

The Balanced Connected k -Partition Problem: Polyhedra and Algorithms

Matheus Jun Ota

Institute of Computing, University of Campinas

Av. Albert Einstein, 1251, Cidade Universitária, Campinas/SP - Brasil, CEP 13083-852

matheus.ota@students.ic.unicamp.br

Defense Date: October 2020

Advisor: Flávio Keidi Miyazawa

Co-Advisor: Phablo Fernando Soares Moura

RESUMO

O problema da k -partição conexa balanceada (em inglês, “balanced connected k -partition problem” ou BCP_k) consiste em particionar um grafo conexo em subgrafos conexos de pesos semelhantes. Esse problema é conhecido por ser $\mathcal{NP}$ -difícil e possui diversas aplicações, como patrulhamento policial, processamento de imagem e sistemas operacionais. Nesse trabalho propomos duas formulações para o BCP_k , uma baseada em fluxos e outra baseada em cortes. A formulação de fluxos possui um número polinomial de restrições, enquanto a formulação de cortes possui um número exponencial de desigualdades que podem ser separadas em tempo polinomial. Nós introduzimos novas desigualdades válidas para esse último modelo e desenvolvemos rotinas de separação correspondentes. Além disso, nós apresentamos o primeiro estudo poliédrico para o problema. Por fim, nós conduzimos experimentos computacionais em instâncias da literatura e em novas instâncias inspiradas em uma aplicação real. Os resultados mostram que nossas abordagens possuem um desempenho significativamente superior ao estado da arte para o BCP_k .

PALAVRAS CHAVE. Programação inteira, Partição de grafos, Poliedros.

Tópicos (Programação matemática, Otimização combinatória, Teoria dos grafos)

ABSTRACT

The balanced connected k -partition problem (BCP_k) consists in partitioning a connected graph into connected subgraphs with similar weights. This problem is known to be $\mathcal{NP}$ -hard and has several applications, such as police patrolling, image processing, data base and operating systems. We propose flow and cut based formulations for BCP_k . The flow based formulation have a polynomial number of constraints and variables. The cut based formulation contains only binary variables and a potentially exponential number of constraints that can be separated in polynomial time. We introduce new valid inequalities for this last model and design corresponding polynomial-time separation routines. Moreover, based on this cut based formulation, we present the first polyhedral study for BCP_k in the literature. Lastly, we conduct computational experiments in instances of the literature and in new real-world instances. The results show that the exact algorithms based on the proposed formulations significantly outperform the state of the art for BCP_k .

KEYWORDS. Integer programming, Graph partition, Polyhedra.

Paper topics (Mathematical programming, Combinatorial optimization, Graph theory)

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. Let $G = (V, E)$ be a graph modelling this map. 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 $w: V \rightarrow \mathbb{Q}_{\geq}$ 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).

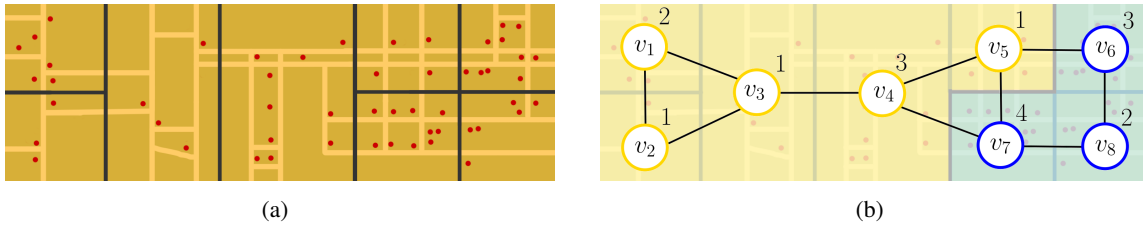


Figure 1: (a) A toy example of a map. The black lines delimit the patrolling areas and the red dots represent crime occurrences. (b) A corresponding instance (G, w) and a solution for $k = 2$. The numbers outside the vertices indicate the weights.

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 $w: V \rightarrow \mathbb{Q}_{\geq}$ 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 $\{1, \dots, t\}$; 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 $\min_{i \in [k]} \{w(V_i)\}$.

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 dissertation for

detailed explanations [Ota, 2020]. We highlight that Chapter 2 of the dissertation was carefully written so that someone without a deep background in integer programming can easily follow the derivations of the main results obtained in this research.

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. Implementation details are discussed in Section 4. In Section 5, we report on computational experiments on instances from the literature and newly proposed instances based on an application in police patrolling. These last instances were generated using 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 ([Matić, 2014], [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 $S = \{s_1, \dots, s_k\}$. 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.

One disadvantage of our flow formulation is that constraints (3) might lead to a weak LP relaxation if the value $w(V)$ is too large. Trying to further improve this model, we designed a multicommodity flow formulation that uncouples the weights of the vertices from the flow variables.

Such a formulation also considers a total ordering of the vertices to reduce the number of symmetries in the solution space. Despite the removal of symmetries, the branch-and-bound algorithm based on this strengthened flow formulation had an inferior performance. Thus, we omit the model from this summary. We refer the reader to Chapter 4 of the dissertation for more details. In the same chapter we also consider the impact of ordering the classes by their weights through inequalities (1).

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.

$$\begin{aligned} \max \quad & \sum_{v \in V} w(v) x_{v,1} \\ \text{s.t.} \quad & \sum_{v \in V} w(v) x_{v,i} \leq \sum_{v \in V} w(v) x_{v,i+1} \quad \forall i \in [k-1], \end{aligned} \quad (8)$$

$$\sum_{i \in [k]} x_{v,i} \leq 1 \quad \forall v \in V, \quad (9)$$

$$x_{u,i} + x_{v,i} - \sum_{z \in S} x_{z,i} \leq 1 \quad \forall uv \notin E, S \in \Gamma(u, v), i \in [k], \quad (10)$$

$$x_{v,i} \in \{0, 1\} \quad \forall v \in V \text{ and } i \in [k]. \quad (11)$$

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 dissertation. Before we proceed, let us define the polytope $\mathcal{P}_k(G, w) = \text{conv}\{x \in \mathbb{B}^{nk} : x \text{ satisfies inequalities (8) -- (10)}\}$. 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 dissertation 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, denoted by $\bar{\mathcal{P}}_k(G)$, which avoids ordering the classes by their weights. This formulation has an extra real variable $z \in \mathbb{R}_{\geq}$ which indicates the weight of the lightest class. Let $\bar{\mathcal{P}}_x$ be the projection of $\bar{\mathcal{P}}_k(G)$ into the x -variables. We prove the following lemma.

Lemma 3.4 *Suppose $\pi x \leq \pi_0$ is an inequality that is valid for both $\mathcal{P}_k(G)$ and $\bar{\mathcal{P}}_x$. Let F and $\bar{F}_x$ be the faces induced by (π, π_0) in the polytopes $\mathcal{P}_k(G)$ and $\bar{\mathcal{P}}_x$, respectively. Then it holds that $\dim(F) \leq \dim(\bar{F}_x)$.*

Using Lemma 3.4 we show that the results in Theorem 3.3 extends naturally to $\bar{\mathcal{P}}_k(G)$. Moreover, we can show that inequalities (10) are always facet defining, and we establish necessary and sufficient conditions so that the cross inequalities induce facets. These results can be seen in Section 5.4 of the dissertation.

4. Implementation Details

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 and Zhou et al. were implemented using only Gurobi 9.0. We used SCIP because its plugin system facilitates the implementation of additional enhancements which we discuss next.

Besides using the already mentioned inequalities, our branch-and-cut algorithm separates lifted minimal cover inequalities. Since we impose an ordering over the classes, we can derive upper bounds for the total weight of each class. Specifically, for every $i \in [k]$, the inequality $\sum_{v \in V} w(v)x_{v,i} \leq w(V)/(k-i+1)$ is valid for $\mathcal{P}_k(G, w)$. These inequalities can be interpreted as knapsacks with capacities $w(V)/(k-i+1)$. Therefore, we can take advantage of the known strong inequalities for the 0/1-knapsack polytope. In our implementation, we used the built-in heuristics of SCIP to separate lifted minimal cover inequalities.

Another enhancement that we used, called *domain propagation*, refers to presolving techniques that one can use to reduce the domain of the variables in each node of the branch-and-bound tree. Suppose that we are currently exploring a node of the tree. For every $j \in [k]$, let $F_j \subseteq V$ be the set of vertices $v \in V$ such that $x_{v,j}$ was fixed to one. Fix $i \in [k]$ and assume $F_i \neq \emptyset$. Consider the graph G_i obtained from G by removing the vertices that were fixed to classes distinct from i . Let $\{C_j\}_{j \in [t]}$ be the connected components of G_i , and suppose that $F_i \cap V(C_t) \neq \emptyset$. Now for every vertex $u \in V(C_j)$, with $j \in [t-1]$, we either set $x_{u,i} = 0$; or, if variable $x_{u,i}$ was already fixed to one, then the current branch-and-bound node is infeasible and it can be pruned.

5. 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 grids and random connected graphs, as 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. Figure 3 shows an example. 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.

All of our experiments were executed with a time limit of 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. Note that the cross inequalities were very effective in the grids. Also, in both grid and random graphs, our best algorithms were significantly superior to the methods in the literature.

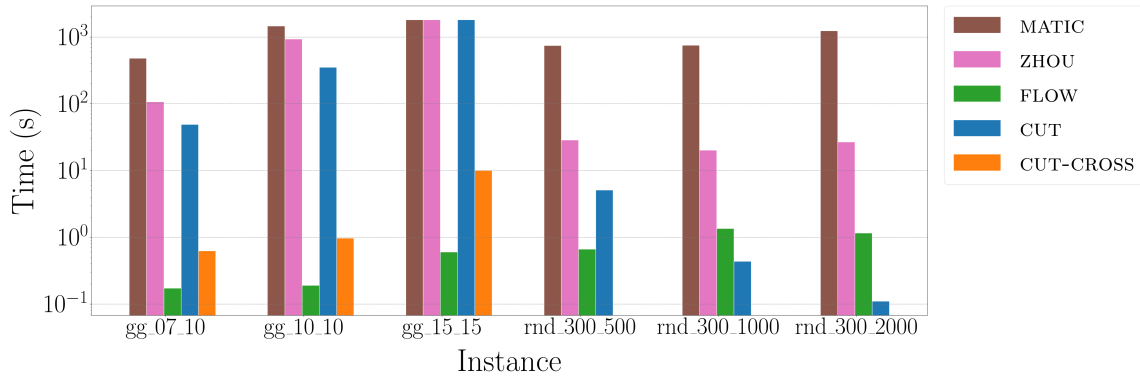


Figure 2: Computational results for BCP_2 on grid and random graphs. Time is in logarithmic scale. Grid and random graphs have names in the format *gg_height_width* and *rnd_n_m*, respectively.

To further assess the performance of our methods, we generated larger graphs (Table 1). In this table, the cross inequalities were used only in grid graphs (as the random graphs may not be planar). Our experiments show that FLOW was able to solve grids with over 400 times more vertices than the grids solved by the previous state of the art (Zhou et al.). On the other hand, CUT was not able to solve any of those instances. The situation changes for larger random graphs and different density of edges. For those instances CUT had a better behavior when $m \geq n^{1.2}$. In addition, in some of these graphs, CUT solved the problem in the root node of the branch-and-bound tree. This indicates that the cut-based formulation may yield a better approximation of the convex hull.

We also conducted experiments for $k > 2$ (see Table 2). We remark that Zhou et al. formulation was proposed for MIN-MAX BCP_k , a version of BCP_k where the goal is to minimize the weight of the heaviest class. One can easily check that BCP_2 and MIN-MAX BCP_2 are equivalent; however, for $k > 2$, BCP_k and MIN-MAX BCP_k are different problems. Hence, we adapted our algorithms for the min-max case by altering the objective function. Observe that for grid and random graphs FLOW was able to solve all of the 60 instances considered, while the previous state of the art solved only 16 of these instances. Finally, only FLOW was able to solve the police patrolling instances. Table 3 present the execution times in some of these instances.

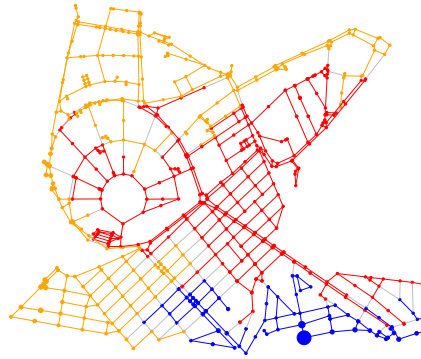


Figure 3: An instance of BCP_3 based on the map of the University of Campinas.

Instance	CUT			FLOW		
	Sol	Nodes	Time	Sol	Nodes	Time
gg_120_120	0	-	-	10	<u>575</u>	148.93
gg_150_150	0	-	-	10	<u>546</u>	302.97
gg_180_180	0	-	-	10	<u>305</u>	571.58
gg_210_210	0	-	-	10	<u>400</u>	1100.32
rnd_n1000_m1995	10	<u>26</u>	53.74	10	2,482	7.21
rnd_n1000_m3981	10	<u>12</u>	3.91	10	1,331	16.21
rnd_n1000_m7943	10	<u>1</u>	0.66	10	1,510	34.83
rnd_n1000_m15849	10	<u>1</u>	0.76	10	45	7.43
rnd_n1000_m31623	10	<u>1</u>	1.21	10	342	36.80

Table 1: Performance of CUT and FLOW to solve BCP_2 on larger instances.

Instance	k	CUT (MIN-MAX)			FLOW (MIN-MAX)			Zhou et al.		
		Sol	Nodes	Time	Sol	Nodes	Time	Sol	Nodes	Time
gg_07_10	3	0	-	-	10	4,177	0.93	2	1,072,107	603.97
gg_07_10	4	0	-	-	10	25,242	3.86	1	99,880	68.95
gg_07_10	5	0	-	-	10	1,578,669	247.61	0	-	-
rnd_100_150	3	10	9,922	223.21	10	1,933	0.74	9	109,799	70.04
rnd_100_150	4	0	-	-	10	13,404	3.14	4	1,019,886	641.36
rnd_100_150	5	0	-	-	10	627,636	149.27	0	-	-

Table 2: Computational results for MIN-MAX BCP_k when $k \in \{2, 3, 4, 5\}$.

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 3: 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.

6. 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. The computational results show that the exact algorithms based on the proposed formulation significantly outperform the state of the art for BCP_k . On the theoretical side, we studied the facial structure of polytopes based on the cut formulation. We remark that this is the first polyhedral study for BCP_k in the literature.

This work gave rise to two published papers. One is a short version of the dissertation 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. e Uchoa, E. (1999). The γ -connected assignment problem. *European Journal of Operational Research*, 118(1):127–138. ISSN 0377-2217.
- Dyer, M. e 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 trees 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. S., Ota, M. J., e Wakabayashi, Y. (2020). Cut and flow formulations for the balanced connected k -partition problem. In Baïrou, M., Gendron, B., Günlük, O., e Mahjoub, A. R., editors, *Combinatorial Optimization*, volume 12176 of *Lecture Notes in Computer Science*, p. 128–139. Springer International Publishing. ISBN 978-3-030-53262-8.
- Miyazawa, F. K., Moura, P. F., Ota, M. J., e Wakabayashi, Y. (2021). Partitioning a graph into balanced connected classes: Formulations, separation and experiments. *European Journal of Operational Research*, 293(3):826–836. ISSN 0377-2217. URL <https://www.sciencedirect.com/science/article/pii/S0377221720311218>.
- Ota, M. J. (2020). The balanced connected k -partition problem : polyhedra and algorithms. Master's thesis, Universidade Estadual de Campinas. URL <http://repositorio.unicamp.br/jspui/handle/REPOSIP/356940>.
- Zhou, X., Wang, H., Ding, B., Hu, T., e 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.