Mathematical Formulations for the Balanced Vertex k-Separator Problem

Denis Cornaz*, Fabio Furini*, Mathieu Lacroix†, Enrico Malaguti‡, A. Ridha Mahjoub* and Sébastien Martin§
* LAMSADE, Université Paris-Dauphine
Place du Maréchal de Lattre de Tassigny, 75775 Paris, France
Email: denis.cornaz@dauphine.fr, fabio.furini@dauphine.fr, mahjoub@lamsade.dauphine.fr
† LIPN, Université Paris 13
99 Avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
Email: mathieu.lacroix@lipn.univ-paris13.fr
‡ DEI, Università di Bologna
Viale Risorgimento 2, 40136 Bologna, Italy
Email: enrico.malaguti@unibo.it
§ LCOMS, Université de Lorraine
Ile du Saulcy, 57045 Metz cedex
Email: sebastien.martin@univ-lorraine.fr

***Abstract*—Given an indirected graph $G = (V, E)$, a *Vertex k-Separator* is a subset of the vertex set V such that, when the separator is removed from the graph, the remaining vertices can be partitioned into k subsets that are pairwise edge-disconnected. In this paper we focus on the Balanced Vertex k-Separator Problem, i.e., the problem of finding a minimum cardinality separator such that the sizes of the resulting disconnected subsets are balanced. We present a compact Integer Linear Programming formulation for the problem, and present a polyhedral study of the associated polytope. We also present an Exponential-Size formulation, for which we derive a column generation and a branching scheme. Preliminary computational results are reported comparing the performance of the two formulations on a set of benchmark instances.**

I. Introduction

Graph partitioning (GP) is a fundamental family of problems with applications in many areas. For this, it has received within the past few years a great attention from both theoretical and practical points of view. The graph partitioning problems for an undirected graph $G = (V, E)$ ask for a partition of the vertex set V into pairwise disjoint subsets with some additional properties. For instance, given a positivie integer k, the k-way partition problem consists in partitioning the vertex set into k elements while minimizing the number of edges whose ends belong to different elements of the partition. In the *k-terminal way version*, k vertices are also given and each element of the searched partition must contain one of these vertices (terminal). Important classes of graph partitioning problems are the ones taking into consideration the equity and the uniformity of the subsets of vertices. They include the problem of partitioning the graph G into connected components of about the same size or the problem of finding a partition with an element of a fixed size. The GP problem has important applications particularly in parallel computing, scheduling in multi-processor systems, data analysis, risk management of energy and social networks, clustering and detection of cliques in social, pathological and biological networks, image processing and VLSI physical design (see, e.g., [2], [13]). The GP problems related to these areas have been widely studied in the literature, but typically the developed algorithms try to find heuristic or approximated solutions. Most exact methods are restricted to the bipartization case, and rely on the branch-and-bound framework.

978-1-4799-6773-5/14/$31.00

Another well-known family of problems are the Vertex k-separator problems. Given a simple undirected graph $G = (V, E)$ a *Vertex k-Separator (VKS)* in G, for $k \geq 2$, is a subset of vertices $V_0 \subseteq V$ such that $V \setminus V_0$ can be partitioned into k subsets $V_1, ..., V_k$ that are pairwise disconnected, i.e., there is no edge between two subsets V_i and V_j for all $i \neq j \in \{1, \ldots, k\}$. If the cardinality of each subset V_i, $i \in \{1, \ldots, k\}$, is larger than or equal to an integer $b(n)$, then V_0 is called a *Cardinality Constraint VKS (MCVKS)* [1],[3],[4]. If the difference of cardinality between all pairs of subsets V_i, V_j, $i \neq j \in \{1, ..., k\}$, is bounded by an integer q, then V_0 is called a *Balanced VKS (BVKS)*. The *CCVKS (resp. BVKS) Problem* (CCVKSP) (resp. (BVKSP)) consists in finding a CCVKS (resp. BVKS) of minimum cardinality. Both problems are NP-hard in general.

The BVKSP has interesting applications in many fields, for example in parallel simulation of physical movements. The movement of a physical system (electric circuit, motor for example) can be modeled by a differential algebraic equation system. To solve such a system using a parallel algorithm, it is necessary to decompose the associated Jacobian matrix. This consists in determining a minimum group of variables (called interface variables) whose removal allows to reorganize the rest of the Jacobian into k (diagonal) blocks. In other words, the remaining variables can be divided into k subgroups with no interaction between the elements of the different groups. The variables of the same block lead to a subproblem that is then treated by a processor. Minimizing the number of interface variables allows maximum reduction of the interaction time between the different processors and, accordingly, energy and computing time savings. This problem therefore boils down to the BVKSP in an appropriate graph associated with the

Jacobian matrix of the system.

The aim of this paper is to provide exact formulations for the BVKSP, as well as efficient approaches for solving it. In Section II, we provide a compact formulation and present a polyhedral study of the associated polytope. Then, in Section III, we present a formulation with an exponential number of variables but a polynomial number of constraints. We also give a column generation scheme to solve the linear relaxation and prove the effectiveness of this approach by showing that the subproblem is polynomial-time solvable. Section IV reports the preliminary experimental results we obtain by solving the two formulations, the first one by a branch-and-bound algorithm and the second by a branch-and-price algorithm. The rest of this introduction is devoted to notation and definitions.

Let K denote the set of integers $\{1, ..., k\}$. Given a simple undirected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, for an edge $uv \in E$, we say that u and v are *neighbors*. Then, the *neighborhood* of a vertex $w \in W$ is defined as follows: $N(w) = \{w' \mid w' \in V, ww' \in E\}$. We extend this notion to a set of vertices and define the *neighborhood* of W by $N(W) = \bigcup_{w \in W} N(w) \setminus W$, where $W \subseteq V$. We denote by $\tilde{N}(W)$ the set $W \cup N(W)$. For all $v \in V$, we denote by $\bar{N}'(v)$ the set of vertices in $V \setminus \{v\}$ such that they are not adjacent to v. $\bar{N}'(v) = V \setminus (N(v) \cup \{v\})$.

II. Compact Integer Linear Programming formulation

Consider a simple undirected graph $G = (V, E)$ and two integers k and q. For all vertices $v \in V$ and for all integers $i \in K$, let us associate a binary variable x_v^i which takes 1 if the vertex v is in V_i, and 0 otherwise. Remark that if $\sum_{i \in K} x_v^i = 0$ then the vertex v is in V_0.

The BVKSP is equivalent to the following integer linear program (P).

$$\max \sum_{i \in K} \sum_{v \in V} x_v^i \tag{1}$$

$$\sum_{i \in K} x_v^i \leq 1 \qquad \forall\, v \in V, \tag{2}$$

$$x_u^i + x_v^j \leq 1 \qquad \forall\, i \neq j \in K, \forall\, uv \in E, \tag{3}$$

$$\sum_{v \in V} (x_v^i - x_v^j) \leq q \qquad \forall\, i \neq j \in K, \tag{4}$$

$$x_v^i \geq 0 \qquad \forall\, i \in K, \forall\, v \in V, \tag{5}$$

$$x_v^i \in \{0, 1\} \qquad \forall\, i \in K, \forall\, v \in V. \tag{6}$$

Inequalities (2) states that vertex v cannot be assigned to both subsets V_i, V_j, where $i \neq j \in K$. Inequalities (3) prevent the ends of any edge to be assigned one to V_i, the other to V_j where V_i, V_j, $i \neq j \in K$. Inequalities (4) ensure the maximum unbalance between the cardinality of all subsets V_i, $i \in K$ cannot exceed q.

The convex hull associated with the program (P), defined as

$$P(G, k, q) = conv(\{x \in \{0, 1\}^{kn} |\ x \text{ satisfies } (2) - (5)\}).$$

In the following, we directly give different propositions.

Proposition 1: $P(G, k, q)$ is full dimensional if $q > 0$.

Proposition 2: Inequalities (5) define facets of $P(G, k, q)$.

Proposition 3: The inequality (4), associated with $v \in V$, defines a facet of $P(G, k, q)$ if and only if either $q > 1$ or for all edges $uv \in E$, there exists in $V \setminus \tilde{N}(\{u, v\})$ an independent set of cardinality greater or equal to $k - 1$.

Inequalities (3) do not define facet. In the following proposition, we present a new family of valid inequalities for the $P(G, k, q)$ which dominate (3). Moreover, Proposition 5 gives necessary and sufficient conditions for the new inequalities to be facet-defining.

Proposition 4: Let $uv \in E$ be an edge and $i \in K$. The inequality

$$x_u^i + \sum_{j \in K \setminus \{i\}} x_v^j \leq 1, \tag{7}$$

is valid for $P(G, k, q)$.

Proposition 5: The inequality (7) associated with $uv \in E$ and $i \in K$ defines a facet of $P(G, k, q)$ if and only if either $q > 1$ or G contains an independent set of cardinality greater or equal to $k - 1$ in $V \setminus \tilde{N}(\{u, v\})$.

We propose an inequality family, hereafter called W-*balanced inequalities*, that dominates the balanced inequalities (4) given in the formulation.

Proposition 6: Let $W \subseteq V$ be the set of vertices such that for all $v \in W$, either $|\bar{N}'(v)| < q + k - 1$ or $\bar{N}'(v)$ doesn't contain an independent set of cardinality $k - 2$. The following inequalities

$$\sum_{v \in V} x_v^t - \sum_{u \in V \setminus W} x_u^r \leq q \qquad \text{for all } t \neq r \in K, \tag{8}$$

are valid for $P(G, k, q)$.

We now give the necessary and sufficient conditions for the W-balanced inequalities to be facet-defining.

Proposition 7: Inequalities (8) define facets of $P(G, k, q)$ if and only if every vertex of W has a degree less than or equal to $n - q - 1$.

The linear relaxation of formulation (1)–(6) have a trivial fractional solution of value n, which is obtained by splitting each vertex between 2 subsets so as to constraints (3). A fractional solution of the same value is obtained even if the model is strengthened by inequalities (7); in the latter case each vertex is equally split among k subsets. Despite the weakness of the linear relaxation, the computational experiments of Section IV show that the strengthened constraints (7) are effective in helping a Mixed Integer Linear Programming solver in finding good integer solutions. A formulation with a stronger linear relaxation is presented in the next section.

III. Exponential-size Integer Linear Programming formulation

In this section, we derive an alternative formulation for the BVKSP having an exponential number of variables with respect to the input size. Let $\mathcal{S} = \{S \subseteq V\}$ be the family of all possible subsets of vertices of V. With every $S \in \mathcal{S}$ and every $i \in K$, we associate a binary variable z_S^i which is equal to 1 if S corresponds to V_i, and 0 otherwise.

The exponential-size model for the BVKSP reads as follows:

$$\max \sum_{i \in K} \sum_{S \in \mathcal{S}} |S| z_S^i \tag{9}$$

$$\sum_{i \in K} \sum_{S \in \mathcal{S}: v \in S} z_S^i \leq 1 \qquad \forall\, v \in V, \tag{10}$$

$$\sum_{i \in K} \sum_{S \in \mathcal{S}} a_S^{uv} z_S^i \leq 1 \qquad \forall\, (u,v) \in E, \tag{11}$$

$$\sum_{S \in \mathcal{S}} z_S^i = 1 \qquad \forall\, i \in K, \tag{12}$$

$$\sum_{S \in \mathcal{S}} |S| z_S^i - \sum_{S \in \mathcal{S}} |S| z_S^j \leq q \qquad \forall\, i \neq j \in K, \tag{13}$$

$$z_S^i \in \{0,1\} \qquad \forall\, i \in K, \forall\, S \in \mathcal{S}, \tag{14}$$

where:

$$a_S^{uv} = \begin{cases} 1 & u \in S \vee v \in S, \\ 0 & \text{otherwise.} \end{cases}$$

The objective function (9) maximizes the sum of the cardinalities of the selected subsets, i.e, it minimizes the size of the separator. Constraints (10) impose that each vertex appears in at most one subset. Constraints (11) impose that no edge exists between subsets, and (12) impose that exactly one set is selected for each V_i, $i \in K$. Constraints (13) impose the partitions to be balanced, and finally, constraints (14) impose the variables to be binary.

Model (9)–(14) has exponential size, thus we need a *column generation* procedure to solve its continuous relaxation. The model is initialized with a subset of the variables, and then the additional variables necessary to solve its linear relaxation are generated by separating the associated dual constraints (see, e.g., [3] for more details on this topic). Given the values of the dual variables $\lambda_v^*, \gamma_i^*, \pi_{uv}^*, \rho_{ij}^*$, associated with constraints (10), (11), (12) and (13), respectively, the separation of a dual constraint for $i \in K$ is to find a subset $S^* \in \mathcal{S}$ such that:

$$\sum_{v \in S^*} \lambda_v^* + \sum_{(u,v) \in E} a_{S^*}^{uv} \pi_{uv}^* + {\gamma_i}^* + |S^*| \sum_{j \in K:\ j \neq i} (\rho_{ij}^* - \rho_{ji}^*) < |S^*|.$$

If such a subset exists, the corresponding variable is added to the linear relaxation of model (9)–(14), and the procedure is iterated; otherwise, the linear relaxation is optimally solved.

Defining:

$$b_i^* = \sum_{j \in K:\ j \neq i} (\rho_{ij}^* - \rho_{ji}^*),$$

the separation associated with i is then to find a set S^* such that:

$$\sum_{v \in S^*} (\lambda_v^* + b_i^* - 1) + \sum_{(u,v) \in E} a_{S^*}^{uv} \pi_{uv}^* < -{\gamma_i}^*. \tag{15}$$

The separation problem associated with $i \in K$ can be tackled as an optimization problem, denoted as SP in the following. SP can be modeled as a Binary Linear Program using variables x_v, which determine whether vertex v belongs to S^*, and variables y_{uv}, which model coefficient $a_{S^*}^{uv}$, i.e., the fact then at least one between vertices u and v is in subset S^*:

$$x_v = \begin{cases} 1 & v \in S^*, \\ 0 & \text{otherwise,} \end{cases} \quad \text{for all } v \in V,$$

$$y_{uv} = \begin{cases} 1 & u \in S^* \vee v \in S^*, \\ 0 & \text{otherwise,} \end{cases} \quad \text{for all } (u,v) \in E.$$

The Binary Linear Program for the SP is given by:

$$\max \sum_{v \in V} \nu_v^* x_v - \sum_{(u,v) \in E} \pi_{uv}^* y_{uv} \tag{16}$$

$$y_{uv} \geq x_u \qquad (u,v) \in E, \tag{17}$$

$$y_{uv} \geq x_v \qquad (u,v) \in E, \tag{18}$$

$$x_v \in \{0,1\} \qquad v \in V, \tag{19}$$

$$y_{uv} \in \{0,1\} \qquad (u,v) \in E, \tag{20}$$

where:

$$\nu_v^* = -\lambda_v^* - b_i^* + 1. \tag{21}$$

Note that, since $\pi_{uv} \geq 0$, $(u,v) \in E$, and because the variables x_v ($v \in V$) are binary, we do not to have to explicitly impose the binary condition (20) for the variable vector y.

The SP can be interpreted as follows: given $G = (V, E)$, a profit ν_v^* for each $v \in V$ and a penalty $\pi_{uv}^* \geq 0$ for each $(u,v) \in E$, the problem aims at selecting the subset of vertices of maximum profit; if one vertex v is selected, the penalty π_{uv} associated with edges $(u,v) \in E$ is paid.

Without loss of generality, we may assume $\nu_v > 0$ for each $v \in V$. Actually, a vertex v with $\nu_v \leq 0$ can be removed together with its incident edges. The penalties associated with a removed edge, say (u,v), is then summed to the profit ν_u of vertex u. We can assume as well that $\pi_{uv} > 0$ for each $(u,v) \in E$, actually, if $\pi_{uv} = 0$ we can remove the edge.

Proposition 8: The SP is polynomial-time solvable.

Indeed, the transpose of the constraint matrix of model (16)–(20) has at most one coefficient of value 1 and one coefficient of value -1 per column. Thus, it is a *totally*

unimodular matrix (TUM) . Since the transpose of a TUM is a TUM, the constraint matrix is TU and the linear relaxation of the SP has integer optimal solution.

Finally, in order to obtain a stronger linear relaxation of model (9)–(14), we have to exclude the empty set from the family of all possible subsets of vertices of V. This can be obtained by solving the SP up-to n times, each time by explicitly imposing that one of the n vertices is included the subset s^*.

A. Branching scheme for the exponential-size formulation

When the optimal solution of the linear relaxation of Model (9)–(14) is fractional, a branching scheme is necessary in order to obtain an integer solution.

One option is to branch on the variables of formulation (1)–(6), i.e., the branching imposes that either a vertex v belongs to a partition i, or it does not. Thus, if the solution is fractional it exist i and v such that:

$$0 < \sum_{s \in S: v \in s} x_s^i < 1. \tag{22}$$

Then the branching generates two nodes by imposing either

$$\sum_{s \in S: v \in s} x_s^i = 0, \tag{23}$$

or

$$\sum_{s \in S: v \in s} x_s^i = 1. \tag{24}$$

Proposition 9: The branching scheme is complete.

This branching scheme does not affect the structure of the SP: in the first case vertex v is no longer available for partition i, in the latter, vertex v must be included into partition i and it is no longer available in partitions $j \in K,\ j \neq i$. The variables in Model (9)–(14) which are not consistent with the performed branching must then be removed.

IV. Experimental results

In this section, we assess the computational performances of the formulations discussed in this paper, denoted as the *Basic Model* ((1)-(6)), the *Strengthen Model* ((1),(2),(4)-(6),(7)) and the *Exponential-Size Model* ((9)-(14)). We first describe the *test problems* and then we report the *tables' discussion*. The results we are going to present are preliminary and we plan to extend the analysis in the journal version of this manuscript.

Test problems. Our primary aim is to investigate the strength of the different formulations and the dimension of the problems that could be solved to proven optimality by at least one of the formulations presented. In addition, two important parameters are investigated, i.e. k which corresponds to the number of different partitions and q which is the balanced coefficient controlling the maximum difference in terms of cardinality of the different partitions. Precisely, our preliminary experiments aim at investigating their impact on the CPU time. In our experiments we decided to test the following values of $k \in \{2,4,6,8,10\}$ and the following values of $q \in \{5, 10\}$, in their combinations. We decided to use four different instances from the DIMACS library , i.e. `myciel3`, `myciel4`, `myciel15`, `miles250`. These instances are widely used in the literature in order to test algorithms and formulations for problems based on graphs. They are a subset of instances already used by [1] for their computational tests on the MCVKS (where $k = 2$). The dimension of the instances is relatively low, i.e. the number of vertices goes from 11 to 128 and the number of edges goes from 20 to 774. Since solving these small problems is computationally challenging, this fact suggests that the BVKSP is a particularly difficult combinatorial optimization problem. In the journal version of the present manuscript we plan to enlarge the test bed of instances.

Tables' discussion. We run the tests on a PC with an Intel(R) Core2 Duo CPU E6550 at 2.33GHz and 2 GB RAM memory, under Linux Ubuntu 12, 64-bit. In Table I, the instance features are reported. In addition, we report the model size, in terms of variables and constraints of the compact formulations respectively. It is worth mentioning that the number of variables is the same for the Basic and the Strengthen Formulation; while the number of constraints is lower for the latter. In the remaining part of this section, we discuss two sets of experiments aiming respectively at comparing the LP-relaxation of the different formulations and the computational behaviour for solving the test problems to proven optimality.

For each run, we set a time limit of 3600 seconds, and we used `CPLEX` 12.6 [19] with default parameter settings, for solving the Basic and Strengthen Model; while we used SCIP [20] for the Exponential-Size Model. In case the time limit is reached we report "$\geq$ 1h". In case of memory limit instead we report `mem` .

In Table II we report the comparison between the Basic and the Strengthened Models. Analyzing the table, it is clear that a considerable speed up in terms of CPU time can be achieved using the Strengthened Model. The performance of both formulations is affected by the number of partitions k, thing which is not particularly surprising since k has a direct impact on the size of the formulations themselves. When $k = 10$ the CPU time considerably increases and for the largest instance a time limit of one hour is reached. On the contrary the absolute value of q does not seem to considerably affect the performances of the formulations.

In Table III we assess the performance of the Exponential-Size Formulation. We claim from the beginning that this formulation is still not competitive with the others and the reason we see is twofold. The implementation tested is still basic and the used solver is less competitive. In implementing the Branch-and-Price we used SCIP which is the best non-commercial solver which allows the direct implementation of a Column Generation procedure integrated with a Branching scheme. Despite the implementation is basic, the Exponential-Size Formulation is able to solve `myciel3` and `myciel4`. In addition the table stresses the point of strengths of this formulation, i.e., a stronger the LP relaxation than the one of the Basic and Strengthen Models, and the fact that the CPU time is less dependent on the number of partitions k. For these reasons we hope that an additional effort in improving the performance of the Exponential-Size Formulation could lead to improved results, outperforming the Compacts Models at least for some subclasses of instances.

The entries in the table I are:
Name : name of instance,
n : number of nodes,
m : number of edges,
k : number of sets in the partition,
q : balanced coefficient,
variables : number of variables in the compact formulation,
cons (2), (3), (6) : number of inequalities (2), (3), (6) in the basic formulation,
cons (2), (7), (6) : number of inequalities (2), (7), (6) in the strengthen formulation.

TABLE I. DESCRIPTION OF INSTANCES AND SIZE OF THE COMPACT FORMULATION

Instance	n	m	k	q	variables	cons (2), (3), (6)	cons (2), (7), (6)
myciel3	11	20	2	5	22	32	32
	11	20	2	10	22	32	32
	11	20	4	5	44	137	97
	11	20	4	10	44	137	97
myciel4	23	71	2	5	46	95	95
	23	71	2	10	46	95	95
	23	71	4	5	92	455	313
	23	71	4	10	92	455	313
	23	71	6	5	138	1103	464
	23	71	6	10	138	1103	464
	23	71	8	5	184	2039	619
	23	71	8	10	184	2039	619
	23	71	10	5	230	3263	778
	23	71	10	10	230	3263	778
myciel5	47	236	2	5	94	284	284
	47	236	2	10	94	284	284
	47	236	4	5	188	1469	997
	47	236	4	10	188	1469	997
	47	236	6	5	282	3602	1478
	47	236	6	10	282	3602	1478
	47	236	8	5	376	6683	1963
	47	236	8	10	376	6683	1963
	47	236	10	5	470	10712	2452
	47	236	10	10	470	10712	2452
miles250	128	774	2	5	256	903	903
	128	774	2	10	256	903	903
	128	774	4	5	512	4778	3230
	128	774	4	10	512	4778	3230
	128	774	6	5	768	11753	4787
	128	774	6	10	768	11753	4787
	128	774	8	5	1024	21828	6348
	128	774	8	10	1024	21828	6348
	128	774	10	5	1280	35003	7913
	128	774	10	10	1280	35003	7913

The entries in the table II are:
Name : name of instance,
k : number of sets in the partition,
q : balanced coefficient,
nodes : number of generated nodes in the Branch-and-Bound tree,
CPU : total CPU time in seconds,
$|W|$: size of node set to improve the balanced inequalities.

TABLE II. RESULTS FOR COMPACT FORMULATION

Instances			Basic model		Strengthen model		
Name	k	q	nodes	CPU	nodes	$\|W\|$	CPU
myciel3	2	5	14	0	12	1	0
	2	10	11	0	11	11	0
	4	5	82	0	21	11	0
	4	10	52	0	13	11	0
myciel4	2	5	47	0	28	0	0
	2	10	31	0	31	0	0
	4	5	1754	0	182	0	0
	4	10	118	0	47	2	0
	6	5	2482	1	142	0	0
	6	10	281	0	48	7	0
	8	5	2340	7	78	1	0
	8	10	1685	7	631	13	2
	10	5	2465	11	2000	7	6
	10	10	2264	13	627	23	2
myciel5	2	5	202	0	283	0	0
	2	10	144	0	117	0	0
	4	5	18294	26	536	0	1
	4	10	32742	37	1543	0	3
	6	5	47331	166	1530	0	9
	6	10	39642	104	2659	0	23
	8	5	379912	2485	3238	0	55
	8	10	150598	1061	4275	0	81
	10	5	193490	$>$ 1h	16929	0	319
	10	10	80910	mem	6366	0	181
miles250	2	5	15	0	9	0	0
	2	10	11	0	12	0	0
	4	5	3096	24	138	0	4
	4	10	3071	22	142	0	4
	6	5	3298	85	1674	0	96
	6	10	10817	185	428	0	26
	8	5	20710	1053	2464	0	343
	8	10	16056	606	3018	0	517
	10	5	30414	$>$ 1h	25299	0	$>$ 1h
	10	10	23371	$>$ 1h	3220	0	1288

The entries in the table III are:
Name : name of instance,
k : number of sets in the partition,
q : balanced coefficient,
LP : value of the LP relaxation,
Gap : the gap between the LP relaxation and the optimal value,
Col_{LP} : number of generated column for the relaxation,
CPU : total CPU time in seconds for the Branch-and-Price,
$\text{Col}_{\text{B\&P}}$: number of generated column for the Branch-and-Price,
nodes : number of nodes for the Branch-and-Price.

V. CONCLUSION

In this paper we investigate different mathematical formulations for the the Vertex k-Equi-Separator Problem, i.e., the problem of finding a minimum cardinality separator such that the sizes of the resulting disconnected subsets are balanced. We present a compact Integer Linear Programming formulation and discuss some possible improvement which brought to a strengthen model. Then we introduce an exponential-size formulation, for which we derive a column generation and a branching scheme. Preliminary computational results comparing the performance of the two formulations on a set of benchmark instances are reported which give some insight about the difficulty of the problem and the dimension of the instance which can be solved to proven optimality. We especially evaluated the effect of the number of partitions on the CPU time. Future developments include further strengthening of the compact formulation, as well as testing alternative algorithmic solutions of the column generation problem (e.g., as a flow problem) and alternative branching schemes.

REFERENCES

[1] E. Balas and C. de Souza, *The vertex separator problem: a polyhedral investigation*, Mathematical Programming, 103(3):583–608, 2005.

TABLE III. RESULTS FOR EXTENDED FORMULATION

Instance	k	q	LP	gap	$\mathrm{Col_{LP}}$	CPU	$\mathrm{Col_{B\&P}}$	nodes
myciel3	2	5	9	22,22	56	0	779	131
	2	10	9,7	17,53	52	0	204	32
	4	5	6,7	10,45	140	0	146	17
	4	10	7,3	17,81	100	0	146	17
myciel4	2	5	18,5	18,92	440	120	13508	215
	2	10	20,6	22,33	220	238	18640	321
	4	5	16,2	19,75	1024	mem	-	-
	4	10	17,5	8,57	840	308	19437	497
	6	5	15,1	13,91	1086	mem	-	-
	6	10	16	0,00	1206	1	544	30
	8	5	13,6	4,41	888	9	1687	291
	8	10	14,4	9,72	1408	6	1989	113
	10	5	12,4	11,29	1350	3	1069	298
	10	10	12,7	13,39	1840	3	960	322

[2] A. Bulu, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz, *Recent Advances in Graph Partitioning*, preprint Cornell University, 2013.

[3] G. Desaulniers, J. Desrosiers, and M.M. Solomon, editors. *Column Generation*. Springer-Verlag, Berlin, 2005.

[4] C. de Souza and E. Balas, *The vertex separator problem: algorithms and computations*, Mathematical Programming, 103(3):609–631, 2005.

[5] M. Didi Biha and M.J. Meurs, *An Exact Algorithm for Solving the Vertex Separator Problem*, Journal of Global Optimization, 49:425–434, 2010.

[6] H. N. Djidjev, *Partitioning Planar Graphs with Vertex Cost: Algorithms and Applications*, Algorithmica, 28:51–75, 2000.

[7] C.M. Fiduccia and R.M. Mattheyses, *A linear-time heuristic for improving network partition*, In Proceedings of the 19th Design Automation Conference, 19:175–181, 1982.

[8] N. Garg, H. Saran, and V. V. Vazirani, *Finding separator cuts in planar graphs within twice the optimal*, SIAM J. Computing, 35:159–179, 1999.

[9] Y. Kamidoi, S. Wakabayashi, and N. Yoshida, *A divide-and-conquer approach to the minimum k-way cut problem*, Algorithmica, 32:262–276, 2002.

[10] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM Journal on Scientific Computing, 20:359-392, 1998.

[11] G. Karypis and V. Kumar, *Multilevel k-way Partitioning Scheme for Irregular Graphs*, Journal of Parallel and Distributed Computing, 48:96–129, 1998.

[12] G. Karypis and V. Kumar, *Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs*, SIAM Review, 41:278–300, 1999.

[13] B. W. Kernighan and S. Lin, *An Efficient Heuristic Procedure for Partitioning Graphs*, The Bell System, Technical Journal, 49(1):291-307, 1970.

[14] R. J. Lipton and R.E. Tarjan, *A Separator Theorem for Planar Graphs*, SIAM J. Appl.Math., 36:177–189, 1979.

[15] S. Martin, *Analyse structurelle des systèmes algébro-différentiels conditionnels : complexité, modèles et polyèdres*, PhD Thesis, Paris Dauphine University, 2011.

[16] A. Pothen, *Graph partitioning algorithms with applications to scientific computing*, In Parallel Numerical Algorithms, Kluwer Academic Press, 1996.

[17] P. Sanders and C. Schulz, *Engineering Multilevel Graph Partitioning Algorithms*, In 19th European Symposium on Algorithms (ESA), volume 6942 of LNCS, pages 469-480, Springer, 2011.

[18] C. Walshaw, *Multilevel Refinement for Combinatorial Optimisation Problems*, Annals of Operations Research, 131(1):325-372, 2004.

[19] http://ampl.com/products/solvers/solvers-we-sell/cplex

[20] http://scip.zib.de/