アルゴリズム 52－7
（1996．7．24）

重みなし連結度問題に対する効率的近似アルゴリズム

陳 致中

東京電機大学理工学部数理学科

本稿では、二つのNP困難な問題に対する効率的な近似アルゴリズムを提案する。一つ目の問題は次のとおりである：整数 k と k-辺連結グラフ $G=(V,E)$ が与えられたとき、グラフ (V,E') が k-辺連結であるような最小サイズの $E' \subseteq E$ を求めよ。この問題に対して、以下の性質を持ったアルゴリズムを提案する：

(1) すべての k に対して、近似率は 1.924 である。

(2) 逐次的には $O(k(n+m))$ 時間である。これは、近似率が 2 より小さく かつ 時間量が $O(k(n+m))$ である、最初のアルゴリズムである。

(3) 並列には $n+m$ 個のプロセッサを用い、$O(k\log^3 n)$ 時間である。これは、近似率が 2 より小さく かつ 時間量が n の対数多項式である、最初の並列アルゴリズムである。

(4) G が単純グラフでかつ $k \geq 0.38n$ であるとき、近似率は既知のベスト 1.85 より小さい。

二つ目の問題は次のとおりである：整数 k と k-頂点連結グラフ $G=(V,E)$ が与えられたとき、グラフ (V,E') が k-頂点連結であるような最小サイズの $E' \subseteq E$ を求めよ。この問題に対して、近似率が 1.96 で線形時間の近似アルゴリズムを提案する。これは 近似率が 2 より小さい最初の多項式時間アルゴリズムである。

Efficient Approximation Algorithms for Unweighted Connectivity Problems

Zhi-Zhong Chen

Department of Mathematical Sciences, Tokyo Denki University

This paper presents efficient approximation algorithms for two NP-hard problems of finding minimum spanning subgraphs with a given connectivity requirement. One problem considered is the following: Given an integer k and a k-edge-connected graph $G=(V,E)$ with n vertices and m edges, find an $E' \subseteq E$ of minimum size such that the graph (V,E') is k-edge-connected. Our algorithm for this problem has the following properties:

(1) It achieves an approximation factor of 1.924 for all k.

(2) It has an $O(k(n+m))$-time sequential implementation, and is the first sequential algorithm that both achieves an approximation factor less than 2 and runs in $O(k(n+m))$ time.

(3) It can also be implemented in $O(k\log^3 n)$ time with a linear number of processors, and is the first parallel algorithm that both achieves a constant approximation factor less than 2 for all k and runs in time polylogarithmic in n.

(4) If G has no multiple edge and k is large (namely, $k \geq 0.38n$), the approximation factor achieved by the algorithm is smaller than the previous best 1.85.

The other problem considered is the following: Given a 3-vertex connected graph $G=(V,E)$ with n vertices and m edges, find an $E' \subseteq E$ of minimum size such that the graph (V,E') is 3-vertex-connected. Our algorithm for this problem achieves an approximation factor of 1.96 and runs in linear time. It is the first polynomial-time approximation algorithm for this problem that achieves an approximation factor less than 2. Moreover, it can be implemented in $O(\log^3 n)$ time with $O(m(n+m))$ processors.

1 Introduction

The problems. Let $G = (V, E)$ be a connected graph without self-loops (but possibly with multiple edges), and k be a positive integer. A *k-cut* of G is a set of k edges whose removal disconnects G. G is said to be *k-edge-connected* if it has no $(k-1)$-cut. Similarly, G is said to be *k-vertex-connected* if it has no set of $k-1$ vertices whose removal disconnects G. The *minimum edge-connectivity problem* (MECP for short) is the following: Given a positive integer k and a k-edge-connected graph G, find a minimum k-edge-connected spanning subgraph of G. Similarly, the *minimum vertex-connectivity problem* (MVCP for short) is the following: Given a positive integer k and a k-vertex-connected graph G, find a minimum k-vertex-connected spanning subgraph of G. Sometimes we want to fix the input integer k to a constant; in this case, we denote the problems by k-MECP and k-MVCP to make the constant k explicit. Hereafter, n and m denotes the numbers of vertices and edges in the input graph G, respectively.

MECP, MVCP, and their restricted versions (k-MECP and k-MVCP for various k) have applications in diverse areas of computer science. For example, they can be used to design communication networks that can tolerate a required number of link or site failures.

Existing work on MECP and k-MECP. MECP and k-MECP for $k \geq 2$ are NP-hard. Thus, it is of interest to design approximation algorithms for them. It is rather trivial to obtain a polynomial-time approximation algorithm for MECP (and hence its restricted versions) that achieves an approximation factor of 2. Khuller and Vishkin were the first to give a polynomial-time approximation algorithm for 2-MECP that achieves an approximation factor less than 2 [8]. Subsequently, Garg *et al.* gave better approximation algorithms for 2-MECP [5]. Recently, Khuller and Raghavachari succeeded in giving the first polynomial-time approximation algorithm for MECP that achieves a constant approximation factor less than 2 (namely, 1.85) [7]. Their algorithm runs in $O(k^2n^2)$ time and is of high complexity.

In the parallel setting, there have been several approximation algorithms for MECP and k-MECP [1, 3]. Cheriyan and Thurimella gave a parallel approximation algorithm for MECP [1]. Their algorithm achieves an approximation factor of 2 and runs in time polylogarithmic in n (but not in k) using a polynomial number of processors. Chong and Lam were the first to give an NC approximation algorithm for 2-MECP that achieves an approximation factor less than 2 (namely, $1.5 + \epsilon$ for any $\epsilon > 0$) [3].

Our results on MECP and k-MECP. Although Khuller and Raghavachari's approximation algorithm for MECP achieves an approximation factor of 1.85, its time complexity is rather high. A natural question is to ask if there is a more efficient approximation algorithm for MECP that achieves a constant approximation factor less than 2. In this paper, answering this question affirmatively, we present an approximation algorithm for MECP that achieves an approximation factor of 1.924 and runs in $O(k(n+m))$ time. An interesting property of our algorithm is that its approximation factor is good when k is rather large and G has no multiple edge. More precisely, its approximation factor is smaller than 1.841 if $k \geq 0.38n$ and G has no multiple edge; this improves on the previous best factor 1.85 due to Khuller and Raghavachari [7]. In addition, the approximation factor decreases as k increases from $0.38n$ to $n-1$.

Previous to our work, even for 3-MECP, there had been no NC approximation algorithm achieving a constant approximation factor less than 2. Our algorithm mentioned above can be implemented to run in $O(k \log^3 n)$ time with a linear number of processors on a PRIORITY PRAM. This is the *first* parallel algorithm for MECP that both achieves a constant approximation factor less than 2 and runs in time polylogarithmic in n. Although our algorithm does not run in polylogarithmic time when k is not polylogarithmic in n, we think that the running time of our algorithm is the best possible for the following reason: It is still an open question whether there is an NC approximation algorithm for MECP achieving an $O(1)$ (i.e., an arbitrary constant) approximation factor [2].

Our algorithm can be viewed as a *nontrivial* mixture of Khuller and Raghavachari's sequential approximation algorithm for MECP [7] and Chong and Lam's NC approximation algorithm for 2-MECP [3]. Suppose k is an even integer. Like Khuller and Raghavachari's sequential algorithm, our algorithm starts with an empty subgraph G' and then works in $k/2$ phases: In each phase, the edge-connectivity of G' is increased by 2. However, our algorithm differs from that of Khuller and Raghavachari in how to perform each phase. (*Note:* Each phase of Khuller and Raghavachari's algorithm performs a depth-first search for which there is no known NC algorithm.) Each phase of our algorithm is similar to Chong and Lam's algorithm; in each phase, a *maximal* matching M in $G - G'$ is found and then a maximal

spanning forest containing M together with some other edges in $G - G'$ is added to G'. However, since $G - G'$ may be not 2-edge-connected, we need to modify Chong and Lam's algorithm appropriately. Although our algorithm is a mixture of the two algorithms above, its analysis is completely new. An important part of its analysis is to use the size of the matching M found in each phase to prove a lower bound on the size of the minimum k-edge-connected spanning subgraph of G.

Existing work on MVCP and k-MVCP. MVCP and k-MVCP for $k \geq 2$ are also NP-hard. Cheriyan *et al.* gave efficient sequential and parallel approximation algorithms for MVCP achieving an approximation factor of 2. Khuller and Vishkin were the first to give a polynomial-time approximation algorithm for 2-MVCP that achieves an approximation factor less than 2 [8]. Subsequently, Garg *et al.* gave better approximation algorithms for 2-MVCP [5]. Recently, Chong and Lam presented the first NC approximation algorithm for 2-MVCP that achieves an approximation factor less than 2 [3].

Our result on 3-MVCP. Previous to our work, even for 3-MVCP, there had been no polynomial-time approximation algorithm achieving an approximation factor less than 2. In this paper, we present the first such algorithm for 3-MVCP. Our algorithm runs in linear time and achieves an approximation factor of 1.96. It can also be implemented to run in $O(\log^3 n)$ time with $O(m(n+m))$ processors on a PRIORITY PRAM. The algorithm is based on new properties of scan-first search trees (these trees are introduced by Cheriyan *et al.* in [2]). It also uses Chong and Lam's approximation algorithm for 2-MVCP as a subroutine.

2 An approximation algorithm for MECP

We start by giving several basic definitions. Let $G = (V, E)$ be an undirected (possibly disconnected) graph. For $E' \subseteq E$, we denote the graph (V, E') by $G[E']$. For $V' \subseteq V$, the *subgraph induced by* V' is the graph (V', E') with $E' = \{\{u, v\} \in E \mid \{u, v\} \subseteq V'\}$. For $U \subseteq V$, we denote by $G - U$ the subgraph induced by $V - U$. When U consists of a single vertex u, we write $G - u$ instead of $G - \{u\}$.

A subset M of E is a *matching* in G if no two edges in M have a common endpoint. Let M be a matching. A vertex is *matched* by M if it is incident to an edge in M, and *unmatched* otherwise. An *augmenting path* for M is a simple path in G whose endpoints are both unmatched and whose edges are alternately in $E - M$ and in M. A matching is *maximal* if it is not properly included in any other matching. A matching is *maximum* if it has the maximum cardinality among all matchings in G.

A *bridge* of G is an edge whose removal increases the number of connected components in G. Define an equivalence relation $\equiv$ as follows: For every two vertices u and v of G, $u \equiv v$ if and only if $u = v$ or G contains an edge-disjoint cycle in which both u and v appear. The vertices of G are partitioned into equivalence classes by $\equiv$, and the subgraphs of G induced by these classes are called the *2-edge-connected components* of G. It is clear that no bridge of G appear in a 2-edge-connected component of G. By merging each 2-edge-connected component of G into a super-vertex (with resulting self-loops being deleted), we obtain a forest whose edges are the bridges of G. Consequently, the number of bridges of G is less than the number of 2-edge-connected components of G. (*Note:* Our definitions of bridges and 2-edge-connected components are not standard, as they are usually defined only for connected graphs.)

2.1 The algorithm and its running time

We first give three lemmas that are helpful to understand our algorithm. The first two lemmas are known and their proofs can be found in [7].

Lemma 2.1 Let $G = (V, E)$ be a graph which is at least k-edge-connected. Let E' be a subset of E such that $G[E']$ is $(k-1)$-edge-connected, and let E'' be the edge set of a maximal spanning forest in $G[E - E']$. Then, $G[E' \cup E'']$ is k-edge-connected.

Lemma 2.2 Let $G = (V, E)$ be a graph which is at least k-edge-connected. Let E' be a subset of E such that $G[E']$ is $(k-2)$-edge-connected, and let E'' be the edge set of a maximal spanning forest in $G[E - E']$. Then, every $(k-1)$-cut of $G[E' \cup E'']$ must contain exactly one edge of E''.

Lemma 2.3 Let $G = (V, E)$ be a graph which is at least k-edge-connected, and let E' be a subset of E such that $G[E']$ is $(k-2)$-edge-connected. Let A be a subset of $E - E'$ such that for every edge $e \in E - (E' \cup A)$, $G[A]$ has the same bridges as $G[A \cup \{e\}]$. Then, $G[E' \cup A]$ is k-edge-connected.

We first describe our algorithm for even values of k. Let $G = (V, E)$ be the input k-edge-connected graph. The algorithm starts by setting $E' = \emptyset$. Then, it proceeds in phases. In each phase, some edges in $E - E'$ are added to E' so that the edge-connectivity of $G[E']$ is increased by 2. After $k/2$ phases, the edge-connectivity of $G[E']$ is k and the algorithm outputs $G[E']$.

Below, we describe the procedure used by our algorithm in each phase. It is a modification of Chong and Lam's NC approximation algorithm for 2-MECP, and consists of the following steps.

1. Compute a maximal matching M in $G[E - E']$.

2. Construct a maximal spanning forest $F = (V, E'')$ in $G[E - E']$ with $M \subseteq E''$, and set $A = E''$.

3. In parallel, for each edge $e \in M$, if there is an edge $f \in E - (E' \cup A)$ such that $G[A \cup \{f\}]$ contains a cycle in which both e and f appear, then add one such edge f to A. (*Comment:* e and f belong to the same 2-edge-connected components of $G[A]$ (and of every supergraph of $G[A]$).)

4. Let C_1, ..., C_t be those 2-edge-connected components of $G[A]$ whose vertices are all matched by M. Compute n_j $(1 \leq j \leq t)$, the number of vertices in C_j. (*Comment:* C_j must contain $n_j - 1$ edges of F among which exactly $n_j/2$ ones are edges in M. Thus, by step 3 and the comment after it, C_j contains at most $3n_j/2 - 1$ edges. According to [3], if C_j has exactly $3n_j/2 - 1$ edges, then C_j must contain at least one *redundant edge* whose removal from C_j does not destroy the 2-edge-connectivity of C_j.)

5. In parallel, for each C_j $(1 \leq j \leq t)$ with exactly $3n_j/2 - 1$ edges, find one redundant edge in C_j and remove it from A. (*Comment:* Clearly, $G[A]$ still contains a maximal spanning forest of $G[E - E']$ as a subgraph.)

6. In parallel, for each bridge e in $G[A]$, if there is an edge $f \in E - (E' \cup A)$ such that $G[A \cup \{f\}]$ contains a cycle in which both e and f appear, then add one such edge f to A. (*Comment:* We claim that after step 6, $G[A]$ has the same bridges as $G[A \cup \{g\}]$ for every edge $g \in E - (E' \cup A)$. Assume, on the contrary, that this is not true. Then, by the comment on step 5, there are some bridge e in $G[A]$ and some edge $g \in E - (E' \cup A)$ such that $G[A \cup \{g\}]$ contains a cycle in which both e and g appear. Since no edge added to A in step 6 can be a bridge in $G[A]$, e must have been a bridge in $G[A]$ before step 6. Now, the existence of g implies that some edge f must have been added to A in step 6 so that e can be no longer a bridge in $G[A]$ after step 6, a contradiction.)

7. Compute a maximal spanning forest (V, E_1) of $G[A]$ and add the edges in E_1 to E'. Further compute a maximal spanning forest (V, E_2) of $G[A - E_1]$ and add the edges in E_2 to E'. (*Comment:* Before step 7, the edge-connectivity of $G[E' \cup A]$ is at least 2 larger than that of $G[E']$, by Lemma 2.3 and the comment on step 6. So, by Lemma 2.1, the edge-connectivity of $G[E']$ increases by at least 2 after step 7.)

Now, consider the case where k is odd. In this case, we first use the above procedure repeatedly to find a $(k-1)$-edge-connected spanning subgraph $G[E']$ of G and then add a maximal spanning forest of $G[E - E']$ to $G[E']$ to make $G[E']$ k-edge-connected.

Theorem 2.4 The algorithm is correct and runs in $O(k \log^3 n)$ time with $O(n + m)$ processors. Moreover, it can be turned into an $O(k(n + m))$-time sequential algorithm.

2.2 Performance analysis

Define $k_2 = \lfloor k/2 \rfloor$. For $1 \leq i \leq k_2$, let M_i and A_i be the matching M and the edge set A computed in phase i of the algorithm, respectively. For $1 \leq i \leq k_2$, let $E'_i = \cup_{1 \leq j \leq i} A_j$ and $x_i = |M_i|$. For convenience, let $E'_0 = \emptyset$ and E'_{out} be the output of the algorithm. If k is even, then $E'_{out} \subseteq E'_{k_2}$; otherwise, E'_{out} includes a subset of E'_{k_2}.

Lemma 2.5 If k is even, then $|E'_{out}| \leq \sum_{i=1}^{k_2} (2n - x_i - 2)$. In case k is odd, $|E'_{out}| \leq (n-1) + \sum_{i=1}^{k_2} (2n - x_i - 2)$. Consequently, $|E'_{out}| \leq kn - \sum_{i=1}^{k_2} (x_i + 2)$.

Let Opt be the number of edges in a minimum k-edge-connected spanning subgraph of G.

Fact 1 $Opt \geq \lceil kn/2 \rceil$.

Lemma 2.6 For $1 \leq i \leq k_2$, $Opt \geq (k-2i+2)n - 2(k-2i+2)x_i + 2(i-1)$. ∎

Theorem 2.7 Let R_k be the approximation factor achieved by our algorithm. Then, $R_k < 1.924$ for all $k \geq 2$.

Proof. Let $k_4 = \lfloor k/4 \rfloor$. By Lemma 2.5, Fact 1, and Lemma 2.6, we have

$$\begin{aligned}
R_k &\leq \frac{kn - \sum_{i=1}^{k_2}(x_i+2)}{\max_{1\leq i\leq k_2}\{\lceil kn/2\rceil, (k-2i+2)n - 2(k-2i+2)x_i + 2(i-1)\}} \\
&\leq \frac{kn - \sum_{i=1}^{k_4+1} x_i}{\max_{1\leq i\leq k_4+1}\{\lceil kn/2\rceil, (k-2i+2)n - 2(k-2i+2)x_i\}} \\
&= \frac{(kn - \frac{k_4+1}{2}n) + \sum_{i=1}^{k_4+1}(\frac{1}{2}n - x_i)}{\max_{1\leq i\leq k_4+1}\{\lceil kn/2\rceil, (k-2i+2)n - 2(k-2i+2)x_i\}} \\
&\leq \frac{kn - \frac{k_4+1}{2}n}{\lceil kn/2\rceil} + \frac{\sum_{i=1}^{k_4+1}(\frac{1}{2}n - x_i)}{(k-2i+2)n - 2(k-2i+2)x_i} \\
&\leq 2 - \frac{k_4+1}{k} + \sum_{i=1}^{k_4+1}\frac{1}{2k-4i+4} \qquad (1) \\
&\leq 2 - \frac{k_4+1}{k} + (\frac{1}{2k-4(k_4+1)+4} + \int_1^{k_4+1}\frac{1}{2k-4x+4}dx) \qquad (2) \\
&= 2 - \frac{k_4+1}{k} + \frac{1}{2k-4k_4} + \frac{1}{4}\int_{k-2k_4}^{k}\frac{1}{y}dy \qquad (3) \\
&= 2 - \frac{k_4+1}{k} + \frac{1}{2k-4k_4} + \frac{1}{4}\ln\frac{k}{k-2k_4} \\
&\leq 2 - \frac{1}{4} + \frac{1}{2k-4k_4} + \frac{1}{4}\ln 2 \\
&< 1.9233 + \frac{1}{2k-4k_4} < 1.9233 + \frac{1}{k}. \qquad (4)
\end{aligned}$$

For $k \geq 1429$, $\frac{1}{k} \leq 0.0007$ and $R_k < 1.924$ by (4). For small values of k, we resort to the inequality $R_k \leq 2 - \frac{k_4+1}{k} + \sum_{i=1}^{k_4+1}\frac{1}{2k-4i+4}$ (see (1) above). By executing a computer program for computing the quantity $2 - \frac{k_4+1}{k} + \sum_{i=1}^{k_4+1}\frac{1}{2k-4i+4}$, we can verify that this quantity is less than 1.9232 for all $k < 1429$. Therefore, no matter whether k is large or small, we always have $R_k < 1.924$. ∎

2.3 A variation for large values of k

Throughout this subsection, we assume that the input graph G has no multiple edge. Suppose that we modify the algorithm in subsection 2.1 as follows. In step 1 of *each phase*, we compute a *special* maximal matching M in $G[E-E']$ for which there is no augmenting path with 3 edges. The other steps of each phase remain unchanged.

Lemma 2.8 [6] If S is a matching in an n'-vertex and m'-edge graph without multiple edges such that there is no augmenting path with 3 edges for S, then $|S| \geq \frac{m'}{n'-1}$ and S can be found in linear time.

We adopt the notations and definitions in subsection 2.2. For simplicity, we assume that k is even; similar results can be proved for odd k. Clearly, Lemma 2.5 and Fact 1 still hold. By Lemma 2.8, $x_1 \geq \frac{m}{n}$ and $|A_1| \leq 2n - x_1 - 2 \leq 2n - \frac{m}{n} - 2$. From this and Lemma 2.8, we have $x_2 \geq \frac{m-|A_1|}{n}$ and $|A_2| \leq 2n - x_2 - 2 < 2n - \frac{m}{n}$. Repeating this for all $3 \leq i \leq k_2$, we get $|A_3| < 2n - \frac{m}{n} + 2$, ..., $|A_{k_2}| < 2n - \frac{m}{n} + 2(k_2 - 2)$. Thus, $E'_{out} = \sum_{i=1}^{k_2}|A_i| < 2k_2n - \frac{k_2m}{n} + (k_2-1)(k_2-2)$. Therefore,

$$R_k < \frac{\min\{2k_2n - \frac{k_2m}{n} + (k_2-1)(k_2-2), m\}}{\lceil \frac{kn}{2} \rceil} < \frac{4n+k}{2n+k}.$$

Using this, we can verify that $R_k < 1.841$ if $k \geq 0.38n$. Also, the upper bound on R_k decreases as k increases. In particular, when $k = \frac{n}{2}$, we have $R_k < 1.8$. These approximation factors are better than the previous best 1.85 [7]. It can also be verified that $R_k \leq 1.923$ if $k \geq 0.17n$. This improves on the approximation factor 1.924 shown in subsection 2.2. The modified step 1 can also be done in $O(\log^3 n)$ time with $O(n^6)$ processors [4].

Theorem 2.9 There is an approximation algorithm for MECP which runs in linear time and achieves an approximation factor of $\frac{4n+k}{2n+k}$. In particular, the approximation factor is less than 1.841 for all $k \geq 0.38n$. Moreover, the algorithm can be implemented in $O(\log^3 n)$ time with $O(n^6)$ processors.

2.4 A variation for small values of k

Throughout this subsection, we fix k as a constant. Suppose that we modify the algorithm in subsection 2.1 as follows. Fix an arbitrary constant $\epsilon > 0$. In step 1 of *phase 1*, we compute a matching M in G whose size is at least $(1 - \epsilon)$ optimal, and then extend M to a maximal matching. According to [4], M can be found in polylogarithmic time using a polynomial number of processors. The other steps of phase 1 and all steps of the other phases of the algorithm remain unchanged.

Clearly, the modified algorithm is still correct and runs in polylogarithmic time using a polynomial number of processors (recall that k is fixed). In the remainder of this subsection, we prove that it achieves a better approximation factor than 1.924 if k is sufficiently small.

We adopt the notations and definitions in subsection 2.2. Clearly, Lemma 2.5 and Fact 1 still hold. To obtain a new lower bound on Opt, let x_0 be the size of the maximum matchings in G.

Lemma 2.10 A minimum k-edge-connected spanning subgraph of G must contain at least $kn - 2(k-1)x_0 - 1$ edges.

Therefore, the approximation factor R_k achieved by the modified algorithm satisfies:

$$R_k \leq \frac{kn - \sum_{i=1}^{k_2}(x_i + 2)}{\max_{2 \leq i \leq k_2}\{\lceil kn/2 \rceil, kn - 2(k-1)x_0 - 1, (k-2i+2)n - 2(k-2i+2)x_i\}}.$$

By the modified algorithm, $x_1 \geq (1-\epsilon)x_0$. Using this and modifying the proof of Theorem 2.7, we have

$$\begin{aligned} R_k &\leq \frac{kn - ((1-\epsilon)x_0 + 2) - \sum_{i=2}^{k_2}(x_i + 2)}{\max_{2 \leq i \leq k_2}\{\lceil kn/2 \rceil, kn - 2(k-1)x_0 - 1, (k-2i+2)n - 2(k-2i+2)x_i\}} \\ &\leq 2 - \frac{1-\epsilon}{2(k-1)} - \frac{k_4}{k} + \sum_{i=2}^{k_4+1} \frac{1}{2k-4i+4} \\ &= 2 - \frac{1}{2(k-1)} - \frac{k_4}{k} + \sum_{i=2}^{k_4+1} \frac{1}{2k-4i+4} + \frac{\epsilon}{2(k-1)}. \end{aligned} \tag{5}$$

Since ϵ can be set arbitrarily small as long as it is a constant, (5) can be rewritten as $R_k \leq 2 - \frac{1}{2(k-1)} - \frac{k_4}{k} + \sum_{i=2}^{k_4+1} \frac{1}{2k-4i+4} + \delta$ for any constant $\delta > 0$. Using this, we obtain $R_2 \leq 1.5 + \delta$, $R_3 \leq 1.75 + \delta$, and $R_4 \leq 1.834 + \delta$. To mention more, we have $R_k < 1.9$ for all $k \leq 11$, and $R_k < 1.92$ for all $k \leq 78$.

3 An approximation algorithm for 3-MVCP

In this section, we present an approximation algorithm for 3-MVCP. We will describe it as a *parallel algorithm*; at the end of this section, we will also mention how to turn it into a sequential algorithm. We need the following result proved by Chong and Lam [3].

Lemma 3.1 [3] There is a parallel algorithm such that given a 2-vertex-connected graph $G = (V, E)$ and a matching M in G, finds a 2-vertex-connected spanning subgraph of G with at most $2|V| - \lfloor \frac{|M|}{2} \rfloor - 3$ edges in $O(\log n)$ time using $O(m(n+m))$ processors.

We start by giving several basic definitions. Let $G = (V, E)$ be an undirected (possibly disconnected) graph. For $E' \subseteq E$, the *subgraph induced by* E' is the graph (V', E'), where V' is the set of all endpoints of edges in E'. Define an equivalence relation $\equiv_e$ as follows: For every two edges e_1 and e_2 of G, $e_1 \equiv_e e_2$ if and only if $e_1 = e_2$ or G contains a vertex-disjoint cycle in which both e_1 and e_2 appear. The edges of G are partitioned into equivalence classes by $\equiv_e$. The subgraphs induced by these classes together with the isolated vertices in G are called the *biconnected components* of G. A vertex may be contained in two or more biconnected components and such a vertex is called a *cut point*. (*Note:* Our definitions of biconnected components and cut points are not standard, as they are usually defined only for connected graphs G.)

Lemma 3.2 Let p and q be the numbers of connected components and biconnected components of G, respectively. Then, the sum of the numbers of vertices in the biconnected components of G is $|V| + q - p$.

3.1 Scan-first search

The notion of scan-first search was introduced by Cheriyan *et al.* [2]. For the sake of completeness, we give the definition here. Given a connected graph G and a specified vertex r, a *scan-first search* in G starting at r is a systematic way of marking the vertices. The main marking step is called *scan*: to scan a marked vertex means to mark all previously unmarked neighbors of that vertex. At the beginning of the search, only the specified starting vertex is marked. Then, the search iteratively scans a marked and unscanned vertex until all vertices are scanned.

A scan-first search in G produces a *rooted* spanning tree defined as follows. At the beginning of the search, the tree is empty. Then, for each vertex v in G, when v is scanned, all the edges between v and its previously unmarked neighbors are added to the tree; the edges between v and its previously marked neighbors are not added to the tree.

We next prove several useful properties of scan-first search trees. Some of the properties have been implicitly proved in [2] but our proofs are much simpler. To begin, let $G = (V, E)$ be a connected graph and r be a vertex in G. Fix a scan-first search tree $T = (V, E_T)$ of G rooted at r. We refer to the edges in E_T as *tree edges* and the edges in $E - E_T$ as *nontree edges*.

Fact 2 There is no nontree edge $\{u, v\}$ such that u is an ancestor of v in T.

Lemma 3.3 For every $u \in V$, $T - u$ has at most one connected component C such that there is a nontree edge connecting u to a vertex of C.

Lemma 3.4 Let u_1 and u_2 be two vertices in G such that u_1 is scanned earlier than u_2 by T. Then, $T - \{u_1, u_2\}$ has at most one connected component C such that there is a nontree edge connecting u_1 to a vertex of C.

3.2 The algorithm

We first prove two lemmas which are helpful to understand our algorithm.

Lemma 3.5 Let $G = (V, E)$ be a graph which is at least 2-vertex-connected, and let $T = (V, E')$ be a rooted scan-first search tree of G. Let E'' be the edge set of a maximal spanning forest of $G[E - E']$. Then, $G[E' \cup E'']$ is 2-vertex-connected.

Lemma 3.6 Let $G = (V, E)$ be a graph which is at least 3-vertex-connected, and let $T = (V, E')$ be a rooted scan-first search tree of G. Let A be a subset of $E - E'$ such that for every edge $e \in E - (E' \cup A)$, $G[A]$ has the same number of biconnected components as $G[A \cup \{e\}]$. Then, $G[E' \cup A]$ is 3-vertex-connected.

Let $G = (V, E)$ be a 3-vertex-connected graph. We next use Lemma 3.6 to design an approximation algorithm for computing a small 3-vertex-connected spanning subgraph in G. The algorithm consists of two phases. In phase 1, it computes a rooted scan-first search tree $T = (V, E')$ in G. In phase 2, it computes a subset A of $E - E'$ satisfying the condition in Lemma 3.6. The output of the algorithm is the subgraph $(V, E' \cup A)$. The correctness of the algorithm follows from Lemma 3.6 immediately. We need to specify how to perform phase 2. Phase 2 consists of the following five steps.

1. Compute a maximal matching M in $G[E - E']$.

2. Compute the biconnected components B_1, ..., B_q of $G[E - E']$.

3. In parallel, for each B_j $(1 \leq j \leq q)$, compute the number n_j of vertices in B_j and compute M_j, the set of all edges in both M and B_j.

4. In parallel, for each B_j $(1 \leq j \leq q)$ with at least one edge, use M_j to find a 2-vertex-connected spanning subgraph B'_j of B_j with at most $2n_j - \lfloor \frac{|M_j|}{2} \rfloor - 3$ edges.

5. Set A to be the union of the edge sets of B'_1, ..., B'_q.

We claim that the set A computed in step 5 satisfies the condition in Lemma 3.6. To see this, first note that B'_1, ..., B'_q are the biconnected components of $G[A]$. Fix an arbitrary edge e in $E - (E' \cup A)$. e must belong to some B_j. By step 4, the biconnected components of $G[A \cup \{e\}]$ must be B'_1, ..., B'_{j-1}, B''_j, B'_{j+1}, ..., B_q, where B''_j is obtained from B'_j by adding e to it. Thus, $G[A]$ has the same number of biconnected components as $G[A \cup \{e\}]$. Therefore by Lemma 3.6, $G[E' \cup A]$ is 3-vertex-connected. This shows the correctness of the algorithm.

We next analyze the approximation factor achieved by the algorithm. Let $|M| = x$.

Lemma 3.7 $|E' \cup A| < 3n - \frac{x}{2}$.

Proof. Without loss of generality, we may assume that B_1, ..., $B_{q'}$ are the biconnected components of $G[E - E']$ that are not isolated vertices. Then, $|A| \leq \sum_{j=1}^{q'} (2n_j - \lfloor \frac{|M_j|}{2} \rfloor - 3)$. Using $\sum_{j=1}^{q'} |M_j| = x$, we get $|A| \leq 2\sum_{j=1}^{q'} n_j - \frac{x}{2} - \frac{5}{2}q'$. The total number of vertices in B_1, ..., B_q is $\sum_{j=1}^{q} n_j = \sum_{j=1}^{q'} n_j + (q - q')$, which is at most $n + q - 1$ by Lemma 3.2. Thus, $|A| \leq 2(n + q' - 1) - \frac{x}{2} - \frac{5}{2}q' < 2n - \frac{x}{2}$. Therefore, $|E' \cup A| = (n - 1) + |A| < 3n - \frac{x}{2}$. ∎

Lemma 3.8 Let Opt be the number of edges in a minimum 3-vertex-connected spanning subgraph of G. Then, $Opt > 2n - 4x$.

By Lemma 3.7 and Lemma 3.8, we see that the approximation factor achieved by the algorithm is at most $\frac{3n - x/2}{\max\{1.5n, 2n - 4x\}} \leq 2 - 1/24 < 1.96$.

Theorem 3.9 There is an NC approximation algorithm for 3-MVCP that achieves an approximation factor of 1.96 and runs in $O(\log^3 n)$ time with $O(m(n+m))$ processors. Moreover, it can be turned into a linear-time sequential algorithm.

References

[1] J. Cheriyan and R. Thurimella, Algorithms for Parallel k-Vertex Connectivity and ..., *STOC'91*.

[2] J. Cheriyan, *et al.*, Algorithms for Parallel k-Vertex Connectivity and ..., *SIAMJC* 22 (1993).

[3] K.W. Chong and T.W. Lam, Approximating Biconnectivity in Parallel, *SPAA'95*.

[4] T. Fischer, *et al.*, Approximating Matchings in Parallel, *IPL* 46 (1993).

[5] N. Garg, *et al.*, Improved Approximation Algorithms for Biconnected Subgraphs ..., *SODA'93*.

[6] Y. Kajitani, *et al.*, New Approximation Results on Graph Matching and Related Problems, *WG'94*.

[7] S. Khuller and B. Raghavachari, Improved Approximation Algorithms for Uniform Connectivity Problems, *STOC'95*.

[8] S. Khuller and U. Vishkin, Biconnectivity Approximations and Graph Carvings, *JACM* 41 (1994).