MAX-PLANCK-INSTITUT
FÜR
INFORMATIK

Generalized Topological Sorting in Linear Time

Torben Hagerup and Martin Maas

MPI–I–93–119 May 1993

Im Stadtwald
W 6600 Saarbrücken
Germany

Generalized Topological Sorting in Linear Time

Torben Hagerup and Martin Maas

MPI–I–93–119 May 1993

Generalized Topological Sorting in Linear Time

Torben Hagerup[1] and Martin Maas[2]

[1] Max-Planck-Institut für Informatik, W–6600 Saarbrücken, Germany
[2] CAP debis, Division Industrie, Erich-Herion-Straße 13, W–7012 Fellbach, Germany

Abstract. The generalized topological sorting problem takes as input a positive integer k and a directed, acyclic graph with some vertices labeled by positive integers, and the goal is to label the remaining vertices by positive integers in such a way that each edge leads from a lower-labeled vertex to a higher-labeled vertex, and such that the set of labels used is exactly $\{1, \ldots, k\}$. Given a generalized topological sorting problem, we want to compute a solution, if one exists, and also to test the uniqueness of a given solution. The best previous algorithm for the generalized topological sorting problem computes a solution, if one exists, and tests its uniqueness in $O(n \log\log n + m)$ time on input graphs with n vertices and m edges. We describe improved algorithms that solve both problems in linear time $O(n + m)$.

1 Introduction

Motivated by an application in the VLSI layout system HILL (Lengauer and Mehlhorn, 1984), Hagerup and Rülling (1986) introduced the generalized topological sorting (GTS) problem and considered two variants of the problem. We focus here on the so-called constrained GTS problem, deferring a discussion of the unconstrained GTS problem to the final section of the paper. Given a partial function $\widehat{f}$, $Dom(\widehat{f})$ denotes the domain of $\widehat{f}$, i.e., $Dom(\widehat{f}) = \{v \mid \widehat{f}(v) \text{ is defined}\}$.

Definition. An instance of the *constrained GTS problem* is a tuple $P = (V, E, \widehat{f}, k)$, where (V, E) is a directed, acyclic graph, $\widehat{f}$ is a partial function from V to $\mathbb{N} = \{1, 2, \ldots\}$, and $k \in \mathbb{N}$. A solution to P is a total function $f : V \to \mathbb{N}$ with the following properties:
(1) f is *monotonic*, i.e., for all $(u, v) \in E$, $f(u) < f(v)$.
(2) f is an extension of $\widehat{f}$, i.e., for all $v \in Dom(\widehat{f})$, $f(v) = \widehat{f}(v)$.
(3) $f(V) = \{1, \ldots, k\}$.

In the application in the HILL system, an instance of the GTS problem is formulated by a user of the system, whose intent is to describe a certain desired

Supported by the ESPRIT Basic Research Actions Program of the EC under contract No. 7141 (project ALCOM II). Part of the research was carried out while both authors were with the Universität des Saarlandes.

vertex labeling to the system in a convenient manner. If the instance has no solution, the user has made a mistake. If it has more than one solution, on the other hand, the labeling is underdetermined, and the user should be prompted for more information. We are therefore interested not only in deciding solvability and computing solutions to GTS problems, but also in testing the uniqueness of a given solution.

In a more general perspective, the same question can be asked for other computational problems: If a given instance has a solution, is that solution unique? Related questions play a certain role in complexity-theoretic investigations of more difficult problems, cf. the definition of UP (Valiant, 1976) as the class of languages accepted in polynomial time by nondeterministic Turing machines with at most one accepting path for each input (a single accepting path corresponds in a natural way to a unique solution). However, little work seems to have been done along these lines for problems that can be solved in polynomial time. It can be shown that there are problems for which testing the uniqueness of a solution is harder than producing an arbitrary solution, but we are not aware of any natural examples of this phenomenon.

This paper describes linear-time algorithms for the following tasks: Given an instance P of the constrained GTS problem, (A) decide whether P has a solution and, if so, compute one; (B) decide whether a given solution to P is unique. It turns out that (A), the *existence problem*, is essentially solved by a known algorithm for convex bipartite matching, whereas our handle on (B), the *uniqueness problem*, is a careful analysis of the special solution produced by our algorithm for (A). The best previous algorithms for problems (A) and (B), due to Hagerup and Rülling (1986), have running times of $O(n \log\log n + m)$ for input graphs with n vertices and m edges.

2 Preliminaries

Consider from now on a fixed constrained GTS problem $P = (V, E, \hat{f}, k)$ and let $n = |V|$, $m = |E|$ and $K = \{1, \ldots, k\}$. The functions *Low* and *High*, defined below, are fundamental to our discussion.

Definition. Let *Low* and *High* be the functions defined on V by

$$\begin{aligned} \mathit{Low}(v) &= \max\{f_{\mathrm{L}}(u) + \mathit{length}(p) \mid u \in V \text{ and } p \text{ is a path in } G \text{ from } u \text{ to } v\} \\ \mathit{High}(v) &= \min\{f_{\mathrm{H}}(w) - \mathit{length}(p) \mid w \in V \text{ and } p \text{ is a path in } G \text{ from } v \text{ to } w\}, \end{aligned}$$

for all $v \in V$, where $\mathit{length}(p)$ denotes the number of edges on the path p, and f_{L} and f_{H} are the trivial extensions of $\hat{f}$ with

$$f_{\mathrm{L}}(v) = \begin{cases} \hat{f}(v), & \text{if } v \in \mathit{Dom}(\hat{f}) \\ 1, & \text{if } v \in V \backslash \mathit{Dom}(\hat{f}) \end{cases}$$

$$f_{\mathrm{H}}(v) = \begin{cases} \hat{f}(v), & \text{if } v \in \mathit{Dom}(\hat{f}) \\ k, & \text{if } v \in V \backslash \mathit{Dom}(\hat{f}). \end{cases}$$

$Low(v)$ and $High(v)$ can be computed for all $v \in V$ in $O(n+m)$ time by a procedure, similar to usual topological sorting, that processes the vertices once in topological order and once in inverse topological order. If f is a solution to P, clearly $Low(v) \leq f(v) \leq High(v)$ for all $v \in V$. In particular, we will assume throughout that $1 \leq Low(v) \leq High(v) \leq k \leq n$ for all $v \in V$, since otherwise P fails in a trivial way to be solvable. It is then easy to see that Low and $High$ are monotonic extensions of $\hat{f}$, i.e., the functions Low and $High$ are solutions to P, except that their ranges may not include all of K.

3 The Existence Problem

Consider the bipartite graph H on the vertex sets K and V that contains an edge (i, v), for $i \in K$ and $v \in V$, exactly if $Low(v) \leq i \leq High(v)$. H is *convex*, i.e., the vertices connected to each $v \in V$ form a set of consecutive integers. We make use of maximum matchings in H, but for reasons of convenience phrase the discussion in terms of what we call *pairings*.

Definition. A *pairing* is an injective partial function $g : V \hookrightarrow \mathbb{N}$ with $Low(v) \leq g(v) \leq High(v)$ for all $v \in Dom(g)$. The *size* of a pairing g is $|Dom(g)|$, and g is *maximum* if $|Dom(g)| \geq |Dom(g')|$ for all pairings g'.

Lipski and Preparata (1981) gave an algorithm for computing maximum matchings in convex bipartite graphs, which is trivially converted into an algorithm for computing maximum pairings. As demonstrated by Gabow and Tarjan (1985), the algorithm can be implemented to run in linear time $O(n)$. The algorithm below, where implementation details have been ignored, computes the same maximum pairing as the linear-time algorithm. Identify the partial function g with the set $\{(v, g(v)) \mid v \in Dom(g)\}$.

```
Sort the elements of V by their High values, i.e., compute a bijection
  σ : {1,...,n} → V such that for all i, j ∈ ℕ with 1 ≤ i < j ≤ n,
  High(σ(i)) ≤ High(σ(j));
g := ∅;
for i := 1 to n do
  begin (* assign a value to σ(i), if possible *)
    J := {Low(σ(i)),...,High(σ(i))}\g(Dom(g));
      (* possible candidates for g(σ(i)) *)
    if J ≠ ∅
    then g := g ∪ {(σ(i), min J)};
  end;
```

The vertices are hence processed in the order of increasing $High$ values, and each vertex is mapped to the smallest allowed value still available, if any. In addition to being maximum, the pairing computed by the algorithm is also *regular* in the sense of the following definition.

Definition. A pairing g is called *regular* if for all $u, v \in V$ with $v \in Dom(g)$ and $Low(u) \leq g(v) \leq High(u) < High(v)$, we have $u \in Dom(g)$ and $g(u) < g(v)$.

Lemma 1. *Any pairing computed by the above algorithm is regular.*

Proof. Let g be a pairing computed by the algorithm and let $u, v \in V$ be as in the definition of a regular pairing, i.e., $v \in Dom(g)$ and $Low(u) \leq g(v) \leq High(u) < High(v)$. Then $\sigma^{-1}(u) < \sigma^{-1}(v)$, i.e., u is processed before v. Since $g(v)$ is included in $g(Dom(g))$ during the processing of v, it belongs to J during the processing of u. In particular, $J \neq \emptyset$ during the processing of u, so $u \in Dom(g)$. The claim now follows, since $g(u)$ is chosen as $\min J$, yet is different from $g(v)$. □

Definition. For any partial function g from V to $\mathbb{N}$, the total function $f : V \rightarrow \mathbb{N}$ given by

$$f(v) = \begin{cases} g(v), & \text{if } v \in Dom(g) \\ High(v), & \text{if } v \in V \backslash Dom(g) \end{cases}$$

is called the *High extension* of g.

Lemma 2. *Let f be the High extension of a regular pairing g. Then for all $u, v \in V$,*

$$Low(u) \leq f(v) \leq f(u) \quad \Rightarrow \quad High(v) \leq High(u).$$

Proof. If $v \notin Dom(g)$, then $High(v) = f(v) \leq f(u) \leq High(u)$. Suppose therefore that $v \in Dom(g)$ and that $High(v) > High(u)$. Then

$$Low(u) \leq f(v) = g(v) \leq f(u) \leq High(u) < High(v),$$

and the regularity of g implies that $f(u) = g(u) < g(v) = f(v)$, a contradiction. □

Lemma 3. *Any High extension f of a regular pairing of size k is a solution to P.*

Proof. We need only verify that f is monotonic. Let $(u, v) \in E$. Since $Low(u) < Low(v) \leq f(v)$ and $High(v) > High(u)$, Lemma 2 implies that $f(v) > f(u)$. □

Theorem 1. *Given an instance $P = (V, E, \widehat{f}, k)$ of the constrained GTS problem with $|V| = n$ and $|E| = m$, the solvability of P can be tested in $O(n + m)$ time, and if P is solvable, a solution to P can be computed in $O(n + m)$ time.*

Proof. First compute the functions *Low* and *High* corresponding to P in $O(n+m)$ time. Then find a regular maximum pairing g. As stated above, this can be done in $O(n)$ time. If g is not of size k, clearly P is not solvable. Otherwise, by Lemma 3, P is solvable, and a solution to P can be obtained in $O(n)$ time as the *High* extension of g. □

4 The Uniqueness Problem

This section develops an algorithm to test whether a given constrained GTS problem has more than one solution. Define a *standard solution* to be the *High* extension of a regular pairing of size k (by Lemma 3, each such function is indeed a solution to P). We begin by investigating the properties of standard solutions, one major goal being to show that such solutions maximize the sum of all vertex labels.

Definition. For any solution f to P, let $S(f) = \sum_{v \in V} f(v)$. f is called *maximal* if $S(f) \geq S(f')$ for all solutions f' to P, and *minimal* if $S(f) \leq S(f')$ for all solutions f' to P.

Definition. For any two solutions f and f' to P, let $D(f', f)$ denote the directed multigraph (K, A), where $A = \{(f(v), f'(v)) \in K \times K \mid v \in V\}$ should be taken as a multiset, i.e., it is formed without elimination of duplicates, and its cardinality is exactly n.

Given two solutions f and f' to P, $D(f', f)$, informally, expresses the changes necessary to turn f into f'. We now decompose $D(f', f)$ into simpler parts.

Definition. Let $D = (U, A)$ be a directed multigraph. A vertex in D is called *outdominant* in D if its outdegree in D (strictly) exceeds its indegree in D. A path p in D is called *complete* in D exactly if p is simple and the first vertex on p is outdominant in D. A *path decomposition* of D is a set Π of paths in D such that each edge in A lies on exactly one path in Π, and such that each path in Π is either cyclic or complete in D.

Lemma 4. *Every directed multigraph $D = (U, A)$ has a path decomposition.*

Proof. By induction on $|A|$. If $|A| = 0$, there is nothing to show. Otherwise choose p_1 as a cycle in D or, if D is acyclic, as a maximal path in D, which is necessarily complete in D, and apply the induction hypothesis to $D_1 = (U, A \backslash A_1)$, where A_1 is the set of edges on p_1. The lemma follows, since every complete path in D_1 is a complete path in D. □

Definition. For every directed multigraph $D = (K, A)$ on the vertex set K, let $\Delta(D) = \sum_{(i,j) \in A} (j - i)$. D is called *incrementing* if $\Delta(D) > 0$.

If D is a cycle, clearly $\Delta(D) = 0$, and if D is a complete path from i to j, then $\Delta(D) = j - i$. Furthermore, for any two solutions f and f' to P, it is easy to see that $\Delta(D(f', f)) = S(f') - S(f)$. Using the latter characterization of $S(f') - S(f)$ as well as the following lemma, we are finally able to demonstrate the maximality of standard solutions.

Lemma 5. *For all standard solutions f and for all $v \in V$ with $|f^{-1}(f(v))| \geq 2$, we have $f(v) = High(v)$.*

Proof. Let f be the *High* extension of a regular pairing g, and suppose that $v \in V$ has $|f^{-1}(f(v))| \geq 2$, but $f(v) < High(v)$. Then $v \in Dom(g)$, and for some $u \in V$ with $u \notin Dom(g)$, $f(u) = High(u) = f(v)$. But then

$$Low(u) \leq f(u) = f(v) = g(v) = High(u) < High(v),$$

contradicting the regularity of g. □

Lemma 6. *Let f' be a solution to P and let f be a standard solution. Then no path decomposition of $D(f', f)$ contains an incrementing path.*

Proof. Let p be a complete path in some path decomposition of $D(f', f)$ consisting of the edges

$$(i_0, i_1), (i_1, i_2), \ldots, (i_{t-1}, i_t),$$

and choose $w_1, \ldots, w_t \in V$ such that for $j = 1, \ldots, t$,

$$f(w_j) = i_{j-1} \quad \text{and} \quad f'(w_j) = i_j.$$

For $s = 1, \ldots, t$, consider now the assertion

$$Q(s): \quad High(w_s) \leq i_0.$$

We will prove $Q(s)$ for $s = 1, \ldots, t$ by complete induction on s. $Q(t)$ implies $i_t = f'(w_t) \leq High(w_t) \leq i_0$, showing that p cannot be incrementing.

Since i_0 is outdominant in $D(f', f)$ and $f'(V) = K$, $|f^{-1}(i_0)| > |(f')^{-1}(i_0)| \geq 1$. Hence by Lemma 5, $High(w_1) = f(w_1) = i_0$, providing the basis $Q(1)$ for the induction.

For the inductive step, fix s with $2 \leq s \leq t$ and assume that $Q(j)$ is true for $j = 1, \ldots, s-1$. Choose r with $1 \leq r \leq s-1$ such that $i_{r-1} > i_{s-1} \geq i_r$. This is possible since $i_j > i_{s-1}$ holds for $j = 0$, by the induction hypothesis $Q(s-1)$ and the fact that $i_0 \neq i_{s-1}$, but not for $j = s-1$. Now

$$Low(w_r) \leq f'(w_r) = i_r \leq i_{s-1} = f(w_s) < i_{r-1} = f(w_r),$$

and Lemma 2 implies that $High(w_s) \leq High(w_r)$ and hence, by the induction hypothesis $Q(r)$, that $High(w_s) \leq i_0$, i.e., $Q(s)$. □

Lemma 7. *Every standard solution f is a maximal solution to P.*

Proof. Let f' be an arbitrary solution to P and let $\{p_1, \ldots, p_r\}$ be a path decomposition of $D(f', f)$. $S(f') - S(f) = \Delta(D(f', f)) = \sum_{i=1}^{r} \Delta(p_i)$, and by Lemma 6, $\Delta(p_i) \leq 0$ for $i = 1, \ldots, r$. Hence $S(f') \leq S(f)$. □

Assume in the remainder of this section that P is solvable. Having shown that a maximal solution to P can be computed in $O(n+m)$ time, we next argue that

a minimal solution to P can be found within the same time bound. Whereas this fact can be demonstrated by simple modifications to the algorithms described in Section 3, a cleaner argument proceeds as follows: Consider the constrained GTS problem $P_{\text{rev}} = (V, E_{\text{rev}}, k+1-\hat{f}, k)$, where $E_{\text{rev}} = \{(v,u) \in V \times V \mid (u,v) \in E\}$ and $k+1-\hat{f}$ is the partial function from V to $\mathbb{N}$ with the same domain as $\hat{f}$ and given by $(k+1-\hat{f})(v) = k+1-\hat{f}(v)$, for all $v \in Dom(\hat{f})$. It is easy to see that for any given function $f : V \to \mathbb{N}$, f is a minimal solution to P if and only if $k+1-f$ is a maximal solution to P_{rev}. $O(n+m)$ time hence suffices to compute a standard (maximal) solution $f_{\max}$ and a minimal solution $f_{\min}$ to P. If $f_{\min} \neq f_{\max}$, then obviously f has more than one solution. On the other hand, if $f_{\min} = f_{\max}$, then all solutions to P are maximal. We will show that in this case, if f is a standard solution to P, then P has a solution different from f if and only if such a solution can be obtained from f simply by interchanging the labels of two vertices. Subsequently we show how to identify two such vertices in linear time.

For all $v \in V$, call $u \in V$ a *predecessor* of v if $(u,v) \in E$, and call $w \in V$ a *successor* of v if $(v,w) \in E$.

Definition. Let f be a solution to P. An *interchangeable pair* in f is a pair $(v_1, v_2) \in (V \backslash Dom(\hat{f}))^2$ such that

(1) $f(v_1) < f(v_2)$.
(2) $f(u) < f(v_1)$ for all predecessors u of v_2.
(3) $f(w) > f(v_2)$ for all successors w of v_1.

Lemma 8. *Let f be a standard solution. Then P has a maximal solution different from f if and only if there is an interchangeable pair in f.*

Proof. If (v_1, v_2) is an interchangeable pair in f, then the function $f' : V \to K$ given by

$$f'(v) = \begin{cases} f(v_2), & \text{if } v = v_1 \\ f(v_1), & \text{if } v = v_2 \\ f(v), & \text{if } v \in V \backslash \{v_1, v_2\} \end{cases}$$

obviously is a solution to P different from f.

Assume now that P has a maximal solution f' different from f and let $D = D(f', f)$. By Lemma 6, the indegree of each vertex in D equals its outdegree.

Choose $v_2 \in V$ with $f'(v_2) \neq f(v_2)$ such that $f'(v_2)$ is minimal, subject to the condition $f'(v_2) \neq f(v_2)$. By formulating the corresponding property in terms of D, it is easy to see that $f'(v_2) < f(v_2)$ ($f'(v_2)$ is the smallest nonisolated vertex in D). Then choose $v_1 \in V$ with $f(v_1) < f(v_2)$ and $f'(v_1) \geq f(v_2)$ such that $f(v_1)$ is maximal, subject to the conditions $f(v_1) < f(v_2)$ and $f'(v_1) \geq f(v_2)$. The existence of v_1 follows from the properties of D: Since at least one edge in D (namely, $(f(v_2), f'(v_2))$) leads from $K_{\text{H}} = \{f(v_2), \ldots, k\}$ to $K_{\text{L}} = \{1, \ldots, f(v_2) - 1\}$, at least one edge must lead from K_{L} to K_{H}. Furthermore, $f(v_1) \geq f'(v_2)$. The situation is depicted in Fig. 1. Note that $f'(v_2) = f(v_1)$ and/or $f(v_2) = f'(v_1)$ is possible.

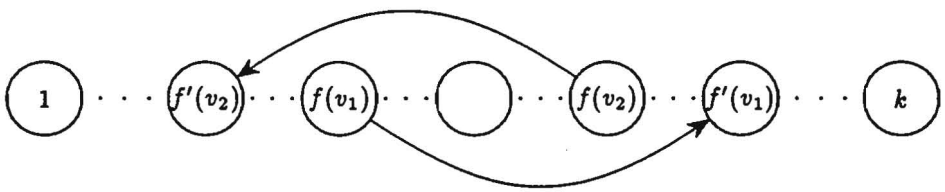

Fig. 1. The identification of an interchangeable pair (v_1, v_2).

We next verify that (v_1, v_2) satisfies the conditions stated in the definition of an interchangeable pair. First of all, it is clear that $v_1, v_2 \notin Dom(\widehat{f})$ and that $f(v_1) < f(v_2)$. In order to verify condition (2), let u be a predecessor of v_2. Since $f'(u) < f'(v_2)$, it follows from the choice of v_2 that $f(u) = f'(u) < f'(v_2) \leq f(v_1)$, as required.

Finally consider condition (3) and let w be a successor of v_1. Since f and f' are solutions to P, $f(w) > f(v_1)$ and $f'(w) > f'(v_1)$. If $f(w) < f(v_2)$, we have a contradiction to the choice of v_1. On the other hand, it follows from condition (2) that $w \neq v_2$. Hence if $f(w) = f(v_2)$, Lemma 5 implies that $f(w) = \mathit{High}(w)$, which leads to the contradiction $f'(w) > f'(v_1) \geq f(v_2) = \mathit{High}(w)$. We may therefore conclude that $f(w) > f(v_2)$. □

Lemma 9. *Given a solution f to P containing an interchangeable pair, an interchangeable pair in f can be identified in $O(n+m)$ time.*

Remark. We cannot compute all interchangeable pairs, since their number may be $\Omega(n^2)$.

Proof. Consider the following algorithm:

```
U_1 := ∅;
for j := 2 to k do
  begin
    U_j := (U_{j-1} ∪ {v ∈ V\Dom(f̂) | f(v) = j − 1})
      \{v ∈ V | f(w) = j for some successor w of v};
    if U_j ≠ ∅
    then
      begin
        Choose z_1 ∈ U_j with f(z_1) maximal;
        for all z_2 ∈ f^{-1}(j)\Dom(f̂)
        do if f(u) < f(z_1) for all predecessors u of z_2
            then output('An interchangeable pair: ', z_1, z_2);
      end;
  end;
```

It is easy to see by induction that for $j = 1, \ldots, k$,

$$U_j = \{v \in V \backslash Dom(\widehat{f}) \mid f(v) \leq j-1 \text{ and } f(w) > j \text{ for all successors } w \text{ of } v\}.$$

Hence every pair (z_1, z_2) reported by the algorithm indeed is an interchangeable pair. On the other hand, suppose that (v_1, v_2) is an interchangeable pair in f. By the characterization of U_j given above, it is easy to see that $v_1 \in U_{f(v_2)}$. Hence in that execution of the loop in which j has the value $f(v_2)$, z_1 will be chosen as some vertex with $f(z_1) \geq f(v_1)$, and the algorithm will report the interchangeable pair (z_1, v_2) (and possibly other pairs). The algorithm hence correctly computes an interchangeable pair, if there is one.

We now consider the complexity of the algorithm. First of all, the elements of V can initially be (bucket-) sorted in $O(n)$ time by their images under f (i.e., labels). If $U_1, \ldots, U_k$ are the successive values assumed by a single variable U implemented as a doubly-linked linear list with its elements sorted by their images under f, it is then possible to carry out the insertions into U in $O(n)$ overall time, the deletions from U in $O(n+m)$ overall time, and the selection of z_1 in $O(1)$ time per execution of the loop. Since the remaining parts of the algorithm can also be executed in $O(n+m)$ time, the lemma follows. □

Theorem 2. *Given an instance $P = (V, E, \widehat{f}, k)$ of a constrained GTS problem with $|V| = n$ and $|E| = m$, it is possible to test in $O(n+m)$ time whether P has more than one solution. If so, two distinct solutions to P can be computed in $O(n+m)$ time.*

Proof. If P is solvable, compute a standard solution $f_{\max}$ and a minimal solution $f_{\min}$ to P. As previously described, this can be done in $O(n+m)$ time. If $f_{\min} \neq f_{\max}$, the two desired solutions to P have been produced. If $f_{\min} = f_{\max}$, compute an interchangeable pair in $f_{\max}$, if one exists. By Lemma 9, this can be done in $O(n+m)$ time. If an interchangeable pair exists, it directly implies a second (maximal) solution to P. If not, it follows from Lemma 8 that $f_{\max}$ is the only solution to P. □

5 Unconstrained GTS Problems

In this section we review the connection between the constrained GTS problem studied in this paper and the original *unconstrained GTS problem* of (Hagerup and Rülling, 1986).

An instance of the unconstrained GTS problem is a tuple $(V, E, \widehat{f})$, where V is a directed, acyclic graph and $\widehat{f}$ is a partial function from V to $\mathbb{N}$. A solution to P is a total function $f : V \to \mathbb{N}$ such that for some $k \in \mathbb{N}$, f is a solution to the constrained GTS problem $(V, E, \widehat{f}, k)$ (informally, the choice of a suitable value for k is left to the algorithm).

Given an unconstrained GTS problem $P = (V, E, \widehat{f})$, denote by P_k the corresponding constrained GTS problem $(V, E, \widehat{f}, k)$, for all $k \in \mathbb{N}$. Further let $M = \max_{v \in V} Low(v)$ (computed with respect to an arbitrary P_k). It is implied in (Hagerup and Rülling, 1986) that the set $\{k \in \mathbb{N} \mid P_k \text{ is solvable}\}$ is either empty or of the form $\{M, M+1, \ldots, k_{\max}\}$, for some integer $k_{\max} \geq M$. A proof

of this fact needs only show that if P_{k+1} is solvable, for some $k \geq M$, then so is P_k. To see the truth of the latter claim, suppose that f is a solution to P_{k+1}, let

$$U = \{v \in V \mid \text{there is a path in } (V,E) \text{ of length } k+1-f(v) \text{ from } v \text{ to a vertex } w \text{ with } f(w) = k+1\}$$

and note that since $k \geq M$, U contains neither vertices in $Dom(\hat{f})$ nor vertices v with $f(v) = 1$. Now consider the function $f' : V \to \mathbb{N}$ with

$$f'(v) = \begin{cases} f(v)-1, & \text{if } v \in U \\ f(v), & \text{if } v \in V \backslash U. \end{cases}$$

We will show that f' is a solution to P_k. The fact noted above shows that f' is an extension of $\hat{f}$. If some vertex $v \in V$ belongs to U, then so does any predecessor u of v with $f(u) = f(v)-1$; hence f' is monotonic. Finally, if some vertex $v \in V$ with $f(v) \leq k$ belongs to U, then so does at least one successor w of v with $f(w) = f(v)+1$, i.e., $f'(V) = \{1,\ldots,k\}$. Hence f' is indeed a solution to P_k.

As argued in (Hagerup and Rülling, 1986), it is now easy to compute solutions to unconstrained GTS problems and to test their uniqueness by answering the same questions for constrained GTS problems. Specifically, with the notation from above, P is solvable if and only if P_M is, and a solution to P is unique if and only if P_M has a unique solution, while P_{M+1} is not solvable.

Acknowledgment. We are grateful to Günter Rote, who pointed out the possible relevance of convex bipartite matching.

References

Gabow, H. N., and Tarjan, R. E. (1985), A Linear-Time Algorithm for a Special Case of Disjoint Set Union, *J. Comput. System Sci.* **30**, pp. 209–221.

Hagerup, T., and Rülling, W. (1986), A Generalized Topological Sorting Problem, *in* Proc. 2nd Aegean Workshop on Computing, Springer Lecture Notes in Computer Science, Vol. 227, pp. 261–270.

Lengauer, T, and Mehlhorn, K. (1984), The HILL System: A Design Environment for the Hierarchical Specification, Compaction, and Simulation of Integrated Circuit Layouts, *in* Proc. Conference on Advanced Research in VLSI, MIT, pp. 139–149.

Lipski, W., Jr., and Preparata, F. P. (1981), Efficient Algorithms for Finding Maximum Matchings in Convex Bipartite Graphs and Related Problems, *Acta Inform.* **15**, pp. 329–346.

Valiant, L. G. (1976), Relative Complexity of Checking and Evaluating, *Inform. Process. Lett.* **5**, pp. 20–23.