Improved approximations for tour and tree covers

Jochen Könemann[1*], Goran Konjevod[2**],
Ojas Parekh[2***], and Amitabh Sinha[1†]

[1] Graduate School of Industrial Administration,
Carnegie Mellon University, Pittsburgh, PA 15213-3890

[2] Department of Mathematical Sciences,
Carnegie Mellon University, Pittsburgh, PA 15213-3890

Abstract. A tree (tour) cover of an edge-weighted graph is a set of edges which forms a tree (closed walk) and covers every other edge in the graph.
Arkin, Halldórsson and Hassin (Information Processing Letters 47:275–282, 1993) give approximation algorithms with ratio 3.55 (tree cover) and 5.5 (tour cover). We present algorithms with worst-case ratio 3 for both problems.

1 Introduction

1.1 Problem statement and notation.

Let $G = (V, E)$ be an undirected graph with a (nonnegative) weight function $c : E \to \mathbb{Q}_+$ defined on the edges. A *tree cover* (*tour cover*) of G is a subgraph $T = (U, F)$ such that **(1)** for every $e \in E$, either $e \in F$ or F contains an edge f adjacent to e: $F \cap N(e) \neq \emptyset$, and **(2)** T is a tree (closed walk). (We allow the tour cover to be a closed walk in order to avoid restricting the weight function c to be a metric. Our algorithm for tour cover produces a closed walk in G, but if the weight function c satisfies the triangle inequality, this walk may be short-cut into a simple cycle which covers all edges in E without increasing the weight.)

The tree cover (tour cover) problem consists in finding a tree cover (tour cover) of minimum total weight:

$$\min \sum_{e \in F} c_e,$$

over subgraphs $H = (U, F)$ which form a tree cover (tour cover) of G.

* Supported in part by the W. L. Mellon Fellowship. (e-mail: `jochen@andrew.cmu.edu`)
** Supported in part by the NSF CAREER grant CCR-9625297. (e-mail: `konjevod@andrew.cmu.edu`)
*** (e-mail: `odp@andrew.cmu.edu`)
† Supported in part by the W. L. Mellon Fellowship. (e-mail: `asinha@andrew.cmu.edu`)

For a subset of vertices $S \subseteq V$, we write $\delta(S)$ for the set of edges with exactly one endpoint inside S. If $x \in \mathbb{R}^{|E|}$ is a vector indexed by the edges of a graph $G = (V, E)$ and $F \subseteq E$ is a subset of edges, we use $x(F)$ to denote the sum of values of x on the edges in the set F, $x(F) = \sum_{e \in F} x_e$.

1.2 Previous work.

The tree and tour cover problems were introduced by Arkin, Haldórsson and Hassin [1]. The motivation for their study comes from the close relation of the tour cover problem to vertex cover, watchman route and traveling purchaser problems. They provide fast combinatorial algorithms for the weighted versions of these problems achieving approximation ratios 5.5 and 3.55 respectively (3.55 is slightly lower than their claim—the reason being the recent improvements in minimum Steiner tree approximation [8]). For unweighted versions their best approximation ratios are 3 (tour cover) and 2 (tree cover), and they also show how to find a 3-approximate tree cover in linear time. Finally, they give approximation-preserving reductions to vertex cover and traveling salesman problem, showing that tree and tour cover are MAXSNP-hard problems.

Our methods are similar to those used by Bienstock, Goemans, Simchi-Levi and Williamson [2], also referred to by Arkin et al. as a possible way of improving their results; however, our algorithms were developed independently and were in fact motivated primarily by the work of Carr, Fujito, Konjevod and Parekh [3] on approximating weighted edge-dominating sets.

1.3 Algorithm overview.

Both our algorithms run in two phases. In the first phase we identify a subset of vertices, and then in the second phase we find a walk or a tree on these vertices. Very informally, the algorithms can be described as follows.

(1) Solve the linear programming relaxation of the tour cover (tree cover) problem.
(2) Using the optimal solution to the linear program, find a set $U \subseteq V$, such that $V \setminus U$ induces an independent set.
(3) Find an approximately optimal tour (tree) on U.

Part **(3)** above reduces to the invocation of a known algorithm for approximating the minimum traveling salesman tour or the minimum Steiner tree.

2 Tour cover

2.1 Linear program.

We first describe an integer programming formulation of tour cover.

Let $\mathcal{F}$ denote the set of all subsets S of V such that both S and $V \setminus S$ induce at least one edge of E,

$$\mathcal{F} = \{S \subseteq V \mid E[S] \neq \emptyset,\ E[V \setminus S] \neq \emptyset\}.$$

Note that if C is a set of edges that forms a tour cover of G, then at least two edges of C cross S, for every $S \in \mathcal{F}$. This observation motivates our integer programming formulation of tour cover. For every edge $e \in E$, let the integer variable x_e indicate the number of copies of e included in the tour cover. We minimize the total weight of edges included, under the condition that every cut in $\mathcal{F}$ be crossed at least twice. In order to ensure our solution is a tour we also need to specify that each vertex has even degree; however, we drop these constraints and consider the following relaxation.

$$\begin{aligned} &\min \sum_{e \in E} c_e x_e \\ &\sum_{e \in \delta(S)} x_e \geq 2 \qquad \text{for all } S \in \mathcal{F} \\ &x \in \{0,1,2\}^{|E|}. \end{aligned} \tag{1}$$

Note that since the optimum tour may use an edge of G more than once, we cannot restrict the edge-variables to be zero-one. However, it is not difficult to see that under a nonnegative weight function the minimal solution will never use an edge more than twice. This follows since an Eulerian tour T_1 on a subset $U \subseteq V$ of vertices may be transformed into an Eulerian tour T_2 on U such that **(1)** no edge is used in T_2 more times than in T_1 and **(2)** no edge is used in T_2 more than twice.

Replacing the integrality constraints by

$$0 \leq x \leq 2,$$

we obtain the linear programming relaxation. We use $\mathrm{ToC}(G)$ to denote the convex hull of all vectors x satisfying the constraints above (with integrality constraints replaced by upper and lower bounds on x).

To show that $\mathrm{ToC}(G)$ can be solved in polynomial time we appeal to the ellipsoid method [7] and construct a separation oracle. We interpret a given candidate solution x as the capacities on the edges of the graph G. For each pair of edges $e_1, e_2 \in E$ we compute the minimum capacity cut in G that separates them. The claim is that x is a feasible solution iff for each pair of edges $e_1, e_2 \in E$ the minimum-capacity e_1, e_2-cut has value at least 2. Clearly, if x is not a feasible solution then our procedure will find a cut of capacity less than 2 having at least one edge on either side. On the other hand if our procedure returns a cut of value less than 2 then x cannot be feasible.

Notice that the dual of $(\mathrm{ToC}(G))$ fits into the packing framework and the above oracle enables us to use fast combinatorial packing algorithms [4, 5]. That is, we avoid using the ellipsoid method, reducing the time complexity but at the cost of losing a $(1+\epsilon)$-factor in the approximation guarantee.

2.2 The subtour polytope.

Let $G = (V, E)$ be a graph whose edge-weights satisfy the triangle inequality: for any u, v, and $w \in V$,

$$c_{uv} + c_{vw} \geq c_{uw}.$$

The *subtour polytope* $\mathrm{ST}(G)$ is defined as

$$\begin{aligned}\mathrm{ST}(G) = \{x \in [0,1]^{|E|} \mid x(\delta(S)) \geq 2 \;\; \forall S \subseteq V,\; \emptyset \neq S \neq V,\\ \text{and } x(\delta(\{v\})) = 2 \;\; \forall v \in V\}.\end{aligned}$$

In fact, the upper-bound constraints $x \leq 1$ are redundant and

$$\begin{aligned}\mathrm{ST}(G) = \{x \geq 0 \mid x(\delta(S)) \geq 2 \;\; \forall S \subseteq V,\; \emptyset \neq S \neq V,\\ \text{and } x(\delta(\{v\})) = 2 \;\; \forall v \in V\}.\end{aligned}$$

2.3 The parsimonious property.

Let $G = (V, E)$ be a complete graph with edge-weight function c. For every pair of vertices i, $j \in V$, let a nonnegative integer r_{ij} be given. The *survivable network design problem* consists in finding the minimum-weight subgraph such that for every pair of vertices i, $j \in V$, there are at least r_{ij} edge-disjoint paths between i and j. A linear programming relaxation of the survivable network design problem is given by

$$\begin{aligned}&\min \sum_{c \in E} c_e x_e \\ &\sum_{e \in \delta(S)} x_e \geq \max_{ij \in \delta(S)} r_{ij} \qquad \text{for all } S \subseteq V,\; \emptyset \neq S \neq V \\ &x \geq 0.\end{aligned} \tag{2}$$

Goemans and Bertsimas [6] prove the following.

Theorem 1. *If the weight function c satisfies the triangle inequality then for any $D \subseteq V$ the optimum of the linear program* (2) *is equal to the optimum of*

$$\begin{aligned}&\min \sum_{c \in E} c_e x_e \\ &\sum_{e \in \delta(S)} x_e \geq \max_{ij \in \delta(S)} r_{ij} \qquad \text{for all } S \subseteq V,\; \emptyset \neq S \neq V \\ &\sum_{e \in \delta(\{v\})} x_e = \max_{j \in V \setminus \{v\}} r_{vj} \qquad \text{for all } v \in D \\ &x \geq 0.\end{aligned} \tag{3}$$

2.4 Algorithm.

We are now ready to state our algorithm for tour cover.

(1) Let x^* be the vector minimizing cx over $\mathrm{ToC}(G)$.
(2) Let $U = \{v \in V \mid x^*(\delta(\{v\})) \geq 1\}$.
(3) For any two vertices $u, v \in U$, if $uv \notin E$, let c_{uv} be the weight of the shortest u-v path in G.
(4) Run Christofides' heuristic to find an approximate minimum traveling salesman tour on U.

The algorithm outputs a tour on U. Since U is a vertex cover of G, this tour is in fact a tour cover of G.

We note that there are some trivial cases which our algorithm will not handle. However, they can be processed separately, and we briefly mention them here. If the input graph is a star, the central node is a solution of weight zero. If the input graph is a triangle, doubling the cheapest edge gives us an optimal solution. All other cases can be handled by our algorithm.

2.5 Performance guarantee.

Theorem 2. *Let x^* be the vector minimizing cx over* $\mathrm{ToC}(G)$ *and $U = \{v \in V \mid x^*(\delta(\{v\})) \geq 1\}$. Let F denote the (complete) graph with vertex-set U and edge-weights c as defined by shortest paths in G. Then*

$$\min\{cy \mid y \in \mathrm{ST}(F)\} \leq 2\min\{cx \mid x \in \mathrm{ToC}(G)\}.$$

Proof. Let $y = 2x^*$. Then, y is feasible for

$$\begin{aligned} A = \{x \geq 0 \mid\ & x(\delta(\{v\})) \geq 0 \ \ \forall v \in V \setminus U \\ & x(\delta(\{u\})) \geq 2 \ \ \forall u \in U \\ & x(\delta(S)) \geq 2 \ \ \forall S \subseteq V,\ S \cap U \neq \emptyset,\ U \setminus S \neq \emptyset,\ \emptyset \neq S \neq V \\ & x(\delta(S)) \geq 0 \ \ \forall S \subseteq V \setminus U,\ S \neq \emptyset\}. \end{aligned}$$

Notice that A corresponds to the survivable network polytope (2) with requirement function

$$r_{uv} = \begin{cases} 2 & , \quad u, v \in U \\ 0 & , \quad \text{otherwise.} \end{cases}$$

Now let

$$\begin{aligned} B^0 = \{x \geq 0 \mid\ & x(\delta(\{v\})) = 0 \ \ \forall v \in V \setminus U \\ & x(\delta(\{u\})) = 2 \ \ \forall u \in U \\ & x(\delta(S)) \geq 2 \ \ \forall S \subseteq V,\ S \cap U \neq \emptyset,\ U \setminus S \neq \emptyset,\ \emptyset \neq S \neq V \\ & x(\delta(S)) \geq 0 \ \ \forall S \subseteq V \setminus U,\ S \neq \emptyset\}. \end{aligned}$$

By the parsimonious property (Theorem 1),

$$\min\{cx \mid x \in A\} = \min\{cx \mid x \in B^0\}.$$

We define

$$\begin{aligned} B = \{x \geq 0 \mid\ & x(\delta(\{v\})) = 0 \ \ \forall v \in V \setminus U \\ & x(\delta(\{u\})) = 2 \ \ \forall u \in U \\ & x(\delta(S)) \geq 2 \ \ \forall S \subset U, \ \ \emptyset \neq S \neq U\}, \end{aligned}$$

that is, B is the subtour polytope $ST(F)$. We next show that $B = B^0$, from which it follows that

$$\min\{cx \mid x \in B\} = \min\{cx \mid x \in A\}. \tag{4}$$

Claim. $B = B^0$.

Proof. It is clear that $B^0 \subseteq B$. Let $x \in B$. Clearly, for $\emptyset \neq S \subset V \setminus U$ we have $x(\delta(S)) \geq 0$. Now, consider some set S with a requirement of 2. We show that $x(\delta(S)) = x(\delta(S \cap U))$. The claim then follows from $x \in B$.

In the following we use $\bar{U}$ to denote $V \setminus U$. We also use $U : V$ to denote the set of edges with exactly one end point in each of U and V, that is, $U : V = \{uv \in E \mid u \in U, \ v \in V\}$. Notice that we can express the difference $x(\delta(S)) - x(\delta(S \cap U))$ in the following way

$$x(S \cap \bar{U} : \bar{S} \cap \bar{U}) + \tag{5}$$

$$x(S \cap \bar{U} : \bar{S} \cap U) - \tag{6}$$

$$x(S \cap \bar{U} : S \cap U). \tag{7}$$

Since $x \in B$ we know that $x(\delta(v)) = 0$ for all $v \in \bar{U}$. Hence the terms (5), (6), and (7) above evaluate to zero. □

The right-hand side of (4) is equal to $\min\{cx \mid x \in \mathrm{ST}(F)\}$. Now, putting together all of the above, we have

$$\begin{aligned} \min\{cx \mid x \in \mathrm{ST}(F)\} &= \min\{cx \mid x \in B\} = \min\{cx \mid x \in A\} \\ &\leq cy = 2cx^* = 2\min\{cx \mid x \in \mathrm{ToC}(G)\}. \end{aligned}$$

The first equality here follows from the definition of B. The second equality is equation (4), and the inequality is true because y is feasible for A. The final two equalities follow from the definitions of y and x^*. □

Wolsey [11] and Shmoys and Williamson [9] prove the following theorem.

Theorem 3. *Let $G = (V, E)$ be a graph with edge-weight function c satisfying the triangle inequality. Then the weight of the traveling salesman tour on G output by Christofides' algorithm is no more than $\frac{3}{2}\min\{cx \mid x \in ST(G)\}$.*

From Theorems 2 and 3, and the fact that $\min\{cx \mid x \in \mathrm{ToC}(G)\}$ is a lower bound on the weight of an optimal tour cover, it follows that the approximation ratio of our algorithm for tour cover can be upper-bounded by 3.

Corollary 1. *The algorithm above outputs a tour cover of weight no more than 3 times the weight of the minimum tour cover.*

3 Tree cover

3.1 Bidirected formulation.

For tree cover, we follow essentially the same procedure as for tour cover, with one difference. We use a bidirected formulation for the tree cover. That is, we first transform the original graph into a directed graph by replacing every undirected edge uv by a pair of directed edges $(u \to v), (v \to u)$ each having the same weight as the original undirected edge. We then pick one vertex as the *root*, and search for a minimum-weight branching which also covers all the edges of the graph. We denote this directed graph by $\overrightarrow{G} = (V, \overrightarrow{E})$.

We do not know which vertex to pick as the root. However, we can simply repeat the whole algorithm for every possible choice of the root, and pick the best solution. It is easy to see that such a branching has a direct correspondence with a tree cover in the original undirected graph, having the same weight.

3.2 Linear program.

For a fixed root r, define $\mathcal{F}$ to be the set of all subsets S of $V \setminus \{r\}$ such that S induces at least one edge of $\overrightarrow{E}$,

$$\mathcal{F} = \{S \subseteq V \setminus \{r\} \mid \overrightarrow{E}[S] \neq \emptyset\}.$$

If C is a set of edges forming a tree cover of G and containing r, then let $\overrightarrow{C}$ denote the branching obtained by directing all edges of C towards the root r. Now for every $S \in \mathcal{F}$, $\overrightarrow{C}$ must contain at least one edge leaving S. We use $\delta^+(S)$ to denote the set of directed edges leaving the set S. Hence we have the following IP formulation.

$$\begin{aligned} &\min \sum_{e \in \overrightarrow{E}} c_e x_e \\ &\sum_{e \in \delta^+(S)} x_e \geq 1 \qquad \text{for all } S \in \mathcal{F} \\ &x \in \{0,1\}^{|\overrightarrow{E}|}. \end{aligned} \tag{8}$$

Replacing the integrality constraints by

$$x \geq 0,$$

we obtain the linear programming relaxation. We use $\mathrm{TrC}(\overrightarrow{G})$ to denote the convex hull of all vectors x satisfying the constraints above.

3.3 Quasi-bipartite bidirected Steiner tree polytope.

A graph $G = (V, E)$ on which an instance of the Steiner tree problem is given by specifying the set $R \subseteq V$ of terminals is called *quasi-bipartite* if $S = V \setminus R$ induces an independent set. Rajagopalan and Vazirani [10] give a $\frac{3}{2}$-approximation algorithm for the quasi-bipartite Steiner tree problem using a *bidirected cut relaxation*.

For a specific choice of a root vertex r, the *quasi-bipartite bidirected Steiner tree polytope* $\mathrm{QBST}(\overrightarrow{G[R]})$ is defined as

$$\mathrm{QBST}(\overrightarrow{G[R]}) = \{x \in [0,1]^{|\overrightarrow{E}|} \mid x(\delta^+(S)) \geq 1 \ \ \forall S \subseteq V \setminus \{r\},\ S \cap R \neq \emptyset\}.$$

3.4 Algorithm.

We are now ready to state our algorithm for tree cover.

(1) For every vertex $r \in V$, let x_r^* be the vector minimizing cx over $\mathrm{TrC}(\overrightarrow{G})$ with r as the root.
(2) Let $U = \{v \in V \mid x_r^*(\delta^+(\{v\})) \geq \frac{1}{2}\}$.
(3) For any two vertices $u, v \in U$, if $uv \notin E$, let c_{uv} be the weight of the shortest u-v path in G.
(4) Run the Rajagopalan-Vazirani algorithm to find an approximate minimum Steiner tree on $\overrightarrow{G}$, with U as the set of terminals, and call this T_r.
(5) Pick the cheapest such T_r.

Note that we are able to solve the linear program in step **(1)** in essentially the same way as the tour cover LP, appealing to the ellipsoid method and using a min-cut computation as a separation oracle. Trivial cases exist for this problem too; they can be handled similar to the way we handle the tour cover trivial cases. The algorithm initially yields a branching in the bidirected graph. We map this in the obvious way to a set of edges in the original undirected graph. Some of the edges in this set may be redundant since we were working on the metric completion of the directed graph; we prune the solution to get a tree without any increase in weight.

The algorithm outputs a tree which spans U (and possibly other vertices). Since U is a vertex cover of G, this tree is in fact a tree cover of G.

3.5 Performance guarantee.

Theorem 4. *Let x^* be the vector minimizing cx over $\mathrm{TrC}(\overrightarrow{G})$ and $U = \{v \in V \mid x^*(\delta^+(\{v\})) \geq \frac{1}{2}\}$. Then*

$$\min\{cy \mid y \in \mathrm{QBST}(\overrightarrow{G[U]})\} \leq 2 \min\{cx \mid x \in \mathrm{TrC}(\overrightarrow{G})\}.$$

Proof. Consider an edge $\overrightarrow{e} = uv \in \overrightarrow{E}$. Since $x^* \in \mathrm{TrC}(\overrightarrow{G})$, we have that $x^*(\delta^+(\{u,v\})) \geq 1$. Hence, either $x^*(\delta^+(\{u\})) \geq \frac{1}{2}$ or $x^*(\delta^+(\{v\})) \geq \frac{1}{2}$, and U is a vertex cover of G. Note that $V \setminus U$ is an independent set because for all $u, v \in V \setminus U$, we have $x(\delta^+(u)) < \frac{1}{2}$ and $x(\delta^+(v)) < \frac{1}{2}$ so that $uv \notin E$.

Now consider the vector $y = 2x^*$. Clearly $cy = 2cx^*$. Also clearly $y \in \mathrm{QBST}(\overrightarrow{G[U]})$. Hence if y^* is the minimizer of $\{cy \mid y \in \mathrm{QBST}(\overrightarrow{G[U]})\}$, then $cy^* \leq cy = 2cx^*$. □

Rajagopalan and Vazirani[10] prove the following.

Theorem 5. *Let $G = (V, E)$ be a graph with edge-weight function c satisfying the triangle inequality. Let $V = R + S$ be a partition of the vertex set such that G has no edges both of whose end points are in S. Then we can find in polynomial time a Steiner tree spanning R of weight no more than $\frac{3}{2}\min\{cx \mid x \in QBST(\overrightarrow{G[R]})\}$.*

From Theorems 4 and 5 it follows that the approximation ratio of our algorithm for tree cover can be upper-bounded by 3.

Corollary 2. *The algorithm above outputs a tree cover of weight no more than 3 times the weight of the minimum tree cover.*

4 Conclusion

4.1 Gap examples: linear program, algorithm.

We do not have examples where the worst-case performance of our algorithm is actually achieved. However, we do have examples where the ratio of our solution to the LP solution is equal to the performance guarantee.

For the tour cover problem, consider the unit complete graph. It is easy to see that an optimal LP solution is obtained by setting $x_e = \frac{1}{n-2}$ for each edge in the graph. This solution has value $\frac{n(n-1)}{2(n-2)} \approx \frac{n}{2}$. Our algorithm will round this to a tree, which could yield a star having $n-1$ edges and all nodes of odd degree. The second stage will then yield a tour having roughly $\frac{3}{2}(n-1)$ edges, which is of weight 3 times the LP solution.

We are not aware of any graph for which the Rajagopalan-Vazirani algorithm achieves its worst case bound of $\frac{3}{2}$. Hence for the tree cover, we do not have an example where the ratio of our solution to even the LP optimum is 3. However, for the complete unit graph, it is easy to see that the integrality gap is at least 2.

4.2 Further open questions.

Obtaining approximation algorithms with better approximation guarantees is an obvious open question. We note that we do not have examples where either algorithm actually achieves its worst-case performance bound, so it may be possible to improve the performance guarantees of our algorithms with tighter analyses. The directed version of both problems remains wide open.

We also note that we use a two stage procedure to solve these problems. A single procedure which directly puts us in the desired polytopes might yield a better approximation ratio.

References

1. E. M. Arkin, M. M. Halldórsson, and R. Hassin. Approximating the tree and tour covers of a graph. *Information Processing Letters*, 47:275–282, 1993.
2. D. Bienstock, M. X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Math. Programming*, 59:413–420, 1993.
3. R. D. Carr, T. Fujito, G. Konjevod, and O. Parekh. A $2\frac{1}{10}$-approximation algorithm for a generalization of the weighted edge-dominating set problem. In *procof "ESA '00"*, 2000.
4. L. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 24–31, 1999.
5. N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *Proceedings of the 39th Annual IEEE Symposium on Foundations of Computer Science*, pages 300–309, 1998.
6. M. X. Goemans and D. J. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Math. Programming*, 60:145–166, 1993.
7. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.
8. G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–779, 2000.
9. D. B. Shmoys and D. P. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. *Information Processing Letters*, 35:281–285, 1990.
10. V. V. Vazirani and S. Rajagopalan. On the bidirected cut relaxation for the metric Steiner tree problem. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 742–751, 1999.
11. L. A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Math. Programming Stud.*, 13:121–134, 1980.