Computers & Operations Research 41 (2014) 174–184

Contents lists available at ScienceDirect

Computers & Operations Research

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

A branch-and-bound algorithm for the acyclic partitioning problem

Jenny Nossack*, Erwin Pesch

Department of Management Information Science, University of Siegen, Hölderlinstraße 3, D-57068 Siegen, Germany

ARTICLE INFO

Available online 20 August 2013

Keywords:
Graph partitioning
Branch-and-bound
Constraint propagation
Acyclic graph
Container transshipment

ABSTRACT

We focus on the problem of partitioning the vertex set of a directed, edge- and vertex-weighted graph into clusters, i.e., disjoint subsets. Clusters are to be determined such that the sum of the vertex weights within the clusters satisfies an upper bound and such that the sum of the edge weights within the clusters is maximized. Additionally, the graph is enforced to partition into a directed, acyclic graph where the clusters define the vertices. This problem is known as the acyclic partitioning problem and is NP-hard. Real-life applications arise, for example, in VLSI design and at rail–rail transshipment yards. We propose an integer programming formulation for the acyclic partitioning problem and suggest an exact solution approach based on a branch-and-bound framework that integrates constraint propagation. Computational results are reported to confirm the strength of our solution proposal.

1. Introduction

In this paper, we consider a variant of the classical *graph partitioning problem*, or simply *partitioning problem*, where the vertex set of an edge-weighted graph is to be partitioned into k disjoint subsets (also referred to as *clusters*), such that the sum of the edge weights within the clusters is maximized (or equivalently the sum of the edge weights between different clusters is minimized). The *acyclic partitioning problem*, defined on a directed, edge-, and vertex-weighted graph, additionally restricts the size of the clusters and enforces the graph to partition into a directed, acyclic graph, i.e., a graph that contains no directed cycle. The acyclic partitioning problem belongs to the class of NP-hard problems and remains NP-hard even if all vertex weights and all edge weights are equal to 1, cf. [1].

Graph partitioning problems subject to additional constraints arise in a wide area of applications, including VLSI (Very Large Scale Integration) design [2], qualitative data analysis [3,4], computer register allocation [5], and finite element computation [6]. The problem instances that have motivated this research are encountered at rail–rail transshipment yards where gantry cranes allow for an efficient transshipment of containers between different freight trains. A rail–rail transshipment yard is typically operated in a so-called *train bundle* which is a set of trains that simultaneously enters the transshipment yard. This train bundle is then served by cranes within a certain time slot and jointly leaves the yard only after all container transshipments have been processed. An important problem that emerges during the daily operations is the *train scheduling problem* [7] which decides on the bundling of trains under the objective to minimize the number of *split moves* and *train revisits* to the yard. Split moves appear whenever containers have to be exchanged between trains of different bundles, whereas revisits occur if a train has to enter the yard twice, because some container dedicated to this train was not available during its first visit. A modification of this problem forbids train revisits and solely meets the objective to minimize the number of crane movements. This outlined problem can be modeled as an acyclic partitioning problem.

A considerable amount of literature has been published on graph partitioning problems. These studies investigate different problem variants, as well as different solution approaches. Variants of the partitioning problem, e.g., restrict the number of clusters k, the size of the clusters, or are defined on specific graph topologies. Specifications include the *bi-partitioning* ($k:=2$), the *node-capacitated partitioning*, as well as the *hypergraph partitioning*. For references to these and other related problems we refer, e.g., to Alpert and Kahng [2]. Solution algorithms are based on local search approaches, clustering methods, geometric representations, and combinatorial formulations. We only mention some of these solution approaches and refer the reader to the detailed algorithm surveys by Alpert and Kahng [2] and Fjällström [8]. Most prominent is the variable depth search algorithm by Kernighan and Lin [9] for bi-partitioning problems. Fiduccia and Mattheyses [10] modify this algorithm by restricting cluster exchanges to single vertices and Sanchis [11] adapt the Fiduccia–Mattheyses algorithm to general graph partitioning problems.

A problem that is closely related to the acyclic partitioning problem is known in the literature as the *clique partitioning problem*. This problem is to partition the vertex set of an undirected graph with real-valued edge weights into cliques, i.e.,

* Corresponding author. Tel.: +49 271 740 3402; fax: +49 271 740 2940.
E-mail addresses: jenny.nossack@uni-siegen.de (J. Nossack), erwin.pesch@uni-siegen.de (E. Pesch).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.08.013

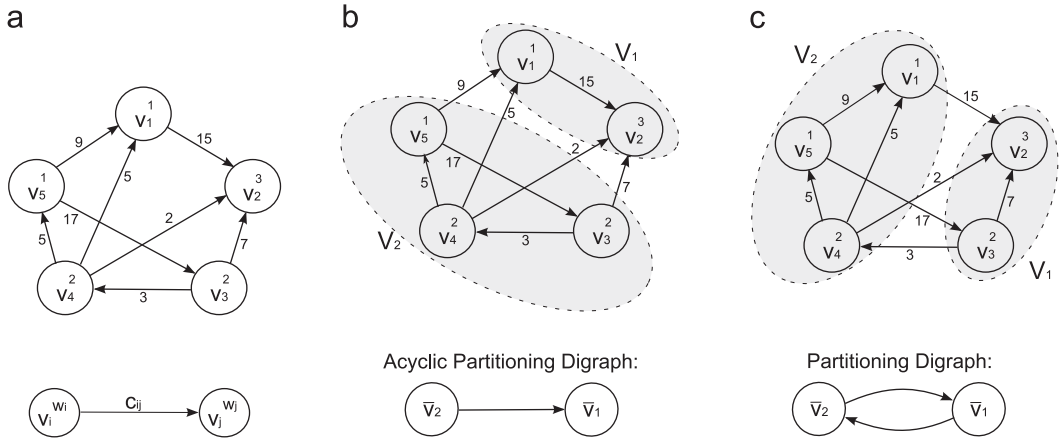

Fig. 1. Partition and acyclic partition. (a) Digraph, (b) acyclic partition and (c) partition.

complete subgraphs, such that the sum of the edge weights within the clusters is maximized. Grötschel and Wakabayashi [4] analyze the polyhedral structure of the clique partitioning polytope and develop a cutting plane solution approach [3]. Oosten et al. [12] extend the cutting plane method of Grötschel and Wakabayashi [3,4] by defining further facet-defining inequalities. An ejection chain heuristic, as well as an exact solution method based on a branch-and-bound framework are presented by Dorndorf and Pesch [13] for the same problem. Jaehn and Pesch [14] suggest new bounds, a different branching strategy, and an accelerated constraint propagation for the branch-and-bound algorithm of Dorndorf and Pesch [13].

Another further related problem is the *node-capacitated partitioning problem* which imposes so-called *knapsack constraints* on the clusters of an edge- and vertex-weighted graph to restrict the sum of the vertex weights within the clusters. Holm and Sørensen [15] consider the problem where the number of clusters k is prespecified. They implement a branch-and-cut algorithm that includes cuts to reduce the symmetric nature of the suggested problem formulation in order to keep the size of the branch-and-bound tree as small as possible. Johnson et al. [16] suggest an augmented set partitioning formulation for the same problem and solve it by a branch-and-price approach. The pricing problem is thereby solved by a branch-and-cut algorithm. Mehrotra and Trick [17] also address this problem by a branch-and-price method and solve the subproblem as combinatorial optimization problem. Ferreira et al. [18] suggest a multi-cut formulation for the node-capacitated partitioning problem and provide several cutting planes which they embed in Ferreira et al. [6] in a branch-and-cut framework. Ji and Mitchell [19] consider the clique partitioning problem where each clique is constrained to hold a minimum number of vertices. They suggest a branch-and-price-and-cut method to solve this problem class.

The literature on the acyclic partitioning problem, however, is rather limited. Lukes [20] presents a pseudo-polynomial time algorithm for the acyclic partitioning problem on a tree graph topology. If all edge weights [21], equivalently, if all vertex weights [1] are equal, the acyclic partitioning problem on trees can even be solved in polynomial time. Cong et al. [22] modify the partitioning algorithm by Sanchis [11] to heuristically solve the acyclic partitioning problem on directed graphs with unit vertex and unit edge weights. The authors randomly generate initial solutions based on topological ordered vertices and restrict the vertex exchanges to those that maintain the acyclic property, i.e., a topological vertex order. A precise definition of a topological ordered graph is presented in Section 3.

The main contribution of our work is the presentation of an exact solution algorithm, as well as an integer model formulation for the acyclic partitioning problem on directed graphs with non-negative, integer vertex and edge weights. Our suggested algorithm is based on a thorough problem analysis and a branch-and-bound framework. To our knowledge this is the first attempt to exactly solve the acyclic partitioning problem on directed graphs.

The remainder of the paper is organized as follows. In Section 2, we give an overview of the notation used in this paper, a detailed problem description, and an integer model formulation. A thorough problem analysis of the acyclic partitioning problem is followed in Section 3. The customized components of our branch-and-bound algorithm are discussed in Section 4. In Section 5, we summarize the results of our computational study which we conduct on randomly generated instances. Finally, we conclude our research in Section 6.

2. Notation, problem definition, model formulation

The following notation is used throughout the paper and follows standard combinatorial optimization books such as Korte and Vygen [23] and Schrijver [24]. Let $D=(V,A)$ denote a directed graph (or simply digraph) with vertex set $V=\{v_1,\ldots,v_n\}$ and edge set $A\subseteq\{(v_i,v_j)|v_i,v_j\in V\}$. The considered digraph D is assumed to be finite, loopless, and without multiple edges. Throughout the paper n denotes the number of vertices ($n:=|V|$) and m the number of edges ($m:=|A|$). We associate with each vertex $v_i\in V$ a vertex weight $w_i\in\mathbb{N}^+$ and with each edge $(v_i,v_j)\in A$ an edge weight $c_{ij}\in\mathbb{N}^+$.

A *directed walk* (or simply *walk*) of a digraph D is defined as a sequence of edges $W=((v_i,v_{i+1}),(v_{i+1},v_{i+2}),\ldots,(v_{j-2},v_{j-1}),(v_{j-1},v_j))$. If vertices $v_i,\ldots,v_j$ are all distinct, the walk is called a *directed path* (or simply *path*). If $v_i=v_j$ and $v_i,\ldots,v_{j-1}$ are all distinct, we refer to a walk as a *directed cycle* (or simply *cycle*). A digraph that contains no directed cycle is referred to as an *acyclic digraph* and a digraph with at least one directed cycle is referred to as a *cyclic digraph*.

A *partition* $P=\{V_1,\ldots,V_k\}$ of a digraph D is defined as the collection of k disjoint subsets (i.e., *clusters*) of vertices, $V_1,\ldots,V_k$, such that $\bigcup_{s=1,\ldots,k}V_s=V$ and $V_s\cap V_t=\varnothing$ for all $s,t=1,\ldots,k$ and $s\neq t$. The set of edges connecting vertices of different clusters is called a *cut* and is denoted by $\delta(P):=\{(v_i,v_j)\in A|v_i\in V_s,v_j\in V_t;s,t=1,\ldots,k;s\neq t\}$. The sum of the edge weights defined within the clusters is denoted as the *value* of a partition, i.e., *value*(P):=