Theoretical Computer Science 469 (2013) 92–104

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

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

Confining sets and avoiding bottleneck cases: A simple maximum independent set algorithm in degree-3 graphs

Mingyu Xiao [a,*], Hiroshi Nagamochi [b]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

[b] Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo, Kyoto 606-8501, Japan

A R T I C L E I N F O

Article history:
Received 17 May 2012
Received in revised form 20 September 2012
Accepted 25 September 2012
Communicated by V. Paschos

Keywords:
Exact algorithm
Independent set
Measure and conquer

A B S T R A C T

We present an $O^*(1.0836^n)$-time algorithm for finding a maximum independent set in an n-vertex graph with degree bounded by 3, which improves all previous running time bounds for this problem. Our approach has the following two features. Without increasing the number of reduction/branching rules to get an improved time bound, we first successfully extract the essence from the previously known reduction rules such as domination, which can be used to get simple algorithms. More formally, we introduce a procedure for computing "confining sets", which unifies several known reducible subgraphs and covers new reducible subgraphs. Second we identify those instances that generate the worst recurrence among all recurrences of our branching rules as "bottleneck instances" and prove that bottleneck instances cannot appear consecutively after each branching operation.

1. Introduction

The *maximum independent set* problem (MIS), to find a maximum set of vertices in a graph such that there is no edge between any two vertices in the set, is one of the basic NP-hard optimization problems and has been extensively studied in the literature, in particular in the line of research on worst-case analysis of algorithms for NP-hard optimization problems. In 1977, Tarjan and Trojanowski [15] designed the first nontrivial algorithm for this problem, which runs in $O^*(2^{n/3})$ time and polynomial space. Later, the running time was improved to $O^*(1.2346^n)$ by Jian [9]. Robson [13] obtained an $O^*(1.2278^n)$-time polynomial-space algorithm and an $O^*(1.2109^n)$-time exponential-space algorithm. In a technical report [14], Robson also claimed better running times. Fomin et al. [6] got an $O^*(1.2210^n)$-time polynomial-space algorithm by using the "Measure and Conquer" method. Recently Kneis et al. [10] and Bourgeois et al. [2] improved the running time bound to $O^*(1.2132^n)$ and $O^*(1.2127^n)$ respectively. There is also a considerable amount of contributions to the maximum independent set problem in sparse graphs, especially in degree-3 graphs [1,4,18,3]. Chen et al. [4] showed that MIS3 (the maximum dependent set problem in degree-3 graphs) can be solved in $O^*(1.1254^n)$ time. Xiao et al. [18] used the number of degree-3 vertices as a measure to analyze algorithms and got an $O^*(1.1034^n)$-time algorithm for MIS3. Razgon [11] also designed another $O^*(1.1034^n)$-time algorithm for this problem. Fürer [8] designed an algorithm for MIS3 by measuring the running time in terms of $m - n$, where m is the number of edges. Based upon a refined branching with respect to Fürer's algorithm, Bourgeois et al. [3] got an $O^*(1.0977^n)$-time algorithm for MIS3. Razgon [12] and Xiao [17] further improved the running time bound to $O^*(1.0892^n)$ and $O^*(1.0885^n)$ respectively. Currently, the best result on this problem is Bourgeois et al.'s $O^*(1.0854^n)$-time algorithm designed by carefully checking the worst cases [2]. See Table 1 for a summary on the currently published results on low-degree graphs as well as general graphs.

* Corresponding author. Tel.: +86 15828165155; fax: +86 28 61831656.
E-mail addresses: myxiao@gmail.com (M. Xiao), nag@amp.i.kyoto-u.ac.jp (H. Nagamochi).

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2012.09.022

Table 1
Exact algorithms for the maximum independent set problem.

Authors	Running times	References	Notes
Tarjan & Trojanowski	$O^*(1.2600^n)$ for MIS	1977 [15]	n: number of vertices
Jian	$O^*(1.2346^n)$ for MIS	1986 [9]	
Robson	$O^*(1.2109^n)$ for MIS	1986 [13]	Exponential space
Beigel	$O^*(1.0823^m)$ for MIS	1999 [1]	m: number of edges
	$O^*(1.1259^n)$ for MIS3		
Chen et al.	$O^*(1.1254^n)$ for MIS3	2003 [4]	
Xiao et al.	$O^*(1.1034^n)$ for MIS3	2005 [18]	Published in Chinese
Fomin et al.	$O^*(1.2210^n)$ for MIS	2006 [6]	
Fomin & Høie	$O^*(1.1225^n)$ for MIS3	2006 [7]	
Fürer	$O^*(1.1120^n)$ for MIS3	2006 [8]	
Razgon	$O^*(1.1034^n)$ for MIS3	2006 [11]	
Bourgeois et al.	$O^*(1.0977^n)$ for MIS3	2008 [3]	
Razgon	$O^*(1.0892^n)$ for MIS3	2009 [12]	
Kneis et al.	$O^*(1.2132^n)$ for MIS	2009 [10]	
Xiao	$O^*(1.0885^n)$ for MIS3	2010 [17]	
Bourgeois et al.	$O^*(1.2127^n)$ for MIS	2012 [2]	
	$O^*(1.0854^n)$ for MIS3		
Xiao & Nagamochi	$O^*(1.0836^n)$ for MIS3	This paper	

One reason why MIS3 has been extensively studied is that MIS in low-degree graphs are usually the bottlenecks to get improvement for the problem in general graphs. Most previous result for MIS in general graphs are obtained by carefully analyzing the problems in low-degree graphs. Bourgeois et al. [2] presented a bottom-up method for MIS, which shows that the improvements on MIS for low-degree graphs can be used to derive improved algorithms for MIS in general graphs. Then they got the current best result for MIS in general graphs by designing an improved algorithm for MIS3 and so on.

Most fast algorithms for the maximum independent set problem are obtained via careful examinations of the structures in the graph. In those algorithms, a long list of reduction and branching rules are used, which is derived from a somewhat complicated case analysis. In this paper, we introduce some uniform reduction and branching rules for the maximum independent set and vertex cover problems, which can be used to design simple algorithms. To catch more properties of the graphs, we use the sum of $\max\{0, \delta(v) - 2\}$ over all vertices v as the measure of a graph to analyze the algorithm, where $\delta(v)$ is the degree of a vertex v. When the graph is a degree-3 graph, the measure is the number of degree-3 vertices in the graph. To get improvement on MIS3, we use an idea of avoiding the worse cases. Finally, our algorithm runs in $O^*(1.0836^n)$ time, which improves previous algorithms for MIS3 and can derive improved algorithm for MIS in general graphs by using the bottom-up method introduced in [2].

Based on our new result on MIS3, we recently designed an $O^*(1.1446^n)$-time algorithm to MIS4 (the maximum independent set problem in degree-4 graphs) [20], which improves the results of $O^*(1.1571^n)$ [2] and $O^*(1.1526^n)$ [19].

2. Preliminaries

Let V denote the set of all vertices in an instance and let $n = |V|$. We may simply use v to denote the set $\{v\}$ of a single vertex v. For a set X of vertices, let $N(X)$ denote the *neighbors* of X, i.e., the vertices $y \in V - X$ adjacent to a vertex $x \in X$, and denote $N(X) \cup X$ by $N[X]$. For a vertex $v \in V$, let $N_2(v)$ denote the set of vertices at distance exactly 2 from v, and $\delta(v)$ $(= |N(v)|)$ denote the degree of v. Define $\rho(v) = \max\{0, \delta(v) - 2\}$. For a graph $H = (V_H, E_H)$, we denote $\rho(H) = \sum_{v \in V_H} \rho(v)$. We also denote $\rho(X) = \sum_{v \in X} \rho(v)$ for a set X of vertices in G.

We say that an edge e is incident on a vertex set X, if at least one endpoint of e is in X. Let $G - X$ denote the graph obtained from G by removing the vertices in X and the edges incident to X. *Contracting* X is to identify all vertices in X as a single vertex s, where any resulting self-loops and multiple edges will be removed. Hence s is adjacent to a vertex $v \in V - X$ in the resulting graph if and only if v is adjacent to a vertex in X. Let G/X denote the graph obtained from G by contracting a subset X of vertices.

A subgraph of G is called a *k-path* (or *path*) if it consists of a sequence of $k + 1$ distinct vertices $v_1, v_2, \ldots, v_{k+1}$ such that v_i and v_{i+1} are adjacent for each $i = 1, 2, \ldots, k$. A $(k - 1)$-path $v_1, v_2, \ldots, v_k$ $(k \geq 3)$ together with an edge $v_k v_1$ called a *k-cycle* (or *cycle*). A path $v_1, v_2, \ldots, v_{k+1}$ in a graph G is called a *pure path* if each non-endpoint v_i in the path has no neighbor other than v_{i-1} and v_{i+1} in G. A pure path is called an *o-path* (resp., *e-path*) if the two endpoints are of degree ≥ 3 and the number of non-endpoints (of degree 2) in it is odd (resp., even), where we allow the two endpoints being a same vertex. A component of a graph means a maximal connected subgraph of the graph.

Our algorithms are based on the branch-and-reduce paradigm. We will first apply some reduction rules to reduce the size of instances of the problem. Then we apply some branching rules to branch on the instance by including some vertices in the independent set or excluding some vertices from the independent set. In each branch, we will get a maximum independent set problem in a graph instance with a smaller measure. Next, we introduce the reduction rules and branching rules that will be used in our algorithm.