^{n}\sum_{u=i}^{n} z_{ui} & \\
\text{s.t.:} \quad \displaystyle\sum_{v=u+1}^{n} t_{(u,v)} x_{vi} \geq z_{ui} & i = 1..n, \quad u = 1..n \\
\text{constraints } (17), (2), (3) & \\
\displaystyle\sum_{i=1}^{n}\sum_{u=1}^{i-1} x_{ui} = 0 & \\
0 \leq z_{ui} \leq \left(\displaystyle\sum_{v=u+1}^{n} t_{(u,v)}\right) x_{ui} & i = 1..n, \quad u = 1..n \\
x_{ui} \in \{0,1\} & 1 \leq i < u \leq n
\end{cases}
$$

3.2.0.2 .

PROPOSITION 3.3 *The linear programming relaxation of (BL-NC) is weaker than the one of (L-NC).*

Proof. We can see that the linear programming relaxation of (L-NC) can be rewritten as follows.

$$
\begin{array}{lll}
\min & \dfrac{1}{2}\displaystyle\sum_{u=1}^{n} T_u - \sum_{i=1}^{n-1}\sum_{u=i}^{n-1}\sum_{v=u+1}^{n} t_{(u,v)} \min(x_{ui}, x_{vi}) & \\
\text{s.t.:} & \displaystyle\sum_{u=i}^{n} x_{ui} T_u - \sum_{u=i}^{n-1}\sum_{v=u+1}^{n} t_{(u,v)} \min(x_{ui}, x_{vi}) \leq C & i = 1..n \\
& \text{constraints } (2), (3) & \\
& x_{ui} \in [0,1] & 1 \leq i \leq u \leq n
\end{array}
$$

and the one of (BL-NC) is equivalent to

$$
\begin{array}{lll}
\min & \frac{1}{2}\sum_{u=1}^{n} T_u - \sum_{i=1}^{n}\sum_{u=1}^{n-1} \min(\sum_{v=u+1}^{n} t_{(u,v)} x_{vi}, \left(\sum_{v=u+1}^{n} t_{(u,v)}\right) x_{ui}) & \\
\text{s.t. :} & \displaystyle\sum_{u=i}^{n} x_{ui} T_u - \sum_{u=1}^{n-1} \min(\sum_{v=u+1}^{n} t_{(u,v)} x_{vi}, \left(\sum_{v=u+1}^{n} t_{(u,v)}\right) x_{ui}) \leq C & i = 1..n \\
& \text{constraints } (2), (3) & \\
& \displaystyle\sum_{i=1}^{n}\sum_{u=1}^{i-1} x_{ui} = 0 & \\
& x_{ui} \in [0,1] & 1 \leq i < u \leq n
\end{array}
$$

As we have $\sum_{v=u+1}^{n} t_{(u,v)} \min(x_{ui}, x_{vi}) \leq min(\sum_{v=u+1}^{n} t_{(u,v)} x_{vi}, \left(\sum_{v=u+1}^{n} t_{(u,v)}\right) x_{ui})$ where $x_{ui} \in [0,1]$ for all $1 \leq i < u \leq n$, (L-NC) has a tighter capacity constraint and a tighter objective than (BL-NC). ■

3.3 *Improved estimates of upper-bound parameters and the model (S-BL-NC)*

Clearly, to improve the quality of the n bounds $\lambda^u_{\max}$ (more precisely $n-1$ since $\lambda^n_{\max}=0$ (see Proposition 3.1)), we should take a set $\bar{X}$ more complicated than the set $\{0,1\}^n$. For each $1 \le u \le n-1$, by Propostion 3.1, we can estimate $\lambda^u_{\max}$ by fixing any $1 \le i \le u$ and by maximizing $\sum_{v=u+1}^n t_{vi}x_{vi}$ with the variables x_{vi} for $1 \le v \le n$, over a subset of constraints $\bar{X}^u_i$ of (NC) which involves these variables. It is interesting to remark that in view of the separable character of (NC), such a subset $\bar{X}^u_i$ will contain at most one quadratic capacity constraint. The following proposition tells us how to choose $\bar{X}^u_i$ as tight as possbile.

Proposition 3.4 *The set $\bar{X}^u_i$ defined as*

$$\bar{X}^u_i = \{\sum_{v=i}^n x_{vi}T_v - \sum_{v=i}^{n-1}\sum_{v'=v+1}^{n} x_{vi}x_{v'i}t_{(v,v')} \le C \text{ and } x_{vi} \in \{0,1\} \textit{ for } v{=}i,..,n\}$$

is as tight as possible for the estimation of $\lambda^u_{\max}$.

Proof. We can see that all the constraints of (NC) are separable in i, the second index of the variables x_{ui} except the constraints (2). But when i is fixed, the constraints (2) together with the constraints (3) can be reduced to just $x_{vi} \le 1$ for $v = i, \ldots, n$ which is implied by the constraints $x_{vi} \in \{0,1\}$. ■

The constraints in $\bar{X}^u_i$ can be linearized by classical linearization. As the 0/1 constraints are imposed, we obtain an equivalent set. Hence, denoting $\bar{\lambda}^u_{\max}$ the parameter estimated by maximizing $\sum_{v=u+1}^n t_{vi}x_{vi}$ over $\bar{X}^u_i$, $\bar{\lambda}^u_{\max}$ can be obtained by solving the following 0/1 linear program.

$$\begin{aligned}
&\bar{\lambda}^u_{\max} = \max \sum_{v=u+1}^{n} t_v x_v \\
&\text{s.t.: } \sum_{v=i}^{n} x_v T_v - \sum_{v=i}^{n-1}\sum_{v'=v+1}^{n} y_{vv'}t_{(v,v')} \le C \\
&y_{vv'} \le x_v \\
&y_{vv'} \le x_{v'} \text{ for all } i \le v < v' \le n \\
&x_v \in \{0,1\} \text{ for all } u \le v \le n
\end{aligned}$$

where x_v denotes x_{vi} for $v = i,..,n$ as i is fixed.
In the sequel, we denote (S-BL-NC) the model deduced from (BL-NC) in Section 3.2 by substituting the bounds $\sum_{v=u+1}^n t_{vi}$ with $\bar{\lambda}^u_{\max}$. Finally, we establish theoretical relationships between the continuous relaxations of (BL-NC) and (S-BL-NC) in the following proposition.

Proposition 3.5 *The linear programming relaxation of (S-BL-NC) is stronger than the one of (BL-NC).*

Proof. This simply follows from the fact that $\bar{X}^u_i \subset \{0,1\}^{n-u+1}$. ■

We will see in the next section that in fact in all numerical experiments that we have

conducted, the linear relaxation of (S-BL-NC) turns out to be signicantly stronger than the one of (L-NC). A possible explanation (though not a formal proof) of this is as follows. We can see that the sum $\sum_{v=u+1}^{n} t_{vi}x_{vi}$ is a part to be maximized of both the objective function and the capacity constraints. This fact together with the zero-one constraints (considered collectively) are taken into account in the computation of $\bar{\lambda}^{u}_{\max}$ while in classical linearization, only the zero-one constraints have been considered. In general, this advantage could be balanced by the fact that the classical linearization use better the zero-one constraints as it considers them invidually. But in the case of GPCC, it is particularly favorable for applying the Sherali-Smith's method as the quadratic terms are the same in both the objective function and the constraints.

4. Computational results

In this section, we present computational results comparing the three linearized models (L-NC), (BL-NC) and (S-BL-NC). First, we compare the three models in term of computational efficiency and strength of their linear programming relaxations. Next, we compare the efficiency of the three models at getting exact 0/1 optimal solutions by branch-and-bound.
The test set is composed of 9 randomly generated instances of GPCC with a number of nodes $n \in \{30, 40, 50\}$. These instances have been generated as follows.

- First, n points with coordinates (a_i, b_i) in 2-dimensional space are generated randomly by drawing independently a_i and b_i from the uniform distribution on the interval $[0, 1000]$.
- Next, a positive weight w_i drawn from uniform distribution on the interval $[w_{\min}, w_{\max}]$ and assigned to each of the n above points.
- Then the capacity of each link (i, j) is computed as:

$$t_{(i,j)} = \frac{w_i \times w_j}{(dist(i,j))^2} \times \beta_{ij},$$

  where $dist(i, j) = \sqrt{(a_i - a_j)^2 + (b_i - b_j)^2}$ (the euclidian distance between j and i) and β_{ij} is randomly chosen from the uniform distribution on $[\beta_{\min}, \beta_{\max}]$, $\beta_{\min}$ and $\beta_{\max}$ being to given positive numbers.
- Finally, the capacity upper-bound C is chosen experimentally in order to correspond to the hardest possible instances for the considered methods.

We note that this way of choosing the capacity values $t_{(i,j)}$ is inspired from the so-called "gravity model" which is considered as realistic in the fields of transportation and telecommunications [12]. For each value of $n \in \{30, 40, 50\}$, we generate three instances (labelled v_1, v_2, v_3 in the tables).
All experiments are run on a machine with i7-3820 Intel 3.60GHz 8 cores processor and 16 GB of RAM. The solver CPLEX 12.3 is used to solve all linear programs and mixed integer programs. CPLEX is set to run using 8 threads in deterministic mode and using the branch-and-bound algorithm only.

4.1 *Comparing continuous relaxations*

In this first experiment, we only seek to compute the continuous relaxations of (L-NC), (BL-NC) and (S-BL-NC). For each instance, we report the CPU time, the objective

function value and the integrality gap. The numerical results presented in Table 4.1

Table 4.1. Continuous relaxations comparisons

Ins.	(L-NC)			(BL-NC)			(S-BL-NC)		
	CPU	value	gap	CPU	value	gap	CPU	value	gap
$30v1$	4.11	44674.0	64.6%	0.47	29571	76.6%	0.48	52460.0	58.4%
$30v2$	4.55	20146.0	73.1%	0.45	15220	86.5%	0.44	63015.8	44.1%
$30v3$	3.58	58933.8	69.7%	0.43	42461	78.1%	0.44	82027.0	57.8%
$40v1$	12.37	6014.7	92.8%	0.75	5950	92.9%	0.82	47319.0	43.5%
$40v2$	16.34	28328.0	75.5%	0.78	21376	81.5%	0.80	76540.9	33.9%
$40v3$	15.19	14684.8	79.7%	0.74	10502	85.4%	0.83	44302.6	38.6%
$50v1$	29.02	6356.5	93.9%	1.50	3737	96.4%	1.32	31814.1	69.7%
$50v2$	28.55	1707.9	95.6%	1.21	1465	96.2%	1.29	16770.1	57.0%
$50v3$	37.66	24869.5	82.9%	1.52	17022	88.3%	1.57	64723.9	55.5%

confirm the theoretical result in Proposition 3.3 that the linear programming relaxation of (BL-NC) is slightly weaker than the one of (L-NC). In spite of this, we can observe that solving the linear programming relaxation of (BL-NC) is ten to twenty times faster than solving the one of (L-NC). We will see in the next section that this turns out to be a major advantage in branch-and-bound algorithms for solving GPCC. The linear programming relaxation of (S-BL-NC) presents all the advantages:

- It is the strongest in all the tests. For most instances, the integrality gap is divided by a factor 2 on average;
- The time for solving it is very short, nearly the same as for the linear programming relaxation of (BL-NC).

These good characteristics of (S-BL-NC) result in reducing signicantly the time for solving GPCC by branch-and-bound algorithms and thus leading to a significant increase in the size of the instances solved to optimality as compared with the tests in [1].

4.2 *Comparing exact solutions*

We now present results on the computations for exact solutions for GPCC using respectively the models (L-NC), (BL-NC) and (S-BL-NC). We solve the three models by using the CPLEX 12.3 solver. To ensure that comparisons will not be biased, we switch off the CPLEX pre-solve and inactivate all its generic MIP cuts. Thus the algorithm used to solve the three models is in fact a branch-and-bound algorithm based on its linear relaxation. We set the CPU time limit equal to 7200 seconds. For each model and instance, we report in Table 4.2 the CPU time and the number of nodes in the branch-and-bound search tree. It can be seen from the table that (S-BL-NC) is the most efficient model for branch-and-bound algorithms. The two "compact" models (BL-NC) and (S-BL-NC) generate more nodes in branch-and-bound trees than (L-NC) but as the time required for solving the linear relaxation is much smaller than for (L-NC), these two models can explore more branch-and-bound nodes and reach the exact optimal solutions more quickly than (L-NC). Note that we have included in the CPU time of (S-BL-NC) the additional time for computing the n bound parameters $\lambda^u_{\max}$ for $u = 1, \ldots, n$. Even with this, the reduction of CPU time is important by a factor up to 20 for the larger instances. It is interesting to note that although the (S-BL-NC) model turns out to be stronger than (L-NC) at the root node of the branch-and-bound tree as observed in Section 4.1, the

Table 4.2. Exact solution comparisons

Ins.	(L-NC)		(BL-NC)		(S-BL-NC)	
	CPU	Nodes	CPU	Nodes	CPU	Nodes
$30v1$	19.4	593	4.2	3485	8.3	2211
$30v2$	105.2	9654	17.1	28964	14.8	16860
$30v3$	21.0	733	2.4	2440	7.6	2076
$40v1$	2235.0	14906	342.1	195508	202.6	72989
$40v2$	7200 *	48172	4475.3	2780683	366.9	188515
$40v3$	1835.4	21647	493.4	403661	209.8	151315
$50v1$	4056.6	9671	339.9	73296	221.2	36636
$50v2$	619.8	1081	186.9	42442	156.2	14202
$50v3$	7200 **	10018	2482.2	531678	646.9	85575

** . Time limit exceeded, gap of the best known solution: 7.9%*
***. Time limit exceeded, gap of the best known solution: 36.3%*

former generates more nodes than the latter in branch-and-bound search trees. A possible explanation is that once computed, the parameters $\lambda^u_{\max}$ are fixed thoughout the branch-and-bound process and thus the linear relaxation of (BL-NC) may loose its advantage comparing with the one of (L-NC) once a number of binary variables x_{ui} have been fixed.

5. Conclusions and perspectives

Several models for the graph partitioning problems with cluster capacity constraints have been discussed and compared, highlighting the superiority of models based on bilinear reformulation of the quadratic constraints. Exact solutions of instances involving complete graphs with up to 50 nodes have been reported where for one of the largest instances, the proposed method yields the optimal solution within about 10 minutes while the classical linearized model only gives a feasible 0/1 solution with 36.3% residual gap after two hours. A key ingredient in achieving efficiency is the computation of stronger bounds for the additional variables linearizing the bilinear expressions. In the experiments reported, these bounds are computed only once at the root node of the branch-and-bound tree. An interesting direction for future investigation would be to recompute these bounds in the course of the branch-and-bound process in an attempt at further reducing the number of nodes explored. This is left to future research.

Funding

This work has been funded by the Gaspard Monge Program for Optimization and Operations Research

References

[1] Bonami, P., Nguyen, V., Klein, M., Minoux, M.: On the solution of a graph partitioning problem under capacity constraints. In: Mahjoub, A., Markakis, V., Milis, I., Paschos, V. (eds.) Combinatorial

Optimization, Lecture Notes in Computer Science, vol. 7422, pp. 285–296. Springer Berlin Heidelberg (2012)

[2] Boulle, M.:Compact Mathematical Formulation for Graph Partitioning. Optimization and Engineering. 5(3), 315–333 (2004)

[3] Fan, N., Pardalos, P.M.:Linear and quadratic programming approaches for the general graph partitioning problem. Journal of Global Optimization. 48(1), 57–71

[4] Fan, N., Pardalos, Multi-way clustering and biclustering by the Ratio cut and Normalized cut in graphs. Journal of Combinatorial Optimization. 23, 224–251 (2012)

[5] Fortet, R.: L'algèbre de boole et ses applications en recherche operationnelle. Trabajos de Estadistica 11(2), 111–118 (1960)

[6] Garey, M.R., Johnson, D.S.: Computers and intractability. W. H. Freeman and Co., San Francisco, Calif. (1979)

[7] Goldschmidt, O., Laugier, A., Olinick, E.V.: Sonet/sdh ring assignment with capacity constraints. Discrete Applied Math. 129, 99–128 (June 2003)

[8] Hyafil, L., Rivest, R.: Graph partitioning and constructing optimal deiscion trees are polynomial complete problems. Technical Report 33, INRIA Laboria (1973)

[9] Lisser, A., Rendl, F.,: Graph partitioning using linear and semidefinite programming. Mathematical Programming. 95(1), 91–101 (2003)

[10] Macambira, E.M., Maculan, N., de Souza, C.C.: A column generation approach for sonet ring assignment. Networks 47(3), 157–171 (2006)

[11] Nguyen, D.P., Minoux, M., Nguyen, V.H., Nguyen, T.H., Sirdey, R.: Improved compact formulations for a wide class of graph partitioning problem in sparse graphs. Discrete Optimization (to appear)

[12] Ogden, K.: The distribution of truck trips and commodity flow in urban area: A gravity model analysis. Transportation Research 12(2), 131–137 (1978)

[13] Sherali, H., Smith, J.: An improved linearization strategy for zero-one quadratic programming problems. Optimization Letters 1(1), 33–47 (2007)