Node-Optimal Connected k-Subgraphs

Dorit S. Hochbaum [*]
Department of Industrial Engineering and Operations Research and
Walter A. Haas School of Business,
University of California, Berkeley

Anu Pathria [†]
Department of Industrial Engineering and Operations Research,
University of California, Berkeley

August 4, 1994

Abstract

In this paper we consider the problem of optimally selecting a set of k nodes in a node-weighted graph, under the requirement that the subgraph induced by the set of nodes selected is connected. This problem arises in a Norwegian off-shore oil-drilling application. We show this problem to be NP-complete on general graphs, but provide a polynomial time algorithm to find an optimal solution for trees. We conclude with some approximation algorithms for the problem on general graphs.

0 Introduction

Consider the *Connected k-Subgraph* problem, defined on a graph whose nodes have non-negative weights, in which the objective is to find a connected subgraph on k nodes such that the sum of the weights of the nodes selected is minimized (or, maximized).

This problem models a situation that arises in a Norwegian off-shore oil-drilling application. Given a set of locations where a facility can be set up, each with a certain cost or benefit, the problem is to select (up to) k contiguous facilities to build, such that the overall benefit is maximized. By modeling the facilities as nodes, with the node weight corresponding to the cost or benefit of the associated facility, and the connectivity of two stations represented by an edge in the graph, we can see that the oil-drilling optimization problem can be modeled by the *Connected k-Subgraph* problem. Another problem that can be formulated by *Connected k-Subgraph* comes from forest harvesting: imagine the problem of optimally selecting k cells to harvest, where each cell has an associated benefit, such that the harvested cells must be contiguous.

In Section 1, we show that both the maximization and minimization *Connected k-Subgraph* problems are NP-hard, even if the input graph is assumed to be either bipartite or planar, and

[*]Research supported in part by ONR contract N00014-91-J-1241.

[†]Author supported in part by an *NSERC '67* scholarship provided by the Natural Sciences and Engineering Research Council of Canada.

the node weights are restricted to be from the set $\{0, 1\}$. In Section 2, we provide a polynomial algorithm, based on dynamic programming, for the problem when the input graph is a tree. Approximation algorithms for both the maximization and minimization problems for general graphs are developed in Section 3. The paper concludes with a summary.

1 Problem Complexity

Because the optimal solution to an instance of the *Connected k-Subgraph* problem is unchanged if a constant value is added/subtracted to each node, our restriction that the node weights be non-negative is not limiting. Also observe that a maximization problem can be converted to a minimization problem (and vice-versa) by multiplying the node weights by -1 (and adding a suitable constant so that node weights are non-negative). [1]

We establish that the *Connected k-Subgraph* maximization problem is NP-hard via a reduction from *Steiner Tree*, which is known to be NP-complete (see [Kar72], [GJ79]); from our comments above, this also implies that the *Connected k-Subgraph* minimization problem is NP-hard.

Theorem 1.1 *The Connected k-Subgraph decision problem is NP-complete.*

Proof: It is clear that the problem is in NP.

Given an instance of *Steiner Tree* on a graph $G = (V, E)$, with $R \subseteq V$ the set of nodes to be included in the tree and all edges having unit weight (the problem remains NP-complete even with this assumption of unit weight edges), the objective is to find a tree of minimum weight that spans the nodes in R. Construct a node-weighted graph $\tilde{G} = (V, E)$ where w_v, the weight assigned to node $v \in V$, is defined as:

$$w_v = \begin{cases} 1 & \text{if } v \in R \\ 0 & \text{otherwise.} \end{cases}$$

Observe that, in general, a collection of n nodes induces a connected subgraph if and only if there exists a tree with $n - 1$ edges spanning the n nodes. It is then easy to see that $\tilde{G}$ has a connected $(k + 1)$-subgraph of weight $|R|$ if and only if G has a Steiner tree of weight k. ■

Notice that in the above proof we reduce an instance of *Steiner Tree* to an instance of the *Connected k-Subgraph* maximization problem on a graph with $\{0, 1\}$ node weights (we could also have reduced to a minimization problem with $\{0, 1\}$ nodes weights by toggling the weights assigned to the nodes in our reduction). This observation is used in the following corollaries to Theorem 1.1:

Corollary 1.1 *The Connected k-Subgraph problem is NP-hard even when restricted to bipartite graphs with $\{0, 1\}$ node weights.*

[1] In fact, the optimal solution is unchanged under any linear transformation applied to the node weights; the objective, however, will be converted from maximization to minimization (or vice-versa) if the linear multiplier is negative.

Proof: Follows from the fact that *Steiner Tree* remains NP-hard for bipartite graphs with unit weight edges (attributed to E. R. Berlekamp in [GJ79]). ■

Corollary 1.2 *The Connected k-Subgraph problem is NP-hard even when restricted to planar graphs with* $\{0,1\}$ *node weights.*

Proof: Follows from the fact that *Steiner Tree* remains NP-hard for planar graphs with unit weight edges (see [GJ77]). ■

2 Optimal Solution for Trees

In this section, we show that an optimal connected k-subgraph on a tree can be found in polynomial time. Our algorithm will rely on the ability to solve a *Multiple-Choice Knapsack* problem. So, we first provide a formulation and algorithm for this knapsack problem.

2.1 The Multiple-Choice Knapsack Problem

Instance: *Given a resource bound* R*, and* M *decisions to be made. Corresponding to each decision stage,* m*, a set* D_m *is given:* D_m *consists of ordered pairs of the form* (b_i^m, r_i^m)*, the benefit and resource usage respectively of selecting decision choice* i*. All resource values are assumed to be integers.*

Optimization Problem: *Select exactly one decision choice for each decision, such that no more than* R *resource units are used, so as to maximize the total benefit.*

Note that in the usual integer *Knapsack* problem there are only two possible choices at each decision stage, one of which corresponds to doing nothing (i.e. can achieve 0 benefit using 0 resource units.)

The *Multiple-Choice Knapsack* problem can be formulated as an integer program. Let x_i^m be an indicator variable equal to 1 if and only if choice i is selected for decision stage m.

$$\begin{array}{rll} \max & \sum_{m=1}^{M} \sum_{i=1}^{|D_m|} b_i^m x_i^m & \\ \text{subject to} & \sum_{m=1}^{M} \sum_{i=1}^{|D_m|} r_i^m x_i^m \leq R & \\ & \sum_{i=1}^{|D_m|} x_i^m = 1, & \text{for } m = 1, \ldots, M \\ & x_i^m \in \{0,1\}, & \forall x_i^m \end{array}$$

The first constraint sets the bound on the total resource usage, as in the usual integer knapsack problem, and the second set of constraints ensures that exactly one choice is selected for each decision stage. We can solve this optimization problem using a dynamic programming approach. For $r = 0, 1, \ldots, R$, let $K(m, r)$ be the optimal set of decision choices for decision stages 1 through m, inclusive, with total resource usage r.

Boundary Conditions:

$$\begin{aligned} K(0,r) &= \begin{cases} 0, & \text{if } r = 0 \\ -\infty, & \text{otherwise.} \end{cases} \\ K(m,r) &= -\infty, \quad \text{for } r < 0. \end{aligned}$$

Recurrence: For $m = 1, \ldots, M$ and $r = 1, \ldots, R$,

$$K(m,r) = \max_{(b_i^m, r_i^m) \in D_m} \{K(m-1, r - r_i^m) + b_i^m\}$$

Solution: $K^* = \max_{0 \le r \le R}\{K(M,r)\}$.

The $K()$ values are calculated in increasing values of m. For each fixed decision stage m and resource value r, $K(m,r)$ can be calculated in $O(|D_m|)$ time. It follows that the overall complexity to find K^* is $O(RD)$, where $D = \sum_{m=1}^{M} |D_m|$. This running time is *pseudo-polynomial* in the size of the input.

2.2 Algorithm for Connected k-Subgraph Problem on Trees

We now show that the *Connected k-Subgraph* problem can be solved in polynomial time on trees. Let n be the number of nodes in the graph $T = (V, E)$, and let w_v be the weight of node v. Without loss of generality, we present a dynamic programming solution to the maximization problem.

Root T at any node, r. For all $v \in V$ and $j = 0, 1, \ldots, k$, define $S(v,j)$ as the value of the optimal connected j-subgraph in T_v, where T_v is the subtree (w.r.t. the rooting) of T with v as root, such that v is included in the j-subgraph.

Boundary Conditions: For $v \in V$ a leaf node in T,

$$S(v,j) = \begin{cases} 0, & \text{if } j = 0 \\ w_v, & \text{if } j = 1 \\ -\infty, & \text{for } j = 2, \ldots, k \end{cases}$$

Recurrence: Let $\Gamma(v)$ denote the set of children of v.

$$S(v,0) = 0$$

For $j = 1, \ldots, k$:

$$\begin{aligned} S(v,j) \;=\; & w_v + \max \sum_{y \in \Gamma(v)} \sum_{i=0}^{j-1} S(y,i) x_i^y \\ & \text{subject to} \sum_{y \in \Gamma(v)} \sum_{i=0}^{j-1} i x_i^y = j - 1 \\ & \qquad \sum_{i=0}^{j-1} x_i^y = 1, \quad \forall\, y \in \Gamma(v) \\ & \qquad x_i^y \in \{0,1\}, \quad \forall\, y \in \Gamma(v),\ i = 0, \ldots, j-1 \end{aligned}$$

Solution: $S^* = \max_{v \in V}\{S(v,k)\}$.

Now, for a given node $v \in V$, the optimization problems presented in the recurrence expressions for the $S(v,j)$'s can be solved via a single *Multiple-Choice Knapsack* problem: set $R = k$, and let each decision stage correspond to a child $y \in \Gamma(v)$ (so, $M = |\Gamma(v)|$), where for each child a choice must be made as to what size of subgraph rooted at that child to select (so, $|D_m| = k+1$, for all decisions m).

Then, *for the fixed node* v, $K(M,j)$ corresponds to $S(v,j)$, so that all $S(v,j)$ can be calculated in time $O(k^2\,|\Gamma(v)|)$. Because there are $n-1$ total children in T, it follows that all $S(v,j)$ ($\forall v \in V$, and for all integers $0 \leq j \leq k$) can be computed in time $O(k^2 n)$.

Theorem 2.1 *The complexity of the dynamic programming procedure for calculating* S^**, the optimal solution to the Connected* k*-Subgraph problem, is* $O(k^2 n)$*, which is polynomial in the size of the input.*

3 Approximation Algorithms for General Graphs

We show that, for general graphs $G = (V,E)$, a k-approximate solution can be found to the *Connected k-Subgraph* minimization problem and that a $\frac{1}{k}$-approximate solution can be found to the *Connected k-Subgraph* maximization problem, both in polynomial time. While these may not be very satisfactory approximation guarantees, at least they show that polynomial approximation algorithms do exist whose running time and approximation guarantee do *not* depend upon the actual node weights (we only assume that the nodes weights are non-negative).

3.1 $\frac{1}{k}$-Approximation Algorithm for Maximization Problem

Consider the connected components of G, and let w be the maximum weight of a node in a connected component with at least k nodes. Clearly, the optimal solution to the *Connected k-Subgraph* maximization problem does not exceed kw. But, by including a node of weight w, we can easily construct a solution of weight at least w.

3.2 k-Approximation Algorithm for Minimization Problem

The nature of our k-approximation algorithm for the minimization problem is reminiscent of the approach taken in [HS86] to develop approximation algorithms for certain bottleneck optimization problems.

Let the nodes of G be numbered $1,\ldots,n$ so that $w_1 \leq \ldots \leq w_n$. Define G_i as the subgraph of G induced by nodes whose weight does not exceed w_i. Consider the following test:

Test(G_i)**:** *Does* G_i *have a connected component with at least* k *nodes?*

Clearly, **Test**(G_{i-1}) = "*no*" implies that the optimal solution to the *Connected k-Subgraph* minimization problem is at least w_i. On the other hand, **Test**(G_i) = "*yes*" implies the existence of a solution with weight not exceeding kw_i. The following k-approximation algorithm follows:

```
1. Find $i^* = \min_i\{\mathbf{Test}(G_i) = \text{“}yes\text{”}\}$ .
2. Return a connected subgraph of $G_{i^*}$ on $k$ nodes.
```

The above procedure can be implemented either via a binary search method for finding i^* (fewer calls to $\mathbf{Test}()$, but more work per call), or via an incremental iterative search for i^* (more calls to $\mathbf{Test}()$, but less work per call since G_i's build on each other).

4 Summary

In this paper, we have defined the *Connected k-Subgraph* problem, and established several complexity results. First, we established that the problem is NP-hard, even when the input graph is highly restricted. Second, we have shown that the problem can be solved in polynomial time, using dynamic programming, when the input graph is a tree. Finally, we have provided $O(k)$ approximation algorithms for both the maximization and minimization problems.

We have considered the problem in which exactly k nodes are to be selected; the approximation algorithms also assume that the node weights are non-negative. It would be desirable to establish complexity results, similar to the ones mentioned, for the problem in which *up to k* nodes are to be selected, such that a connected subgraph is induced, where there is no assumption on the sign of the node weights. In particular, it would be desirable to explicitly establish that this problem is NP-hard, and also to derive approximation results; observe, however, that because the $O(k^2n)$ dynamic programming procedure presented for trees does not make any assumption on the sign of the node weights, and that an optimal solution for the *Connected j-Subgraph* problem for all $j = 1, 2, \ldots, k$ is returned, it is also an $O(k^2n)$ algorithm for this problem.

References

[GJ77] M.R. Garey and D. S. Johnson. The Rectilinear Steiner Tree Problem is NP-complete. *SIAM Journal on Applied Mathematics*, 32:826–834, 1977.

[GJ79] M.R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[HS86] D. S. Hochbaum and D. Shmoys. A Unified Approach to Approximation Algorithms for Bottleneck Problems. *Journal of the Association for Computing Machinery*, 33(3):533–550, July 1986.

[Kar72] R. M. Karp. *Reducibility Among Combinatorial Problems*, pages 85–103. Complexity of Computer Computations. Plenum Press, New York, 1972. R. E. Miller and J. W. Thatcher (eds.).