The Balanced Connected k-Partition Problem: Polyhedra and Algorithms

Matheus J. Ota[1], Flávio K. Miyazawa[1] (Advisor), Phablo F. S. Moura[2] (Co-Advisor)

[1]Institute of Computing, University of Campinas, Brazil

[2]Department of Computer Science, Federal University of Minas Gerais, Brazil

matheus.ota@students.ic.unicamp.br, fkm@ic.unicamp.br, phablo@dcc.ufmg.br

Abstract. *The balanced connected k-partition (BCP_k) problem consists in partitioning a connected graph into connected subgraphs with similar weights. This problem arises in multiple practical applications, such as police patrolling, image processing, data base and operating systems. In this work, we address the BCP_k using mathematical programming. We propose a compact formulation based on flows and a formulation based on separators. We introduce classes of valid inequalities and design polynomial-time separation routines. Moreover, to the best of our knowledge, we present the first polyhedral study for BCP_k in the literature. Finally, we report on computational experiments showing that the proposed algorithms significantly outperform the state of the art for BCP_k.*

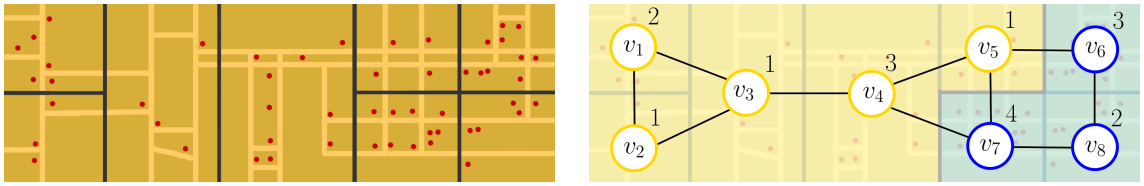
1. Introduction

The problem of partitioning a graph into connected subgraphs with similar weights has been extensively investigated since the late seventies [Lovász 1977]. Besides its innate theoretical interest, such problems have many practical applications in police patrolling, image processing, data base and operating systems. In the following, we elaborate on a police patrolling application.

Suppose we are given a map of a city that is divided into patrolling areas. Suppose also that this map has points representing crime occurrences. We can model this map with a graph $G = (V, E)$. The set of vertices V corresponds to the patrolling areas and an edge $\{p_1, p_2\}$ belongs to E if and only if patrolling area p_1 is adjacent to p_2 in the map. Next, define a function $w\colon V \to \mathbb{Q}_{\geq}$ which indicates the criminality in a patrolling area. Finding k connected subgraphs with similar weights in (G, w) is equivalent to attributing balanced and contiguous patrolling regions to k police teams (see Figure 1).

We are now ready to formalize the balanced connected k-partition problem. To this end, we assume henceforth that $G = (V, E)$ is a connected undirected graph and $w\colon V \to \mathbb{Q}_{\geq}$ is a function that assigns weights to the vertices of G. As usual, we use the notation $n := |V|$ and $m := |E|$. Moreover, for $t \geq 1$, $[t]$ denotes the set $\{1, \ldots, t\}$; and for any subset $V' \subseteq V$, $w(V')$ refers to the sum $\sum_{v \in V'} w(v)$.

Fix k to be a positive integer. A *connected k-partition* of G is a partition $\{V_i\}_{i \in [k]}$ of V, where, for every $i \in [k]$, $V_i \neq \emptyset$ and $G[V_i]$ is connected. We refer to each V_i as a *class* of the partition. We may assume that $|V| \geq k$, otherwise G does not admit a connected k-partition. In the *balanced connected k-partition problem* (BCP_k), we are given a vertex-weighted connected graph, and we seek a connected k-partition that maximizes the weight of a lightest class, i.e., the goal is to maximize $\min_{i \in [k]}\{w(V_i)\}$.



(a) (b)

Figure 1. (a) A toy example of a map. The black lines delimit the patrolling areas and the red dots represent crime occurrences. (b) A corresponding instance (G, w) and a solution for $k = 2$. The numbers outside the vertices indicate the weights.

For $k \geq 2$, the balanced connected k-partition problem is known to be $\mathcal{NP}$-hard even in bipartite graphs [Dyer and Frieze 1985]. Furthermore, BCP$_k$ has been largely investigated under different approaches and perspectives: exact algorithms, approximation algorithms for some values of k or special classes of graphs, close variants of the problem, and inapproximability results. Moreover, besides the already mentioned applications, BCP$_k$ has recently been used to model problems in cluster analysis, education, robotics and metabolic networks. All these different real-world applications indicate the importance of designing algorithms for BCP$_k$ and reporting on the computational experiments with their implementations. Not less important are the theoretical studies of the rich and diverse mathematical formulations BCP$_k$ leads to.

1.1. Contributions and Outline

Our work advances the state of the art for exact algorithms for BCP$_k$. Due to space constraints we present results without details. The reader is referred to the author's thesis for detailed explanations [Ota 2020]. In Section 2 of this summary, we introduce a compact flow based formulation for the problem. Then, in Section 3, we present a novel cut based integer linear programming formulation, and show valid inequalities that have corresponding polynomial-time separation routines. A polyhedral study is also presented in the same section. To the best of our knowledge, this is the first polyhedral study for BCP$_k$ described in the literature. We report on computational experiments in Section 5. In the same section, we also propose new benchmark instances, based on instances generated with OpenStreetMap and real-world public safety data. The computational results show that the exact algorithms based on our formulations are able to solve larger instances in a smaller amount of time than the previous exact solving methods in the literature, namely, the algorithms proposed by Matić [Matić 2014] and Zhou et al. [Zhou et al. 2019].

2. Flow formulation

The flow formulation is based on flows in a digraph D. Given the input graph $G = (V, E)$, the set of vertices of D is $V(D) = V \cup S$, where $S = \{s_1, \ldots, s_k\}$. Each vertex in S is associated to a class and represents a source of flow. The arc set of the digraph D is set to $A(D) = \{(u, v), (v, u) : \{u, v\} \in E\} \cup \{(s_i, v) : i \in [k], v \in V\}$.

For each arc $a \in A(D)$, we associate a real variable $f_a \geq 0$ which represents the amount of flow passing through a, and a binary variable y_a (such that $y_a = 1$ if $f_a > 0$) that allow us to impose that flows from different sources do not mix. This way,

we establish that the flow sent by source s_i reaches exactly the vertices in V_i. Next, we enforce that each vertex $v \in V$ consumes $w(v)$ from the flow. Then, the amount of flow sent by s_i corresponds precisely to the weight of the i-th class. In the below formulation, $y(A')$ (resp. $f(A')$), for arc set $A' \subseteq A(D)$, stands for $\sum_{a \in A'} y_a$ (resp. $\sum_{a \in A'} f_a$).

$$
\begin{aligned}
\max\ & f(\delta^+(s_1)) \\
\text{s.t. } & f(\delta^+(s_i)) \leq f(\delta^+(s_{i+1})) && \forall i \in [k-1], && (1)\\
& f(\delta^-(v)) - f(\delta^+(v)) = w(v) && \forall v \in V, && (2)\\
& f_a \leq w(V)\, y_a && \forall a \in A(D), && (3)\\
& y(\delta^+(s_i)) \leq 1 && \forall i \in [k], && (4)\\
& y(\delta^-(v)) \leq 1 && \forall v \in V, && (5)\\
& y_a \in \{0,1\} && \forall a \in A(D), && (6)\\
& f_a \in \mathbb{R}_{\geq} && \forall a \in A(D). && (7)
\end{aligned}
$$

Constraints (2) enforces that each vertex $v \in V$ consumes $w(v)$ flow units. Inequalities (3) imply that positive flow can only pass through arcs that were selected by the y variables. Constraints (4) impose that at most one arc leaving a source transports positive flow. Similarly, inequalities (5) require that every non-source vertex receives a positive flow from at most one vertex. Lastly, the flows sent by the sources are ordered non-decreasingly by inequalities (1). This explains the objective function. As we shall see in Section 4, a branch-and-bound algorithm using this flow formulation is the fastest exact solving method in general.

3. Cut Formulation

The cut formulation we propose for BCP_k is based on the following central concept. Let u and v be two non-adjacent vertices in G. We say that $S \subseteq V \setminus \{u, v\}$ is a (u, v)-*separator* if u and v belong to different components of $G - S$. We denote by $\Gamma(u, v)$ the collection of all minimal (u, v)-separators in G. In the formulation, we use a binary variable $x_{v,i}$, for every $v \in V$ and $i \in [k]$, that indicates if v belongs to the i-th class.

$$
\begin{aligned}
\max\ & \sum_{v \in V} w(v)\, x_{v,1} \\
\text{s.t. } & \sum_{v \in V} w(v)\, x_{v,i} \leq \sum_{v \in V} w(v)\, x_{v,i+1} && \forall i \in [k-1], && (8)\\
& \sum_{i \in [k]} x_{v,i} \leq 1 && \forall v \in V, && (9)\\
& x_{u,i} + x_{v,i} - \sum_{z \in S} x_{z,i} \leq 1 && \forall uv \notin E, S \in \Gamma(u,v), i \in [k], && (10)\\
& x_{v,i} \in \{0,1\} && \forall v \in V \text{ and } i \in [k]. && (11)
\end{aligned}
$$

Because of the weight ordering imposed by inequalities (8), the objective function maximizes the weight of a lightest class. Constraints (10) guarantee the connectedness of each class. Since the cardinality of each set $\Gamma(u, v)$ may be exponentially large, we treat these inequalities with a cutting-plane approach.

Next, we derive valid inequalities. The corresponding separation routines are discussed in Section 5.2 of the thesis. Before we proceed, let us define the polytope $\mathcal{P}_k(G, w) = \text{conv}\{x \in \mathbb{B}^{nk} : x \text{ satisfies inequalities (8)}-\text{(10)}\}$. The next proposition perform a lifting of inequalities (10) by removing appropriate vertices from the separator.

Proposition 3.1 *Let u and v be two non-adjacent vertices of G, and let S be a minimal (u, v)-separator. Let $i \in [k]$, and let $L = \{z \in S : w(P_z) > w(V)/(k-i+1)\}$, where P_z is a minimum-weight path between u and v in G that contains z. The following inequality is valid for $\mathcal{P}_k(G, w)$: $x_{u,i} + x_{v,i} - \sum_{z \in S \setminus L} x_{z,i} \leq 1$.*

The next class of inequalities are of a different nature and are referred to as *cross inequalities* [de Aragão and Uchoa 1999]. In Chapter 5 of the thesis we introduce a generalization of the cross inequalities. Moreover, given a face F, we designed a dynamic programming algorithm that separates the cross inequalities in time complexity $\mathcal{O}(|F|k^2)$.

Proposition 3.2 *Let G be a planar graph and let F be the boundary of a face with at least 4 vertices. Consider distinct vertices s_1, s_2, t_1 and t_2 that appear in a clockwise order in F. For all $i, j \in [k]$, $i \neq j$, the constraint $x_{s_1,i} + x_{s_2,j} + x_{t_1,i} + x_{t_2,j} \leq 3$ is valid for $\mathcal{P}_k(G, w)$.*

3.1. Polyhedral results

In this section, we denote by 1-BCP$_k$, the special case of BCP$_k$ in which all vertices have unit weight. In this case, the polytope $\mathcal{P}_k(G, w)$ is simply written as $\mathcal{P}_k(G)$.

Theorem 3.3 *Let G be an input for* 1-BCP$_k$*. Then, the following hold.*

(a) The polytope $\mathcal{P}_k(G)$ is full-dimensional, that is, $\dim(\mathcal{P}_k(G)) = kn$;
(b) For every $v \in V$ and $i \in [k]$, the inequality $x_{v,i} \geq 0$ defines a facet of $\mathcal{P}_k(G)$;
(c) For every $v \in V$, the inequality $\sum_{i \in [k]} x_{v,i} \leq 1$ defines a facet of $\mathcal{P}_k(G)$.

Further polyhedral results were obtained during this research. Namely, we characterized when inequalities (10) are facet defining for $\mathcal{P}_k(G)$. We also studied a variation of the cut formulation that avoids ordering the classes by their weights. Then, when studying the facial structure of the associated polytope, we do not require that all vertices have the same weight, as in 1-BCP$_k$. Besides extending the results in Theorem 3.3 for this second polytope, we also characterized when the cross inequalities are facet defining. These results can be seen in Section 5.4 of the thesis.

4. Computational Experiments

We conducted computational experiments to compare the performance of the proposed solving methods with the state of the art, and to test our algorithms on instances from a real-world application. Among the considered instances are grid graphs and random connected graphs, since these were used in previous works [Matić 2014,Zhou et al. 2019]. Moreover, we evaluated the performance of our algorithms on new instances based on a police patrolling application. Using the OSMnx python library we transformed maps from OpenStreetMap into undirected graphs. Next, we collected public safety data for the cities of Chicago, Los Angeles, New York and Campinas. Finally, we designed a function that assigns weights to the vertices in a way that each crime occurrence has an influence over a region according to a gaussian function.

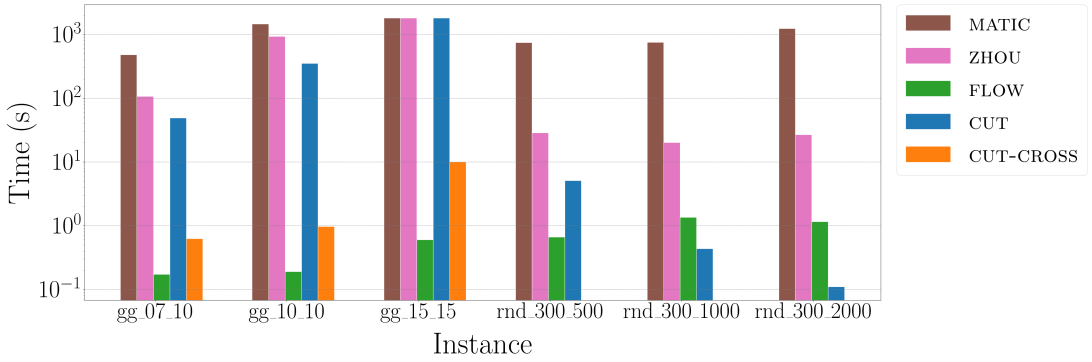


Figure 2. Computational results for BCP_2 on grid and random graphs. Time is in logarithmic scale. Grid and random graphs have names in the format *gg_height_width* and *rnd_n_m*, respectively.

Instance	2	3	4	5
barao_1913_2752	8.18	504.97	-	-
campinas_centro_579_942	2.05	733.79	206.66	-
chicago_loop_624_971	6.97	26.01	1262.51	293.26
la_hollywood_1368_2030	22.71	120.30	878.96	696.60
la_skidrow_1667_2459	51.10	57.30	1648.06	-
nyc_chelsea_822_1228	2.59	84.33	133.40	242.25
nyc_hellskitchen_498_746	1.47	10.02	3.50	-
unicamp_624_901	1.33	95.20	-	-

Table 1. Running time in seconds of FLOW on real-world instances with $k \in \{2, 3, 4, 5\}$. Instances have names in the format *name_n_m*, where *name* refers to the geographic region name.

4.1. Computational results

Our implementation was written in C++ using the graph library Lemon. The branch-and-cut algorithm based on the cut formulation was implemented using SCIP 6.0 and Gurobi 9.0 as the LP solver. The branch-and-bound algorithms for the flow formulation and the models previously proposed by Matić [Matić 2014] and Zhou et al. [Zhou et al. 2019] were implemented using only Gurobi 9.0. Our branch-and-cut algorithm have additional enhancements discussed in Chapter 5 of the thesis; namely, we separate lifted minimal cover inequalities and implement custom domain propagation routines. We use SCIP plugin system to implement these features. The execution time limit in our experiments was set to 1800 seconds.

Figure 2 show histograms with the average execution time in instances of BCP_2. In these plots, we assume that an algorithm took 1800 seconds to solve instances which were not solved within the time limit. For each graph type, we generated 10 instances. CUT-CROSS refer to the cut formulation with the cross inequalities. As one can see, the cross inequalities were very effective in the grid graphs. Also, in both grid and random graphs, our best algorithms were significantly superior to the methods in the literature. To further assess the performance of our methods, we generated larger graphs. Our experiments show that FLOW was able to solve grids with over 400 times more vertices than the grids solved by the previous exact methods in the literature.

We also conducted experiments for $k > 2$. For grid and random graphs with $k \in \{3, 4, 5\}$, FLOW was able to solve all of the 60 instances considered. The previous state of the art (Zhou et al.) was able to solve 16 of these instances. Lastly, only FLOW was able to solve the real-world graphs. Table 1 present the execution times in some of these instances. Notice that the problem became harder for larger values of k.

5. Conclusion

The contributions of this work are twofold. On the practical side, we introduced classes of real-world instances for BCP_k and proposed efficient exact algorithms based on (mixed) integer linear programming formulations. We also introduced classes of valid inequalities and devised polynomial-time separation routines. On the theoretical side, we studied the facial structure of the polytope associated with the cut formulation. We remark that this is the first polyhedral study of BCP_k in the literature.

This work gave rise to two published papers. One is a short version of the thesis and was published in the proceedings of the International Symposium on Combinatorial Optimization 2020 (Qualis B2) [Miyazawa et al. 2020]. The second paper has a more practical emphasis and was published in the European Journal of Operational Research (Qualis A1) [Miyazawa et al. 2021]. A third paper is currently in preparation, and it concerns our theoretical findings regarding the facial structure of the polytopes associated with BCP_k.

References

de Aragão, M. P. and Uchoa, E. (1999). The γ-connected assignment problem. *European Journal of Operational Research*, 118(1):127–138.

Dyer, M. and Frieze, A. (1985). On the complexity of partitioning graphs into connected subgraphs. *Discrete Applied Mathematics*, 10(2):139–153.

Lovász, L. (1977). A homology theory for spanning tress of a graph. *Acta Mathematica Academiae Scientiarum Hungarica*, 30:241–251.

Matić, D. (2014). A mixed integer linear programming model and variable neighborhood search for maximally balanced connected partition problem. *Applied Mathematics and Computation*, 237:85–97.

Miyazawa, F. K., Moura, P. F., Ota, M. J., and Wakabayashi, Y. (2021). Partitioning a graph into balanced connected classes: Formulations, separation and experiments. *European Journal of Operational Research*, 293(3):826–836.

Miyazawa, F. K., Moura, P. F. S., Ota, M. J., and Wakabayashi, Y. (2020). Cut and flow formulations for the balanced connected k-partition problem. In *International Symposium on Combinatorial Optimization*, pages 128–139. Springer.

Ota, M. J. (2020). The balanced connected k-partition problem : polyhedra and algorithms [1]. Master's thesis, Universidade Estadual de Campinas.

Zhou, X., Wang, H., Ding, B., Hu, T., and Shang, S. (2019). Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm. *Expert Systems with Applications*, 116:10–20.

[1] `http://repositorio.unicamp.br/jspui/handle/reposip/356940`