South African
Computer
Journal
Number 12
November 1994

Suid-Afrikaanse
Rekenaar-
tydskrif
Nommer 12
November 1994

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Rekenaarvereniging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083
Email: dkourie@dos-lan.cs.up.ac.za

Subeditor: Information Systems
Prof Lucas Introna
Department of Informatics
University of Pretoria
Hatfield 0083
Email: lintrona@econ.up.ac.za

Production Editor
Dr Riël Smit
Mosaic Software (Pty) Ltd
P.O.Box 23906
Claremont 7735
Email: gds@mosaic.co.za

Subscriptions

	Annual	Single copy
Southern Africa:	R50,00	R25,00
Elsewhere:	$30,00	$15,00

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276

CALL FOR CONTRIBUTIONS

South African Computer Journal (SACJ): Special Issue

Information Technology and Development

South Africa, as a developing country, needs to find ways to harness it's enormous potential in a rapid and sustainable way. It is argued by some that the developments in information technology are increasingly creating opportunities for socio-economic development to be enabled by the diffusion of IT. If this is true, then these opportunities should surely be explored. Consequently, the editorial board of SACJ has decided to devote a special issue to this theme.

Some of the questions that the special issue might address are:

- To what degree can information technology enable (or accelerate) socio-economic development?
- How can the diffusion of information technology be achieved in order to maximize its contribution towards socio-economic development?
- What are the conditions required for such technology enabled development?
- Are there high leverage areas where quick returns can be achieved?
- Should the state play an active role and in what way?
- What are the experiences of other developing countries in information technology enabled socio-economic development?
- What are the moral and ethical issues involved in information technology enabled socio-economics development?

We invite *all* researchers in the field of information technology and other disciplines to contribute to this special issue. All contributions will be reviewed by three independent reviewers. Contributions should be sent in four copies to:

Prof Lucas D. Introna
South African Computer Journal (SACJ)
Special Issue: IT and Development
Dept. of Informatics
University of Pretoria
Pretoria, 0002
SOUTH AFRICA

Contributions should be received by June 5, 1995. Contributions sent by facsimile or e-mail will not be accepted. For any information regarding the special issue contact Prof Lucas Introna at +2712 4203376 (office), +2712 434501 (fax) or Internet: lintrona@econ.up.ac.za.

Important Dates

Contribution deadline:	June 5, 1995
Notification to author(s):	September 1, 1995
Final versions due:	November 1, 1995

LEXICA

an integrated environment for machine translation

LEXICA's capabilities include :

- **Translator's Workbench**
- **Powerful word-based Post Editor**
- **Automatic translator**
- **Integrated on-line dictionaries**
- **Support for most Windows-based wordprocessors**

LEXICA is a language independent platform

New language pairs can be developed on request

LEXICA provides unique support for languages spoken on the African continent, including :

- **Swahili**
- **Tswana**
- **Zulu**
- **Afrikaans**
- **Portuguese**

as well as :

- **German and French**

Eenheid vir Programmatuur-Ingenieurswese
Unit for Software Engineering

in association with
Department of Computer Science
University of Pretoria, Pretoria 0002 South Africa

Telephone: (012) 420-2504
Fax: 43-6454
E-mail: lexica@cs.up.ac.za

A Linear Time Algorithm for the Longest (s-t)–path Problem Restricted to Partial k-trees.

Manrique Mata-Montero *† John A. Ellis‡

†*Department of Computer Science, Memorial University of Newfoundland, NFLD, Canada A1C 5S7.*
email:manrique@cs.mun.ca

‡*Department of Computer Science, University of Victoria, Victoria, BC, Canada.*

Abstract

The Longest (s-t)–path Problem, a known NP*-complete set, is shown to admit a linear time solution when the instances of the problem are restricted to partial k-trees. This class of graphs is defined and some of the properties of partial k-trees, those needed for our algorithm are proved.*

Keywords: *bounded treewidth, partial k-trees, longest path, linear time algorithms.*
Computing Review Categories: *F2.2.*

1 Introduction

The LONGEST (S-T)-PATH PROBLEM is a known *NP*-complete problem. Ellis *et al.* showed a linear time solution for it when the instances of the problem are restricted to outerplanar graphs [7]. Table 1 indicates that outerplanar graphs are partial 2-trees or, equivalently, outerplanar graphs have treewidth bounded by 2, since a graph G is a partial k-tree if and only if G has treewidth bounded by k [14]. In this paper, we show that the LONGEST (S-T)-PATH PROBLEM admits a linear time solution even when the instances are relaxed to partial k-trees, for any fixed value of k.

We start by defining the class of k-trees, partial k-trees and describing some relationships between k-trees and other classes of graphs. The class of k-trees is a generalization of the class of trees, which are themselves 1-trees. In general, a k-tree is built from the complete graph on k vertices by repeated addition of vertices, each connected with k edges to a complete subgraph on k vertices. A partial k-tree is a k-tree from which some edges have been deleted.

It has been shown that deciding whether a graph G is a partial k-tree, given G and k as the input, is *NP*-complete [1]. Additionally, they show that for a fixed k, testing whether a graph G is a partial k-tree can be done in polynomial time. Their algorithm outputs the 'k-tree reduction of G'. It follows from the theory developed by Robertson and Seymour that *there exists* an algorithm with time complexity in $O(n^2)$ for testing membership in the family of partial k-trees for any fixed k [9]. An algorithm with this time complexity has been announced [5].

Every graph is a partial k-tree for some k. So, for those *NP*-complete problems that have polynomial time solutions, when restricted to partial k-trees, for some fixed k, either the constants involved are exponential in k or the algorithms are not polynomialy uniform. If those constants were bounded by a polynomial in k and the algorithms were polynomialy uniform, then $NP = P$, which is unlikely.

Some authors have suggested that the minimum k for which a graph G is a partial k-tree, $k(G)$, is a 'good' measure of the algorithmic properties of the members of the family of such graphs [2]. However, one notes that there are *NP*-complete problems that have computationally easy solutions when restricted to families that do not have bounded $k(G)$, for instance the CHROMATIC NUMBER PROBLEM restricted to cographs [2], and that there are problems that remain *NP*-complete, even for 1-trees, for instance the BANDWIDTH MINIMIZATION PROBLEM [12]. The class of partial k-trees is rich. Table 1 from [4], [14] and [8] shows the relationship between some well known families of graphs with families of partial k-trees, for some fixed k.

2 The Class of k-trees

In this section we define partial k-trees, tree decomposition and related concepts. In what follows k is always a positive integer, and graphs are always finite.

Definition 1 *Let $G = (V, E)$ be a graph. A set $K \subseteq V$, where $|K| = k$, is a* **k-clique** *if K induces a complete graph.*

We give two equivalent definitions of k-trees, of which the second is usually the most convenient for our purposes.

Definition 2 *A* **k-tree** *is:*

- *a complete graph on k vertices, or*
- *the graph $(V \cup \{v\}, E \cup \{\{u_1, v\}, \ldots, \{u_k, v\}\})$ obtained from a k-tree $T = (V, E)$, where $\{u_1, \ldots, u_k\} \subseteq V$, is a k-clique in T and $v \notin V$.*
- *k-trees are the graphs obtained by application of the previous two rules only.*

We denote the **open neighbourhood** of a vertex v; i.e., the set of vertices adjacent to v, by $adj(v)$. The **closed neighbourhood** of a vertex v is $adj(v) \cup \{v\}$.

Definition 3 *A graph $T = (V, E)$ is a* **k-tree** *if:*

- *V is a k-clique or*

*This author's research was supported in part by Department of Computer Science, University of Natal–Pietermaritzburg, Natal, Republic of South Africa, and Memorial University of Newfoundland.

Table 1. Classes of Graphs and Maximum $k(G)$ for Graphs in the Class

Class of graphs	Upper-bound for $k(G)$.
$(k,2)$-partite graphs	$2k-1$
Forests	1
Almost trees with parameter k	$k+1$
Graphs with bandwidth$\leq k$	k
Graphs with cyclic bandwidth$\leq k$	$2k$
Graphs with cutwidth$\leq k$	k
Series-parallel graphs	2
Outerplanar graphs	2
Halin graphs	3
k-outerplanar graphs	$3k-1$
Chordal graphs with max. clique size k	$k-1$
Undirected path graphs with max. clique size k	$k-1$
Directed path graphs with max clique size k	$k-1$
Interval graphs with max. clique size k	$k-1$
Proper interval graphs with max. clique size k	$k-1$
Circular arc graphs with max clique size k	$2k-1$
Proper circular arc graphs with max. clique size k	$2k-2$
k-bounded tree-partite graphs	$2k-1$

- *there exists $v \in V$ such that $degree(v) = k$, $adj(v)$ induces a k-clique and*
  $T' = (V \setminus \{v\}, E \setminus \{\{x,v\} \mid \{x,v\} \in E\}$ is a k-tree.

The process given in Figure 1 shows the construction of a k-tree according to Definition 2. Given the last graph in the sequence, it can be shown, using Definition 3, that it is a k-tree by reversing the steps shown in the sequence.

The characterization of k-trees given in Definition 3 can be used to define a reduction process, since it follows from Definition 3 that a graph T is a k-tree iff its vertices can be removed in rounds, where a vertex can be removed at a given round if it has degree k and it is connected to a k-clique, until a complete graph on k vertices remains [3]. This remaining k-clique is called the **root** of the k-tree, which we denote $root(T)$. The vertices eligible for removal at each round are **k-leaves** of intermediate k-trees. This reduction defines a precedence relation between k-cliques and vertices. These concepts are formalized in the following definitions.

The following definition is based on the characterization of chordal graphs given by Rose, Tarjan, Leuber and Yannakakis [13].

Definition 4 *Let G be a graph. A vertex $v \in V$ is* **simplicial** *if its closed neighbourhood is a complete graph.*

The following definition of reduction is from [11].

Definition 5 *A* **reduction** *of a k-tree $T=(V,E)$, where $|V| = n+1$, is a partial sequence $r_0, \ldots, r_{m=n-k}$ of an ordering $r_0, \ldots, r_n$ of V such that for all $i \in \{0, \ldots, m\}$, vertex r_i is simplicial in the graph induced by $\{r_i, \ldots, r_n\}$, and its closed neighbourhood has cardinality $k+1$.*

See Figure 1 for an example of a reduction of a 3-tree. Notice that in a k-tree T, any vertex of degree k, i.e., k-leaf, has a k-clique as its open neighbourhood, and that the k-clique corresponding to the open neighbourhood of r_m is the root of T.

Given a particular reduction we can define partial orderings on the vertices with respect to the k-cliques in T, so that k-leaves have no descendants and every vertex not in the root of T is a descendant of the root. Below we define one such ordering.

Definition 6 *Let $T = (V,E)$ be a k-tree, $r_0, \ldots, r_m$, be a reduction of T and $|V| = n+1$. A vertex r_i, where $0 \leq i \leq m$, is a* **descendant** *of a k-clique K iff $r_i \notin K$ and r_i, in the graph induced by $\{r_i, \ldots, r_n\}$, has open neighbourhood K or its open neighbourhood is K' and all the vertices in K' not in K are descendants of K.*

Definition 7 *Given a k-tree T and a reduction R, we say that T is* **ordered** *according to the ordering given in Definition 6.*

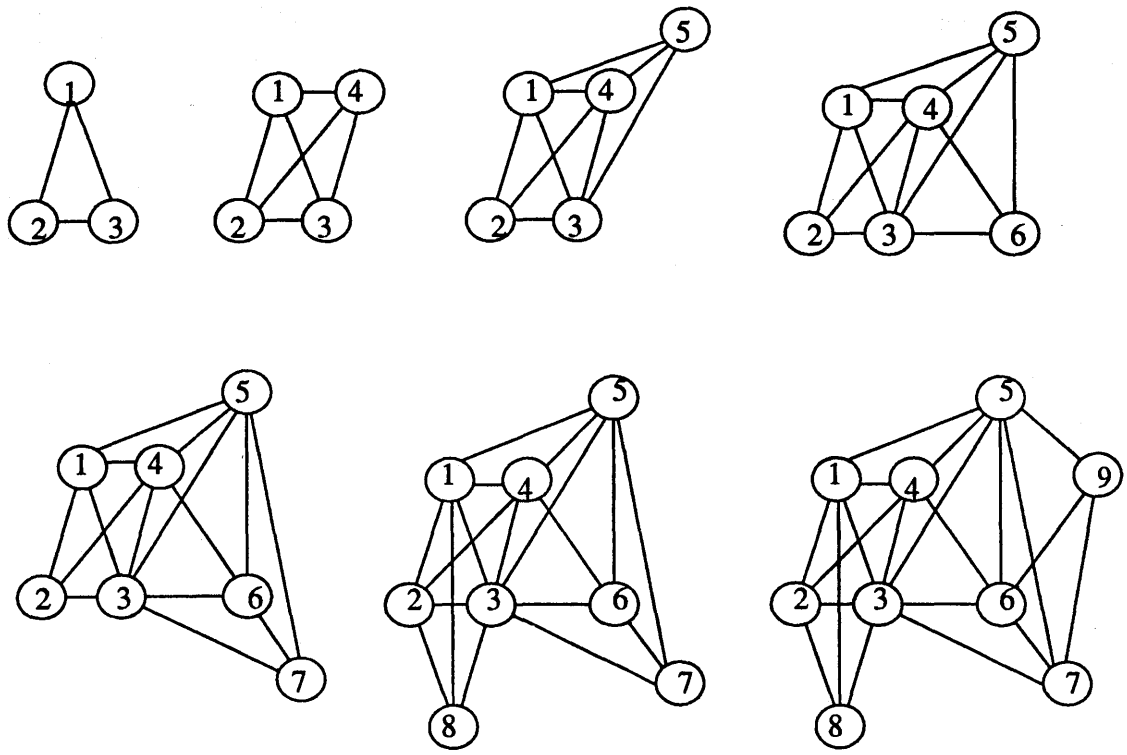
We are interested in the wider class of partial k-trees. We define this class below.

Definition 8 *A* **partial** *k-tree of a k-tree $T = (V,E)$ is a graph $G = (V,E')$ where $E' \subseteq E$; i.e., G is obtained from a k-tree by deleting some of its edges.*

It is clear that if we are given a k-tree T of which G is a partial k-tree, then we can find, in linear time, a reduction for G. This reduction corresponds to the sequence obtained from deleting k-leaves from T until what remains is a k-clique. Likewise, if we have a partial k-tree G and a reduction for it, we can find, in linear time, a k-tree of which G is a a partial k-tree. Notice that this k-tree is not necessarily unique. We can also transfer the ordering of the k-tree T to the partial k-tree G in the natural way.

Definition 9 *Let K be a k-clique in the k-tree T and R a reduction of T. The connected components induced by the descendants of K, $desc(K)$, are the* **branches** *on K, and K is the* **base** *of a branch on K.*

We will use the notation $r \in R$, where R is a sequence, to refer to the fact that r is an element of the sequence



Reductions: 9, 8, 7, 6, 5, 4 or 9, 7, 6, 5, 4, 8

Figure 1. Example of the Generation of a 3-tree and Two of its Reductions

R. The context in which such expressions appear should indicate without ambiguity its meaning.

Definition 10 *Let T be a k-tree, $R = r_0, \ldots, r_m$ be a reduction of it and $|V| = n + 1$. For all $r_i \in R$:*

- *$K(r_i)$ or simply K, when r_i is understood, denotes the open neighbourhood of r_i in the graph induced by $\{r_i, \ldots, r_n\}$,*
- *$K'(r_i)$ is the closed neighbourhood of r_i, i.e., $K(r_i) \cup \{r_i\}$,*
- *$K^u = K' - \{u\}$, for all $u \in K'$ and*
- *$B(K(r_i))$ is the set of vertices u descendants of $K(r_i)$ such that $u = v_j \in R$ and $j < i$.*

Notice that K' induces a $k+1$-clique; i.e., $k+1$ overlapping k-cliques. The k-clique $K(r_i)$ is a node separator in T and no node separator of T has cardinality greater than k [2]. Recall that a set of vertices K is a $u - v$ node separator, where $u, v \in V(T) \setminus K$, if every $(u-v)$-path in T contains at least one vertex in K.

We show two properties of partial k-trees relevant to the arguments for the correctness of our algorithm.

Lemma 1 *Let K be a k-clique in a k-tree T. The graph T' induced by $K \cup V(C_i) \cup \ldots \cup V(C_j)$, where $C_i, \ldots, C_j$ are branches on K, is a k-tree.*

Proof. Follows immediately from Definitions 3, 6 and 9. □

Lemma 2 *If a k-clique K is a node separator in a k-tree T, $R = r_0, \ldots, r_m$ is a reduction of T, and w is a descendant of K such that $K = K(w)$, then for all $u, v \in K'(w)$, $desc(K^u) \cap desc(K^v) = \emptyset$, where $u \neq v$; i.e., descendants of the $k+1$ k-cliques in K' are pairwise disjoint.*

Proof. The proof proceeds by induction on m, the number of nodes in a reduction of T.

For the case $m = 0$ the lemma follows trivially.

Let $m > 0$, $x = r_0$. Suppose, there exists $y \in R$ such that $I = desc(K^u) \cap desc(K^v) \neq \emptyset$, where $u, v \in K'(y)$, and $u \neq v$.

By hypothesis, the Lemma holds for the tree induced by $V(T) \setminus \{x\}$,. Adding x to the tree induced by $V(T) \setminus \{x\}$; i.e., connecting x to $K(x)$, does not change the descendants of any k-clique other than $K(x)$. So, $K(x)$ is K^u or K^v, otherwise, in the tree induced by $V(T) \setminus \{x\}$, there would be a $z \in K(x)$ such that $z \in I$, which contradicts the hypothesis.

If $K(x) = K^u$ and x is a descendant of both K^u and K^v then $degree(x) > k$. But $degree(x)$ can not be more than k, since x was the last node added to T. We reach the same conclusion if $K(x) = K^v$. □

3 The Longest (s-t)–path Problem

In this section we show that, for fixed k, the LONGEST (S-T)–PATH PROBLEM admits a linear time solution when the input is restricted to partial k-trees, together with their corresponding reductions.

The LONGEST (S-T)–PATH PROBLEM (LP) is a basic NP-complete problem [10]. It remains NP-complete even when the length of the edges is 1 [10]. The LP

problem is defined below.

THE LONGEST (S-T)–PATH PROBLEM (LP)

Instance: Graph $G = (V, E)$, for all $e \in E$, lengths $l(e) \in \mathcal{Z}^+$, a lenght L, and specified vertices $s, t \in V$.
Question: Is there a simple path in G from s to t of length L or more; i.e., whose edge lengths sum to at least L?

A set $P \subseteq V$ is a **simple path** in the graph G if and only if $|P| \leq 1$ or $P = \{p_1, \ldots, p_l\}$, where for all $i, j \in \{1, \ldots, l\}$, $p_i \neq p_j$, whenever $i \neq j$, and there are edges $\{p_i, p_{i+1}\}$, where $1 \leq i < l$.

The notation that will be adopted, unless it is explicitly said otherwise is:

- $R = r_0, \ldots, r_m$ is a reduction of T,
- for all k-cliques K in T, $desc(K)$ denote the descendants of K with respect to a reduction R.
- if P is a path, $end(P)$ is the set containing its endpoints.

Definition 11 *Given an ordered k-tree T and its reduction R, two simple paths P and P' are* **disjoint with respect to a k-clique** *K, if they don't have any nodes in common in the graph induced by $desc(K)$.*

Notice that two disjoint simple paths (dsp) with respect to a k-clique K may share nodes in K.

An inductive argument shows that if $P = \{v_1, \ldots, v_l\}$ is an *(s-t)*-path of maximum length in the partial k-tree G, there is at least a node r in the reduction of G such that $desc(K(r)) \cap \{v_1, \ldots, v_l\} \neq \emptyset$ and P includes at least one node in $K'(r)$, or path P is fully contained in the root of T.

The following lemma shows that *(s-t)*-paths have optimal substructure.

Lemma 3 *Let r be in a reduction of a partial k-tree G, and $P = \{v_1, \ldots, v_l\}$ be an* (s-t)*-path of maximum length in the partial k-tree G, not fully contained in $root(T)$, and including nodes in $K'(r)$, and in $desc(K(r))$.*

- *If P includes nodes in $K(r)$, let paths $\{P_1, \ldots, P_m\}$ be the set of maximal subpaths of P contained in the graph induced by $desc(K) \cup K$ with no nodes in K except at least one of their endpoints, or*
- *if P does not include any nodes of $K(r)$, but includes r, let path P consist of disjoint path segments $\{v_1, \ldots, v_{m-1}, r\}$ and $\{r, v_{m+1}, \ldots, v_l\}$, for some $1 \leq m \leq l$, belonging to graphs induced by $desc(K^u) \cup K^u$ and $desc(K^v) \cup K^v$, respectively, for some $u, v \in K$.*

Path segments $\{P_1, \ldots, P_m\}$ have maximum accumulated length among all sets of dsp in graph $K \cup desc(K)$, or the two path segments $\{v_1, \ldots, v_{m-1}, r\}$ and $\{r, v_{m+1}, \ldots, v_l\}$ have maximum accumulated length among all path segments in the graph $(desc(K^u) \cup K^u) \cup (desc(K^v) \cup K^v)$ having the same endpoints.

Proof. If the path segments don't have maximum accumulated length among all the sets of dsp in their respective subgraphs having the same endpoints, then by replacing a set of dsp having the same points and greater accumulated length, it is possible to build another *(s-t)*-path of length greater that that of P. This contradicts the hypothesis concerning the maximality of P. So, the Lemma holds. □

Lemma 3 asserts that the *LP* Problem exhibits optimal substructure [6]. Using this fact we will use a dynamic programming approach to solve the problem.

Definition 12 *Given an ordered k-tree T and two distinguished nodes s and t, the set $\mathcal{P}(K(r))$, where r is in the reduction of T, includes all sets of disjoint simple paths (dsp) with respect to k-clique K in the graph induced by $K \cup B(K(r))$, where paths in each set of dsp:*

- *do not induce cycles, and*
- *do not include any nodes in K, but possibly their endpoints, of which at least one must be in K, and*
- *if they have only one endpoint in K, then their other endpoint is in $\{s, t\}$, and*
- *if they share endpoints, they share at most one, the shared endpoint is in K and is not s nor t, and it is shared by at most two paths.*

Notice that a longest *(s-t)*-path fully contained in $root(T)$ is not in any set member of $\mathcal{P}(K)$, for any k-clique K, for all $k > 2$.

For each set of dsp, it will be sufficient for our purpose to keep track of the endpoints of each of its simple paths. So, for each set of dsp $\{P_1, \ldots, P_m\} \in \mathcal{P}(K)$, define $\{\hat{P}_1, \ldots, \hat{P}_m\}$, where $\hat{P}_i = [X_i, Z_i]$, for all $i \in \{1, \ldots, m\}$, such that

$$X_i = end(P_i) \cap K,$$

and

$$Z_i = end(P_i) \cap \{s, t\}.$$

Clearly, there may be more than one set of dsp yielding the same set of $\hat{P}_i$'s, i.e., including nodes of $K \cup \{s, t\}$ in the 'same manner'. For our purpose, it suffices to keep track of the set of dsp whose accumulated length is maximum, since solutions to the problem have optimal substructure. We formalize these concepts below.

Definition 13 *We say that two elements of $\mathcal{P}(K)$, for any particular K, say $\{P'_1, \ldots, P'_m\}$ and $\{P_1, \ldots, P_m\}$, use the nodes of K in the* **same manner** *if and only if for all $i \in \{1, \ldots, m\}$*

$$\hat{P}'_i = \hat{P}_i$$

Note that *same manner* is an equivalence relation. An equivalence class containing $\{P_1, \ldots, P_m\}$ can be represented (indexed) by the set $\{\hat{P}_1, \ldots \hat{P}_m\}$; i.e., the index $\{\hat{P}_1, \ldots \hat{P}_m\}$, represents all the sets of dsp in $\mathcal{P}(K)$, with the same cardinality, and having identical endpoints.

Lemma 4 *Let r be in a reduction of a partial k-tree G, and $P = \{v_1, \ldots, v_l\}$ be an* (s-t)*-path of maximum length in the partial k-tree G, not fully included in $root(K)$, and including nodes in $K'(r)$, and $desc(K(r))$.*

- *If P includes nodes in $K(r)$, let paths $\{P_1, \ldots, P_m\}$ be the set of maximal subpaths of P contained in the graph induced by $desc(K) \cup K$ with no nodes in K except at least one of their endpoints, or*
- *if P does not include any nodes of $K(r)$, but includes r, let path P consist of disjoint path segments $\{v_1, \ldots, v_{m-1}, r\}$ and $\{r, v_{m+1}, \ldots, v_l\}$, for some $1 \leq m \leq l$, belonging to graphs induced by $desc(K^u) \cup K^u$ and $desc(K^v) \cup K^v$, respectively, for some $u, v \in K$.*

The path segments $\{P_1, \ldots, P_m\}$ belong to an equivalence class in the set $\mathcal{P}(K)$ or paths $\{v_1, \ldots, r\}$ and $\{r, \ldots, v_l\}$ are members of equivalence classes of $\mathcal{P}(K^u)$ and $\mathcal{P}(K^v)$, for some u and v in K.

Proof. Follows immediately from the definition of the equivalence classes. □

We reduce the LP problem to the more general problem of finding for each k-clique K, a representative of each equivalence class in the set $\mathcal{P}(K)$, and for each equivalence class the accumulated length of a member set of paths maximizing this value. When the information for the root has been found, then a root optimization procedure processes this information to detect the possible existence of an *(s-t)*-path including nodes in the root.

For each $\{\hat{P}_1, \ldots, \hat{P}_m\}$ which is an index of an equivalence class in $\mathcal{P}(K)$, we define a function f with value $f(K, \{\hat{P}_1, \ldots, \hat{P}_m\})$ corresponding to the maximum

$$\sum_{j=1}^{m} length(P_j)$$

where $\{P_1, \ldots, P_m\}$ varies over the members of the equivalence class represented by such an index.

Using this notation, if $\{P_1, \ldots, P_m\}$ are the subpaths of a longest *(s-t)*-path in the graph induced by $desc(K) \cup K$ whose intersection with $desc(K)$ is not empty and including no nodes of K except at least one of their endpoints, then

$$\sum_{j=1}^{m} length(P_j) = f(K, \{\hat{P}_1, \ldots, \hat{P}_m\})$$

where $\{\hat{P}_1, \ldots, \hat{P}_m\})$ is the index of the equivalence class including $\{P_1, \ldots, P_m\}$.

In the algorithm for solving the LP problem, we associate with each k-clique K of T a state S. This state $S(K)$ is the set of indices of the relation *same manner* relating sets of dsp with respect to K.

So, an index $SP = \{[X_1, Z_1], \ldots, [X_m, Z_m]\}$ is in the state $S(K)$ if and only if there is a set of dsp with respect to K, denoted $\{path([X_1, Z_1]), \ldots, path([X_m, Z_m])\}$ such that:

- $end(path([X_i, Z_i])) \cap K = X_i$, where $i \in \{1, \ldots, m\}$, and
- $end(path([X_i, Z_i])) \cap \{s, t\} = Z_i$, where $i \in \{1, \ldots, m\}$, and
- $f(K, \{[X_1, Z_1], \ldots, [X_m, Z_m] = \sum_{i \in \{1, \ldots, m\}} length(path([X_i, Z_i]))$, and this sum is the maximum accumulated value of any sets of dsp belonging to the class represented by SP.

The cardinality of the state $S(K)$ is bounded by a function of k alone, since the cardinality of an index SP is bounded by a function of k as described below.

$$|SP| \leq \left(\binom{k}{2} + k \right) \times 2^2, \text{ for all } SP \in S(K) \quad (1)$$

Since the maximum number of indices bounds the cardinality of the state, we have the bound for the cardinality of state S given below.

$$|S(K)| \leq 2^{\left(\binom{k}{2} + k \right) \times 2^2}, \text{ for all } S. \quad (2)$$

Consequently, the cardinality of both sets, $S(K)$ and SP are constants, because k is constant.

The algorithm for the LP problem consists of:

- a data structure that represents the state S of each k-clique K in T, and the value of a function f indicating for each index in S the accumulated path length of a member of its class having maximum value,
- a procedure that updates the state S of each k-clique $K(r)$, where r is in the reduction of T, given the states associated with the $k+1$ k-cliques in K', or finds an *(s-t)*-path including r but no nodes of K (this path does not belong to $\mathcal{P}(K(r))$), and
- and a root optimization procedure that takes the state associated with $root(T)$ and finds a solution for the LP problem.

The states S associated with k-cliques K of T are updated by following the reduction $R = r_0, \ldots, r_m$. All states are initially empty. The reduction R is traversed from r_0 to r_m. When visiting node r_i, the state update procedure updates the state associated with $K(r_i)$ and deletes node r_i from G, or finds an *(s-t)*-path including r_i but no nodes of $K(R_i)$. At this point, all the states associated with the k k-cliques K^u, where $u \neq r_i$ and $u \in K$, have their final values. This is so because the descendants r_j of all these k-cliques are such that $j < i$, by the definition of reduction (Definition 5). These k-cliques disappear from G when r_i is deleted. The state associated with K may already have some information from branches on K not including r_i.

Before we show how to update states, we define *merging* of equivalence classes from sets $\mathcal{P}(K^u)$, where u is in $K'(r_i)$, for any r_i in R. This operation is defined in terms of the classes indices, and it is the core of the *Update* procedure.

For convenience, let us assume that set $\mathcal{P}(K)$, for each k-clique K, includes an empty class represented by the empty index with an f value of 0, i.e., $S(K) = \{ \{[\emptyset, \emptyset]\} \}$, for all k-cliques K.

Definition 14 *Let $(\mathcal{C}_j)_{j \in J}$ be equivalence classes with indices $(P_j)_{j \in J}$, in sets $(\mathcal{P}(K^{u_j}))_{j \in J}$, where for all j, node $u_j \in K'(r_i)$, and for all $l \neq j$, $u_j \neq u_l$, for some r_i in the reduction of partial k-tree G. The* **merging** *of classes $(\mathcal{C}_j)_{j \in J}$, denoted $\bigoplus_{j \in J} \mathcal{C}_j$, is represented by the sequence of indices created by the algorithm given in Figures 2, 3, 4, and 5.*

It follows from the definition of *merging* that *merging* classes may yield indices of more that one class in $\mathcal{P}(K(r_i))$.

The following claim can be verified easily using the definition of *merging*.

Claim 1 *If the cardinality of J is bounded by a constant and $(\mathcal{C}_j)_{j \in J}$ are in $\mathcal{P}(K^{u_j})$, where $u_j \in K'$ then $\bigoplus_{j \in J} \mathcal{C}_j$ can be computed in constant time.*

Procedure *Merge*$((P_j)_{j\in J})$;

Input: Sequence of indices of classes of dsp sets and node r_i.
Output: Sequence of indices of equivalence classes in $\mathcal{P}(K(r_i))$

if there are no path segments in $(P_j)_{j\in J}$ sharing both endpoints, and
  whenever there is a node in $K(r_i)$ shared by path segments, it is shared
  by at most two path segments,
  and there are no path segments represented in $(P_j)_{j\in J}$ inducing a cycle
then
  if r_i is not endpoint of any path segment represented by any index in $(P_j)_{j\in J}$
    then *MergeNoEndpoint*
  else
    if r_i is endpoint of one path segment represented by an index in $(P_j)_{j\in J}$
      then *MergeNoEndpoint*
    else
      if r_i is endpoint of two path segments represented by indices in $(P_j)_{j\in J}$
        then *MergeNoEndpoint*
      endif
    endif
  endif
endif
end

Figure 2. Algorithm for merging *classes of dsp*

Procedure *MergeNoEndpoint;*

Create index $\bigcup_{j\in J} P_j$ with an f value given by $\sum_{j\in J} f(K^{u_j}, P_j)$
if $r_i \in \{s,t\}$ **then**
  for all $u \in K(r_i)$, where
    $u \notin \{s,t\}$ and there is an edge $\{r_i, u\}$ and
      u is not endpoint of more than one path segment,
    or $u \in \{s,t\}$ and u is not the endpoint of any other path segment
  do
    Create $\bigcup_{j\in J} P_j \cup \{[\{u\}, \{r_i,u\} \cap \{s,t\}]\}$
      with an f value is given by $\sum_{j\in J} f(K^{u_j}, P_j) + l(\{r_i,u\})$
  endfor
endif
if $r_i \notin \{s,t\}$ **then**
  for all paths $\{u, r_i, u'\}$, where there are edges $\{u, r_i\}$ and $\{r_i, u'\}$,
    and $u \neq u'$ and both are in $K(r_i)$,
    and nodes $\{u, u'\}$ are not the endpoints of any path represented in $(P_j)_{j\in J}$,
    and neither u nor u' are endpoints of more than one other path segment,
    and there are no cycles induced by path segments represented in
    $\bigcup_{j\in J} P_j \cup \{[\{u,u'\}, \{u',u\} \cap \{s,t\}]\}$,
    and if any u or u' is in $\{s,t\}$, it is not an endpoint of any other path segment
  do
    Create $\bigcup_{j\in J} P_j \cup \{[\{u,u'\}, \{u',u\} \cap \{s,t\}]\}$ with an f value given by
      $\sum_{j\in J} f(K^{u_j}, P_j) + l(\{r_i,u\}) + l(\{r_i,u'\}$
  endfor
endif
end

Figure 3. merging *classes of dsp (r_i is no endpoint of any path segment)*

Procedure *MergeOneEndpoint;*

Let r_i be the endpoint of a path represented by $PS = [X, Z]$
if $r_i \in \{s,t\}$, and there are no cycles induced by path segments represented in
  $(\bigcup_{j\in J} P_j \setminus \{PS\}) \cup \{[X \setminus \{r_i\}, Z]\}$, and
    X includes a node in $K(r_i)$,
    or X does not include any nodes in $K(r_i)$ but $Z = \{s,t\}$, and
      PS is the only path segment represented in $(P_j)_{j\in J}$
then
  Create $(\bigcup_{j\in J} P_j \setminus \{PS\}) \cup \{[X \setminus \{r_i\}, Z]\}$ with f value
    given by $\sum_{j\in J} f(K^{u_j}, P_j)$
endif
if $r_i \notin \{s,t\}$ and there is an edge $\{u, r_i\}$, where $u \in K(r_i)$ **then**
  for all $u \in K(r_i)$ such that
    there is an edge $\{u, r_i\}$, and
      $(X \setminus \{r_i\}) \cup \{u\}$ are not the endpoints of any path represented in $(P_j)_{j\in J}$, and
        u is an endpoint of no more than one other path segment,
        or node u is in $\{s,t\}$ and it is not endpoint of any path segment, and
      there are no cycles induced by path segments represented in
      $(\bigcup_{j\in J} P_j \setminus \{PS\}) \cup \{[(X \setminus \{r_i\}) \cup \{u\}, Z \cup (\{s,t\} \cap \{u\})]\}$
  do
    Create $(\bigcup_{j\in J} P_j \setminus \{PS\}) \cup \{[(X \setminus \{r_i\}) \cup \{u\}, Z \cup (\{s,t\} \cap \{u\})]\}$
      with f value given by $\sum_{j\in J} f(K^{u_j}, P_j) + l(\{u, r_i\}$
  endfor
endif
end

Figure 4. merging *classes of dsp (r_i is endpoint of one path segment)*

Procedure *MergeTwoEndpoints;*

Let r_i be the endpoint shared by $PS = [X, Z]$ and $PS' = [X'Z']$
if $r_i \notin \{s,t\}$, and there are no cycles induced by path segments in
  $(\bigcup_{j\in J} P_j \setminus \{PS, PS'\}) \cup \{[X \cup X' \setminus \{r_i\}, Z \cup Z'\}$, and
  $X \cup X' \setminus \{r_i\} = \{s,t\}$, and
    nodes in $X \cup X' \setminus \{r_i\}$ are not the endpoint of any other path segment,
  or nodes in $X \cup X' \setminus \{r_i\}$ are in $K(r_i)$, and
    they are not the endpoints of any other path segment,
  or $X \cup X' \setminus \{r_i\}$ includes a node u, and the node not in $K(r_i)$ is in $\{s,t\}$, and
    if node u is not in $\{s,t\}$, it is shared by at most one other path segment,
    or if node u is in $\{s,t\}$, then it is not common to any other path segment
then
  Create $(\bigcup_{j\in J} P_j \setminus \{PS, PS'\}) \cup \{[X \cup X' \setminus \{r_i\}, Z \cup Z'\}$, with f value given by
    $\sum_{j\in J} f(K^{u_j}, P_j)$
endif
end

Figure 5. merging *classes of dsp (r_i is endpoint of two path segment)*

We now show that updating a state S can be done efficiently.

Lemma 5 *Let G be a partial k-tree, s and t two specified nodes, R its reduction, and $r \in R$. The state S associated with $K(r)$ in the graph induced by $K \cup B(K(r))$ can be found in constant time from the states associated with the $k+1$ k-cliques K^u, where $u \in K'$.*

Proof. We describe the procedure *Update* in Figure 6 for updating states S associated with k-cliques in a k-tree T.

The correctness of the *Update* procedure follows from the fact that the indices of the equivalence classes of set $\mathcal{P}(K(r))$ can be expressed in terms of indices of classes in sets $\mathcal{P}(K^u(r'))$, where $u \in K'$ and r' appears before r in the reduction.

Let $\{\hat{P}_1, \ldots, \hat{P}_n\}$ be an index of a class in $\mathcal{P}(K(r))$, where r is the last descendant of K in the reduction $r_0, \ldots, r_m$ of T.

Node r may or may not be included in a path segment represented by $\hat{P}_i$, for some $1 \leq i \leq n$.

If r is not included in any such path segments then it follows that $\{\hat{P}_1, \ldots, \hat{P}_n\}$ can be partitioned, so that each subset of the partition is an index of an equivalence class of a set $\mathcal{P}(K^u(r'))$, for some $u \in K$, and r' appearing before r in the reduction of T.

If r is included in $path(\hat{P}_i)$, for some $1 \leq i \leq n$, then we have two cases:

- $path(\hat{P}_i)$ has both endpoints in $K(r)$, or
- $path(\hat{P}_i)$ has only one endpoint u in $K(r)$.

In the second case r is s or t, and $path(\hat{P}_i)$ is an edge included in K', or there is a $u \in K(r)$ such that r is the endpoint of a path represented in an index of a class of $\mathcal{P}(K^u(r'))$, where r' appears before r in the reduction of T.

In the first case, $path(\hat{P}_i)$ falls in one of the following three categories.

- It consists of nodes $\{u, r, v\}$, where u and v are in $K(r)$ and there are edges $\{u, r\}$ and $\{r, v\}$, i.e., $path(\hat{P}_i)$ is included in $K'(r)$, or
- $path(\hat{P}_i)$ consists of a path segment belonging to a class in the set $\mathcal{P}(K^u(r'))$, where $u \in K(r)$ and r' appears before r in the reduction, extended by an edge $\{r, v\}$, where $v \in K(r)$, or
- $path(\hat{P}_i)$ consists of two path segments members of classes in sets $\mathcal{P}(K^u(r'))$ and $\mathcal{P}(K^v(r''))$, where u and v (not necessarily distinct) are in K and r' and r'' appear before r in the reduction of G. If $u = v$ then both subpaths belong to the same class of $\mathcal{P}(K^u(r'))$ (this guarantees that the subpaths are simple and disjoint), and if $u \neq v$ then the subpaths belong to classes of sets $\mathcal{P}(K^u(r'))$ and $\mathcal{P}(K^v(r''))$, where r' and r'' appear before r in the reduction of G, which, by Lemma 2, include classes of paths that are disjoint with exception of possibly their endpoints. In any case, path segments are represented by indices of their corresponding classes.

It follows that an index of a class of $\mathcal{P}(K(r))$ can be build from indices of classes in sets $\mathcal{P}(K^u(r'))$, where $u \in K$ and r' appears before r in the reduction, or by inspecting nodes and edges in the graph induced by K'. This is the definition of *merging* of equivalence classes.

Also, procedure *Update* detects *(s-t)*-paths not including nodes in K but including node r, and maintains the value stored in *LongPath*.

The complexity of the procedure is constant since all the loops have constant range and the tests and operations can all be done in constant time, as stated in Claim 1. □

The algorithm proceeds bottom-up from k-leaves towards the root by following a reduction R of the K-tree T of which G is a subgraph. It finds the state associated with each k-clique of T. When all the vertices in G, but those in the $root(T)$, have been removed, it finds the solution for the LP problem from the information collected in the states associated with the k-cliques of T. Figure 7 shows an algorithm for the LP Problem.

The correctness and time complexity of this algorithm is established in the following theorem.

Theorem 1 *The algorithm for the LP problem is correct and has time complexity $O(n)$.*

Proof. The procedure *OptimizeRoot* considers the following cases:

- nodes s and t are both in $root(T)$, or
- only one of nodes s or t are in $root(T)$, or
- none of s nor t are in $root(T)$.

In the first case, the procedure scans all the indices in state S associated with $root(T)$ and for each index, it attempts to build an *(s-t)*-path using all the path segments represented in such an index and possibly some edges in the graph induced by $root(T)$.

In the second case, the procedure scans all the indices in state S associated with $root(T)$ containing the representation of one path segment which includes the complementary endpoint and for each index, it attempts to build an *(s-t)*-path using all the path segments represented in such an index and possibly some edges in the graph induced by $root(T)$.

In the third case, the procedure scans all the indices in state S including the representation of paths with endpoints s and t and for each index, it attempts to build an *(s-t)*-path using all the path segments represented in such an index and possibly some edges in the graph induced by $root(T)$.

Whenever this procedure finds an *(s-t)*-path it updates the value *LongPath*.

Clearly, the procedure *OptimizeRoot* can be implemented so that it can be executed in constant time, since all the information that it uses has constant size. The Theorem follows from this observation and the following facts.

A longest path can be built in a bottom–up fashion, because the descendants of the k-cliques K^u, where $u \in K$ and disjoint, as shown in Lemma 2. States associated with k-cliques can be updated in constant time, as stated in Lemma 5. Any solution to the LONGEST *(s-t)*-PATH PROBLEM exhibits optimal substructure, given by Lemma 3. A longest *(s-t)*-path is fully contained in the $root(T)$, or includes subpaths belonging to equivalence classes of some sets $\mathcal{P}(K(r))$, $r \in R$, and intersecting $K(r)$, or including r, as stated in Lemma 4. □

Procedure *Update(S,r);*

Input: State S to be updated and node r, where S is the state associated with $K(r)$
Output: State S containing indices of equivalence classes in $\mathcal{P}(K(r))$

for each sequence of classes $(C_j)_{j\in J}$, from $\bigcup_{v\in K'} \mathcal{P}(K^v)$ and
  no two classes from the same set $\mathcal{P}(K^v)$
do
  if $\bigoplus_{j\in J} C_j$ includes index $\{[\emptyset, \{s,t\}]\}$ and has f value V **then**
    UpdatePathSoFar(LongPath,V)
    put in S all indices $\bigoplus_{j\in J} C_j$ except index $\{[\emptyset, \{s,t\}]\}$ and
      update their f values
  else
    put in S all indices $\bigoplus_{j\in J} C_j$ and update their f values
  endif
endfor
end

Figure 6. Algorithm for Updating a state S

Procedure *LongPath(G,T,R,s,t);*

INPUT: $G = (V, E), T = (V, E')$, G a partial k-tree and T its embedding k-tree,
  $R = \{r_0, \ldots, r_n\}$ a reduction of T and s, t two designated vertices

$kC \leftarrow$ set of k-cliques in T
Let R be the root of T
LongPath $\leftarrow 0$
for $kC_i \in kC$ **do**
  set state S associated with k-clique kC_i to $\{\{[\emptyset, \emptyset]\}\}$
  set the f value of the index $\{[\emptyset, \emptyset]\}$ in S to 0
endfor
for $r = r_0$ **upto** r_n **do**
  $Update(S, r)$;
endfor
OptimizeRoot(S(root(T)))
if *LongPath* > 0 **then** output 'The Longest *(s-t)*-path has length *LongPath*
else output 'There is no *(s-t)*-path in G'
endif
end

Figure 7. Algorithm for the LP Problem for Partial k-trees

4 Example: Finding a longest path in a partial 3-tree

Let G be the partial 3-tree given in Figure 8, where node $s = 3$ and node $t = 9$. In this example, only states with nonempty information are shown, namely, $S(\{1,2,3\})$, $S(\{1,3,4\})$, $S(\{3,4,5\})$, $S(\{3,5,6\})$.

Let $r = 9,7,6,5,4,8$. In this case, the root optimization procedure finds a longest *(s-t)*-path of length 6. For simplicity, all the edges of the graph in this example are assumed to have weight 1.

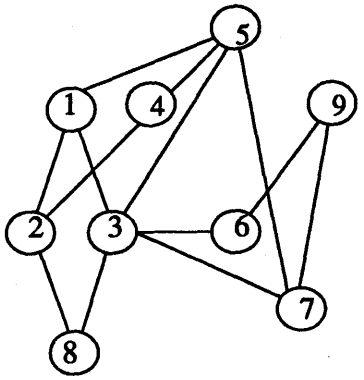


Figure 8. Partial 3-tree

Node to be eliminated: 9
$K(9) = \{5,6,7\}$
$S = \{\{[\emptyset,\emptyset]\},$
$\{[\{6\},\{9\}]\},$
$\{[\{7\},\{9\}]\}\}$
each index with f values 0, 1, 1 respectively.

Node to be eliminated: 7
$K(7) = \{3,5,6\}$
$S = \{\{[\emptyset,\emptyset]\},$
$\{[\{6\},\{9\}]\},$
$\{[\{3\},\{3,9\}]\},$
$\{[\{5\},\{9\}]\},$
$\{[\{3,5\},\{3\}]\},$
$\{[\{3,5\},\{3\}],[\{6\},\{9\}]\}\}$
each index with f values 0, 1, 2, 2, 2, 3 respectively. Notice that at this stage a $(3-9)$–path has been found, this is represented by the index $\{[\{3\},\{3,9\}]\}$. This path is detected when node 3 is to be eliminated or, as in this case when the root optimization procedure finds it, because 3 does not belong to the reduction of the graph.

Node to be eliminated: 6
$K(6) = \{3,4,5\}$
$S = \{\{[\emptyset,\emptyset]\},$
$\{[\{3\},\{3,9\}]\},$
$\{[\{5\},\{9\}]\},$
$\{[\{3,5\},\{3\}]\}\}$
each index with f values 0, 2, 2, 2 respectively.

Node to be eliminated: 5
$K(5) = \{1,3,4\}$
$S = \{\{[\emptyset,\emptyset]\},$
$\{[\{1,4\},\emptyset]\},$
$\{[\{1,3\},\{3\}]\},$
$\{[\{3\},\{3,9\}]\},$
$\{[\{4\},\{9\}]\},$
$\{[\{1\},\{9\}]\},$
$\{[\{1,3\},\{3\}]\},$
$\{[\{4,3\},\{3\}]\}\}$
each index with f values 0, 2, 2, 3, 3, 3, 3, 3 respectively.

Node to be eliminated: 4
$K(4) = \{1,2,3\}$
$S = \{\{[\emptyset,\emptyset]\},$
$\{[\{1,2\},\emptyset]\},$
$\{[\{2,3\},\{3\}]\},$
$\{[\{3\},\{3,9\}]\},$
$\{[\{2\},\{9\}]\},$
$\{[\{1\},\{9\}]\},$
$\{[\{1,3\},\{3\}]\}\}$
each index with f values 0, 3, 3, 3, 4, 3, 3 respectively.

Node to be eliminated: 8
$K(8) = \{1,2,3\}$
$S = \{\{[\emptyset,\emptyset]\},$
$\{[\{1,2\},\emptyset]\},$
$\{[\{2,3\},\{3\}]\},$
$\{[\{3\},\{3,9\}]\},$
$\{[\{2\},\{9\}]\},$
$\{[\{1\},\{9\}]\},$
$\{[\{1,3\},\{3\}]\}\}$
$\{[\{1,2\},\emptyset],[\{2,3\},\{3\}]\},$
$\{[\{2\},\{9\}],[\{2,3\},\{3\}]\},$
$\{[\{1\},\{9\}],[\{2,3\},\{3\}]\}\}$
each index with f values 0, 3, 3, 3, 4, 3, 3, 5, 6, 5 respectively.

The root optimization procedure finds index $\{[\{2\},\{9\}],[\{2,3\},\{3\}]\}$, in state $S(\{1,2,3\})$ and builds the (3-9)-path with value 6. It also finds (3-9)-paths with value 6, from indices $\{[\{2\},\{9\}]\}$, and $\{[\{1\},\{9\}],[\{2,3\},\{3\}]\}$.

5 Conclusions

We showed a linear time algorithm for solving the LONGEST *(s-t)*-PATH PROBLEM, when the instances are restricted to partial k-trees together with their reduction. This algorithm has restricted applicability because the constants involved in the time complexity function are exponential in k. As mentioned, these exponential constants may be inherent to the time complexity function, because if they could be eliminated and the algorithms for each value of k are polynomialy uniform, the consequence is that $P = NP$. No effort was taken in minimizing the magnitude of the constants involved in the time complexity.

References

1. S Arnborg, D G Corneil, and A Proskurowski. 'Com-

plexity of finding embeddings in a k-tree'. *SIAM J. of Alg. and Disc. Math.*, 8(2):277–284, (1987).
2. S Arnborg and A Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. To appear Discr. Applied Math.
3. S Arnborg and A Proskurowski. 'Algorithms on graphs with bounded decomposability'. *Congressus Num.*, (53):161–170, (1986).
4. H L Bodlaender. 'Classes of graphs with bounded decomposability'. Technical Report RUU-CS-86-22, Computer Science Dept., University of Utrecht, (1986).
5. H L Bodlaender. 'Improved self-reduction algorithms for graphs with bounded treewidth'. Technical Report RUU-CS-88-29, Computer Science Dept., University of Utrecht, (1988).
6. T H Cormen, C E Leiserson, and R L Rivest. *Introduction to Algorithms*. MIT Press, 1991.
7. J A Ellis, G MacGillivray, and M Mata-Montero. 'A linear time algorithm for the longest (s-t) path in weighted outerplanar graphs'. *Information Processing Letters*, (32):199–204, (1989).
8. J A Ellis, M Mata-Montero, and H Muller. 'Serial and parallel algorithms for $(k, 2)$-partite graphs'. *Journal of Parallel Processing*. to appear.
9. M R Fellows and M A Langston. Nonconstructive tools for proving polynomial-time decidability. To appear J. of Algorithms.
10. M R Garey and D S Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., San Francisco, 1979.
11. E Mata-Montero. 'Resilience of partial k-tree networks with node and edge failures'. Technical Report CIS-TR-89-15, University of Oregon, Eugene, (1989).
12. B Monien. 'The bandwidth minimization problem for caterpillars with hair length 3 is NP-complete'. *SIAM J. Alg. Discr. Math.*, 7(4), (October 1986).
13. J Naor, M Naor, and A Sch¨affer. 'Fast parallel algorithms for chordal graphs'. *Proc. of the 19th Annual ACM Symp. on the Theory of Computing*, (1987).
14. J van Leeuwen. *Algorithms and Complexity*, volume A of *The Handbook of Theoretical Computer Science*. MIT Press/Elsevier, 1990.

Received: 12/93, Accepted: 3/94, Final copy: 6/94.

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for *review* should be prepared according to the following guidelines.

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be numbered and titled. Figures should be submitted as original line drawings/printouts, and not photocopies.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text in square brackets [1–3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As (a) LaTeX file(s), either on a diskette, or via e-mail/ftp – a LaTeX style file is available from the production editor;
2. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.

   Further instructions on how to reduce page charges can be obtained from the production editor.
3. In camera-ready format – a detailed page specification is available from the production editor;
4. In a typed form, suitable for scanning.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R30-00 per final page for LaTeX or camera-ready contributions that require no further attention. The maximum is R120-00 per page for contributions in typed format (charges include VAT).

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers in categories 2 and 4 above may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Camera-ready submissions will only be accepted if they are in strict accordance with the detailed guidelines. It is the responsibility of the authors to ensure that their submissions are error-free.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to less than about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

References

1. E Ashcroft and Z Manna. 'The translation of 'goto' programs to 'while' programs'. In *Proceedings of IFIP Congress 71*, pp. 250–255, Amsterdam, (1972).
2. C Bohm and G Jacopini. 'Flow diagrams, turing machines and languages with only two formation rules'. *Communications of the ACM*, **9**:366–371, (1966).
3. S Ginsburg. *Mathematical theory of context free languages*. McGraw Hill, New York, 1966.

South African Computer Journal

Number 12, November 1994
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 12, November 1994
ISSN 1015-7999

Contents

UP Drukkers Printers