Discrete Applied Mathematics 210 (2016) 103–111

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

On the monotonicity of process number

Nicolas Nisse [a,b], Ronan Pardo Soares [a,b,c,*]

[a] Inria, France
[b] Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271, 06900 Sophia Antipolis, France
[c] ParGO - Univ. Federal do Ceará, Brazil

ARTICLE INFO

Article history:
Received 16 December 2013
Received in revised form 11 October 2014
Accepted 29 January 2015
Available online 7 March 2015

Keywords:
Graph searching
Process number
Monotonicity

ABSTRACT

Graph searching games involve a team of searchers that aims at capturing a fugitive in a graph. These games have been widely studied for their relationships with the tree- and the path-decomposition of graphs. In order to define decompositions for directed graphs, similar games have been proposed in directed graphs. In this paper, we consider a game that has been defined and studied in the context of routing reconfiguration problems in WDM networks. Namely, in the *processing game*, the fugitive is invisible, arbitrarily fast, it moves in the opposite direction of the arcs of a digraph, but only as long as it can access to a strongly connected component free of searchers. We prove that the processing game is monotone which leads to its equivalence with a new digraph decomposition.

1. Introduction

During the last few years, an important research effort has been done in order to design digraph decompositions that are as powerful as the path-decomposition or the tree-decomposition in the case of undirected graphs (e.g., see [13]). Because graph searching games are equivalent to path- and tree-decompositions in undirected graphs, several proposals have been done to define such games in directed graphs [2,15,3]. In this paper, we study the *processing game* and show its equivalence with a new digraph decomposition.

1.1. Graph searching and monotonicity

In graph searching games, a team of *searchers* aims at capturing a fugitive that stands at the vertices of a connected graph G (see [11] for a survey). Initially, no searchers are occupying the vertices of G and any vertex may host the fugitive. The vertices are initially said *contaminated*. The fugitive can move by following the paths of G while it does not meet any searcher. The fugitive is arbitrarily fast in the sense that its moves are instantaneous whatever be the length of the paths it follows. A *strategy* for the searchers is a sequence of the following two possible actions: ($Place(v)$) place a searcher at a vertex v of G, or ($Remove(v)$) remove a searcher from a vertex v. When a searcher occupies a vertex, this vertex becomes *clear*. However, a clear vertex v is *recontaminated* if it is not occupied and there is a path without searchers from v to a contaminated vertex. In other words, a vertex $v \in V(G)$ remains clear if all paths from v to a contaminated vertex contain a vertex occupied by a searcher. In particular, a vertex occupied by a searcher is clear.

A strategy is *winning* if it allows to capture the fugitive whatever it does or, equivalently, if all vertices are eventually clear. That is, in a winning strategy, a searcher eventually occupies the same vertex as the fugitive and the fugitive cannot

* Corresponding author at: ParGO - Univ. Federal do Ceará, Brazil. Tel.: +55 8594846666; fax: +55 8594846666.
E-mail addresses: nicolas.nisse@inria.fr (N. Nisse), ronan.soares@gmail.com (R. Pardo Soares).

http://dx.doi.org/10.1016/j.dam.2015.01.038
0166-218X/

move anymore (i.e., all the neighbors of its current position are occupied by searchers). The number of searchers used by a strategy is the maximum number of occupied vertices throughout all steps of the strategy. Any graph G has a trivial winning strategy that consists in placing a searcher at every vertex of G. Such a strategy uses n searchers where n is the number of vertices of G. A natural question is then to design a winning strategy using the lowest number of searchers. The *search number* of a graph G is the smallest integer $k \geq 1$ such that there is a winning strategy using k searchers in G. As an example, consider the cycle with $n \geq 3$ vertices $\{v_1, \ldots, v_n\}$. A possible strategy is as follows: first three searchers are placed at v_1, v_2 and v_3. Then, for $i = 2$ to $n - 2$, remove the searcher at v_i and place it at v_{i+2}. This strategy is winning and uses 3 searchers and it is easy to see that there are no winning strategy using at most two searchers in the cycle. Hence, the search number of any cycle with at least three vertices is 3.

There are many variants of this game arising due to different properties, or behaviors, given to the fugitive or to the searchers. For instance, the fugitive may be visible, i.e., its location is known at any moment of the game and the strategy may take advantage of this knowledge, or it may be invisible, i.e., the fugitive may be at any contaminated vertex. If the fugitive is visible, the corresponding search number of a graph equals its *tree-width* plus one [24]. On the other hand, if the fugitive is invisible, the search number is equal to the *path-width* plus one [19]. The relationship between graph decompositions and search strategies mainly relies on the *monotonicity* property of these variants of graph searching. A strategy is said *monotone* if the area reachable by the fugitive is never increasing, i.e., once a vertex is clear it never becomes contaminated anymore. Equivalently, in the case of an invisible fugitive, a searcher cannot be removed from a vertex if it has a neighbor that is neither occupied nor has been occupied before, i.e., once a vertex has been occupied, the fugitive must not be able to reach it anymore. A variant of graph searching is said *monotone* if "recontamination does not help", i.e., for any graph with search number k, there is a winning monotone strategy using k searchers. That is, the number of searchers necessary to capture a fugitive considering only monotone strategies is not bigger than without this consideration.

The visible and invisible variants of graph searching were proven to be monotone in [21] (invisible) and [24] (visible). A simpler proof in the invisible case has been proposed by Bienstock and Seymour [4], and a unified proof for both visible and the invisible case can be found in [22]. Note that there are graph searching variants that are not monotone in undirected graphs, i.e., imposing the monotonicity of strategies may increase the number of searchers required to capture the fugitive. In *connected graph searching*, the area that cannot be reached by the fugitive is restricted to be connected along all stages of the strategy. The connected graph searching variant has been proved to be not monotone when the fugitive is invisible [30] neither when the fugitive is visible [12].

1.2. Graph searching in directed graphs

In [16], Johnson et al. defined the first variant of graph searching in directed graphs, related to directed tree-width. This variant, where the visible fugitive can move along directed cycles that are free of searchers, is however not monotone [1]. In [3], a variant is proposed where the visible fugitive can move along directed paths without searchers and [20] defined a variant where the invisible fugitive can move along directed paths without searchers only when a searcher is about to land at the vertex that the fugitive is currently occupying. Both these variants, respectively related to DAG-width and Kelly-width, are not monotone [20].

In some other cases, considering an invisible fugitive, more positive results have been provided. Barát defined the *directed path-width* related to a graph searching variant where the invisible fugitive is constrained to follow the direction of the arcs, i.e., it can move along directed paths free of searchers [2]. Barát adapted the framework of Bienstock and Seymour [4] to show that, in this variant, the monotonicity cannot increase the number of searchers by more than one [2]. Hunter then completed this proof to show the monotonicity of this variant [14]. Other variants that generalize the *edge-graph searching* (see [11]) to directed graphs have been defined. Yang and Cao proved the monotonicity of strong and weak graph searching variants where the searchers can moreover slide along arcs either in both directions (strong) or in the direction of arcs (weak) [28,29].

1.3. Process number

Surprisingly, a variant of graph searching in directed graphs has been defined in the context of routing reconfiguration in Wavelength-Division Multiplexing (WDM) networks [9]. WDM networks are currently becoming more flexible, offering new on-demand services for provisioning new light-paths, but also allowing better management of maintenance operations and equipment failures. A building block for flexibility and reliability is the possibility to *reconfigure* the routing, that is to compute new optical paths for some connection requests and then to switch the traffic from former to new optical paths. Such process may however affect the quality of service by inducing potential traffic disruptions. Thus, the routing reconfiguration process must be carefully optimized (see, e.g., [23,17,5,27]).

An instance of the routing reconfiguration problem is defined by a network, a set of connections, an initial routing and a final routing. The network is represented by a directed graph N. The set of connections is given by $C \subseteq V(N) \times V(N)$. An initial routing of these connections is given by a set I of directed paths in N joining each pair $(a, b) \in C$, with the restriction that two different paths do not share an arc of N, that is, these paths are arc-wise disjoint. Similarly, the final routing F is a set of arc-wise disjoint paths joining each pair $(a, b) \in C$. The *routing reconfiguration problem* consists in sequentially reroute the connections from the initial routing to the final one and minimizing the traffic disruption.