On Parallelizability of α-Connectivity*

3T−9

Chuzo IWAMOTO *and* Kazuo IWAMA
Department of Computer Science and Communication Engineering
Kyushu University, Fukuoka 812, Japan

1 Introduction

Consider the common algorithm [5] obtaining connected components of undirected graphs. As some connected component U grows, a nearby vertex v is absorbed into U if v is connected to U, i.e., if an edge exists between v and some vertex in U. An obvious problem in this connectivity is that it does not reflect the strength of the connection. This naturally leads us to another graph connectivity which we call *α-connectivity*. Unlike the conventional connectivity, α-connectivity requires that the vertex v should have at least $\lceil d(v)\cdot\alpha/n\rceil$ neighbors in U, where $d(v)$ denotes the degree of vertex v. (Suppose for example that $n = 100$ and $\alpha = 10$. Then v can join U only if at least one tenth of v's neighbors are included in U.) One can imagine the force of gravity between a star and a meteorite and a certain kind of clustering problems for its application. When $\alpha = 1$, α-connectivity is the same as the conventional connectivity.

In this paper, we discuss the parallel complexity of this α-connectivity problem (obtaining α-connected components). Our result is remarkable in that the complexity gradually increases as the value of α grows: (i) α-connectivity is in NC^t when $\alpha = c(\log n)^{\frac{t-2}{2}}$ for positive constants c and $t \geq 2$. (ii) The complexity jumps if the value α becomes a bit larger: It is P-complete when α is given by two positive constants c and (any small) ϵ as $\alpha = cn^{\epsilon}$.

In the field of P and NP, "gradually intractable problems" whose complexity changes from P to NP-complete are not rare. Most problems prefixed by "k–", such as k-clique, fall into this category. However, "gradually unparallelizable problems" *are* rare. As the gradually intractable problems nicely exhibit fundamental differences between P and NP-complete, α-connectivity is expected to play the same role inside P.

Obviously, k-connectivity, which is also a generalization of graph connectivity, is related to our α-connectivity. [2] shows that k-connectivity can be solved in $O(k^2 \log n)$ time by a polynomial number of processors. Thus the bound continuously grows with k as α-connectivity, but it is open whether it finally becomes P-complete. [1] gives the partial negative answer: Constructing a maximum s-t flow in a directed graph whose edge weights are given in unary is in RNC; namely, k-connectivity for an undirected graphs is in RNC. Probably, α-connectivity provides the first example of gradually increasing complexities from NC to P-complete.

2 α-Connectivity

Let $G = (V, E)$ be a graph of n vertices and m edges. We denote by $d(v)$ the degree of vertex v. Let v_1 be a vertex in V and $V_1 \subseteq V$ be a vertex set which does not contain v_1. v_1 is said to be *α-close* to V_1 if at least $\lceil d(v_1)\cdot\alpha/n\rceil$ edges out of v_1's edges go to vertices in V_1. A vertex set $V_i \subseteq V$ is said to be *α-connected* if (i) $|V_i| = 1$ or (ii) there are α-connected sets V_i' and V_i'' such that $V_i = V_i' \cup V_i''$ and V_i' contains a vertex α-close to V_i''. A vertex set $V_k \subseteq V$ is said to be an *α-connected component* if (i) V_k is α-connected and (ii) no vertex outside V_k is α-close to V_k. The α-connectivity problem requires us to partition V into α-connected components. Although details are omitted, we can show that the partition into α-connected components is unique. Note that the definition of α-connectivity when $\alpha = 1$ is the same as that of graph connectivity because $\lceil d(v_1)\cdot\alpha/n\rceil = 1$.

Theorem 1. *α-connectivity can be solved in $O(\alpha^2 \log n)$ time by a polynomial number of CRCW PRAM processors.*

It is well known that a problem is in NC^{t+1} if it is solved in time $O(\log^t n)$ by a polynomial number of CRCW processors. Therefore:

Corollary 1. *α-connectivity is in NC^t when $\alpha = c(\log n)^{\frac{t-2}{2}}$ for positive constants c and $t \geq 2$.*

Theorem 2. *α-connectivity is P-complete when α is given by two constants c and ϵ as $\alpha = cn^{\epsilon}$, where c may be any positive value and $0 < \epsilon < 1/2$.*

Remark 1. Why does the complexity of

*This work was done while the first author was visiting McGill University. This work was in part supported by Grant in Aid Scientific Research for the Ministry of Education, Science and Culture of Japan, No. 1149.

α-connectivity change like this? Suppose that the value of α is larger than one (more precisely larger than $n/(n-1)$). Then, vertices of the complete graph K_n cannot be merged at all, since $\lceil d(v)\cdot\alpha/n\rceil > 1$. In terms of the space model mentioned previously, that means the forces between the objects are completely balanced (and hence each cannot join another). In other words, the joining process of α-connectivity makes full use of "unbalanced" structure of the graph. If α is large, the process depends on more unbalanced structure, which one can imagine finally leads to the inherent sequentiality causing P-completeness.

3 Parallel Algorithms

For simplicity, we assume that there are no vertices of degree 0. α-connectivity can be solved by the following parallel algorithm:

(1) Start with n one-vertex sets, $V_1 = \{v_1\}$, $V_2 = \{v_2\}, \ldots, V_n = \{v_n\}$.

(2) Assume that there are currently p sets, $V_1, V_2, \ldots, V_p$. Construct a graph $G' = (V', E')$ such that $V' = \{V_1, V_2, \ldots, V_p\}$ and $(V_i, V_j) \in E'$ if one of the two sets V_i and V_j contains a vertex v_1 α-close to the other set.

(3) Merge all vertices of every connected component of G' into a single set.

(4) Repeat (2) and (3) until $E' = \emptyset$.

Using $d(v)$ processors, we can decide whether V_i (resp. V_j) containing v can be merged with V_j (resp. V_i) in $O(\log n)$ time. In step (2), G' is thus constructed in $O(\log n)$ time by $2mn$ processors. In step (3), we use the parallel graph connectivity algorithm [4] which runs in $O(\log n)$ time using $n + 2m$ processors. Due to space limitations, we omit the proof of the following lemma.

Lemma 1. *The number of iterations is* $O(\alpha^2)$.

4 P-Completeness

Since α-connectivity can be solved by a sequential version of the algorithm in Section 3, α-connectivity is in P. In the following, only a brief overview of the reduction will be presented. We show that the solvable path system problem (SPS) [3] is log-space reducible to α-connectivity when $\alpha = cn^\epsilon$. The instance of SPS is $Q = (X, R, A)$, called a *path system*, where X is a finite set of *sentences*, R is a set of *inference rules*, $A \subseteq X$ is a set of *axioms*. If $(a, b, c) \in R$, then $a \neq b$, $b \neq c$, and $c \neq a$. Let S be the least subset of X such that (i) $A \subseteq S$ and (ii) if $a, b \in S$ and $(a, b, c) \in R$ then $c \in S$. SPS asks whether $S = X$.

We reduce $Q = (X, R, S)$ to $G = (V, E)$. We first define a vertex set as $V_X = \{v_a | a \in X\}$. We then replace all sentences in X by the vertices in V_X. For each rule $(a, b, c) \in R$, we apply the following reduction. Let $m = |R|$. Two sets of vertices are introduced as $V^x_{abc} = \{x^1_{abc}, \ldots, x^{k\delta-1}_{abc}\}$ and $V^y_{abc} = \{y^1_{abc}, \ldots, y^{k\delta-2}_{abc}\}$, where $\delta = \lfloor m^{(1-\epsilon)/\epsilon} \rfloor$ and k is a sufficiently large constant. Then we connect $\{v_a, v_b\}$ with V^x_{abc} by $2\times|V^x_{abc}|$ edges. Similarly, we connect V^x_{abc} with V^y_{abc} and then connect V^y_{abc} with $\{v_c\}$. By replacing all the inference rules with the vertices and the edges mentioned above, we get a graph, say G_1.

Now we add new vertices (divided into four families F_0 through F_3) to G_1. F_0 includes a single element, i.e., $F_0 = \{s\}$. F_1 and F_2 are two $(\delta-1)$-vertex sets and F_3 is a $(k\delta - 1)$-vertex set. We connect F_0 with F_1, F_1 with F_2, and F_2 with F_3. For every axiom $d \in A$, we then connect F_3 with $\{v_d\}$. We denote the resulting graph by G_2.

For some technical reason, we wish to construct G such that the values n and k satisfy $1/k\delta < cn^\epsilon/n \leq 2/k\delta$. Our strategy for obtaining such G is as follows: We first construct a graph G_3 such that n and k satisfy $2/k\delta \leq cn^\epsilon/n$. Such G_3 can be obtained by constructing G_2 whose k is sufficiently large. Then we add a new vertex z_1 and edge (s, z_1) to G_3, and then add z_2 and (z_1, z_2), and so on, up until the number of vertices satisfies $1/k\delta < cn^\epsilon/n \leq 2/k\delta$.

Although details are omitted, G can be constructed by a log-space program. The proof of the following lemma is omitted.

Lemma 2. *All the vertex of* G *can be merged in a single set if and only if all the sentences are provable.*

Acknowledgment. We would like to thank David Avis for his helpful comments.

References

[1] R. Karp, E. Upfal, and A. Wigderson, "Constructing a perfect matching is in Random NC", *Combinatorica* **6** (1) (1986) 35-48.

[2] S. Khuller and B. Schieber, "Efficient parallel algorithms for testing k-connectivity and finding disjoint s-t paths in graphs", *SIAM J. Comput.* **20** 2 (1991) 352-375.

[3] S. Miyano, S. Shiraishi, and T. Shoudai, "A list of P-complete problems", Tech. Rept. RIFIS-TR-CS-17, Research Institute of Fundamental Information Science, Kyushu University, 1990.

[4] Y. Shiloach and U. Vishkin, "An $O(\log n)$ parallel connectivity algorithm", *J. Algorithms* **3** (1982) 57-67.

[5] R. Tarjan, "Depth-first search and linear graph algorithms", *SIAM J. Comput.* **1** (1972) 146-160.