Approximate Counting of Given Length Cycles

Carsten Dorgerloh[§] Jürgen Wirtgen[¶]

Abstract

We present a method for approximate counting simple cycles of a given length in graphs. The main result of this paper is an (ϵ, δ)-approximation to the number of simple cycles of length k which runs in $\mathcal{O}(M(n) \log n/\epsilon^2 \log(1/\delta))$ time, where $\mathcal{O}(M(n))$ is the complexity of matrix multiplication.

[§]Institut für Informatik, Universität Bonn, Römerstr. 164, D-53117 Bonn, Germany, email: carsten@cs.uni-bonn.de

[¶]Institut für Informatik, Universität Bonn, Römerstr. 164, D-53117 Bonn, Germany, email: wirtgen@cs.uni-bonn.de

1 Introduction

The literature on detection and finding simple cycles of a given length has a long and varied history (see e.g. [Mo 85], [RL 85], [Ri 86], [YZ 94], [Do 96], [DW 96], [DW 97a], [DW 97b]). However, there are only a few fast algorithms for counting the number of simple cycles of a given fixed length. [AYZ 94] showed that for any $k \leq 7$ the number of simple cycles of length k in a graph can be counted in $\mathcal{O}(M(n))$ time, where $M(n) < n^{2.376}$ is the complexity of matrix multiplication [CW 87]. We present an (ϵ, δ)-approximation to this number that works for any fixed k and which runs in $\mathcal{O}(M(n) \log n/\epsilon^2 \log(1/\delta))$ time. Our algorithm for that problem, while using a result from [AYZ 95], takes advantage of the covering method introduced by [KLM 89]. [KLM 89] make use of this method to approximate the number of distinct satisfying assignments of a given boolean formula in disjunctive normal form (DNF).
The paper is organized as follows. Section 2 contains some definitions and notations used throughout the paper. In Section 3 we present the (ϵ, δ)-approximation algorithm for counting simple cycles of a given length in graphs. Finally, we conclude in Section 4 with open problems.

2 Notations and Definitions

The graph terminology used in this paper follows that of Even [Ev 79]. Let $G = (V, E)$ be a graph. A *simple cycle* of length k is a sequence $v_0, v_1, ..v_{k-1}$ of vertices with $v_i \neq v_j$ for $i \neq j$ and $\{v_i, v_{(i+1) \bmod k}\} \in E$ for $0 \leq i < k$. Furthermore, we will need the notion of an approximation algorithm. Consider a problem f and let $\#f$ denote the number of distinct solutions of f.

Definition 1 (ϵ-approximation) *An algorithm A for a counting problem f is an ϵ-approximation if it takes an input instance and a real value ϵ and produces an output y such that*

$$(1 - \epsilon) \cdot \#f \leq y \leq (1 + \epsilon) \cdot \#f$$

Definition 2 ((ϵ, δ)-approximation) *An algorithm A for a counting problem f is an (ϵ, δ)-approximation if it takes an input instance and two real values ϵ, δ and produces an output y such that*

$$Pr[(1 - \epsilon) \cdot \#f \leq y \leq (1 + \epsilon) \cdot \#f] \geq 1 - \delta$$

3 An (ϵ, δ)-Approximation for Counting Simple Cycles

In this section we develop an (ϵ, δ)-approximation algorithm for counting the simple cycles of length k in a given graph G in $\mathcal{O}(M(n) \log n/\epsilon^2 \log(1/\delta))$ time. To achieve this running time we will make heavily use of the following lemma which is implicitly in [AYZ 95].

Lemma 3 *There is an explicit construction of a list of $k^{O(k)} \log n$ colorings that has the property that every sequence $v_1, \ldots v_k$ of k vertices from V is consecutively colored by $0, \ldots, k-1$ in at least one coloring of the list. Such a list may be computed in $\mathcal{O}(n \log n)$ time.*

Let G_j be the directed graph obtained from G by removing all edges between vertices which are non-consecutively colored in the j-th coloring of a list having the properties of Lemma 3 and by directing all edges from a vertex having color h to its neighbor having color $h+1$ (with respect to mod k). Now let $A = A_{G_j}$ be the adjacency matrix of G_j. Denote by $a_{\alpha\beta}^{(k)}$ the elements of the k-th power of A and by $C(G_j)$ the set of the simple directed cycles of length k of G_j. The value $a_{ii}^{(k)}$ clearly determines the number of simple directed cycles of lenght k in G_j which start in the vertex i. Thus computing

$$|C(G_j)| = \sum_{i:\text{the vertex } i \text{ is colored with 0 in } G_j} a_{ii}^{(k)} \tag{1}$$

yields the number of simple directed cycles of length k in G_j. Thus $\sum |C(G_j)|$ is a $\mathcal{O}(\log n)$-approximation for $|C|$ (where $C = C(G)$ is the set of simple undirected cycles of length k in G) with a running time of $\mathcal{O}(M(n)\log n)$, since each simple cycle of length k will be discovered at least once in some direction and at most in each coloring of the list. To obtain the desired (ϵ, δ)-approximation we use the covering-method of [KLM 89] to find an estimate for $|C|$. In order to use this method we need a superset (the universe) U of C. The algorithm involves N independent trials, where each of this trials consists of the following steps:

1. Choose an element u_i from U uniformly at random.
2. Set
$$Y_i := \begin{cases} |U| & \text{if } u_i \in C \\ 0 & \text{otherwise.} \end{cases}$$

The expected value of $Z = |U| \sum_{i=1}^{N} Y_i/N$ will be $|C|$. To ensure that this estimate is with high probability an ϵ-approximation the following lemma is used. A proof can be found e.g. in [MR 95].

Lemma 4 (Bernstein) *Let $\mu = |C|/|U|$. Then the above described method yields an (ϵ, δ)-approximation to $|C|$ provided*

$$N \geq \frac{4}{\epsilon^2 \mu} \ln(2/\delta)$$

Now we construct a suitable universe U. We could choose U to be the set of all subsets of V of size k, but then $1/\mu$ might be in the order of $\mathcal{O}(n^k)$ and the number of trials would be to large if $|C|$ is small. In order to solve this problem we will make use the covering-method of [KLM 89]. Let U_i be the set of directed simple cycles in G_i, then $\bigcup_i U_i = C$, if we replace each directed cycle in U_i by its undirected version. The goal is to estimate the size of this union. The brute-force method is inefficent if C and the U_i are large. The inclusion-exclusion formula is also inefficent, since the number of sets U_i will be in the order of $k^{O(k)} \log n$ and therefore we have to compute $n^{k^{O(k)}}$ terms. In our approach this would be even worse than examining all subsets of size k of V. In general the covering-method requires that the following assumptions are valid to achieve a polynomial running time.

1. For all i, $|U_i|$ is computable in polynomial time.
2. It is possible to sample uniformly at random from any U_i.

3. For all $c \in C$, it can be determined in polynomial time whether $c \in U_i$.

Let r_1, r_2, r_3 be the running time of the 3 steps mentioned above. In our case we have to perform these steps much faster than just in polynomial time. Define the universe U to be the multiset union of the sets U_i: $U = \{(c,i)|c \in U_i\}$ and let $l = k^{O(k)} \log n$. Then we have $|C| \leq |U| = \sum_{i=1}^{l} |U_i| \leq l|C|$. Define $\zeta : U \to \{0,1\}$ as

$$\zeta((c,i)) := \begin{cases} 1 & \text{if } i = \min\{j|c \in U_j \text{ (or its reverse) }\} \\ 0 & \text{otherwise.} \end{cases}$$

Let $\tilde{C} = \{(c,i) \in U|\zeta((c,i)) = 1\}$. Clearly $|C| = |\tilde{C}|$.
Now we are ready to use the covering-method to approximate the size of $\tilde{C}$ and therefore the size of C. We start by showing how to sample uniformly at random from U. This is done in two steps:

1. Choose U_i with probabilty $|U_i|/|U|$.
2. Choose an element c (a simple directed cycle) uniformly at random from this U_i.

The first step can be perfomed by first calculating the values $A_i = \sum_{j=1}^{i} |U_i|$ in $\mathcal{O}(lr_1(n))$ time. Then we choose an $j \in_R \{1, ..., |U|\}$ and find by binary search the i for which $A_{i-1} < j \leq A_i$. The second step is described in the proof of the following lemma and can be done in $r_2(n) = \mathcal{O}(M(n))$ time.

Lemma 5 *Let G_j be a directed graph obtained from G as described in the beginning of this section. Then we can sample uniformly at random a simple directed cycle of length k from U_j in $\mathcal{O}(M(n))$ time.*

PROOF: We first raise the adjacency matrix $A = A_{G_j}$ of G_j to the k-th power. As mentioned before, $a_{ii}^{(k)}$ determines the number of simple cycles in G_j which start in the vertex i. Then we choose a vertex i having color 0 with probability $a_{ii}^{(k)}/|C(G_j)|$, where $|C(G_j)|$ is the value of the sum (1). Denote the vertex we have choosen by v_{c_1}. We now raise the adjacency matrix of G_j to the $k-1$-th power. This gives us all pairs of vertices connected by directed paths of length $k-1$ in G_j. In particular, we have all the pairs of vertices having v_{c_1} as second endpoint and the number $P(v_{c_1}, w)$ of directed paths $v_{c_1}, \ldots, w$ of length $k-1$ for each vertex w. We choose one of those pairs v_{c_1}, w with $(w, v_{c_1}) \in E(G_j)$ and $v_{c_1}, \ldots, w$ is a directed path of length $k-1$, each with probability $P(v_{c_1}, w)/a_{v_{c_1} v_{c_1}}^{(k)}$. Let us denote the choosen pair by (v_{c_1}, v_{c_k}) and define $p_k = P(v_{c_1}, v_{c_k})$. Clearly the color of v_{c_k} is $k-1$. We proceed by raising the adjacency matrix $k-2$-th power and by choosing a pair v_{c_k}, w with $(w, v_{c_k}) \in E(G_j)$ and $v_{c_1}, \ldots, w$ is a directed path of length $k-2$, each with probability $P(v_{c_1}, w)/p_k$. We denote the choosen pair by $(v_{c_{k-1}}, v_{c_k})$ and define $p_{k-1} = P(v_{c_1}, v_{c_{k-1}})$. By iterating this process by performing analogeously computations we clearly sample a simple directed cycle of length k uniformly at random in k phases, since

$$\begin{aligned} \Pr[(v_{c_1}, ..., v_{c_k}) \text{ is the choosen cycle }] &= \frac{a_{v_{c_1} v_{c_1}}^{(k)}}{|C(G_j)|} \frac{p_k}{a_{v_{c_1} v_{c_1}}^{(k)}} \frac{p_{k-1}}{p_k} \ldots \frac{1}{p_3} \\ &= \frac{1}{|C(G_j)|} \end{aligned}$$

Since the time of each phase is dominated by the complexity of the matrix multiplication and k is constant, the running time of the whole procedure is $\mathcal{O}(M(n))$. ∎

To test whether this sample c of step 2 above is in $\tilde{C}$, we evaluate $\zeta((c,i))$. This can be done in $\mathcal{O}(lr_3(n))$ time as described later. Thus, by the *Bernstein*-Lemma we have an (ϵ,δ)- approximation algorithm for counting the number of simple cycles of length k in G with a running time of $O(lr_1(n) + l1/\epsilon^2 \log(1/\delta)(r_2(n) + lr_3(n)))$. We summarize the description of the algorithm by a listing:

Algorithm 6 *(ϵ,δ)-approximation algorithm for counting simple cycles of length k*

Input: *Graph G with m edges and n vertices, fault-tolerance ϵ, error-probability δ and $k \in \mathcal{O}(1)$.*

Step 1: *Compute $|U_j|$, the number of simple cycles of lenght k in G_j (which can be constructed as described before).*

Step 2: *Compute $A_i = \sum_{j=1}^{i} |U_j|$ for $1 \leq i \leq l$ and set $A_0 = 0$.*

Step 3: $\tilde{Y} := 0$.

Step 4: *Execute $N = l \cdot \frac{4\ln(2/\delta)}{\epsilon^2}$ times*

- *a) Choose an i with probability $|U_i|/|U|$. This is done by choosing an $j \in_R \{1 \ldots |U|\}$ and finding the i with $A_{i-1} < j \leq A_i$ using binary search.*
- *b) Choose an element c (a simple directed cycle of length k) uniformly at random from U_i.*
- *c) If $\zeta((c,i)) = 1$ then $Y := |U|$, else $Y := 0$.*
- *d) $\tilde{Y} := \tilde{Y} + Y$.*

Output: $\tilde{Y}/N$.

Remark 7 *Steps 4 a) and 4 b) choose an $(c,i) \in U$ with probability $1/|U|$.*

Theorem 8 *The algorithm above is an (ϵ,δ)-approximation and has a running time of $\mathcal{O}(M(n)\log n/\epsilon^2 \log(2/\delta))$.*

PROOF: If we choose a right value for N, then Lemma 4 ensures us the (ϵ,δ)-property, if the random variable Y of step 4 c) has the expected value $E[Y] = |C|$. Let $\mathrm{cov}(c) = \{i \mid c$ (or its reverse) is a simple directed cycle of length k in $G_i\}$. Then $\sum_{i \in \mathrm{cov}(c)} \zeta((c,i)) = 1$ if $\mathrm{cov}(c) \neq \emptyset$, since for each cycle $c \in C$ there exists exactly one i with $\zeta((c,i)) = 1$. Thus, $E[Y] = \sum_{(c,i)} \zeta((c,i)) = \sum_c \sum_{i \in \mathrm{cov}(c)} \zeta((c,i)) = \sum_c 1 = |C|$. Therefore the algorithm is an (ϵ,δ)-approximation.
The running time follows by analysing the number of executions of step 4 and by determing the values for r_1, r_2 and r_3. The value $|U_j|$ may be computed in $r_1(n) = \mathcal{O}(M(n))$ time for each j by raising the adjacency matrix of G_j to the k-th power and computing the sum (1) from the beginning of this section. Thus we can compute step 1 and 2 in $\mathcal{O}(lM(n))$ time, since we can construct the list of colorings having the properties of Lemma 3 in $\mathcal{O}(n \log n)$ time and each G_j in $\mathcal{O}(n^2)$ time. Steps 4 a) and b) take $r_2(n) = \mathcal{O}(M(n))$ time by Lemma 5. In step 4 c) we have to determine the i^*, for which $\zeta((c,i^*)) = 1$. This means that we have to check for each $i^* \leq i$ whether c (or its reverse) is in G_{i^*}. Each such verification can easily be done in $r_3(n) = \mathcal{O}(1)$ time. Thus, step 4 c) takes $lr_3(n) = \mathcal{O}(l)$ time. Since the most expensive step in the loop of step 4 is step 4 b) and $l = k^{O(k)} \log n$, we obtain an $\mathcal{O}(M(n)\log n/\epsilon^2 \log(1/\delta))$ time algorithm. ∎

4 Open Problems and Further Research

We have presented an (ϵ, δ)-approximation to the number of simple cycles of a given length k in graphs which runs in $\mathcal{O}(M(n) \log n/\epsilon^2 \log(1/\delta))$ time. For counting small simple cycles, i.e. of size $k \leq 7$, one can use the algorithm of [AYZ 94] to exactly count the number of such cycles in $\mathcal{O}(M(n))$ time. Can the number of simple cycles of length $k > 7$ be counted exactly within $\tilde{\mathcal{O}}(M(n))$ time? Maybe there are faster algorithms for special graph classes like e.g. dense graphs.

References

[AYZ 94] Alon, N., Yuster, R., Zwick, U., *Finding and counting given length cyles*, Proc. 2^{nd} European Symposium on Algorithms (1994), pp. 354–364.

[AYZ 95] Alon, N., Yuster, R., Zwick, U., *Color-coding*, Proc. 42^{nd} Journal of the ACM (1995), pp. 844–856.

[CW 87] Coppersmith, D., Winograd, S., *Matrix multiplication via arithmetic progressions*, Proc. 19^{th} ACM STOC (1987), pp. 1–6.

[Do 96] Dorgerloh, C. F., *A Fast Randomized Parallel Algorithm for Finding Simple Cycles in Planar Graphs*, Research Report 85150-CS, Institut für Informatik der Universität Bonn, 1996.

[DW 96] Dorgerloh, C. F., Wirtgen, J., *A note on improving the running time of a class of parallel algorithms using randomization*, Research Report 85159-CS, Institut für Informatik der Universität Bonn, 1996.

[DW 97b] Dorgerloh, C. F., Wirtgen, J., *Once again: Finding simple cycles*, Research Report 85165-CS, Institut für Informatik der Universität Bonn, 1997.

[DW 97a] Dorgerloh, C., Wirtgen, J., *Faster Finding of Simple Cycles in Planar Graphs on a randomized EREW-PRAM*, Proc. 2^{nd} Workshop on Randomized Parallel Computing (1997), held in conjunction with IPPS'97.

[Ev 79] Even, S., *Graph Algorithms*, Computer Science Press, 1979.

[KLM 89] Karp, R. M., Luby, M., Madras, N., *Monte-Carlo Approximation Algorithms for Enumeration Problems*, Journal of Algorithms **10** (1989), pp. 429–448.

[Mo 85] Monien, B., *How to find long paths efficiently*, Annals of Discrete Mathematics **25** (1985), pp. 239–254.

[MR 95] Motwani, R., Raghavan, P., *Randomized Algorithms*, Cambridge University Press, 1995.

[Ri 86] Richards, D., *Finding short cycles in a planar graph using seperators*, Journal of Algorithms **7** (1986), pp. 382–394.

[RL 85] Richards, D., Liestman, A. L., *Finding cycles of a given length*, Annals of Discrete Mathematics **27** (1985), pp. 249–256.

[YZ 94] Yuster, R., Zwick, U., *Finding even cycles even faster*, Proc. 21^{st} International Colloquium on Automata, Languages and Programming (1994), pp. 532–543.