Some polynomial special cases for the Minimum Gap Graph Partitioning Problem

Maurizio Bruglieri[1], Roberto Cordone[2], Isabella Lari[3], Federica Ricca[3], Andrea Scozzari[4]

[1] Politecnico di Milano
maurizio.bruglieri@polimi.it
[2] Università degli Studi di Milano
roberto.cordone@unimi.it
[3] Sapienza Università di Roma
{isabella.lari,federica.ricca}@uniroma1.it
[4] Università degli Studi Niccolò Cusano, Roma
andrea.scozzari@unicusano.it

Abstract

We study various polynomial special cases for the problem of partitioning a vertex-weighted undirected graph into p connected subgraphs with minimum gap between the largest and the smallest vertex weight.

Keywords : *Graph partitioning, min-sum gap optimization, min-max gap optimization.*

1 Introduction

The *Minimum Gap Graph Partitioning Problem* (*MGGPP*) is a graph partitioning problem [1] introduced in [2]. Let $G = (V, E)$ be an undirected connected graph, w_v an integer *weight* coefficient defined on each vertex $v \in V$, and $p \leq |V|$ a positive integer. Given a vertex subset $U \subseteq V$, we denote by $m_U = \min_{u \in U} w_u$ and $M_U = \max_{u \in U} w_u$ the minimum and maximum weight in U, respectively, and define the *gap* of U as $\gamma_U = M_U - m_U$ (if U is a singleton, $\gamma_U = 0$). The *MGGPP* consists in partitioning G into p vertex-disjoint connected subgraphs $G_r = (V_r, E_r)$, $r = 1, \ldots, p$. We consider also the *nondegenerate* problem (*MGGPPnd*), in which all subgraphs must have at least two vertices. The *min-sum* version of both problems minimizes the sum of all gaps $f^{MS} = \sum_{r=1}^{p} \gamma_{V_r}$, while the *min-max* version minimizes $f^{MM} = \max_{r=1,\ldots,p} \gamma_{V_r}$. The *MGGPP* is related to uniform graph partitioning [4] which has applications, for example, in agriculture (divide a land into parcels with small height difference [7]) and in social network analysis.

The computational complexity and the approximability of the *MGGPP* are studied in [2]. A Tabu Search metaheuristic and a Mixed Integer Linear Programming (MILP) formulation for the *min-sum* version are proposed in [3]. We note that the *min-max MGGPP* can be seen as a special case of the following more general graph partitioning problem. Given a graph G and a $n \times n$ matrix of *dissimilarities* between any pair of vertices, find a partition of G into p connected components that minimizes the maximum dissimilarity of a pair of vertices in the same component. This problem is $\mathcal{NP}$-complete even on star graphs [5]. However, the *min-max MGGPP* might be easier because the dissimilarity for any given pair of vertices u and v is computed as $|w_u - w_v|$.

In this paper, we investigate some polynomial cases of the *MGGPP* concerning special graph topologies, such as paths, spiders, stars, caterpillars and complete graphs.

2 Polynomial cases

First of all, we introduce two useful properties of the *MGGPP* (but not of the *MGGPPnd*):

P1. the optimal value does not increase as the number p of components increases;

P2. given a partition π in p components with maximum gap equal to γ, another partition π' with $p' > p$ components and a gap less than or equal to γ always exists.

These properties hold for both objectives, because a singleton with zero gap can always be disconnected from a subgraph, increasing the number of components, but not the objective value. On the basis of the above properties we are able to solve the *min-max MGGPP* by a binary search over all the $O(n^2)$ possible values γ of the optimal maximum gap, that correspond to the differences between pair of vertices' weights. Therefore, we can follow an approach based on the solution of a polynomial number of instances of the following auxiliary problem.

Definition 1 (Feasibility problem) *Find, if any, a connected partition of G having the minimum number q of components and such that each component has gap at most γ.*

If q is greater than p, the value of γ must be increased; otherwise it must be decreased. At the end of the binary search, the partition having the maximum $q \leq p$ is considered and in case $q < p$, a partition in exactly p components with the same gap value is found by further dividing some components of the partition until p components are obtained. The binary search requires a pre-sorting of the $O(n^2)$ possible values of γ implying an overall time complexity of $O(n^2 \log n)$. This complexity can be reduced as follows. First sort the weights of the vertices, in $O(n \log n)$; then consider implicitly the ordered matrix of the differences between weights and apply the $O(n)$ time procedure for finding the k^{th} smallest value in a $n \times n$ matrix of reals with sorted rows and columns [6]. With this approach the whole solution procedure requires $O(n \log n + T_F(G) \log n)$ time, where $T_F(G)$ is the time for solving the feasibility problem. We apply this approach to paths and spiders.

2.1 Paths

If G is a path, we assume its vertices to be numbered progressively considering an arbitrary direction along the path: $P = \{v_1, \ldots, v_n\}$. Every feasible solution is a partition into p vertex-disjoint subpaths. Hence, the problem requires to identify and remove $p - 1$ edges from G.

Theorem 1 *When G is a path, the* min-max MGGPP *can be solved in $O(n \log n)$ time.*

Proof : For a given value γ, we find a partition of P into the minimum number of components such that the gap is less than or equal to γ by scanning P and removing some edges. Starting from one end vertex of P, say v_1, edge (v_{i-1}, v_i) is removed as soon as a vertex v_i is found such that $|w_{v_i} - w_{v_j}| > \gamma$ for at least one vertex v_j in the current subpath. This procedure is then repeated starting from vertex v_i. Exploiting the general binary search approach the overall time complexity is $O(n \log n)$. □

For the other versions, the following theorem provides a polynomial approach.

Theorem 2 *When G is a path, all versions of the* MGGPP *can be solved in $O(n^2 p)$ time.*

Proof : (Sketch) We build an auxiliary directed graph with a source node u_0, p nodes $u_{j,1}, \ldots, u_{j,p}$ for each vertex v_j of V, an arc from u_0 to each node $u_{j,1}$ and an arc $(u_{i,r-1}, u_{j,r})$ for each pair of vertices v_i and v_j with $i < j$ and for $r = 2, \ldots, p$. The arcs correspond to candidate subpaths and their costs to the corresponding gaps. Therefore, the optimum of the *MGGPP* on G is equal to the minimum cost of a path from u_0 to $u_{n,p}$ on the auxiliary graph. Since the graph is acyclic, this problem can be solved in $O(n^2 p)$. □

2.2 Spiders

A spider is a tree with at most one vertex of degree at least 3.

Theorem 3 *When G is a spider, the* min-max MGGPP *can be solved in $O(n \log^2 n)$ time.*

Proof : We apply the procedure described in Theorem 1 to each of the d paths connecting the leaves of G to the root (the only vertex of degree $d \geq 3$), visiting G bottom-up. Let $P_1, \ldots, P_d$ be the last formed components in the partitions of such paths. We try to merge as many subpaths as possible so that their gap is $\leq \gamma$ by first ordering them w.r.t. their maximum and minimum vertex weights in $O(n \log n)$, and then checking the feasibility of the component under construction with a data structure that scans all the ordered minima and maxima in linear time. Considering the time required by the binary search, the overall time complexity is $O(n \log^2 n)$. □

2.3 Stars

A star is a connected graph with at most one vertex of degree at least 2.

Theorem 4 *When G is a star, the* MGGPP *admits only degenerate solutions, and can be solved in $O(n \log n)$ time.*

Proof : Any solution is a partition of the star where $p-1$ components are leaves (singletons) and one component contains all the other vertices. An $O(n \log n)$ time algorithm can be obtained as follows: i) sort the leaves by nonincreasing weights; ii) visit the leaves according to such an ordering to detect a non-singleton component with minimum gap. □

2.4 Caterpillars

A caterpillar is a tree formed by a central path with n' vertices and n'' leaves attached to it.

Theorem 5 *When G is a caterpillar, the* MGGPP *can be solved in $O(n^3 p^2 \log n)$ time; the* MGGPPnd *in $O(n^2 p)$ time.*

Proof : The *MGGPPnd* amounts to partitioning the central path $\{v_1, \ldots, v_{n'}\}$. We build an auxiliary graph similar to that used in Theorem 2. An arc represents a feasible subgraph, i.e., a portion of the central path and all the attached leaves. Its cost is the gap of the subgraph. The optimal *min-sum* or *min-max* path from u_0 to $u_{n'p}$ identifies the optimal solution. Building the auxiliary graph, the cost function and detecting the optimal path take $O(n^2 p)$.

In the general case, the leaves can be isolated from the central path, but the auxiliary graph can be modified accordingly, i.e. introducing also arcs between non consecutive layers. Each arc represents a central subgraph including a portion of the central path and possibly some leaves, plus some isolated leaves. The cost of an arc is the gap of the central subgraph. For each arc, we determine the central subgraph with minimum gap adapting the $O(n \log n)$ algorithm used to solve the *MGGPP* on stars. The optimal *min-sum* or *min-max* path from u_0 to $u_{n'p}$ identifies the optimal solution. Detecting the optimal path takes $O(n^2 p^2)$, but computing the arc costs requires an overall $O(n^3 p^2 \log n)$ time. □

2.5 Complete graphs

In this case, the connectivity constraint is trivially satisfied. We sort the vertices by nondecreasing weights and rename them so that $i < j \Rightarrow w_{v_i} \leq w_{v_j}$. Then, we can restrict the search for the optimum to the solutions in which for every pair of subgraphs the vertices of one strictly precede the vertices of the other. In fact, every other feasible solution can be transformed into a non-worsening one of this family: for each pair of vertex subsets V' and V'' violating this condition, merge the two subsets and split the result in two, assigning the first $|V'|$ elements to the first subset and the last $|V''|$ elements to the second. Given the path that visits all vertices in nondecreasing weight order, the optimal solution can be detected as in Theorems 1 and 2. However, the vertex ordering allows more efficient algorithms for some cases.

Theorem 6 *When G is a complete graph, the* min-sum MGGPP *can be solved in $O(n \log n)$ time. The* min-sum MGGPPnd *can be solved in $O\left(\min\left(n\sqrt{n}\log\gamma_V, n^2\log n, n^2 p\right)\right)$ time, where γ_V is the gap of the whole graph.*

Proof : Thanks to the ordering of the vertex weights, partitioning the auxiliary path into p subpaths of minimum total gap is equivalent to selecting $p-1$ edges such that the sum of the weight differences between their extreme vertices is maximum. This can be done by saving in a *max-heap* the weight differences between adjacent vertices and extracting from it the $p-1$ largest differences. The dominating time is given by the weight ordering.

In the nondegenerate case, it is forbidden to select two consecutive edges. The problem reduces to the search for a maximum weight matching of cardinality $p-1$ on the path. The Enhanced Capacity Scaling algorithm solves the problem in $O\left(n\sqrt{n}\log\gamma_V\right)$ time, while a reduction to the minimum cost flow problem solves it in $O\left(n^2\log n\right)$ time. Finally, the algorithm of Theorem 2 solves it in $O\left(n^2 p\right)$ time. □

3 Conclusions and perspectives

Table 1 summarizes the special cases discussed in this paper: "NA" marks the non applicable cases (stars admit only degenerate solutions), "?" marks the open cases.

	min-max		*min-sum*	
Topology	*MGGPP*	*MGGPPnd*	*MGGPP*	*MGGPPnd*
Stars	$O(n \log n)$	NA	$O(n \log n)$	NA
Paths	$O(n \log n)$	$O(n^2 p)$	$O(n^2 p)$	$O(n^2 p)$
Spiders	$O(n \log^2 n)$	?	?	?
Caterpillars	$O(n^3 p^2 \log n)$	$O(n^2 p)$	$O(n^3 p^2 \log n)$	$O(n^2 p)$
Complete	$O(n \log n)$	$O(n^2 p)$	$O(n \log n)$	$O(min(n\sqrt{n}\log\gamma_V, n^2\log n, n^2 p))$

TAB. 1: Summary of the computational complexity results

References

[1] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, eds. *Graph Partitioning and Graph Clustering*, v. 588 of *Contemporary Mathematics*. AMS, 2013.

[2] M. Bruglieri, R. Cordone Partitioning a graph into minimum gap components. *Electronic Notes in Discrete Mathematics*, vol. 55: 33–36, 2016.

[3] M. Bruglieri, R. Cordone, V. Caurio A metaheuristic for the minimum gap Graph Partitioning Problem. *Proceedings of CTW2017*, pp. 23–26, Cologne, Germany, June 6-8 2017.

[4] I. Lari, J. Puerto, F. Ricca, A. Scozzari Partitioning a graph into connected components with fixed centers and optimizing cost-based objective functions or equipartition criteria. *Networks*, vol. 67: 69–81, 2016.

[5] M. Maravalle, B. Simeone, R. Naldini Clustering on trees. *Computational Statistics & Data Analysis*, vol. 24: 217–234, 1997.

[6] A. Mirzaian, E. Arjomandi Selection in X+Y and matrices with sorted rows and columns. *Information Processing Letters*, vol. 20: 13–17, 1985.

[7] Li Xiao, Li Hongpeng, Niu Dongling, Wang Yan, and Liu Gang. Optimization of GNSS-controlled land leveling system and related experiments. *Transactions of the Chinese Society of Agricultural Engineering*, 31(3):48–55, 2015.