# An Optimal Algorithm for the Indirect Covering Subtree Problem

Joachim Spoerhase

*Lehrstuhl für Informatik I, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany*

*Key words:* Graph algorithm, coverage, medianoid, tree, efficient algorithm

**Introduction** Suppose that a company wants to open a number $r$ of facilities in a given network, modeled by an edge-weighted graph. A potential customer $u$, located at some node of the graph, is willing to use the service provided by that company if the cost thereby incurred is limited by some bound $\varrho(u)$. The costs are modeled by shortest-path distances in the underlying graph, that is, the cost for customer $u$ equals the distance to the closest server of the company. For example, the distances may represent transportation costs or travel times in a logistical network, but also response times in a communication network. If the company serves customer $u$—that is, his distance to the closest server does not exceed $\varrho(u)$—it earns a profit of $w(u)$, which generally corresponds to the demand of the customer. The goal of the company is to identify $r$ locations (nodes of the graph) for its facilities such that the total profit is maximized.

The resulting optimization problem, called *maximum coverage location* [MZH83], is NP-hard for arbitrary graphs. It can, however, be solved in time $O(rn^2)$ on *trees* [Tam96]. As trees form the sparsest (and thus cheapest) networks connecting a given set of nodes they play an important role in many areas of application such as logistical and communication networks. For example, we may think of backbones of a computer network, which are often tree-shaped. This is the case that we investigate in this paper.

In the past years there has been an increasing interest in location problems combined with *connectivity* requirements (for example, connected dominating set, or connected facility location). We follow this line of research and consider a variant of maximum coverage location on a tree where the facilities are

*Email address:* `joachim.spoerhase@uni-wuerzburg.de` (Joachim Spoerhase).

required to form a *connected* subgraph, that is, a subtree of the input tree. Albeit consisting of a plurality of nodes, such a subtree is considered as a *single*, *tree-shaped* facility. The cost of this facility is given by its total edge weight. (The number of nodes within the facility is not bounded.) The goal is to identify a tree-shaped facility that maximizes the net profit, that is, the income produced by the customers minus the setup cost of the facility. As possible application we may think of a company that wants to establish a *high-bandwidth core* (tree-shaped facility) in a given communication network in order to provide some service to potential customers. A customer is only willing to pay for the service if the response time is sufficiently low.

Kim et al. [KLTW96] introduce the *indirect covering subtree problem*, which can be used to model the above scenario. But instead of assigning to each customer a *profit* $w(u)$ they assume that a *penalty* $\pi(u)$ is imposed on the company if customer $u$ is *not* served. The company aims at minimizing the sum of setup cost and the total penalty cost. It should be clear that both formulations are equivalent.

**Problem Definition** The input of the indirect covering subtree problem is an undirected tree $T = (V, E)$ with non-negative edge weights $c\colon E \to \mathbb{R}_{\geq 0}$. The edge weights induce a distance function $d\colon V \times V \to \mathbb{R}_{\geq 0}$ on the node set. Each node is associated with a radius $\varrho(u)$ and a non-negative penalty $\pi(u)$. Consider a subtree $Y$ of $T$. A node $u$ is said to be *covered* (indirectly) by $Y$ if $d(u, Y) \leq \varrho(u)$, that is, if $u$ lies within distance $\varrho(u)$ from $Y$. If $u$ is *not* covered by $Y$, then $u$ imposes a penalty $\pi(u)$ on $Y$. If $U \subseteq V$ is a set of nodes then $p(U, Y) := \sum\{\, \pi(u) \mid u \in U \text{ and } d(u, Y) > \varrho(u) \,\}$ denotes the penalty imposed on $Y$ by $U$. The total penalty imposed on $Y$ is given by $p(Y) := p(V, Y)$. The indirect covering subtree problem asks for a subtree $Y$ of $T$ such that the total cost $c(Y) + p(Y)$, given by the sum of setup and penalty cost, is minimum among all subtrees of $T$.

If we require that $Y$ be a node rather than a subtree we obtain the *single maximum coverage location problem* [MZH83,SW09a]. It is not hard to see that single maximum coverage location is a special case of indirect covering subtree. (Scale all edge lengths and radii with a sufficiently large factor while leaving the penalties unchanged.)

**Previous Results** The maximum coverage location problem (allowing the placement of an arbitrary set of $r$ nodes) is NP-hard on general graphs [MZH83] while it can be solved in time $O(rn^2)$ on trees [Tam96]. This leads to an $O(n^2)$ algorithm for the *single* maximum coverage location problem on trees by setting $r = 1$. Kim et al. [KLTW96] provide a faster algorithm running in $O(n \log^2 n)$. Their algorithm works even for the more general indirect covering subtree problem. Recently a slightly faster $O(n \log^2 n / \log\log n)$-time algorithm for single maximum coverage location has been reported [SW09a].

**Our Contribution** We propose an $O(n \log n)$ algorithm for indirect covering subtree, which is faster than the previously best algorithms for that problem and single maximum coverage location on trees. We complement this result with a matching lower bound on the running showing that our algorithm is optimal.

Our result also implies faster algorithms for competitive location problems [Hak83]. Specifically, we obtain an $O(n \log n)$ algorithm for $(1, X)$-medianoid and $O(n^2 \log n \log w(T))$ and $O(n^2 \log n \log w(T) \log D)$ algorithms for the discrete and absolute $(1, p)$-centroid problems on trees, respectively. Here, $w(T)$ denotes the total node weight of the tree and $D$ is the maximum edge weight. The previously best algorithms are slower by factor of $O(\log n / \log\log n)$ [SW09c].

**Sketch of the Algorithm** We employ the algorithmic framework used by Kim et al. [KLTW96]. We improve, however, one of their core routines by using a more sophisticated technique to subdivide trees. This technique, which we call *two-terminal subtree subdivision*, is a simplification of the recursive coarsening strategy [SW09a] used for solving single maximum coverage location on a tree. The source of our speedup is that we manage to avoid explicitly sorting the nodes according to their distances and radii during the recursion, which has been necessary in the coarsening approach and also in the original algorithm of Kim et al. A further advantage of our algorithm is that it is a lot simpler than the recursive coarsening algorithm.

The two-terminal subtree technique has proved successful also for other location problems [SW10,SW09b]. We believe that there are further problem classes where it can be applied.

Our algorithm is based on the so-called *bitree model* [KLTW96]. A bitree is a directed graph $T'$ that can be derived from an undirected tree $T$ by replacing any edge $(u, v)$ of $T$ with a pair of anti-parallel arcs $(u, v)$ and $(v, u)$. With each arc $(u, v)$ of such a bitree $T'$ we associate a cost $c_{T'}(u, v)$. But in contrast to the edge costs in the indirect covering subtree problem we allow these arc costs to be negative and asymmetric. This induces a distance function $d_{T'} : V \times V \to \mathbb{R}$ where $d_{T'}(u, v)$ is the length of the unique $u$-$v$-path in $T'$. If $v$ is a node and $U$ a set of nodes of $T'$ we define $p'(U, v) := \sum\{\, \pi(u) \mid u \in U \text{ and } d_{T'}(u, v) > \varrho(u) \,\}$ and $p'(v) := p'(V, v)$ similar to the undirected case.

Kim et al. reduce the indirect covering subtree problem to the computation of $p'(v)$ for all nodes of a given bitree. Specifically, they show that if $p'(v)$ can be computed in time $O(h(n))$ for all nodes $v$ of an arbitrary bitree then the indirect covering subtree problem can be solved in the same asymptotic running time on undirected trees. They develop a routine to compute the $p'(\cdot)$-values of an bitree in total time $O(n \log^2 n)$. Thus they can also solve indirect covering subtree in $O(n \log^2 n)$.

We suggest an algorithm that computes all $p'(\cdot)$-values of an arbitrary bitree in $O(n \log n)$, which yields our main result.

We assume that any node of the given bitree $T'$ has out-degree at most three. It is not hard to see, that this is no proper restriction. If $s$ and $t$ are distinct nodes then $T'_{st}$ denotes the maximal sub-bitree of $T'$ having $s$ and $t$ as leaves. We call $s$ and $t$ *terminals of* $T'_{st}$ and $T'_{st}$ *two-terminal sub-bitree* (TTSB) of $T'$.

Our algorithm divides the input bitree $T'$ recursively into TTSBs. Since we are dealing with a degree-bounded bitree we can subdivide any TTSB $T'_{st}$ into five (or fewer) TTSBs, called *child TTSBs of* $T'_{st}$, which have bounded size.

**Lemma 1** *Let $T'_{st}$ be a TTSB with maximum out-degree three. Then $T'_{st}$ can be partitioned into five (or fewer) arc-disjoint TTSBs each of which having at most $\frac{1}{2}|T'_{st}| + 1$ nodes. This subdivision can be computed in $O(|T'_{st}|)$ time.* □

Consider a TTSB $T'_{st}$. We introduce the lists $L_{d,s}(T'_{st})$ and $L_{\varrho,s}(T'_{st})$. Both lists contain all nodes $v$ of $T'_{st}$ sorted in increasing order with respect to the values $d_{T'}(s,v)$ and $\varrho(v) - d_{T'}(v,s)$, respectively. The lists $L_{d,t}(T'_{st})$ and $L_{\varrho,t}(T'_{st})$ are defined symmetrically.

Our algorithm computes $p'(v, T'_{st})$ for all $v \in T'_{st}$ as well as the four lists $L_{d,s}(T'_{st})$, $L_{d,t}(T'_{st})$, $L_{\varrho,s}(T'_{st})$ and $L_{\varrho,t}(T'_{st})$ for any TTSB $T'_{st}$ occurring during the recursion. We show that this information can be propagated inductively from child towards parent TTSBs such that we will have computed $p'(\cdot, T') = p'(\cdot)$ at the top of the recursion. One such propagation step takes time linear in the size $|T'_{st}|$ of the current TTSB $T'_{st}$. Together with Lemma 1 we infer.

**Theorem 2** *The indirect covering subtree problem and hence also single maximum coverage location on a tree can be solved in time $O(n \log n)$.* □

We complement our algorithm with a lower bound $\Omega(n \log n)$ on the running time for solving single maximum coverage location on a tree. To this end we introduce a variant of the set disjointness problem, which needs $\Omega(n \log n)$ time to be recognized on certain real-number RAMs [BAG01]. Finally, we provide a linear time reduction from this problem to single maximum coverage location on a tree.

## References


[BAG01] A. M. Ben-Amram and Z. Galil. Topological lower bounds on algebraic random access machines. *SIAM Journal on Computing*, 31(3):722–761, 2001.

[Hak83] S. L. Hakimi. On locating new facilities in a competitive environment. *European Journal of Operational Research*, 12:29–35, 1983.

[KLTW96] T. U. Kim, T. J. Lowe, A. Tamir, and J. E. Ward. On the location of a tree-shaped facility. *Networks*, 28(3):167–175, 1996.

[MZH83] N. Megiddo, E. Zemel, and S. Hakimi. The maximum coverage location problem. *SIAM Journal on Algebraic and Discrete Methods*, 4(2):253–261, 1983.

[SW09a] J. Spoerhase and H.-C. Wirth. An $O(n\,(\log n)^2/\log\log n)$ algorithm for the single maximum coverage location or the $(1, X_p)$-medianoid problem on trees. *Information Processing Letters*, 109(8):391–394, 2009.

[SW09b] J. Spoerhase and H.-C. Wirth. Optimally computing all solutions of Stackelberg with parametric prices and of general monotonous gain functions on a tree. *Journal of Discrete Algorithms*, 7(2):256–266, 2009.

[SW09c] J. Spoerhase and H.-C. Wirth. $(r, p)$-centroid problems on paths and trees. *Theoretical Computer Science*, 410(47–49):5128–5137, 2009.

[SW10] J. Spoerhase and H.-C. Wirth. Relaxed voting and competitive location under monotonous gain functions on trees. *Discrete Applied Mathematics*, 158:361–373, 2010.

[Tam96] A. Tamir. An $O(pn^2)$ algorithm for the $p$-median and related problems on tree graphs. *Operations Research Letters*, 19:59–64, 1996.