Efficient Sequential and Parallel Algorithms for the Negative Cycle Problem* (to appear in ISAAC'94)

Dimitris Kavvadias[1,2], Grammati E. Pantziou[1,4], Paul G. Spirakis[1,3] and Christos D. Zaroliagis[1,5]

[1] Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece
[2] Dept of Mathematics, Patras University, 26500 Patras, Greece
[3] Dept of Computer Sc. & Eng., Patras University, 26500 Patras, Greece
[4] Dept of Math. & Computer Sc., Dartmouth College, Hanover NH 03755, USA
[5] Max-Planck Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany

Abstract. We present here an algorithm for detecting (and outputting, if exists) a negative cycle in an n-vertex planar digraph G with real edge weights. Its running time ranges from $O(n)$ up to $O(n^{1.5}\log n)$ as a certain topological measure of G varies from 1 up to $\Theta(n)$. Moreover, an efficient CREW PRAM implementation is given. Our algorithm applies also to digraphs whose genus γ is $o(n)$.

1 Introduction

Let $G = (V, E)$ be an n-vertex, m-edge digraph with real edge weights and P be a simple path in G between two vertices v and w. The *cost* of P (denoted by $c(P)$) is the sum of the weights of all edges in P. A simple cycle C in G is a simple path starting and ending at the same vertex v. If $c(C) < 0$, then C is called *negative*. The negative cycle problem is the following: given a digraph $G = (V, E)$ with real edge weights, find whether G has any negative cycle. If such a cycle exists, then output the cycle.

The problem of finding a negative cycle in a digraph is a fundamental problem in combinatorial optimization with a lot of applications: shortest paths [3, 22], two dimensional package element [17], min-cost flows [20], minimal cost-to-time ratio [14], checking constraints in VLSI layout [15], etc. Perhaps the most important application concerns the shortest path problem. In a digraph, there is a shortest path from a vertex v to a vertex u if and only if no path from v to u contains a negative cycle (see e.g. [22], Theorem 7.1). Much of the old and new results on the shortest path problem make the assumption that either the digraph has non-negative edge weights [3, 10, 16] or does not contain a negative cycle if negative edge weights are present [6, 8, 9, 19]. While efficient algorithms for the shortest path problem exist in this case (e.g. $O(m + n\log n)$ time for the single-source shortest path problem (sssp) [10]), one needs $O(nm)$ time (Bellman-Ford

* Supported by the EEC ESPRIT Basic Research Action No. 7141 (ALCOM II), by the Ministry of Education of Greece and by the NSF grant No. CDA-9211155. Email: kavadias/spirakis@cti.gr, pantziou@cs.dartmouth.edu, zaro@mpi-sb.mpg.de.

method [3]) if negative edge weights are present in order to solve the sssp problem or to detect a negative cycle. Also, different algorithms for the negative cycle problem have been proposed in [21, 22], but the complexity remains $O(nm)$. In the case that the input digraph is planar, the best previous algorithm is due to Mehlhorn & Schmidt [18] and runs in $O(n^{1.5} \log n)$ time. In parallel computation, the main tool used was matrix powering (using e.g. the method described in [14]), which means that we need $O(\log^2 n)$ time and $M_s(n)$ EREW PRAM processors. (The best value for $M_s(n)$ up to now is $O(n^3(\log\log n)^{1/3}/(\log n)^{7/6})$ [11].) In the case of planar digraphs, the best previous algorithm was given by Cohen [2] and runs in $O(\log^5 n)$ time using $O(n^2)$ work (i.e. the total number of operations) on a CREW PRAM.

The main contribution of this paper is an efficient algorithm for the negative cycle problem in planar digraphs which is parameterized in terms of a topological measure $\tilde{\gamma}$ of the input digraph G (Section 4). More precisely, the sequential implementation of our algorithm runs in $O(n + \tilde{\gamma}^{1.5} \log \tilde{\gamma})$ time. Here $\tilde{\gamma}$ is a topological measure of G and is proportional to the cardinality of a minimum set of faces covering all vertices of G (among all embeddings of G in the plane). The value of $\tilde{\gamma}$ ranges from 1 up to $\Theta(n)$ depending on the particular topological structure of G. (Note for example, that if G is outerplanar then $\tilde{\gamma} = 1$.) A parallel implementation of our algorithm on a CREW PRAM runs in $O(\log^2 n + \log^5 \tilde{\gamma})$ time using $O(n + \tilde{\gamma}^2/\log^5 \tilde{\gamma})$ processors. Our results are clear improvements over the best previous ones, in all cases where $\tilde{\gamma} = o(n)$. Our algorithm does not need an embedding of minimum $\tilde{\gamma}$ to work with and also seems to be very efficient in practice, since we prove that $\tilde{\gamma}$ is $O(1)$ for random graphs which are planar according to the $G_{n,p}$ model [7] (Section 5).

Our results are based: (i) on the hammock decomposition technique [8, 9, 13, 19] which decomposes a given digraph into certain outerplanar digraphs called hammocks, and (ii) on the optimal solution of the negative cycle problem in an outerplanar digraph (Section 3). In this case we give an algorithm that runs in $O(n)$ sequential time. Its parallel implementation runs in $O(\log n \log^* n)$ time using $O(n/\log n \log^* n)$ processors on a CREW PRAM. The latter result is of independent interest, since no optimal algorithm was known before for this problem. Note that for the sssp or the apsp (all-pairs shortest paths) problem in outerplanar digraphs there exist an $O(n)$-time sequential algorithm [9] and an $O(\log^2 n)$-time, $O(n)$-processor CREW PRAM one [19].

To see how our results compare with the best known shortest path algorithms, consider for example the $O(n\tilde{\gamma})$-time algorithm of [9] for the apsp problem (which encodes shortest path information in compact routing tables), or an alternative implementation which runs in $O(n+\tilde{\gamma}^2)$ time (with partial use of these tables). In both cases, our algorithm removes the assumption made in [9] for non-existence of a negative cycle, without increasing the time bounds for solving the problem if such cycles are present. Note also that our bounds match the best previous ones [2, 18] only when $\tilde{\gamma}$ reaches its extreme value of $\Theta(n)$. Our results can be extended to digraphs whose genus γ is $o(n)$. Even in this case, the bounds are much lower than the best known ones for solving the apsp problem [8, 13].

2 Preliminaries

A *hammock decomposition* is a decomposition of a digraph G into certain outerplanar digraphs called *hammocks* [8, 9]. Hammocks satisfy certain separator conditions and hammock decomposition employs the following properties: (i) each hammock has at most *four* vertices in common with any other hammock (and therefore with the rest of the graph), called the *attachment vertices*; (ii) the hammock decomposition spans all the edges of G, i.e. each edge belongs only to one hammock; and (iii) the number of hammocks produced is order of the minimum possible among all possible decompositions, and is proportional to a topological measure $\tilde{\gamma}$ of G. In the case of sparse digraphs (i.e. digraphs with $O(n)$ edges), $\tilde{\gamma}$ varies from 1 up to $\Theta(n)$. Actually, as it is defined in [8], $\tilde{\gamma} = \Theta(\gamma(G'))$, where $\gamma(G')$ is the genus of a graph G'. Here G' is G with a new vertex v added and edges from v to every vertex of G. Moreover, $\gamma(G') \leq \gamma(G)+q$ where G is supposed to be embedded into an orientable surface of genus $\gamma(G)$ so as to minimize the number q of faces that collectively cover all vertices. As it is proved in [8, 13], such an embedding of G does not need to be provided by the input, in order to produce a hammock decomposition into $\tilde{\gamma}$ hammocks either in sequential or in parallel computation. Note that $\tilde{\gamma} = q$ if the digraph is planar, and $\tilde{\gamma} = 1$ if the digraph is outerplanar (since $q = 1$ in this case). Hammock decomposition can be computed sequentially in $O(n)$ time [8, 9]. A parallel implementation on a CREW PRAM runs in $O(\log n \log\log n)$ time using $O(n)$ processors [13]. In the case that the input digraph is planar, the time bound can be further improved to $O(\log n \log^* n)$ [19].

3 An optimal work algorithm for detecting negative cycles in outerplanar digraphs

In this section we will show how to solve optimally the negative cycle problem provided that the input digraph is outerplanar and biconnected. (If it is not biconnected, apply the algorithm to every biconnected component.) We shall describe here the parallel implementation. (The sequential one can be obviously derived.) This is achieved by using a novel extension of the fundamental tree-contraction technique to the tree of interior faces of the outerplanar digraph.

Let $G_o = (V, E)$ be the input outerplanar digraph and let $\hat{G}_o$ be its undirected version. It is well known that the dual graph of $\hat{G}_o$ is a tree, called the *tree of faces*. (The exterior face is excluded in this construction.) We assume that all vertices in G_o are named consecutively in clockwise order around the exterior face and the tree of faces in $\hat{G}_o$ is binary. According to [4, 19] the construction of the tree of faces and its binarization, as well as the consecutive naming of the vertices in G_o can be achieved in $O(\log n \log^* n)$ time using $O(n/\log n \log^* n)$ CREW PRAM processors. (More details in [12].)

Let $s_1(f)$ (resp., $s_2(f)$) be the minimum (resp., maximum) numbered vertex in each interior face (or group of neighboring faces) f. We call these vertices, the *associated vertices* of f and the arc(s)[6] $\langle s_1(f), s_2(f)\rangle$ ($\langle s_2(f), s_1(f)\rangle$) the *associated arc(s)* of f. A pair of vertices v, w is called a *separation pair* in $\hat{G}_o$, if their removal disconnects $\hat{G}_o$. If $\{v, w\}$ is an edge then it is called a *separation edge* or *separator*. We shall use the interval notation $[u, z]$ to denote the set of vertices $\{u, u+1, ..., z-1, z\}$, according to the clockwise naming of vertices around the exterior face of G_o. (Such an interval is allowed to wrap around from n back to 1.) A path from a vertex v to a vertex w in a subgraph J of G_o will be denoted by $P(v, w; J)$. A shortest path from v to w in J is denoted by $SP(v, w; J)$. By $SP_z(x, y; J)$ we will denote the shortest path from x to y in J with the restriction that this shortest path consists only of vertices in $[x, y]$ or $[y, x]$ depending on which of the two subintervals z belongs to. In the sequel, by $\{v, w\}$ we denote the singleton $\{\langle v, w\rangle\}$, or $\{\langle w, v\rangle\}$, or their union depending on the existence of those arcs in G_o. Furthermore, all references to $\{v, w\}$ are considered to be references to all its members.

Consider the following problem: Let G be an outerplanar digraph and $\{v, w\}$ be a separation edge, separating G into two subgraphs G_1 and G_2. Suppose also that in each G_i, $i = 1, 2$, there is no negative cycle. Then a negative cycle $N(G)$ for G (if it exists) will consist of a path in G_1 joined with a path in G_2. Also, $N(G)$ will have v and w as two of its vertices. In order to find $N(G)$ it suffices to find $SP(v, w; G_1)$ (or $SP(w, v; G_1)$) and $SP(w, v; G_2)$ (or $(SP(v, w; G_2))$. The union of these two paths will give (the possible) $N(G)$. Therefore we have the following.

Proposition 1. *Let $G = (V, E)$ be an outerplanar digraph and let $\{v, w\}$ be a separator, separating G into two subgraphs G_1 and G_2. Suppose also that in each G_i, $i = 1, 2$, there is no negative cycle and that the two shortest paths between v and w are known. Then G can be tested for a negative cycle in $O(1)$ time.*

The main idea of the algorithm is the following. We assume that at a certain point, a set of neighboring faces has been tested for a negative cycle. In the case of a positive answer, the negative cycle is output and the algorithm stops. Otherwise, this set of faces is joined to a neighboring set of faces that has also been tested, in order to form a new set. Detection of a negative cycle in this union is done according to the rules stated in Proposition 1. Thus, the algorithm proceeds in a bottom-up fashion.

We say that an interior face f has been *evaluated* in the tree of faces T, iff in the subgraph induced by its descendant nodes in T we have tested if there is a negative cycle and in the case of a negative answer, we have computed shortest path information between certain pairs of vertices in f. The main goal is to evaluate the root face of T. The parallel tree-contraction algorithm [1] evaluates the root of a tree T processing a logarithmic number of binary trees $T_0, T_1, ..., T_k$, $k = O(\log|T|)$, $T_0 = T$ and T_k contains only one node. Also, $|T_i| \leq \varepsilon|T_{i-1}|$, $0 < \varepsilon < 1$. The tree T_i is obtained by T_{i-1} by applying a local operation, called *SHUNT*, to a subset of the leaves of T_{i-1}. The *SHUNT* operation consists in turn by two other operations, called *prune* and *bypass* [1]. Let l be a leaf in a

[6] We refer to directed edges as "arcs".

tree T_{i-1}. By "pruning l" we denote the deletion of l from T_{i-1}. Let l' be the unique child of a non-root node v in T_{i-1}. Then by "bypassing l'" we denote the joining of l' and v into a new node v'. To complete the description of the tree-contraction algorithm we have to show how the information concerning the negative cycle is maintained. We can prove the following [12] (see fig. 3.1 and 3.2).

Lemma 1. *Let G be an outerplanar digraph and $\{v, w\}$ be a separator of G, separating it into two subgraphs G_1 and G_2. Let also $\{a, b\}$ and $\{c, d\}$ be edges belonging to G_1 and G_2 respectively, and such that $b = a - 1$ and $d = c - 1$ (assuming consecutive clockwise naming of vertices in G). Suppose that the following shortest paths in G_1, G_2 (as well as their costs) are known: $SP(a, b; G_1)$, $SP(b, a; G_1)$, $SP(v, w; G_1)$, $SP(w, v; G_1)$, $SP(v, w; G_2)$, $SP(w, v; G_2)$, $SP(c, d; G_2)$, $SP(d, c; G_2)$, $SP_{a+1}(a, v; G_1)$, $SP_{a+1}(v, a; G_1)$, $SP_{b-1}(b, w; G_1)$, $SP_{b-1}(w, b; G_1)$, $SP_{d-1}(v, d; G_2)$, $SP_{d-1}(d, v; G_2)$, $SP_{c+1}(c, w; G_2)$ and $SP_{c+1}(w, c; G_2)$. Then in $O(1)$ time we can compute $SP(a, b; G)$, $SP(b, a; G)$, $SP(c, d; G)$ and $SP(d, c; G)$.*

Lemma 2. *Let G be an outerplanar digraph and $\{v, w\}$ be a separator of G, separating it into two subgraphs G_1 and G_2. Let $\{a, b\}$ and $\{c, d\}$ be edges of G_2 such that $b \leq c < d \leq w < v \leq a$ (assuming consecutive clockwise naming of vertices in G). Assume that there are no edges $\{v, z\}$, $\{w, z\}$: $z \in [b, c]$. Suppose that the following shortest paths are provided by the input: $SP(a, b; G_2)$, $SP(b, a; G_2)$, $SP(v, w; G_1)$, $SP(w, v; G_1)$, $SP(c, d; G_2)$, $SP(d, c; G_2)$, $SP_{a-1}(a, v; G_2)$, $SP_{a-1}(v, a; G_2)$, $SP_{b+1}(b, c; G_2)$, $SP_{b+1}(c, b; G_2)$, $SP_{d+1}(w, d; G_2)$, and $SP_{d+1}(d, w; G_2)$. Then in $O(1)$ time we can compute $SP(a, b; G_1 \cup G_2)$, $SP(b, a; G_1 \cup G_2)$, $SP(c, d; G_1 \cup G_2)$ and $SP(d, c; G_1 \cup G_2)$.*

Clearly, the above lemmata hold in the case where the vertices around the exterior face of G constitute – in clockwise order – a constant number of intervals (instead of one). It is worth noting that as the algorithm proceeds and the tree is contracted, each leaf of T corresponds to a set of neighboring faces whose removal does not disconnect G_o. Conversely, an internal node of T corresponds to a set of faces whose removal disconnects G_o. Therefore it remains to explain what information is exchanged and/or updated during a *SHUNT* operation. We will distinguish between two types of *SHUNT* operations depending on what is the sibling of the tree node performing this operation.

TYPE-1: suppose that l_i is performing a *SHUNT* operation (see fig.3.3). Suppose also that l'_i is a leaf. Initially we know all shortest paths: (a) between the associated vertices of f_i, l_i and l'_i inside f_i; (b) between the associated vertices of l_i inside l_i; and (c) between the associated vertices of l'_i inside l'_i. During the prune operation examine if there exists a negative cycle in $l_i \cup f_i$ (using Proposition 1). If it exists then stop, report that a negative cycle was found and output the cycle. Otherwise, compute $SP(s_x(l'_i), s_y(l'_i); l_i \cup f_i)$ and $SP(s_x(f_i), s_y(f_i); l_i \cup f_i)$ (using Lemma 2), where $1 \leq x \leq 2$, $1 \leq y \leq 2$, $x \neq y$, and continue with the bypass operation resulting into a new node f'_i, where $f'_i = l_i \cup f_i \cup l'_i$. Check again if there exists a negative cycle in f'_i and if not, compute $SP(s_x(f_i), s_y(f_i); f'_i)$

(using Lemma 1). Note that after this *SHUNT* operation (and in the case where a negative cycle was not found) we have $s_1(f_i') = s_1(f_i)$ and $s_2(f_i') = s_2(f_i)$.

TYPE-2: suppose that l_k is performing a *SHUNT* operation, and f_j, the sibling of l_k, is an internal node of T (see fig.3.4). Initially we know all shortest paths: (a) between the associated vertices of f_k, l_k and f_j inside f_k; (b) between the associated vertices of l_k inside l_k; and (c) between the associated vertices of f_j, f_i and l_j inside f_j. During the prune operation examine if there exists a negative cycle in $l_k \cup f_k$ (using Proposition 1). If it exists then stop, report that a negative cycle was found and output the cycle. Otherwise, compute $SP(s_x(f_j), s_y(f_j); l_k \cup f_k)$ and $SP(s_x(f_k), s_y(f_k); l_k \cup f_k)$ (using Lemma 2). After that continue with the bypass operation. This results into a new node f_{kj}, where $f_{kj} = l_k \cup f_k \cup f_j$. Check again if there exists a negative cycle in f_{kj} and if not, compute $SP(s_x(f_k), s_y(f_k); f_{kj})$, $SP(s_x(l_j), s_y(l_j); f_{kj})$ and $SP(s_x(f_i), s_y(f_i); f_{kj})$ (using Lemma 1). Note that the computation of $SP(s_x(l_j), s_y(l_j); f_{kj})$ can only be affected by the (possibly new) cost of $SP(s_x(f_j), s_y(f_j); l_k \cup f_k)$ which has been computed during the prune operation. Also, after this *SHUNT* operation (and in the case where a negative cycle was not found) we have $s_1(f_{kj}) = s_1(f_k)$ and $s_2(f_{kj}) = s_2(f_k)$. This completes the description of the *SHUNT* operations.

Let us call the algorithm presented in this section *Out_Neg_Cycle*. The discussion in this section leads to the following.

Theorem 1. *Given an n-vertex outerplanar digraph $G_o = (V, E)$ with real-valued edge costs, algorithm Out_Neg_Cycle detects and outputs a negative cycle in G_o, if it exists, in $O(\log n \log^* n)$ time using $O(n/\log n \log^* n)$ CREW PRAM processors. A sequential implementation runs in $O(n)$ time.*

4 Detecting a Negative Cycle in a Planar Digraph

In this section we give an efficient algorithm for detecting a negative cycle in a planar digraph G (and outputting it if it exists). The algorithm is based on the hammock decomposition technique and on the detection of a negative cycle in an outerplanar digraph presented in the previous section. The algorithm proceeds as follows: (1) Find a hammock decomposition of G into $\tilde{\gamma}$ hammocks using the algorithms of [8, 13]. (2) Run algorithm Out_Neg_Cycle in each hammock H, to detect if there is any negative cycle in H. If a negative cycle is found in any hammock, then output one such cycle and stop. (3) In each hammock H, compute shortest path trees rooted at the four attachment vertices of H. Then, compress each hammock into an $O(1)$-sized graph such that the shortest paths between its attachment vertices are preserved. (This takes $O(n)$ time sequentially by [9], or $O(\log^2 n)$ time using $O(n)$ processors on a CREW PRAM by [19].) This results into a new planar digraph $G_{\tilde{\gamma}}$ of size $O(\tilde{\gamma})$. Examine if there is any negative cycle in $G_{\tilde{\gamma}}$ by running the algorithm of [18] (resp. [2]) for the sequential (resp. parallel) case. (4) If a negative cycle is found, then output the cycle taking into account the subpaths contained in each hammock. Otherwise, output that there is no negative cycle in G. Therefore, we have the following.

Theorem 2. *There exists an algorithm which detects (and outputs, if exists) a negative cycle in an n-vertex planar digraph G with real edge weights, in $O(n + \tilde{\gamma}^{1.5} \log \tilde{\gamma})$ time. A CREW PRAM implementation of the algorithm runs in $O(\log^2 n + \log^5 \tilde{\gamma})$ time using $O(n + \tilde{\gamma}^2 / \log^5 \tilde{\gamma})$ processors.*

Based on the above approach and the results in [2, 5, 8, 13, 18] we can also prove the following in the case of digraphs with genus $\gamma = o(n)$.

Theorem 3. *There exists an algorithm for detecting (and outputting, if exists) a negative cycle in an n-vertex digraph G with genus $o(n)$, in $O(n + \min\{\tilde{\gamma}^{3a} + \tilde{\gamma}^{1+a} \log \tilde{\gamma}, \tilde{\gamma}^2\})$ time, where $(1/2) \le a < 1$ depends on the separator size of G. A CREW PRAM implementation runs in $O(\log^2 n + \log^3 \tilde{\gamma})$ time using $O(n \log^2 n + \tilde{\gamma}^2 + \tilde{\gamma}^{2a+1})$ work, given that the separator is provided by the input.*

5 Expected Number of Hammocks

In order to study the expected behaviour of $\tilde{\gamma}$ we assume that the input graph G is a random one according to the $G_{n,p}$ model [7], with probability for an edge to exist $p \le c/n$, c a constant. (Note that we are interested in sparse digraphs and here we need only their undirected version.) According to this model, G is planar when $p < 1/n$ [7]. Since $\tilde{\gamma} = \Theta(\gamma(G'))$ (recall Section 2), it suffices to find an estimate for $\gamma(G')$. The genus of G' is bounded above by the number of subgraphs homeomorphic to $K_{2,3}$ plus the number of subgraphs homeomorphic to K_4 in the initial graph G. To see this, consider a subgraph S of G that is homeomorphic to $K_{2,3}$ or to K_4 and let S' be the subgraph of G' induced on $S \cup \{v\}$. Then, the minimum number of handles which must be added to a sphere so that S' can be embedded on the resulting surface, is one.

Proposition 2. *Let G be an instance of $G_{n,p}$ and X be the number of subgraphs of G that are homeomorphic to $K_{2,3}$. Then, we have the following: (a) If $p < 1/n$, then $E(X) \longrightarrow 0$ and (b) If $p > 1/n$, then $E(X) \longrightarrow \infty$.*

Proof: Let S_i be a subgraph of G. Let X_i be an indicator random variable such that $X_i = 1$ iff S_i is homeomorphic to $K_{2,3}$ and $X_i = 0$ otherwise. Then, $X = \sum_i X_i$ and $E(X) = \sum_i E(X_i)$. Let x_j, $j \in \{1, \cdots, 6\}$, denote the number of vertices in each one of the six paths between the vertices of S_i that correspond to the five vertices of $K_{2,3}$. Then we have:

$$E(X) \le \binom{n}{5} \left[\sum_{x_1=0}^{n-5} \binom{n-5}{x_1} x_1! p^{x_1+1} \left[\sum_{x_2=0}^{n-5-x_1} \binom{n-5-x_1}{x_2} x_2! p^{x_2+1} \cdots \right.\right.$$

$$\left.\left.\left[\cdots \left[\sum_{x_6=0}^{n-5-\sum_{j=1}^{5} x_j} \binom{n-5-\sum_{j=1}^{5} x_j}{x_6} x_6! p^{x_6+1} \right] \cdots \right]\right]\right]$$

$$\le \binom{n}{5} p^6 \left(\frac{(n-5)^{n-4} p^{n-4} - 1}{(n-5)p - 1} \right)^6 . \text{ The proposition follows. } \blacksquare$$

Using the above and the second moment method we can prove the following.

Lemma 3. *Let G be an instance of $G_{n,p}$ and X be the number of subgraphs of G which are homeomorphic to $K_{2,3}$. The following hold: (a) If $p < 1/n$, then* $\Pr[X > 0] \longrightarrow 0$ *and (b) If $p > 1/n$, then* $\Pr[X = 0] \longrightarrow 0$.

We can similarly prove a lemma analogous to Lemma 3 for the number of subgraphs that are homeomorphic to K_4. Thus, we have the following.

Theorem 4. *Let G be a random graph according to the $G_{n,p}$ model. If $p < 1/n$, then* $\Pr[\gamma(G') > 0] \longrightarrow 0$. *As a consequence, a random graph which is planar according to the $G_{n,p}$ model can be decomposed into an $O(1)$ number of hammocks.*

We conjecture here that the above theorem holds for any graph G with $p = \Theta(1/n)$, but it seems that a different analysis is needed.

Acknowledgements. We are grateful to Shiva Chaudhuri, Torben Hagerup and Kurt Mehlhorn for many helpful discussions.

References

1. K. Abrahamson, N. Dadoun, D. Kirkpatrick and T. Przytycka, "A Simple Parallel Tree Contraction Algorithm", *J. of Algorithms*, 10(1989), pp.287-302.
2. E. Cohen, "Efficient Parallel Shortest-paths in Digraphs with a Separator Decomposition", *Proc. 5th ACM SPAA*, 1993, pp.57-67.
3. T. Cormen, C. Leiserson and R. Rivest, "Introduction to Algorithms", MIT Press & McGraw Hill, 1990.
4. K. Diks, T. Hagerup and W. Rytter, "Optimal Parallel Algorithms for the Recognition and Colouring of Outerplanar Graphs", *Proc. MFCS'89*, LNCS 379.
5. H. Djidjev, "A Linear Algorithm for Partitioning Graphs of Fixed Genus", SERDICA, Vol.11, 1895, pp.369-387.
6. H. Djidjev, G. Pantziou and C. Zaroliagis, "Computing Shortest Paths and Distances in Planar Graphs", in *Proc. 18th ICALP*, 1991, LNCS 510, pp. 327-339.
7. P. Erdős and J. Spencer, "Probabilistic Methods in Combinatorics", Academic Press, 1974.
8. G.N. Frederickson, "Using Cellular Graph Embeddings in Solving All Pairs Shortest Path Problems", *Proc. 30th IEEE Symp. on FOCS*, 1989, pp.448-453.
9. G.N. Frederickson, "Planar Graph Decomposition and All Pairs Shortest Paths", *J. ACM*, Vol.38, No.1, January 1991, pp.162-204.
10. M. Fredman and R. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms", *JACM*, 34(1987), pp. 596-615.
11. Y. Han, V. Pan and J. Reif, "Efficient Parallel Algorithms for Computing All Pair Shortest Paths in Directed Graphs", *Proc. 4th ACM SPAA*, 1992, pp.353-362.
12. D. Kavvadias, G. Pantziou, P. Spirakis and C. Zaroliagis, "Hammock-on-Ears Decomposition: A Technique for Parallel and On-line Path Problems", CTI Technical Report, TR-93.05.22, Patras, 1993.
13. D. Kavvadias, G. Pantziou, P. Spirakis and C. Zaroliagis, "Hammock-on-Ears Decomposition: A Technique for the Efficient Parallel Solution of Shortest Paths and Other Problems", *Proc. 19th MFCS'94*, LNCS, to appear.

14. E.L. Lawler, “Combinatorial Optimization: Networks and Matroids”, Holt, Rinehart and Winston, 1976.
15. T. Lengauer, “Efficient Algorithms for the Constraint Generation and Intergrated Circuit Layout Compaction”, *Proc. 9th WG83*, pp.219-230, 1983.
16. A. Lingas, “Efficient Parallel Algorithms for Path Problems in Planar Directed Graphs”, *Proc. SIGAL’90*, LNCS 450, pp.447-457, 1990, Springer-Verlag.
17. D. Maier, “An Efficient Method for Storing Ancestor Information in Trees”, *SIAM J. on Comp.*, Vol.8, N.4, pp.599-618, 1979.
18. K. Mehlhorn and B. Schmidt, “A Single Source Shortest Path Algorithm for Graphs with Separators”, in *Proc. FCT83*, LNCS 158, pp.302-309, 1983.
19. G. Pantziou, P. Spirakis and C. Zaroliagis, “Efficient Parallel Algorithms for Shortest Paths in Planar Digraphs”, *BIT* 32 (1992), pp.215-236.
20. C.H. Papadimitriou and K. Steiglitz, “Combinatorial Optimization: Algorithms and Complexity”, Prentice-Hall, 1982.
21. P. Spirakis and A. Tsakalidis, “A Very Fast, Practical Algorithm for Finding a Negative Cycle in a Digraph”, in *Proc. of 13th ICALP*, pp. 397-406, 1986.
22. R.E. Tarjan, “Data Structures and Network Algorithms”, SIAM, 1983.

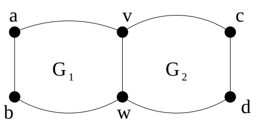

Fig 3.1: Lemma 2.

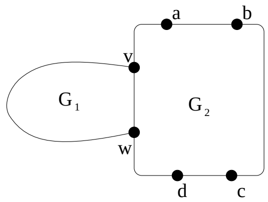

Fig 3.2: Lemma 3.

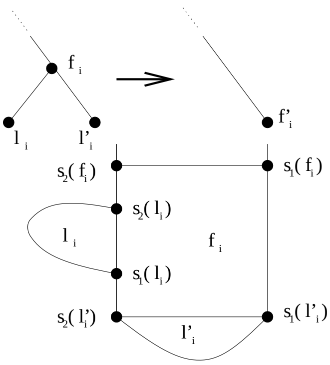

Fig 3.3: SHUNT operation, TYPE-1.

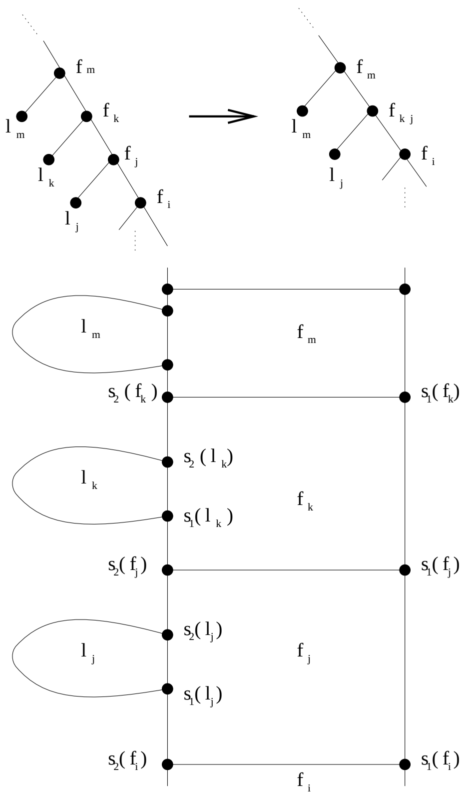

Fig 3.4: SHUNT operation, TYPE-2.