Discrete Applied Mathematics 199 (2016) 89–100

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

A tie-break model for graph search

Derek G. Corneil [a], Jérémie Dusart [b], Michel Habib [b,*], Antoine Mamcarz [b], Fabien de Montgolfier [b]

[a] *Department of Computer Science, University of Toronto, Canada*

[b] *LIAFA, UMR 7089 CNRS & Université Paris Diderot, F-75205 Paris Cedex 13, France*

ARTICLE INFO

Article history:
Received 27 March 2014
Received in revised form 3 June 2015
Accepted 15 June 2015
Available online 7 July 2015

Keywords:
Graph search model
Tie-break mechanisms
BFS
DFS
LBFS
LDFS
Multi-sweep algorithms

ABSTRACT

In this paper, we consider the problem of the recognition of various kinds of orderings produced by graph searches. To this aim, we introduce a new framework, the Tie-Breaking Label Search (TBLS), in order to handle a broad variety of searches. This new model is based on partial orders defined on the label set and it unifies the General Label Search (GLS) formalism of Krueger et al. (2011), and the "pattern-conditions" formalism of Corneil and Krueger (2008). It allows us to derive some general properties including new pattern-conditions (yielding memory-efficient certificates) for many usual searches, including BFS, DFS, LBFS and LDFS. Furthermore, the new model allows easy expression of multi-sweep uses of searches that depend on previous (search) orderings of the graph's vertex set.

1. Introduction

A graph search is a mechanism for systematically visiting the vertices of a graph. It has been a fundamental technique in the design of graph algorithms since the early days of computer science. Many of the early search methods were based on Breadth First Search (BFS) or Depth First Search (DFS) and resulted in efficient algorithms for practical problems such as the distance between two vertices, diameter, connectivity, network flows and the recognition of planar graphs, see [4].

Many variants of these searches have been introduced since, providing elegant and simple solutions to many problems. For example, Lexicographic Breadth First Search (LBFS) [19], and its generalization Maximal Neighbourhood Search (MNS) [21] were shown to yield simple linear time algorithms for chordal graph recognition. More recently, Lexicographic Depth First Search (LDFS), was introduced in [9] based on its symmetrical "pattern-condition" with LBFS. A few years after its discovery it was shown that LDFS when applied to cocomparability graphs yields simple and efficient algorithms for solving various Hamiltonian Path related problems [7,18,8].

Some recent applications of graph searches involve a controlled tie-break mechanism in a series of consecutive graph searches, see [5,10,7,12]. Examples include the strongly connected components computation using double-DFS [20] and the series of an arbitrary LBFS followed by two LBFS$^+$s used to recognize unit interval graphs [5]. Note that a "$^+$ search" breaks ties by choosing (amongst the tied vertices) the vertex that is rightmost with respect to a given ordering of the vertices. This motivates a general study of these graph searches equipped with a tie-break mechanism that incorporates such multi-sweep usage of graph searches. This is the goal of the present paper: to define the simplest framework powerful enough to capture many graph searches either used individually or in a multi-sweep fashion and simple enough to allow general theorems on

* Corresponding author. Tel.: +33 0 1 57 27 92 47; fax: +33 0 1 57 27 94 09.
E-mail address: habib@liafa.univ-paris-diderot.fr (M. Habib).

http://dx.doi.org/10.1016/j.dam.2015.06.011

graph searches. Building on the General Label Search (GLS) framework from [15] we not only simplify their model but also unify their model with the "pattern-conditions" formalism of [9].

This paper is organized as follows. After basic notations and definitions in Section 2 Section 3 introduces the Tie-Breaking Label Search (TBLS) formalism to address graph searches. We then illustrate the TBLS by expressing some classical graph searches in this formalism. We will also show the relationship between our formalism and the "pattern-conditions" of search orderings introduced in [9] thereby yielding some new pattern-conditions for various classical searches. In Section 5 we turn to theoretical issues involving TBLS including showing that the TBLS and GLS models capture the same set of graph searches. We then propose algorithms for recognizing whether a given ordering of the vertices could have been produced by a specific graph search. Finally, in Section 7 we present a general TBLS implementation framework.

2. Preliminaries and notation

In this paper, $G = (V, E)$ always (and sometimes implicitly) denotes a graph with n vertices and m edges. All graphs considered here are supposed to be finite. We identify the vertex-set with $\{1, \dots, n\}$, allowing us to see a total ordering on V as a permutation on $\{1, \dots, n\}$. Note that throughout the paper when we refer to an ordering on V, the ordering is considered to be total.

We define a *graph search* to be an algorithm that visits all the vertices of a graph according to some rules, and a *search ordering* to be the ordering σ of the vertices yielded by such an algorithm. The link between these two notions is an overriding theme of this paper. Vertex $\sigma(i)$ is the ith vertex of σ and $\sigma^{-1}(x) \in \{1, \dots, n\}$ is the position of vertex x in σ. A vertex u is the *leftmost* (respectively *rightmost*) vertex with property X in σ if there is no vertex v such that $X(v)$ and $v <_\sigma u$ (respectively $u <_\sigma v$). Our graphs are assumed to be undirected, but most searches (especially those captured by TBLS) may be performed on directed graphs without any modifications to the algorithm (if xy is an arc then we say that y is a neighbour of x while x is *not* a neighbour of y).

The symmetric difference of two sets A and B, namely $(A - B) \cup (B - A)$ is denoted by $A \triangle B$. Furthermore, $\mathbb{N}^+$ represents the set of integers strictly greater than 0 and $\mathbb{N}_p^+$ represents the set of integers strictly greater than 0 and less than p. $P(\mathbb{N}^+)$ denotes the power-set of $\mathbb{N}^+$ and $P_f(\mathbb{N}^+)$ denotes the set of all finite subsets of $\mathbb{N}^+$. By $\mathfrak{S}_n$ we denote the set of all permutations of $\{1, \dots, n\}$. For finite $A \in P(\mathbb{N}^+)$, let $umin(A)$ be: if $A = \emptyset$ then $umin(A) = \infty$ else $umin(A) = min\{i \mid i \in A\}$; and let $umax(A)$ be: if $A = \emptyset$ then $umax(A) = 0$ else $umax(A) = max\{i \mid i \in A\}$. We always use the notation $<$ for the usual strict (i.e., irreflexive) order between integers, and $\prec$ for a partial strict order between elements from $P_f(\mathbb{N}^+)$ (or from another set when specified). Definitions of most of the searches we will consider appear in [9] or [3].

3. TBLS, a tie-breaking label search

A *graph search* is an iterative process that chooses at each step a vertex of the graph and numbers it (from 1 to n). Each vertex is chosen (also said *visited*) exactly once (even if the graph is disconnected). Let us now define a general *Tie-Breaking Label Search* (TBLS) for search S; see Algorithm 1. It uses *labels* to decide the next vertex to be visited; *label*(v) is a subset of $\{1, \dots, n\}$. A TBLS is defined on:

1. A graph $G = (V, E)$ on which the search S is performed.
2. A strict partial order $\prec_S$ over the label-set $P_f(\mathbb{N}^+)$; ($\prec_S$ encodes S, the specific search being described).
3. An ordering τ of the vertices of V called the *tie-break permutation*.

The output of TBLS$(G, \prec_S, \tau)$ is a permutation σ of V, called a *TBLS−ordering* or also the *search ordering* or *visiting ordering* or *S-ordering*. Let us say a vertex v is *unnumbered* until $\sigma(i) \leftarrow v$ is performed, and then i is its *visiting date*. As shown in Algorithm 1, *label*(v) is always the set of visiting dates of the neighbours of v visited before v. More specifically *label*$_i(v)$ for a vertex v denotes the label of v at the beginning of step i. This formalism identifies a search with the orderings it may produce, as in [9], while extending the formalism of General Label Search (GLS) of [15] by the introduction of a *tie-break* ordering τ, making the result of a search algorithm purely deterministic (no arbitrary decision is taken). Note that nondeterministic searching (where ties can be broken arbitrarily) can be achieved by setting τ to be an arbitrary permutation of $\{1, \dots, n\}$.

Algorithm 1: TBLS$(G, \prec_S, \tau)$

```
foreach v ∈ V do label(v) ← ∅;
for i ← 1 to n do
    Eligible ← {x ∈ V | x unnumbered and ∄ unnumbered y ∈ V such that label(x) ≺_S label(y)};
    Let v be the leftmost vertex of Eligible according to the ordering τ;
    σ(i) ← v;
    foreach unnumbered vertex w adjacent to v do
        label(w) ← label(w) ∪ {i};
```