On the Monotonicity of Process Number

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

[a] *COATI, INRIA, I3S(CNRS/UNSA), France*

[b] *ParGO - Univ. Federal do Ceará, Brazil*

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 tree- and 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 such 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, arbitrary fast, it moves in the opposite direction of the arcs of a digraph, but only as long as it has 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.

Keywords: Graph Searching, Process Number, Monotonicity

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 path-decomposition or tree-decomposition in undirected graphs (e.g., see [9]). Because graph searching games are equivalent to path- and tree-decompositions, several attempts have been done to define such games in directed graphs [2,3,11].

Graph Searching and monotonicity. In graph searching games, a team of *searchers* aims at capturing a fugitive that stands at the vertices of a graph G (see [8] for a survey). The fugitive can move arbitrary fast along the paths of G as long as it does not meet any searcher. A node $v \in V(G)$ is *clear* if all paths from v to the node occupied by the fugitive contain a node occupied by a searcher. In particular, a node occupied by a searcher is clear. A vertex

that is not clear is said *contaminated*. Given a graph G with all nodes initially contaminated, a *strategy* for the searchers is a sequence of the following two possible actions: (R_1) place a searcher at a node of G, or (R_2) remove a searcher from a node. A strategy is *winning* if it allows to capture the fugitive whatever it does or, equivalently, if all nodes are eventually clear. That is, in a winning strategy, a searcher eventually occupies the same vertex as the fugitive and the fugitive cannot 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 and 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.

There are many variants of this problem arising due to different properties, or behaviors, given to the fugitive or to the searchers. If the fugitive is visible, the corresponding search number of a graph equals its *tree-width* plus one [16]. On the other hand, if the fugitive is invisible, the search number is equal to the *path-width* plus one [13]. 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 node is clear it never becomes contaminated at a later stage. Equivalently, a searcher cannot be removed from a node if it has a neighbor that is neither occupied nor is clean, i.e., once a node 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, imposing the monotonicity of the strategies does not increase the number of required searchers. The visible and invisible variants of graph searching were proven to be monotone in [4,16].

Graph searching in directed graphs. In [12], 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 where the visible fugitive can move along directed paths without searchers is proposed and, in [14], a variant where the invisible fugitive can move along directed paths without searchers only when a searcher is about to land at the node currently occupied by the fugitive is defined. Both these variants, respectively related to DAG-width and Kelly-width, are not monotone [14].

In the case of an invisible fugitive, more optimistic 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 [2]. Barát adapted the framework in [4] to show that, in this variant, the monotonicity cannot increases the number of searchers by more than one [2]. Hunter improved this result to show that this variant is monotone [10]. Other

variants that generalize the *edge-graph searching* (e.g., see [8]) to directed graphs have been defined and were proven to be monotone in [17].

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 (e.g., [5–7]). Roughly, the *processing game* is related to the smallest number of interruptions of traffic that are required when modifying the routes of requests in a WDM network.

In the *processing game*, the searchers aim at *processing* all nodes of a digraph. A node is said *safe* if all its out-neighbors are either occupied or already processed. Given $D = (V, A)$ where all nodes are initially unoccupied and not processed, a *monotone process strategy* is a sequence $(s_1, \ldots, s_n)$ of steps that results in processing all nodes of D. Each step s_i is one of the following moves: place a searcher at a node (M_1), process a safe *unoccupied* node (M_2), or process a safe *occupied* vertex and remove the searcher from it (M_3).

The minimum number of searchers used by a monotone process strategy of D is the *monotone process number*, denoted by $\text{monpr}(D)$. The problem to compute the monotone process number is NP-complete [7]. It is polynomial in the class of graphs D with $\text{monpr}(D) \leq 2$ [7] and in trees [6]. In undirected graphs (seen as symmetric digraphs [1]), the monotone processing game is equivalent to the monotone graph searching game where the invisible fugitive is captured if all the neighbors of its position are occupied, i.e., it is not required that a searcher occupies the same node as the fugitive.

In this work, we consider the more general variant of processing game where we allow a processed node to become unprocessed. A *process strategy* for a digraph D is a sequence $(s_1, \ldots, s_n)$ of steps that results in processing all nodes of D, where each step s_i consists of a move M_1 or M_2 or process an *occupied* vertex v and remove the searcher from it (M_3'). After a removal at v, if v was not safe then recontamination occurs: successively, all processed vertices (including v) that have an unoccupied and unprocessed out-neighbor become unprocessed. The fewest number of searchers used by a process strategy of D is the *process number*, denoted by $\text{pr}(D)$. Equivalently, $\text{pr}(D)$ is the smallest number of searchers required to capture an invisible arbitrary fast fugitive that moves backwards the arcs of D and is captured as soon as it cannot access a strongly connected component of at least two nodes and free of searchers. It is known that $\text{pr}(D) = \text{monpr}(D)$ for any symmetric digraph D [6].

Results. In this work [2], we prove that the result holds for any digraph. Moreover, our monotonicity result allows us to interpret the processing game as a new digraph decomposition. Finally, we prove that $\text{pr}(D) = \text{pr}(\overleftarrow{D})$ for any digraph $D = (V, A)$, where $\overleftarrow{D} = (V, \overleftarrow{A})$ and $\overleftarrow{A} = \{(a, b) \ : \ (b, a) \in A\}$.

[1] A digraph $D = (V, A)$ is *symmetric* if, for any $(a, b) \in A$, then $(b, a) \in A$.

[2] Due to lack of space, proofs have been omitted and can be found in [15]

2 Recontamination does not help to process a digraph

In this section, we prove that the process number is monotone, i.e. $\text{monpr}(D) = \text{pr}(D)$ for any directed graph D. We follow the framework introduced in [16]. The main idea is to consider a strategy as a sequence of edge-subsets (*crusade*) where each subset has some *weight*. Then, using the submodularity of the appropriate weight function, it is easy to show the *monotonicity* of crusades. The main difficulty is to define an auxiliary game that will be equivalent to the crusade and therefore monotone. Then, the second technicality is to show the relationship between the auxiliary game and the searching game. The last part is generally done using a graph transformation.

Let $D = (V, A)$ be a digraph. For any $X \subseteq A$, let $\delta(X)$ be the set of vertices that are the head of an arc in X and the tail of an arc in $A \setminus X$. The function δ is submodular, that is, for any $X, Y \subseteq A(D)$, $|\delta(X \cap Y)| + |\delta(X \cup Y)| \leq |\delta(X)| + |\delta(Y)|$ [15].

A *crusade* in $D = (V, A)$ is a sequence $(X_0, X_1, \ldots, X_n)$ of subsets of A such that $X_0 = \emptyset$, $X_n = E$, and $|X_i \setminus X_{i-1}| \leq 1$, for $1 \leq i \leq n$. The crusade has *border* k if $|\delta(X_i)| \leq k$ for $0 \leq i \leq n$. A crusade is *progressive* if $X_0 \subset X_1 \subset \ldots \subset X_n$. Using the submodularity of δ, we can prove that

Lemma 1 *[15] If there is a crusade of $D = (V, A)$ with border k, then there is a progressive crusade with border k.*

Let $D = (V, A)$ be a digraph whose no arcs are initially processed. A *mixed process strategy* of D is a sequence $(s_1, \ldots, s_n)$ that results in processing all arcs in A, where each step s_i is one of the following actions: place a searcher at an unoccupied node (**Place**), remove a searcher from a node (**Remove**), process an arc $(u, v) \in A$ if $v \in V$ is *occupied* (**Head**), slide the searcher at u along $(u, v) \in A$ if u is occupied, v is not occupied and all arcs $e \neq (u, v)$ with tail u are already processed, this process the arc (u, v) (**Slide**), and process an arc $(u, v) \in A$ if all arcs with tail v are already processed (**Extend**).

When a searcher is removed from a node $v \in V$, if there were unprocessed arcs with tail v and v is now unoccupied, then *recontamination* occurs. That is, successively, any processed arc $(u, w) \in A$ such that w is unoccupied and there is an unprocessed arc (w, z) becomes unprocessed.

The *mixed process number*, denoted by $\text{mpr}(D)$, is the fewest number of searchers used by such a strategy. A mixed process strategy is *monotone* if no recontamination occurs, i.e., once an arc has been processed, it must remain processed until the end of the strategy.

The main part of the proof of Theorem 4 consists of the next two lemmas. Let D be a digraph and $\tilde{D}$ be obtained from D by replacing each arc by two "parallel" arcs.

Lemma 2 *[15] If $\text{mpr}(D) \leq k$, then D admits a crusade with border k.*

If there is a progressive crusade of D with border k, then there is a monotone *mixed process strategy using at most k searchers.*

By lemma 2, the mixed processing game is monotone, which allows us to prove

Lemma 3 *[15] For any digraph $D = (V, A)$,* $\text{monpr}(D) \leq \text{mpr}(\tilde{D}) \leq \text{pr}(D)$.

Since, for any digraph D, $\text{pr}(D) \leq \text{monpr}(D)$, Lemma 3 implies:

Theorem 4 *[15] Recontamination does not help to process a digraph, i.e., for any digraph D,* $\text{pr}(D) = \text{monpr}(D)$.

3 Process Decomposition

We now define a digraph decomposition that is equivalent to (monotone) process strategies. This allows us to prove that the process number is invariant when reversing all arcs of a digraph.

A *process decomposition* of a digraph $D = (V, A)$ is a sequence $P = ((W_1, X_1), \cdots, (W_t, X_t))$ of pairs of subsets of V where (1) $(X_1, \cdots, X_t)$ is a partition of $V \setminus \bigcup_{i=1}^{t} W_i$, (2) $\forall i \leq j \leq k$, $W_i \cap W_k \subseteq W_j$, (3) for any $1 \leq i \leq t$, X_i induces a DAG, and (4) $\forall (u, v) \in A$, $\exists j \leq i$ such that $v \in W_j \cup X_j$ and $u \in W_i \cup X_i$.

The width of a process decomposition is given by $\max_{1 \leq i \leq n} |W_i|$, and the *process-width*, denoted by $\text{prw}(D)$, of a digraph D is given by the minimum width over all process decompositions of D. Thanks to Theorem 4, we can prove that

Theorem 5 *[15] For any digraph D,* $\text{pr}(D) = \text{prw}(D)$.

As a corollary of Theorems 4 and 5, we get that

Corollary 1 *For any digraph D,* $\text{pr}(D) = \text{pr}(\overleftarrow{D})$.

4 Conclusion

Both tree-decompositions and path-decompositions have the notion of a dual structure, brambles and blockages respectively. For instance, the tree-width of a undirected graph G equals $k - 1$ iff G has no bramble greater[3] than k [16]. The monotonicity of a game plays an important role in the relationship between the width of a decomposition and its dual. Hence, it will be interesting to use our monotonicity result to define a dual for the process number.

On the other hand, the visible variant of the processing game appears to be an interesting candidate for providing a tree-decomposition for digraphs since the ability of a visible fugitive in the processing game is "between" the ones in the games in [12] and [3] (both having distinct advantages).

[3] Measured by the size of its hitting set.

References

[1] Adler, I., *Directed tree-width examples*, JCTB **97** (2007), pp. 718–725.

[2] Barát, J., *Directed path-width and monotonicity in digraph searching*, Graphs and Combinatorics **22** (2006), pp. 161–172.

[3] Berwanger, D., A. Dawar, P. Hunter, S. Kreutzer and J. Obdrzálek, *The dag-width of directed graphs*, JCTB **102** (2012), pp. 900–923.

[4] Bienstock, D. and P. Seymour, *Monotonicity in graph searching*, J. Algorithms **12** (1991), pp. 239–245.

[5] Cohen, N., D. Coudert, D. Mazauric, N. Nepomuceno and N. Nisse, *Tradeoffs in process strategy games with application in the wdm reconfiguration problem*, Theor. Comput. Sci. **412** (2011), pp. 4675–4687.

[6] Coudert, D., F. Huc and D. Mazauric, *A distributed algorithm for computing the node search number in trees*, Algorithmica **63** (2012), pp. 158–190.

[7] Coudert, D. and J.-S. Sereni, *Characterization of graphs and digraphs with small process number*, Discrete Applied Mathematics **159** (2011), pp. 1094–1109.

[8] Fomin, F. and D. Thilikos, *An annotated bibliography on guaranteed graph searching*, Theo. Comp. Sci. **399** (2008), pp. 236–245.

[9] Ganian, R., P. Hlinený, J. Kneis, D. Meister, J. Obdrzálek, P. Rossmanith and S. Sikdar, *Are there any good digraph width measures?*, in: *5th Int. Symp. on Parameterized and Exact Computation*, LNCS **6478** (2010), pp. 135–146.

[10] Hunter, P., *Losing the +1: Directed path-width games are monotone* (2006).

[11] Hunter, P. and S. Kreutzer, *Digraph measures: Kelly decompositions, games, and orderings*, Theor. Comput. Sci. **399** (2008), pp. 206–219.

[12] Johnson, T., N. Robertson, P. D. Seymour and R. Thomas, *Directed tree-width*, J. Comb. Theory, Ser. B **82** (2001), pp. 138–154.

[13] Kirousis, M. and C. Papadimitriou, *Searching and pebbling*, Theoretical Computer Science **47** (1986), pp. 205–218.

[14] Kreutzer, S. and S. Ordyniak, *Digraph decompositions and monotonicity in digraph searching*, CoRR **abs/0802.2228** (2008).

[15] Nisse, N. and R. P. Soares, *On the Monotonicity of Process Number*, Technical Report RR-, INRIA (2012), http://hal.inria.fr/hal-00745587.

[16] Seymour, P. D. and R. Thomas, *Graph searching and a min-max theorem for tree-width*, J. Comb. Theory Ser. B **58** (1993), pp. 22–33.

[17] Yang, B. and Y. Cao, *On the monotonicity of weak searching*, in: *COCOON*, 2008, pp. 52–61.