Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

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

A lower bound on the Hamiltonian path completion number of a line graph

Paolo Detti [a], Carlo Meloni [b,*], Marco Pranzo [a]

[a] D.I.I. Università di Siena, Via Roma, 56, I-53100 Siena, Italy
[b] D.E.E. Politecnico di Bari, Via E. Orabona, 4, I-70125 Bari, Italy

ARTICLE INFO

Keywords:
Hamiltonian path completion number
Line graphs
Dominating trail
Lower bound

ABSTRACT

Given a line graph $L(G)$ of a graph $G = (V, E)$, the problem of finding the minimum number of edges to add to $L(G)$ to have a Hamiltonian path on $L(G)$ is considered. This problem, related to different applications, is known to be NP-hard. This paper presents an $O(|V| + |E|)$ algorithm to determine a lower bound for the Hamiltonian path completion number of a line graph. The algorithm is based on finding a collection of edge-disjoint trails dominating all the edges of the root graph G. The algorithm is tested by an extensive experimental study showing good performance suggesting its use as a building block of exact as well as heuristic solution approaches for the problem.

1. Introduction

The Hamiltonian path completion number of a graph $G = (V, E)$ is the minimum number of edges which need to be added to G to make it *traceable*, i.e., admitting an Hamiltonian path, and is denoted as *HPCN (G)* [26]. This graph traceability problem is subject of active research [5,11,14,26], and in this paper we investigate the problem restricted to a particular class of graphs, called *line* graphs. The line graph $L(G)$ of $G = (V, E)$ is a graph having $|E|$ nodes, each node of $L(G)$ being associated to an edge of G. If two edges of G are adjacent then there is an edge between the two corresponding nodes of $L(G)$. Linear-time algorithms exist to recognize a *line* graph $L(G)$ and to obtain its *root* graph G, see e.g. [19].

The *HPCN (G)* problem is strongly related to the problem of finding a *minimum dominating trail set* (*MDTS*), which is a dominating trail set of minimum cardinality. In particular, the line graph $L(G)$ of a graph G has a Hamiltonian path if and only if G has a dominating trail. Moreover, for a graph G, the value of *HPCN (L (G))* is determined by the cardinality of a *MDTS* [15].

Finding *HPCN (L (G))* and a *MDTS* on G are known to be *NP*–hard for general [6] and bipartite [2,18] graphs. Polynomial time algorithms have been found for some graph classes [3,22,11,16,10,7], and particular special conditions on G have been found to ensure the existence of a Hamiltonian path on $L(G)$ [24]. Moreover, *MDTS* of G and *HPCN (L (G))* are strongly related to the *total interval number* of G [22].

A study on *HPCN (G)* of random graphs has been proposed by Gamarnik and Sviridenko [14], while Nikolopoulos [21] addresses the study of the problem on quasi-threshold graphs proposing an efficient parallel algorithm for that case.

Applications to the problem of finding a *MDTS* on G and *HPCN (L (G))* are related to different contexts such as, for example, sequencing problems in manufacturing systems [1,2,8], VLSI design [20] and page retrieval problems in relational database systems [18].

In [2], the problem of coordinating two consecutive production departments is addressed, with the objective of minimizing the number of set-ups required to process a given set of production batches. More in details, the batches must be processed by

* Corresponding author.
E-mail addresses: detti@dii.unisi.it (P. Detti), meloni@deemail.poliba.it (C. Meloni), pranzo@dii.unisi.it (M. Pranzo).

0096-3003/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.amc.2013.06.020

the two departments in the same order and each batch is characterized by two attributes, say shape and color. Given two consecutive batches, a set-up is paid by the first (second) department if the shape (color) of the first batch is different from the shape (color) of the second batch. The problem consists in finding a batch sequence such that the total number of set-ups incurred in the two departments is minimized. The input of the problem can be represented as a bipartite graph G, in which each batch corresponds to an edge. Then, the problem is to find an ordering of the edges of the graph G minimizing the number of times two consecutive edges have no node in common. Observe that a dominating trail, if any, on the bipartite graph G, completely specifies a batch schedule in which consecutive edges (i.e., batches) of the dominating trail have a node in common. If no dominating trail exists in G, *HPCN (L (G))* is the number of times in which both departments have to pay a set-up and an *MDTS* specifies the batch schedule. Recently, in [8], this work has been extended considering its two-objective versions, while in [9] it has been extended to consider a general set-up cost structure. In [1], a production system in which two competitive users share a common resource to execute a set of processes is studied. The problem of sequencing the processes on the resource is formulated and efficiently solved by finding a *MDTS* on particular graphs.

In [18], the edge-sequencing on the page connectivity graph of a database is considered to minimize the total number of page accesses. In particular, a graph G represents a relational join operation. Each node represents a data page of a relation which is stored in secondary memory. Two pages are said to be connected and their corresponding nodes are linked by an edge if an information in one page needs to be joined with an information in the other. Before to perform a join operation on two connected pages, they must be fetched in the main memory. The minimum number of fetch operations, i.e., the minimum number of page accesses, required to perform all the join operations corresponds to *HPCN (L (G))*.

In this paper, a linear time algorithm for computing a lower bound of *HPCN (L (G))* is proposed. It is based on finding a (minimum cardinality) set of trails dominating all the edges of G.

The paper is organized as follows. In Section 2, some notation is introduced, the *MDTS* problem is defined discussing also its relationship with *HPCN (G)*. In Section 3, the graph transformations used by the algorithm are described, and, in Section 4, a linear algorithm to compute a lower bound for *HPCN (L (G))* is illustrated discussing also the implementation issues and the complexity aspects. The performance of the algorithm is investigated by an extensive experimental study reported in Section 5. A summary and the conclusions addressing also further research directions form the arguments of the last section.

2. Definitions and notation

Given a graph $G = (V, E)$, a *trail* is a sequence $w := (v_0, e_0, v_1, e_1, v_2, e_2, \dots, e_{k-1}, v_k)$, where $(v_0, v_1, v_2, \dots, v_k)$ are nodes of G, $(e_0, e_1, e_2, \dots, e_{k-1})$ are distinct edges of G, and v_i and v_{i+1} are the endpoints of e_i for $0 \leqslant i \leqslant k-1$. The trail is a *path* if its nodes $(v_0, v_1, v_2, \dots, v_k)$ are distinct, and it may consist of a single node. An edge $e \in E$ is *dominated* by a trail w in G (i.e., w *dominates* e), if e has at least one endpoint belonging to w. A *dominating trail* D_T in G is a trail that dominates all the edges of G. In a given graph G a dominating trail may not exist. A *dominating trail set* $\Sigma(G)$ is a collection of edge-disjoint trails that altogether dominate all the edges of G. A *minimum dominating trail set* (*MDTS*) is a dominating trail set of minimum cardinality. Given a dominating trail D_T (a dominating trail set $\Sigma(G)$) of G, we say that D_T $(\Sigma(G))$ dominates G.

Harary and Nash-Williams [15] link the problem of finding Hamiltonian completion number and the problem of finding a *MDTS*, showing that the line graph $L(G)$ of a graph G has a Hamiltonian path if and only if G has a dominating trail. Moreover, $HPCN(L(G)) = k$ if and only if the cardinality of a *MDTS* of G is $k+1$.

The algorithm we propose provides a lower bound on the cardinality of a *MDTS* of G. Given a general connected graph G, the algorithm is based on transforming the graph into a particular tree, with the property that the cardinality of a *MDTS* of the tree is a lower bound to the cardinality of a *MDTS* of G, and hence to *HPCN (L (G))*. This algorithm has been used to obtain a stopping criterion in an Iterated Local Search (ILS) algorithm [12] for computing the Hamiltonian path completion number of a line graph. Moreover, the proposed lower bound procedure has been also applied to solves different sequencing problems in setup minimization in a two-stage serial production system [13,8,9].

The algorithm we propose for computing a lower bound of *HPCN (L (G))* is based on finding a *MDTS* of G. In Definition 1, the problem of finding a *MDTS* is formally defined.

Definition 1. Given a graph G, a *dominating trail set* $\Sigma(G)$ is a collection of edge-disjoint trails such that $\Sigma(G)$ dominates all the edges of G. The *minimum dominating trail set* (*MDTS*) problem is the problem of finding a dominating trail set of minimum cardinality.

Let $\mu : V \to \{0, 1\}$ be a marking function. A node i such that $\mu(i) = 1$ is called *marked*. Marking a node means that we want to find a *MDTS* on G in which at least one element of the trail set passes through that node of the graph. In the following, we denote as (G, μ) the pair composed by the graph G and the marking function μ. Given a pair (G, μ), the problem of finding a *minimum constrained dominating trail set* (*MCDTS*) is defined in Definition 2.

Definition 2. Given a pair (G, μ), a *constrained dominating trail set* $\Sigma(G, \mu)$ is a collection of edge-disjoint trails such that: (1) $\Sigma(G, \mu)$ dominates all the edges of G; (2) for each marked node i $(\mu(i) = 1)$, a trail $t \in \Sigma(G, \mu)$ containing i exists. The *minimum constrained dominating trail set* (*MCDTS*) problem is the problem of finding a constrained dominating trail set of minimum cardinality.