

A Random-Key Optimizer for a Balanced Edge Partitioning Problem

Leonardo J. C. Vasconcelos

Universidade Federal de Sao Paulo / PPGPO

Avenida Cesare Mansueto Giulio Lattes, 1201, 12247-014, Sao Jose dos Campos, Brazil.

ljcvasconcelos@unifesp.br

Antonio A. Chaves

Universidade Federal de Sao Paulo / PPGPO

Avenida Cesare Mansueto Giulio Lattes, 1201, 12247-014, Sao Jose dos Campos, Brazil.

antonio.chaves@unifesp.br

Mauricio G.C. Resende

Univeristy of Washington

Seattle, WA

mgcr@uw.edu

Larissa P.G. Petroianu

Department of Industrial and Systems Engineering, University of Washington

3900 E Stevens Way NE, Mechanical Engineering Building, Room G6 Seattle, WA 98195 USA.

larissapetroianu@gmail.com

Alberto Francisco Kummer Neto

Mercado Livre

alberto.kummer@gmail.com

ABSTRACT

We present a mixed-integer programming model and a random-key optimizer (RKO) for a balanced edge partitioning problem (BEP) on an undirected graph. In this problem, given a number of partitions into which we assign edges, the number of edges in each partition should be as similar as possible. The objective is to minimize the cost of the partitioning. We use 162 instances to illustrate that for small graphs and a small number of partitions, the mathematical model produces an optimal solution. However, as the size of the graph increases and there are more partitions, the use of a heuristic is required. RKO is shown to be a good option for such large graphs, finding near-optimal solutions.

KEYWORDS. Balanced Edge Partitioning. Random-key optimizer. Metaheuristics.

Paper topics: MH – Metaheuristics, TAG – Graph Theory and Related Algorithms

1. Introduction

Graph partitioning is the process of dividing a graph into smaller subgroups (partitions) so that interactions between the subgroups are minimal. Two approaches are used. Vertex partitioning assigns vertices to different groups and is applicable in integrated circuit design (chips and microchips); edge partitioning assigns edges to different groups and is important in parallel computing. Edge partitioning can lead to a more balanced distribution of subgraphs in size than vertex partitioning and at the same cost [Bourse et al., 2014].

Edge partitioning algorithms can be grouped into two categories: in offline mode, the graph is loaded into memory; in streaming mode, edges are sent as a stream so that partitioning can be performed with only partial knowledge about the graph. According to Guerrieri and Montresor [2014] a good partitioning must be balanced, communicate efficiently between the subgraphs and the information should not pass for many partitions. An example: ideal partitioning should ensure that each processor receives a similar workload, preventing some processors from being idle while others are overloaded.

2. Problem Definition

Let $G = (V, E)$ be an undirected graph, where V is its node set and E its edge set. In the *balanced edge partitioning problem* we seek to partition the edges as well as the nodes into $k > 0$ disjoint sets. While there is no constraint on the nodes other than that each node must be assigned to a single partition, the edge partitioning is required to be balanced. By balanced, we mean that the number of edges in each of the k partitions should be as similar as possible. At worst, the number of edges in two partitions will differ by at most one. The objective is to minimize the cost of partition, computed as the sum of the costs of all edges in the graph. Consider an edge $e = (u, v)$. If vertices u and v are both assigned to the same partition as edge e , then no cost is incurred on edge e , i.e. $c(e) = 0$. If either u or v is assigned to the partition of e and the other vertex is assigned to a different partition, then a cost of one is incurred, i.e. $c(e) = 1$. If both u and v are assigned to the same partition, distinct from the partition to which edge e is assigned, then a cost of two is incurred, i.e. $c(e) = 2$. If u , v , and e are assigned to three different partitions, a cost of three is incurred, i.e. $c(e) = 3$.

Given the sets $E(i, j)$ of edges, V of vertices and K of groups, the decision variables is: x_{ijk} is 1, if edge (i, j) belongs to group k and 0, otherwise and y_{ik} is 1 if vertex i belongs to group k and 0, otherwise. Furthermore, auxiliary variables is: v_{ijk} denotes the cost of having at least one vertex at a different partition for the edge (i, j) , up to two; z_{ijk} denotes the additional cost of one if all the edges and vertices belong to different groups and w_{ijk} is 1, if in the edge (i, j) , both vertices belong to the same partition and 0, otherwise. The mixed integer model is presented below. The objective function is to minimize the cost of partitioning the edges and nodes in groups, $v_{ijk} + z_{ijk}$ represents the cost increase for group k , according to the partitions of the edge (i, j) and its respective vertices. Constraint 1 guarantees that the partitions will be balanced, considering the number of edges in each of them. Constraints 2 and 3 establish that each edge and vertex will belong to only one partition. Constraints 4 to 8 calculate the cost of the partitioning, as it will be explained below. The remain constraints are related to the variables domain.

$$\min \left\{ \sum_{(i,j) \in E} \sum_{k \in K} v_{ijk} + z_{ijk} \right\},$$

Subject to

$$\left\lfloor \frac{|E|}{|K|} \right\rfloor \leq \sum_{(i,j) \in E} x_{ijk} \leq \left\lceil \frac{|E|}{|K|} \right\rceil, \quad \forall k \in K \quad (1)$$

$$\sum_{k \in K} x_{ijk} = 1, \quad \forall (i, j) \in E \quad (2)$$

$$\sum_{k \in K} y_{ik} = 1, \quad \forall i \in V \quad (3)$$

$$v_{ijk} \geq 2x_{ijk} - y_{ik} - y_{jk}, \quad \forall (i, j) \in E \quad \forall k \in K \quad (4)$$

$$w_{ijk} \geq y_{ik} + y_{jk} - 1, \quad \forall (i, j) \in E \quad \forall k \in K \quad (5)$$

$$w_{ijk} \leq y_{ik}, \quad \forall (i, j) \in E \quad \forall k \in K \quad (6)$$

$$w_{ijk} \leq y_{jk}, \quad \forall (i, j) \in E \quad \forall k \in K \quad (7)$$

$$z_{ijk} \geq x_{ijk} - \sum_{k \in K} w_{ijk} - y_{ik} - y_{jk}, \quad \forall (i, j) \in E \quad \forall k \in K \quad (8)$$

$$v_{ijk} \geq 0, \quad z_{ijk} \geq 0, \quad \forall (i, j) \in E \quad \forall k \in K \quad (9)$$

$$x_{ijk} \in \{0, 1\}, \quad y_{ik} \in \{0, 1\}, \quad w_{ijk} \in \{0, 1\}, \quad \forall (i, j) \in E \quad \forall k \in K \quad (10)$$

3. RKO for BEP

Smorodkina et al. [2007] demonstrates that the edge partitioning problem is NP-hard. Therefore, there is a need for applying a heuristic method to calculate a good solution. For this work, we used Random-Key Optimizer (RKO) [Chaves et al., 2024]. This methodology represents solutions as vectors of random keys, which are real numbers uniformly distributed in the range $[0, 1)$. A problem-specific decoder is used to convert these vectors into feasible solutions and evaluate their quality. For the BEP, the decoder presented in Alg. 1 was made. The allocation of nodes to partitions is done by checking connections with other nodes and choosing the best partition based on the number of edges allocated (Balanced form). The allocation of nodes is based on the most common partitions of their connected edges. Finally, the cost is calculated using the allocations (X and Y) made.

Algorithm 1 BEP Decoder

- 1: **Input:** $rk = [r_1, \dots, r_E]$, K partitions, E edges, n_{edges} , n_{nodes}
 - 2: **Output:** edges and vertices partitions
 - 3: **Step 1:** Sort edges based on the random-key vector rk
 - 4: **Step 2:** Initialize allocations for edges and nodes
 - 5: $X = []$; $Y = []$; $lim = \frac{n_{edges}}{K}$
 - 6: **Step 3:** Allocate edges and nodes to partitions
 - 7: **Return** Cost (X,Y)
-

4. Preliminary Computational Results

The tests were run on a computer with an Intel Xeon Silver 4114 2.2 Ghz processor. Gurobi 12.0.1 (Max. Time - 3600 s) was used to solve the model. The Steinlib B set [Koch et al., 2001] was used for the instances, which consists of 18 graphs with $V = [50, 100]$; $E = [63, 200]$ with $K = [2, 10]$, totaling 162 instances. For RKO, the C++ version available on GitHub was used. The Maximum runtime was set to $V + E$, the maximum number of runs to 5 and parameter control in online mode with Q-Learning method to learn the best configuration for each metaheuristic during

the search process. Eight metaheuristics were executed in parallel. The threads run independently, and information about the best solutions is shared through a solution pool. Table 1 presents the results of the 18 graphs grouped by the number of partitions k . In 94 of the 162 instances, RKO found the best known solution and in 6 it found a better solution than Gurobi (3600s) in less time.

Table 1: Mean Results.

K	RKO			Mathematical Model		
	$Best$	Avg	T^* (s)	S	Gap (%)	T (s)
2	16.50	16.51	16.71	16.33	0.00	2.90
3	24.33	24.43	33.36	23.94	0.00	339.55
4	29.67	29.92	42.82	29.22	7.85	1454.09
5	33.83	34.03	46.87	33.39	7.52	1463.44
6	36.72	37.21	51.21	36.28	11.82	1543.40
7	39.89	40.23	46.93	39.00	12.17	1703.41
8	42.39	42.72	56.33	42.28	19.80	1827.54
9	44.83	45.23	55.31	44.44	17.54	2264.91
10	46.83	47.30	53.57	46.06	26.29	2412.47

5. Conclusions

The mathematical model found the optimal solution for problems with a limited number of vertices, edges, and partitions. Increasing the number of partitions or edges made the problem unfeasible to solve mathematically in a reasonable time. Alternatively, RKO did not provide the best known solution for all instances of the problem. However, the difference to the best known solution was relatively small to be considered a good solution and in a much shorter execution time. In the continuation of the work, restarts of the metaheuristics will be analyzed, as well as improvements in the decoder. The work will also be applied to larger graphs that may require GPU processing.

6. Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) and the Fundação de Amparo à Pesquisa do Estado de São Paulo – FAPESP (2024/08848-3).

References

- Bourse, F., Lelarge, M., and Vojnovic, M. (2014). Balanced graph edge partition. In *KDD '14 Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 1456–1465. ISBN 9781450329569.
- Chaves, A. A., Resende, M. G. C., Schuetz, M. J. A., Brubaker, J. K., Katzgraber, H. G., de Arruda, E. F., and Silva, R. M. A. (2024). A random-key optimizer for combinatorial optimization.
- Guerrieri, A. and Montresor, A. (2014). Distributed edge partitioning for graph processing. *CoRR*, abs/1403.6270. URL <http://arxiv.org/abs/1403.6270>.
- Koch, T., Martin, A., and Voss, S. (2001). SteinLib: An Updated Library on Steiner Tree Problems in Graphs. In Du, X. Z. C. D., editor, *Steiner Trees in Industry*, volume 11 of *Combinatorial Optimization*, p. 285–325. Springer, Boston, MA.
- Smorodkina, E., Thakur, M., and Tauritz, D. (2007). Algorithms for the balanced edge partitioning problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 4525 LNCS, p. 311–323. Springer Verlag. ISBN 3540728449.