Detecting Directed 4-Cycles Still Faster

Friedrich Eisenbrand
Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany
eisen@mpi-sb.mpg.de

Fabrizio Grandoni
Dipartimento di Informatica, Sistemi e Produzione
Universita di Roma "Tor Vergata"
Via del Politecnico 1
00133 Roma
Italy
grandoni@disp.uniroma2.it

5th November 2002

Abstract

We present a method to detect simple cycles of length 4 of a directed graph in $O(n^{1/\omega}e^{2-2/\omega})$ steps, where n denotes the number of nodes, e denotes the number of edges and ω is the exponent of matrix multiplication. This improves upon the currently fastest methods for $\alpha \in (2/(4-\omega), (\omega+1)/2)$, where $e = n^{\alpha}$.

1 Introduction

A cycle of a graph is a very simple structure and the detection of given length cycles is one of the most natural problems in algorithmic graph theory. Though the problem is easily stated, fast algorithms to solve it are far from being obvious and progress in terms of faster algorithms has been constantly reported in the last decades.

In this paper we address the problem of finding a C_4, a directed cycle of length 4. It is easy to describe an $O(n^{\omega})$ algorithm for this task, which is based on *fast matrix multiplication*. Here ω is the exponent of the multiplication of two square matrices. Alon, Yuster and Zwick [2] developed an algorithm which detects a C_k, a *simple cycle* of a fixed length k, in $O(e^{2-1/\lceil k/2 \rceil})$ steps. This means that one can detect a C_4 in $O(e^{1.5})$ time. The crucial idea of their algorithm is to partition the vertex set into *high degree* vertices and *low degree* vertices, and first search for cycles which contain at least one high degree vertex. If such cycles do not exist, the high degree vertices are discarded.

Based on this idea Alon et al. [2] presented also a method to detect triangles which runs in $O(e^{2\omega/(\omega+1)})$ steps which uses fast matrix multiplication. The authors pose the question, whether fast matrix multiplication can also be used to speed up their techniques to detect a C_k for $k \geq 4$.

We give a positive answer for the case of detecting a C_4 by describing an algorithm which runs in $O(n^{1/\omega}e^{2-2/\omega})$ steps. This bound is asymptotically smaller than $O(n^{\omega})$ and $O(e^{1.5})$ for graphs with $e = n^{\alpha}$, where $\alpha \in (2/(4-\omega), (\omega+1)/2)$.

1.1 Related work

Itai and Rodeh [5] developed an algorithm which detects a triangle in $O(e^{1.5})$ steps. Alon, Yuster and Zwick [1] described an algorithm which detects a C_k for fixed k in $O(n^{\omega})$ expected time and $O(n^{\omega}\log n)$ worst case time. All of the above methods work in the directed as well as in the undirected case. Yuster and Zwick [7] showed that an even cycle of a fixed length in an undirected graph can be found in $O(n^2)$

steps. Kloks, Kratsch and Müller [6] used the idea to partition the nodes into high and low degree, to efficiently count and detect small induced subgraphs.

2 The Algorithm

As in [2], the set of nodes is decomposed in two subsets: a set L of nodes with degree smaller than Δ (low degree nodes), and the set H of the remaining nodes (high degree nodes). The value of Δ will be determined in the sequel.

We first describe how to detect a C_4 which contains two opposite low degree nodes, i.e., it is of the form v_1, v_2, v_3, v_4, where v_1 and v_3 are members of L. Following [2], we first compute all paths of length 2 which have an intermediate node in L. These paths can be constructed in $O(e\Delta)$ time and space. Next one sorts these paths with respect to their start and end point in lexicographic and reverse lexicographic order with radix sort in linear time. This allows one to group all the two-paths which start in a particular vertex and end in a particular vertex, sorted with respect to their second extremal vertex. With this at hand, it is then straightforward to detect whether there exist two such paths, which form a simple directed cycle of length 4. This procedure runs in $O(e\Delta)$ steps.

Since $\sum_{v \in V} \deg(v) = 2e$ one has that the number of high degree nodes $|H|$ is bounded by $|H| \leq 2e/\Delta$. Thus a C_4 which contains only high degree nodes can be detected in $O((e/\Delta)^\omega)$ steps via fast matrix multiplication.

The other C_4 are of the form

1. C_{LHHH}: three high degree nodes followed by one low degree node;
2. C_{LLHH}: a pair of consecutive low degree nodes followed by a pair of high degree nodes.

Let M_{ABC} be an integer matrix such that, given a node $v \in A$ and a node $w \in C$, $M_{ABC}[v,w]$ is the number of 2-length directed simple paths from v to w which pass trough a node in B, where $A, B, C \in \{L, H\}$. Since there are at most $2e/\Delta$ high degree nodes, one can compute M_{HHH} in time $O(e/\Delta)^\omega)$.

One can compute the matrix M_{HHH} in time $O((e/\Delta)^\omega)$. The matrix M_{HHL} can be computed in time $O((e/\Delta)^{\omega(1,1,\log n/\log(e/\Delta))})$ which is the time to multiply a $2e/\Delta \times 2e/\Delta$-matrix, with a $2e/\Delta \times n$-matrix. Here the number $\omega(1,1,r)$ denotes the exponent of the multiplication of a $n \times n$-matrix, with a $n \times n^r$-matrix.

With these matrices M_{HHH} and M_{HHL} at hand, we can now check whether there exists a C_4 of type 1 or 2. To do so, one generates again all two-paths with intermediate node in L. For each such two path, one then checks whether its start-node u and its end-node v have a positive entry $M_{HHH}[u,v]$ or $M_{HHL}[u,v]$.

2.1 Complexity

The above described procedure runs in time $O(e\Delta + (e/\Delta)^{\omega(1,1,\log n/\log(e/\Delta))})$. This complexity is minimized when $e\Delta = (e/\Delta)^{\omega(1,1,\log n/\log(e/\Delta))}$. The exponent $\omega(1,1,r)$ is bounded by $\omega(1,1,r) \leq r - 1 + \omega$. Via this upper bound one obtains $\Delta = n^{1/\omega} e^{1-2/\omega}$ and thus a running time of $O(n^{1/\omega} e^{2-2/\omega})$.

Theorem 1 *There is an algorithm which detects a C_4 in a directed graph with n nodes and e edges in time $O(n^{1/\omega} e^{2-2/\omega})$.*

This running time is an asymptotic improvement of the $O(n^\omega)$ and $O(e^{1.5})$ bounds if $e = n^\alpha$ for $\alpha \in (2/(4-\omega), (\omega+1)/2)$. The currently best algorithm for matrix multiplication shows that ω is bounded by $\omega < 2.375477$ [3]. This means that our algorithm is the currently fastest algorithm for $\alpha \in [1.2312, 1.6877]$, where $e = n^\alpha$.

2.2 Remarks

Limitations to detecting larger cycles

To speed up the detection of a C_k for $k \geq 5$ one cannot use the same approach. Already to detect a C_5 with the same idea, one would have to spend at least $O((e/\Delta)^\omega)$ steps to detect a C_5 among the high degree nodes and in addition one would have to generate all 3 paths in the low degree nodes, of which there might be $e\Delta^2$ many. Thus the running time would be $\Omega(e^{1+2/3})$, which is not superior to the $O(e^{1+2/3})$ algorithm of Alon et al. [2].

Using fast rectangular matrix multiplication

Above we used the simple upper bound $\omega(1,1,r) \leq r-1+\omega$ to readily compute the optimal value of Δ in terms of n and e. In this way we could also state a closed formula for the worst case complexity of our algorithm.

Huang and Pan [4] described a fast method for rectangular matrix multiplication. The bound on $\omega(1,1,r)$ which results from their algorithm is superior to the $r-1+\omega$ bound that we applied. However, it is not expressed via a closed formula. We numerically found that using Huang and Pan's fast rectangular matrix multiplication algorithm, our method is asymptotically faster than $O(n^\omega)$ and $O(e^{1.5})$ for any α in the interval $[1.2117, 1.6877]$.

References

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the Association for Computing Machinery*, 42(4):844–856, 1995.

[2] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, Mar. 1997.

[3] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[4] X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *Journal of Complexity*, 14(2):257–299, 1998.

[5] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978.

[6] T. Kloks, D. Kratsch, and H. Müller. Finding and counting small induced subgraphs efficiently. *Information Processing Letters*, 74(3-4):115–121, 2000.

[7] R. Yuster and U. Zwick. Finding even cycles even faster. *SIAM Journal on Discrete Mathematics*, 10(2):209–222, 1997.