European Journal of Operational Research 236 (2014) 695–705

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Column-generation based bounds for the Homogeneous Areas Problem

Fabio Colombo, Roberto Cordone *, Marco Trubian

Università degli Studi di Milano, Dipartimento di Informatica, Via Comelico 39, 20135 Milano, Italy

ARTICLE INFO

Article history:
Received 2 May 2012
Accepted 23 December 2013
Available online 31 December 2013

Keywords:
Graph partitioning
Column Generation
Tabu Search

ABSTRACT

Given an undirected graph and a collection of vertex subsets with suitable costs, we consider the problem of partitioning the graph into subgraphs of limited cost, splitting as little as possible the given subsets among different subgraphs. This problem originates from the organization of a region (the graph) including several towns (the vertices) into administrative areas (the subgraphs). The officers assigned to each area take care of activities which involve several towns at a time (the subsets). An activity involving towns from more areas engages the officers of all those areas, leading to redundancies which must be minimized.

This paper introduces a column generation approach to compute a lower bound for the problem. Since the pricing subproblem is $\mathcal{NP}$-hard, we solve it with a Tabu Search algorithm, before applying a suitably strengthened multi-commodity flow formulation. Moreover, we also compute an upper bound for the overall problem with a primal heuristic based on the idea of diving and limited discrepancy search. The computational results refer to two real-world instances, a class of realistic instances derived from them, and two different classes of random instances.

1. Introduction

The *Homogeneous Areas Problem* (*HAP*) can be formulated as follows. Let $G=(V,E)$ be an undirected graph with $V=\{1,\dots,n\}$, $\mathcal{S}\subseteq 2^V$ a collection of subsets of vertices, $q:\ \mathcal{S}\rightarrow\mathbb{R}^+$ a cost function defined on $\mathcal{S}$ and Q a cost threshold. Finally, let k be an integer positive number. Given any subset of vertices $U\subseteq V$, we denote the cost of the subgraph induced by U as $\sum_{S\in\mathcal{S}:S\cap U\neq\emptyset} q_S$, i.e. the sum of the values q_S for all subsets S intersecting U. The problem requires to partition graph G into at most k vertex-disjoint connected subgraphs $G_i=(U_i,E_i)$ such that the cost of G_i does not exceed Q for all i and the total cost

$$\phi=\sum_i\sum_{S\in\mathcal{S}:S\cap U_i\neq\emptyset} q_S \qquad (1)$$

is minimum. The *HAP* is strongly $\mathcal{NP}$-hard (Ceselli, Colombo, Cordone, & Trubian, in press).

This problem derives from a practical requirement, concerning the partitioning of two administrative regions in Northern Italy (the provinces of Milan and Monza) into "homogeneous areas". In that case, the vertices correspond to towns, the edges to pairs of adjacent towns, each subset $S\in\mathcal{S}$ represents an activity involving a subset of towns and requiring from the officers of the province administration a known amount of working hours, q_S. The aim of the problem is to divide the province into connected areas (subgraphs) and to assign a team of officers to each area, in such a way that each activity is split as little as possible among different areas. This is due to the fact that the officers in charge of an area need to be trained on all the activities involving the towns of the area and, therefore, splitting an activity implies a redundancy (more officers trained on the same topics). The cost of a subgraph expresses the number of working hours required from the officers in charge of the corresponding area. The limited number of working hours available for each officer imposes an upper threshold on the cost of each area. The value of this threshold can also be tuned to improve fairness among the teams.

In Fig. 1 we report a sample instance and some of its solutions. Fig. 1(a) provides a graph G with 7 vertices and 9 edges, three subsets with costs $q_{S_1}=10, q_{S_2}=9$ and $q_{S_3}=11$, a cost threshold $Q=25$ and a maximum number of subgraphs $k=3$. If the nodes could be partitioned so as to keep all subsets in $\mathcal{S}$ unsplitted, the overall cost would hit the theoretical lower bound $\sum_{S\in\mathcal{S}} q_S=30$. This value, however, cannot be reached due to the cost threshold imposed on each subgraph. Fig. 1(b) shows an optimal solution, with two subgraphs and a total cost equal to $q_{S_1}+2q_{S_2}+q_{S_3}=39$ (subset S_2 intersects both the subgraphs). Fig. 1(c) shows a suboptimal solution with three subgraphs, in which both S_1 and S_2 intersect two subgraphs, so that the overall cost is $2q_{S_1}+2q_{S_2}+q_{S_3}=49$. Finally, the solution in Fig. 1(d) is unfeasible since the subgraph induced by $U=\{1,2,3,4,5\}$ intersects all three subsets and its cost $q_{S_1}+q_{S_2}+q_{S_3}=30$ exceeds the threshold $Q=25$.

* Corresponding author. Tel.: +39 0250316235; fax: +39 0250316373.
E-mail addresses: fabio.colombo2@unimi.it (F. Colombo), roberto.cordone@unimi.it (R. Cordone), marco.trubian@unimi.it (M. Trubian).

0377-2217/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2013.12.030

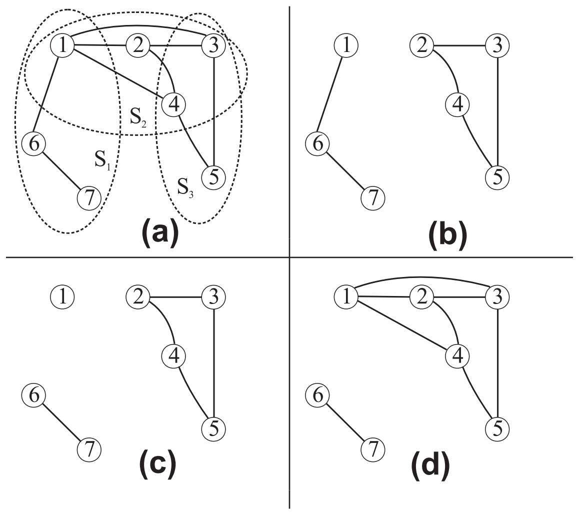

Fig. 1. (a) Sample instance having $q_{S_1} = 10, q_{S_2} = 9, q_{S_3} = 11$ and $Q = 25$; (b) Optimal solution with cost $q_{S_1} + 2q_{S_2} + q_{S_3} = 39$; (c) Sub-optimal solution with cost $2q_{S_1} + 2q_{S_2} + q_{S_3} = 49$; (d) Unfeasible solution: the subgraph induced by $U = \{1,2,3,4,5\}$ has cost $q_{S_1} + q_{S_2} + q_{S_3} = 30 > Q$.

This work proposes a column generation approach to the *HAP*. Since in our decomposition the pricing subproblem is itself $\mathcal{NP}$-hard, we apply a customized heuristic before solving it exactly with an Integer Linear Programming (*ILP*) formulation, for which we introduce some valid inequalities. The heuristic is a Tabu Search algorithm, while the exact approach exploits a multicommodity flow formulation. Section 2 introduces a compact formulation of the problem, and an extended one, solved by a column generation approach. Section 3 deals with the pricing subproblem, discussing its formulation with some strengthenings, its computational complexity and a heuristic approach to solve it. Section 4 presents a primal heuristic for the *HAP*. It is based on the column generation framework and exploits the concepts of diving and limited discrepancy search. The final section presents the computational results.

1.1. On the relationship with graph partitioning problems

The *HAP* can be seen as a variant of the Graph Partitioning Problem (*GPP*). This section provides some references to the huge literature on the *GPP*, a small example to illustrate the specific features of the *HAP* and a discussion of the differences it exhibits with respect to the other related models. A more detailed discussion, with counterexamples to the possibility of easily reducing the *HAP* to a standard *GPP*, can be found in Ceselli et al. (in press).

Given an undirected edge-weighted graph $G = (V, E)$, the most common versions of the *GPP* ask to divide the vertex set V into a given number k of nonempty, pairwise disjoint subsets, such that the edge-cut, i.e. the total weight of the edges that connect vertices in different subsets, is minimized. This basic problem admits a number of variations; see, e.g., the survey in Fjällström (1998). Several different approaches have been proposed to solve them, such as hierarchic multi-level heuristics (Sanders & Schulz, 2011), geometry-based and flow-based methods (Arora, Rao, & Vazirani, 2008), genetic approaches (Kim, Hwang, Kim, & Moon, 2011), spectral methods (Donath & Hoffman, 1973), mathematical programming approaches (Fan & Pardalos, 2010), local search metaheuristics and integrated approaches (Osipov, Sanders, & Schulz, 2012). The *HAP* differs from these classical *GPP*s both in the constraints and in the objective function, which pose specific challenges to a solving algorithm.

1.1.1. Cardinality constraint

In the *GPP*, the number k of vertex-disjoint subsets is usually given, and the subsets are required to be nonempty, since their cardinalities, $n_1, \ldots, n_k$, with $\sum_{j=1}^{k} n_j = |V|$ are explicitly imposed (Guttmann-Beck & Hassin, 2000) or constrained to be approximately of the same size, see e.g. Osipov et al. (2012). In the *HAP*, k is just an upper threshold, so that *the subsets of vertices* U_i *are allowed to be empty*. In fact, merging two subsets into a single one is always profitable, and only the cost threshold Q possibly forbids to do it.

1.1.2. Cost threshold

The *HAP* is related to the *Node-capacitated Graph Partitioning Problem* (Ferreira, Martin, de Souza, Weismantel, & Wolsey, 1998), in which the total weight of each subset in the partition is limited by a threshold. However, the threshold is managed differently in the *HAP*: since it is not associated to single vertices, the cost of a subset U does not increase linearly as new vertices are included, but stepwise as new subsets $S \in \mathcal{S}$ intersect U. Such a nonlinear dependence is much harder to handle.

1.1.3. Connectivity constraint

The connectivity constraint is usually not imposed in *GPP*s, where the edges of the graph are taken into account only when computing the objective function. Quite commonly, the edge costs model a proximity measure, and the subsets end up naturally to be connected in the optimal solution. In the *HAP*, on the contrary, the edges determine the feasibility of the solutions, since each subset must induce a connected subgraph on G, but they have no relation with the objective function. In fact, even considering the smaller benchmark instances, which can be solved exactly, the optimal result obtained relaxing the connectivity constraints is on average 35% lower than the one obtained respecting them (Ceselli et al., in press). This suggests that neglecting the connectivity constraint would not provide meaningful information on the original problem and that classical methods ignoring this constraint would not provide useful solutions.

1.1.4. Objective function

The objective function of the classical *GPP*s depends linearly on the cost of the edges whose extreme vertices belong to different subgraphs. Sometimes, this cost is tuned by a function of the cardinality of the subgraphs; see, e.g., Matula and Shahrokhi (1990). The objective function of the *HAP* is completely independent from the edge set E, and depends nonlinearly on the intersections between the subsets in $\mathcal{S}$ and the subsets of vertices of the subgraphs.

These remarks on the difference between the constraints and the objective function of the *HAP* with respect to other *GPP*s have moved us to develop *ad hoc* methods, instead of straightforwardly adapting algorithms drawn from the literature.

2. Mathematical programming formulations

Hereafter, we present two different formulations of the *HAP*. The first one is a compact multicommodity flow formulation that can be directly solved using a commercial *ILP* solver. The second one is an extended formulation which associates a variable to each feasible subgraph. At the end of the section, we describe the column generation approach used to solve the continuous relaxation of the extended formulation.

2.1. Compact formulation

The *HAP* admits a multicommodity flow formulation based on an auxiliary directed graph $G' = (V, E')$, derived from G replacing