A Recursive Greedy Algorithm for Walks in Directed Graphs

Chandra Chekuri* Martin Pál†

Abstract

Given an arc-weighted directed graph $G = (V, A, \ell)$ and a pair of nodes s, t, we seek to find an s-t walk of length at most B that maximizes some given function f of the set of nodes visited by the walk. The simplest case is when we seek to maximize the number of nodes visited: this is called the orienteering problem. Our main result is a quasi-polynomial time algorithm that yields an $O(\log \mathrm{OPT})$ approximation for this problem when f is a given submodular set function. We then extend it to the case when a node v is counted as visited only if the walk reaches v in its time window $[R(v), D(v)]$.

We apply the algorithm to obtain several new results. First, we obtain an $O(\log \mathrm{OPT})$ approximation for a generalization of the orienteering problem in which the profit for visiting each node may vary arbitrarily with time. This captures the time window problem considered earlier for which, even in undirected graphs, the best approximation ratio known [4] is $O(\log^2 \mathrm{OPT})$. The second application is an $O(\log^2 k)$ approximation for the k-TSP problem in directed graphs (satisfying asymmetric triangle inequality). This is the first non-trivial approximation algorithm for this problem. The third application is an $O(\log^2 k)$ approximation (in quasi-poly time) for the group Steiner problem in undirected graphs where k is the number of groups. This improves earlier ratios [15, 19, 8] by a logarithmic factor and almost matches the inapproximability threshold on trees [20]. This connection to group Steiner trees also enables us to prove that the problem we consider is hard to approximate to a ratio better than $\Omega(\log^{1-\epsilon} \mathrm{OPT})$, even in undirected graphs.

Even though our algorithm runs in quasi-poly time, we believe that the implications for the approximability of several basic optimization problems are interesting.

*Lucent Bell Labs, 600 Mountain Avenue, Murray Hill, NJ 07974. chekuri@reserch.bell-labs.com

†DIMACS Center, CoRE Bldg 4th Floor, Rutgers University, Piscataway, NJ 08854. mpal@acm.org. Part of this work was done while the author was at Lucent Bell Labs.

1 Introduction

A number of interesting combinatorial optimization problems can be cast as a path or tour problem in a graph. The most well known of these is the classical traveling salesman problem (TSP) which asks for a shortest tour that visits all nodes in a graph. Other related problems include the Chinese postman problem, the minimum latency problem, and the k-TSP problem. In these problems we seek to minimize the tour length subject to certain constraints on the nodes visited by the tour. A different class of problems is motivated by *maximizing* some function of the nodes visited, subject to a *budget* on the tour length. The simplest of these is the *orienteering* problem in which the goal is to visit the maximum number of nodes subject to a budget B on the tour length. In particular we are interested in the *rooted* version where the tour or path begins at a given start node s. We are allowed to visit a node multiple times although we obtain a profit only for the first visit. These class of problems are also sometime referred to as *prize-collecting* traveling salesman or repairman problems. These budgeted problems are of much importance in real world applications such as vehicle routing and its variants. They arise from operational issues such as assigning technicians to maintenance jobs or delivering goods to locations. A substantial amount of work on heuristics for these problems can be found in operations research literature. Typically there are several other constraints such as time-windows (each node has an interval $[R(v), D(v)]$ in which it can be visited) and vehicles with capacity constraints. These problems are also known to be difficult to solve (exactly) in practice for even moderately sized instances. The book [25] provides a recent and comprehensive survey on vehicle routing.

Despite the importance of these types of problems, there are few positive algorithmic results in the theoretical computer science literature and some of them are quite recent. One of the difficulties is that the constraints are rigid: budget on the length of the path or time windows or vehicle capacities. A second difficulty is that these problems combine aspects of *scheduling* and *network design* and the algorithmic tools that we have for them do not overlap strongly. Nevertheless, in some recent work [1, 7, 4, 10], progress has been made on some fundamental problems. In [7], Blum *et* *al.* gave the first constant factor approximation algorithm for the orienteering problem and some related problems in metric spaces (induced by undirected graphs). Surprisingly, this was the first non-trivial approximation algorithm for this problem. Prior to this work, a $(2+\epsilon)$-approximation was known for orienteering in geometric settings [1]. The current best ratio for orienteering in metric spaces is 3 due to Bansal *et al.* [4]. Bansal *et al.* [4], building upon the algorithm for orienteering, gave algorithms for the time window version of the problem. They obtain a ratio of $O(\log^2 \mathrm{OPT})$ for the time window problem and a ratio of $O(\log \mathrm{OPT})$ for the simpler problem when all release dates are the same.

The main idea in [7] for the orienteering problem is to transform an approximation algorithm for the rooted k-TSP problem into an approximation algorithm for the orienteering problem. In the rooted k-TSP problem, the objective is to find a minimum length tour that visits at least k-nodes and includes a specified node r. Although the rooted k-TSP and orienteering problems are equivalent in the exact sense, an approximation algorithm for k-TSP does not yield a corresponding approximation algorithm for orienteering. However, in [7], a clever idea is used to combine dynamic programming with the use of k-TSP to obtain an algorithm for orienteering. A constant factor approximation for the rooted k-TSP and related problems such as k-MST are known in undirected graphs. The algorithms in [4] for the time window problem rely on the algorithm for orienteering in a black box fashion and also apply to directed graphs. However, there are no good algorithms known for k-TSP and related problems in directed graphs.

In this paper we take a different approach to these problems. We give a *recursive greedy* algorithm with the recursion reminiscent of the well-known algorithm of Savitch for directed s-t connectivity [24] that relates non-deterministic and deterministic space complexities. We show that this surprisingly simple algorithm has an $O(\log \mathrm{OPT})$ approximation ratio for a number of walk problems in *directed* graphs including the basic orienteering problem. The algorithm works even when we seek to maximize an arbitrary submodular set function on the set of nodes visited by the walk. We discuss some interesting consequences of this. There is however a price to pay for this generality. The algorithm runs in *quasi-polynomial* time. We nevertheless believe that the results are of interest from both an algorithmic and complexity theoretic point of view. We obtain the first non-trivial approximation algorithms for several seemingly difficult optimization problems. Our algorithmic result also yields a lower bound on reduction sizes for inapproximability bounds. And our results add to a growing number of problems for which quasi-polynomial time algorithms yield reasonable approximation ratios but no polynomial time algorithms achieving comparable ratios are known. We now formally define the problems that we consider.

Orienteering: The input to an orienteering problem is given by a directed arc weighted graph $G = (V, A, \ell)$ with $\ell(u, v)$ denoting the length of arc (u, v), two nodes $s, t \in V$, not necessarily distinct, and a budget $B > 0$. Our goal is to find an s-t walk P of length at most B, to maximize reward collected. The reward is determined by the set $V(P)$ of nodes visited by the walk P. For plain orienteering, the reward is simply $|V(P)|$, the number of distinct nodes visited, but we will be interested in more general rewards.

A *reward function* $f : 2^V \mapsto \mathbb{Z}^+$ assigns a non-negative integer reward to every subset of nodes. The function f is *submodular* if for all $A, B \subset V$ and $u \in V$, $f(A \cup \{u\}) - f(A) \geq f(B \cup \{u\}) - f(B)$ whenever $A \subseteq B$. An equivalent definition is that for all $A, B \subset V$, $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$. We say that f is *monotone* if $f(A) \leq f(B)$ for all $A \subseteq B$.

Given an arc weighted directed graph G, two nodes s, t, a budget B and a monotone submodular function f, the *submodular orienteering problem* (SOP) is to find an s-t walk P of length at most B that maximizes $f(V(P))$. We assume that the function f is given by an oracle that returns $f(S)$ when presented with a set $S \subseteq V$.

Time windows: In a time window version of SOP (SOP-TW), each node v has a time window $[R(v), D(v)]$ during which it can be visited: $R(v)$ and $D(v)$ are the release time and deadline of v. We associate a notion of time with walks in G. A *timed walk* P is given by a sequence $s = v_0, v_1, \ldots, v_k = t$ of nodes together with a sequence $T_0, T_1, \ldots, T_k$ of times satisfying $T_0 = 0$, $T_k \leq B$ and $T_i + \ell(v_i, v_{i+1}) \leq T_{i+1}$. Intuitively, T_i is the time at which P visits v_i. It takes $\ell(u, v)$ time to travel from node u to v, but the walk is allowed to travel slower or stall. Given a timed walk P, let $V_T(P)$ be the set of nodes v_i such that $T_i \in [R(v_i), D(v_i)]$. In SOP with time windows (SOP-TW), the goal is to find a timed s-t walk maximizing $f(V_T(P))$ for a given monotone submodular function f.

We note that processing times at nodes can be easily modeled by adding extra arcs and dummy nodes. This is an easy and well-known transformation and we omit the details. Throughout, OPT denotes the value of an optimum solution to the given problem.

Results: Our main result is an algorithm for SOP with time windows (SOP-TW) that runs in quasi-polynomial time and provides a solution of value $\Omega(\mathrm{OPT}/\log \mathrm{OPT})$. More precisely, for any fixed integer $h \geq 2$, we obtain an algorithm with a running time of $(n \log B)^{O(h \log n)}$ that yields an $O(\log \mathrm{OPT}/\log h)$ approximation where B is an upper bound on the length of the tour. Note that this allows us to obtain a *sub-logarithmic* approximation in *sub-exponential* time. We also prove that SOP is hard to approximate to within a factor of $\Omega(\log^{1-\epsilon} \mathrm{OPT})$ unless NP is contained in randomized quasi-polynomial time. Several interesting results follow from the algorithm for SOP-TW. In the following we do not explicitly qualify the approximation bounds as requiring quasi-polynomial time.

First we discuss applications of the algorithm for SOP (without time windows). A first and immediate application is an $O(\log \mathrm{OPT})$ approximation for orienteering in directed graphs. Another application is an $O(\log^2 k)$ approximation for the k-TSP problem in directed graphs. These two results are the first non-trivial approximation algorithms for these problems. A related but surprising consequence is an $O(\log^2 k)$ approximation for the group Steiner problem in *undirected* graphs [15] where k is the number of distinct groups. More precisely, an α-approximation for SOP implies an $O(\alpha \log k)$ approximation for the group Steiner problem. Here we use the submodular property of the reward function. This improves upon the $O(\log^3 k)$ bound achieved in quasi-polynomial time[1] in [8] and the $O(\log^2 k \log n)$ bound achieved in polynomial time[2] [19] (here n is the number of nodes in the graph). The bound we achieve almost matches the inapproximability bound of $\Omega(\log^{2-\epsilon} k)$ on tree instances established by Halperin and Krauthgamer [20]. The inapproximability bound is shown under the assumption that NP has no quasi-polynomial time Las-Vegas algorithms. In fact, as is typical with several PCP based reductions, the reduction reduces 3SAT instances to quasi-polynomial sized instances of the group Steiner problem. It is commonly assumed that with better parameters and methods, such inapproximability results can be improved to obtain a polynomial sized reduction. A consequence of the our algorithm for SOP is that one can obtain an $O(\log^{1+\delta} k)$ approximation for the group Steiner problem for any fixed $\delta > 0$ in sub-exponential time. This shows that an inapproximability bound of $\Omega(\log^{1+\epsilon} k)$ for the group Steiner problem for any fixed $\epsilon > 0$ requires a quasi-polynomial sized reduction from 3SAT (unless NP has sub-exponential time algorithms).

We now discuss the applications of the algorithm for SOP-TW. The first application is to the orienteering problem with time windows for which we obtain an $O(\log \mathrm{OPT})$ approximation. This improves upon the approximation ratio of $O(\log^2 \mathrm{OPT})$ in [4] and works in directed graphs (the algorithm in [4] works in polynomial time so it is not an apples to apples comparison). More interestingly, the combination of single time window and submodularity of the reward function allows us to handle problems in which each node may have multiple time windows during which it is active, and we get a reward for that node only if we visit it during an active window. Our algorithm also applies to problems in which the reward for visiting each node is a function of the time of the visit (if a node is visited more than once, we only get reward for the most profitable visit). An example problem in which the profit of a node varies with time is the discounted-reward TSP [7] in which the profit for visiting a node v at time t is given by $\pi_v \gamma^{t_v}$. We obtain a logarithmic approximation for these problems when the distances are asymmetric.

Related Work: There is a large amount of work on problems related to those that we consider in this paper. Here we briefly discuss some of the more directly related literature. Most of the work on approximation algorithms for route traversal problems is on metric spaces induced by undirected graphs where the distances are symmetric and satisfy the triangle inequality. We will explicitly mention results for directed graphs. A basic problem underlying some of the variants we mentioned is the prize-collecting traveling salesman problem (PC-TSP) [3]. In this problem, each node has a a profit for visiting it and a penalty for not visiting it: the goal is to find a tour that obtains at least a given profit while minimizing the length of the tour and the sum of the penalties of the nodes not visited. Goemans and Williamson [16] gave a primal dual 2-approximation algorithm for a special case of PC-TSP in which there are no profits. Their algorithm and analysis formed the basis for subsequent work on related problems such as the k-MST and k-TSP problems and the minimum latency problem. For the k-TSP problem the current best ratio is 2 due to Garg [14]. The orienteering problem [17] is closely related to PC-TSP. Arkin, Mitchell and Narasimhan gave a $(2+\epsilon)$-approximation when points are in the Euclidean plane and more recently a constant factor approximation was devised for metric spaces by Blum *et al.* [7] with an improvement of the ratio to 3 by Bansal *et al.* [4]. The time window version of orienteering has also been studied. Tsitsiklis [26] showed that the problem is NP-hard even if the metric space is a line. Bar-Yehuda, Even, and Shahar [6] obtained an $O(\log \mathrm{OPT})$ approximation for the line case. Bansal *et al.* [4] gave an $O(\log^2 \mathrm{OPT})$ approximation for general metric spaces and an improved $O(\log \mathrm{OPT})$ approximation if all release dates are the same. Chekuri and Kumar [10] gave a constant factor approximation if the number of distinct time windows is a fixed constant independent of the input size. These two latter results use an algorithm for orienteering in a black box fashion.

The time window problem is related to the problem of scheduling jobs with interval constraints. In this setting, each job has a processing time and can be scheduled in one

[1] In [8] an $O(\log^2 k)$ approximation in quasi-polynomial time is claimed for the directed Steiner tree problem which implies the same ratio for the group Steiner tree problem. However, the analysis in [8] is based on an erroneous lemma of Zelikovsky in [27] and when a correct version of the lemma (see [21]) is used, the ratio obtained from the analysis in [8] becomes $O(\log^3 k)$.

[2] The first poly-logarithmic approximation for the group Steiner problem was given by Garg, Konjevod and Ravi [15]. The bound of $O(\log^2 k \log n)$ is based on subsequent improvements for probabilistic tree embeddings [12] and a more refined analysis in [19]. These results also establish an approximation ratio of $O(\log N \log k \log n)$ where N is the maximum group size. Note that N and k are incomparable.

of several disjoint time windows. Given some number of machines, each of which can process only one job at a time, the goal is to find a non-preemptive schedule to maximize the profit of jobs that are completed. We can also allow the processing times and profits to vary with the interval in which the job is scheduled. A constant factor approximation for this general problem can be found in [5]. We remark that SOP-TW generalizes this scheduling problem in a straight forward way.

The approximability of TSP and related problems have been much less understood in directed graphs. Asymmetric-TSP has an $O(\log n)$ approximation [13, 22] and it is a major open problem to decide whether there is a constant factor approximation. The directed Steiner tree problem with k terminals and the k-MST problem have an $O(i^3 k^{1/i})$ approximation in $n^{O(i)}$ time [8]. Thus we can obtain an $O(\log^3 k)$ approximation in quasi-polynomial time. The related group Steiner problem [15] in undirected graphs has had more success. An $O(\min\{\log k, \log N\} \log n \log k)$ approximation is known [15, 19] where k is the number of groups, N is the maximum group size, and n is the number of nodes in the graph. It is also known that this problem is hard to approximate to within a factor of $\Omega(\log^{2-\epsilon} k)$ unless NP is contained in randomized quasi-polynomial time [20]. We refer the reader to [15, 8, 23, 9, 19, 20] for more details on these problems.

2 Preliminaries

We are interested in walks in directed graphs and the reward function is a monotone submodular function. This allows us to assume without loss of generality that the arc lengths of the given directed graph satisfy the *asymmetric triangle inequality*; that is, for all triples u, v, w of nodes, $\ell(u,v) + \ell(v,w) \geq \ell(u,w)$. With this assumption, we can restrict attention to paths and cycles instead of walks and in particular we can assume that each node is visited only once. We will also assume that all arc lengths are integers.

For simplicity, we will assume that f is an integer valued submodular function. Given a submodular function f on V and a subset $X \subseteq V$ we define a new submodular function f_X on V as $f_X(S) = f(S \cup X) - f(X)$. The following fact will be useful in the analysis.

Fact 2.1 *Let f be a monotone submodular set function on V. Then for any $A \subseteq B \subseteq V$, $f_A(S) \geq f_B(S)$ for all $S \in 2^V$.*

In the SOP-TW problem, we are interested in a timed walk. However, given a path $P = v_0, v_1, \ldots, v_k$, we can check in polynomial time if the nodes can be visited in their time windows while respecting the sequence imposed by P. Therefore, it is sufficient to output a path instead of a timed walk.

Polynomially Bounded Rewards: For computational purpose, given an arbitrary instance of SOP-TW, we transform into an instance in which $f(V)$ and OPT are poly-bounded in n, the number of nodes in the graph. For any given $\epsilon > 0$, we can ensure that this transformation loses only a $(1+\epsilon)$ factor in the approximation ratio. The basic idea is to *guess* the node v with largest profit that an optimum tour visits and restrict attention to nodes $u \in V$ such that $f(\{u\}) \leq f(\{v\})$. Once we do this, if f is modular, we can use standard scaling and rounding ideas to ensure that all values are integers and polynomially bounded. However, for submodular f we cannot do this directly but we can still work with scaling and rounding in an implicit fashion. We define a new function f' where $f'(S) = \lfloor n^2 f(S)/(\epsilon f(\{v\})) \rfloor$ and use $f'(S)$ in place of $f(S)$. We observe that f' need not be submodular, however we can argue that using f' in place of f does not affect the approximation ratio of our algorithms by more than a factor of $(1 + O(\epsilon))$. In this version of the paper, we omit the details and for simplicity assume that f is integer valued and poly-bounded when necessary.

All logarithms in the paper are to base 2.

3 The Recursive Greedy Algorithm

In this section we describe and analyze the algorithm for SOP. Generalization to SOP-TW is relatively straight forward. We first discuss an algorithm that achieves an $O(\log \mathrm{OPT})$ approximation and defer the improvement to a ratio of $O(\log \mathrm{OPT}/\log h)$ to Section 3.3. The algorithm is simple and we describe it at a high level before formalizing it. In the following, it is useful to think of f as the simple function $f(S) = |S|$ which captures the orienteering problem. Let $P^* = s = v_0, v_1, \ldots, v_k = t$ be an optimal walk. We refer to $v_{k/2}$ as the *middle* node. The algorithm *guesses* the middle node *and* the length B' to reach the middle node and recursively calls itself two times. Informally, the algorithm is as follows.

- guess the middle node v and the length B' to reach v from s.
- recursively find a walk P_1 from s to v with budget B'.
- recursively find a walk P_2 from v to t with budget $B - B'$ to *augment* the nodes visited by P_1.
- output the walk obtained by the concatenation of P_1 and P_2.

We note that the second recursive call depends on the first in an essential way since it requires knowledge of which nodes have already been visited by P_1. This is the *greedy* aspect of the algorithm and is critical to the analysis of its performance. The algorithm implements the guessing step by enumerating all possibilities for the middle node v as well as the length B'. In the above description we did not specify a stopping condition for the recursion. The intuition is that we are breaking the optimum walk into two pieces with roughly equal number of nodes, and hence the recursion should have depth $\log k$. As stated, the running time of the algorithm will be $O((2nB)^{\log k})$ which need not be quasi-polynomial if B is super-polynomial. To facilitate the analysis and later present a modified algorithm with an improved running time, we now give a detailed and formal description of the above informal algorithm.

We define a procedure $\mathrm{RG}(s,t,B,X,i)$ that implements the algorithm. We first explain the parameters. The parameters s,t and B indicate that we seek an s-t walk of length at most B. The parameter X indicates that we seek to find a path P so as to maximize $f_X(P) = f(V(P)\cup X) - f(X)$; that is we seek to find a path that augments a set X. The parameter i indicates the depth of the recursion allowed.

Algorithm: $\mathrm{RG}(s,t,B,X,i)$

1. If $(\ell(s,t) > B)$ return **Infeasible**
2. $P \leftarrow s,t$
3. Base case: $i=0$. return P
4. $m \leftarrow f_X(P)$
5. For each $v \in V$ do
   (a) For $1 \le B_1 \le B$ do
      i. $P_1 \leftarrow \mathrm{RG}(s,v,B_1,X,i-1)$
      ii. $P_2 \leftarrow \mathrm{RG}(v,t,B-B_1,X\cup V(P_1),i-1)$
      iii. If $(f_X(P_1\cdot P_2) > m)$
         $P \leftarrow P_1 \cdot P_2$
         $m \leftarrow f_X(P)$
6. return P

Proposition 3.1 *The running time of* $\mathrm{RG}(s,t,B,X,i)$ *is* $O((2nB)^i \cdot T_f)$ *where* T_f *is the maximum time to compute* f *on a given set.*

The heart of the paper is in the analysis of the lemma below. Recall that all logarithms are to base 2. Let $\mathcal{P}(s,t,B)$ denote the set of all s-t paths of length at most B.

Lemma 3.2 *Let* $P^* \in \mathcal{P}(s,t,B)$ *such that* $P^* = (s = v_0, v_1, \ldots, v_k = t)$. *Let* P *be the path returned by* $\mathrm{RG}(s,t,B,X,i)$. *If* $i \ge \lceil 1+\log k\rceil$, *then* $f_X(P) \ge f_X(P^*)/\lceil 1+\log k\rceil$.

Proof. The proof is by induction on k. The base case is when $k=1$ in which case $P^* = s,t$. It is easy to see that the algorithm checks the path s,t and hence $f_X(P) \ge f_X(P^*)$.

Now suppose $k>1$. Let $v = v_{\lceil k/2\rceil}$ be the middle node in P^* and let P_1^* and P_2^* be the subpaths of P^* from s to v and v to t respectively. Let $B_1 = \ell(P_1^*)$ and $B_2 = \ell(P_2^*)$. Consider the procedure $\mathrm{RG}(s,t,B,X,i)$ with $i \ge \lceil 1+\log k\rceil$. From the description of the algorithm, we see that $\mathrm{RG}(s,t,B,X,i)$ calls $P_1 = \mathrm{RG}(s,v,B_1,X,i-1)$ and $P_2 = \mathrm{RG}(v,t,B_2,X\cup V(P_1),i-1)$. We argue now, using induction, that the path $P = P_1\cdot P_2$ has the property that $f_X(P) \ge f_X(P^*)/\lceil 1+\log k\rceil$. This will establish the lemma.

Note that P_1^* and P_2^* have $\lceil k/2\rceil$ and $\lfloor k/2\rfloor$ edges respectively, and that $\lceil 1+\log\lceil k/2\rceil\rceil = \lceil \log k\rceil \le i-1$. Since P_1^* is a candidate for path from s to v of length at most B_1, by induction we have

$$f_X(P_1) \ge \frac{1}{\lceil \log k\rceil} f_X(P_1^*) \tag{1}$$

Let $X' = X \cup V(P_1)$. Since P_2^* is a candidate path from v to t of length at most B_2, again by induction we have

$$f_{X'}(P_2) \ge \frac{1}{\lceil \log k\rceil} f_{X'}(P_2^*). \tag{2}$$

We observe that

$$\begin{aligned} f_{X'}(P_2^*) &= f(P_2^* \cup (P_1 \cup X)) - f(P_1 \cup X) \\ &\ge f_X(P_2^* \cup P_1) - f_X(P_1). \end{aligned}$$

Using this in (2), we obtain that

$$\begin{aligned} f_{X'}(P_2) &\ge \frac{1}{\lceil \log k\rceil}(f_X(P_2^*\cup P_1) - f_X(P_1)) \\ &\ge \frac{1}{\lceil \log k\rceil}(f_X(P_2^*) - f_X(P)). \end{aligned} \tag{3}$$

The inequality above follows from monotonicity of f. Adding (1) and (3) and using submodularity of f_X, we get

$$\begin{aligned} f_X(P) &\ge \frac{1}{\lceil \log k\rceil}(f_X(P_1^*) + f_X(P_2^*) - f_X(P)) \\ &\ge \frac{1}{\lceil \log k\rceil}(f_X(P^*) - f_X(P)). \end{aligned}$$

Rearranging terms, we obtain that

$$f_X(P) \ge \frac{1}{1+\lceil \log k\rceil} f_X(P^*).$$

□

The above lemma is essentially inspired by the algorithm and analysis in [10] for a fixed number of time windows coupled with the idea of recursion.

3.1 Quasi-Polynomial Time Algorithm

To obtain a logarithmic approximation, the algorithm described in the previous section takes $O((2nB)^{\log k})$ time where k is the length of the path of an optimum path. If B is large (super-polynomial in n), the running time need not be quasi-polynomial in the input size. Here we modify the algorithm to obtain an algorithm with a running time of $O((n \cdot \text{OPT} \cdot \log B)^{\log k})$. More precisely, if there is a path P^* with k nodes then the algorithm runs in $O((n \cdot f(P^*) \cdot \log B))^{\log k})$ and yields an $O(\log k)$ approximation. As we mentioned in Section 2, we can assume that $f(P^*)$ is poly-bounded in n. This allows us to obtain a quasi-polynomial time algorithm. We now describe the modified algorithm. The basic idea is to use binary search to guess B_1 instead of enumerating all possible values. The algorithm assumes an upper bound A on the value of $f(P^*)$.

Algorithm: RG-QP(s, t, B, X, i)

1. If $(\ell(s,t) > B)$ return **Infeasible**
2. $P \leftarrow s, t$
3. Base case: $i = 0$. return P
4. $m \leftarrow f_X(P)$
5. For each $v \in V$ do
   (a) For $1 \leq a \leq A$ do
      i. $B_1 \leftarrow \min_b(\text{RG-QP}(s, v, b, X, i-1) \geq a)$
      ii. If $B_1 = \infty$, continue
      iii. $P_1 \leftarrow \text{RG-QP}(s, v, B_1, X, i-1)$
      iv. $P_2 \leftarrow \text{RG-QP}(v, t, B - B_1, X \cup V(P_1), i-1)$
      v. If $(f_X(P_1 \cdot P_2) > m)$
         $P \leftarrow P_1 \cdot P_2$
         $m \leftarrow f_X(P)$
6. return P

The step $B_1 = \min_b(\text{RG-QP}(s, v, b, X, i-1) \geq a)$ in the above algorithm is implemented as a binary search over $[1, B]$ and hence takes $\log B$ recursive calls to RG-QP$(s, v, _, X, i-1)$. The running time analysis is straight forward.

Proposition 3.3 *The running time of* RG-QP(s, t, B, X, i) *is* $O((2 + nA \log B)^i \cdot T_f)$ *where* T_f *is the maximum time to compute* f *on a given set.*

The proof of the following lemma is very similar to that of Lemma 3.2 and hence we omit it in this version.

Lemma 3.4 *Let* $P^* \in \mathcal{P}(s, t, B)$ *and let* k *be the number of edges in* P^*. *Let* P *be the path returned by* RG-QP(s, t, B, X, i). *If* $i \geq \lceil 1 + \log k \rceil$ *and* $A \geq f_X(P^*)$, *then* $f_X(P) \geq f_X(P^*)/\lceil 1 + \log k \rceil$.

Proposition 3.3, the poly-boundedness of $f_X(P^*)$, and Lemma 3.4 yield the theorem below.

Theorem 3.5 *For the submodular orienteering problem (SOP) there is an algorithm with running time* $(n \log B)^{O(\log n)}$ *that yields an* $O(\log \text{OPT})$ *approximation.*

3.2 Time Windows

We now generalize the recursive greedy algorithm to handle time windows on the nodes. Recall that each node v has a time window $[R(v), D(v)]$ during which it can be visited. To make the description of the recursive algorithm cleaner we will now require that the walk starts at s at time σ and has to reach t by time τ. In the algorithm for SOP, σ was 0 and τ was B. The modification to the basic algorithm is simple: we need to ensure that the middle node is visited during its time window. We describe the time window version of RG-QP below.

Algorithm: RG-QP-TW$(s, t, \sigma, \tau, X, i)$

1. If $(\tau < R(t)$ or $\ell(s,t) + \sigma > D(t))$ return **Infeasible**
2. $P \leftarrow s, t$
3. Base case: $i = 0$. return P
4. $m \leftarrow f_X(P)$
5. For each $v \in V$ do
   (a) For $1 \leq a \leq A$ do
      i. $\mu \leftarrow \min_b(\text{RG-QP}(s, v, \sigma, b, X, i-1) \geq a)$
      ii. If $\mu = \infty$, continue
      iii. $P_1 \leftarrow \text{RG-QP}(s, v, \sigma, \mu, X, i-1)$
      iv. $P_2 \leftarrow \text{RG-QP}(v, t, \mu, \tau, X \cup V(P_1), i-1)$
      v. If $(f_X(P_1 \cdot P_2) > m)$
         $P \leftarrow P_1 \cdot P_2$
         $m \leftarrow f_X(P)$
6. return P

The proof of the following theorem is based on similar ideas to those presented earlier for the case without time windows.

Theorem 3.6 *For the submodular orienteering problem with time windows (SOP-TW), there is an algorithm with running time* $(n \log B)^{O(\log n)}$ *that provides an* $O(\log \text{OPT})$ *approximation where* B *is an upper bound on the tour length.*

3.3 Improved Approximation Ratio

We outline how to improve the approximation ratio to $O(\log \text{OPT}/\log h)$ while increasing the running time to $(n \log B)^{O(h \log n/\log h)}$. In the algorithm RG for SOP, we guessed the middle node $v_{k/2}$ of an optimum path $P^* = s = v_0, v_1, \ldots, v_k = t$ and recursed twice in a greedy sequential fashion on subpaths P_1^* and P_2^*. Given an integer parameter $h \geq 2$, to improve the approximation ratio, we guess several nodes on P^*, namely $v_{i_1}, v_{i_2}, \ldots, v_{i_{h-1}}$ where $i_1 = k/h$, $i_2 = 2k/h$ and more generally for $1 \leq j < h$, $i_j = jk/h$. These nodes break up the path P^* into h subpaths $P_1^*, P_2^*, \ldots, P_h^*$ and we also guess the budget used by P^* in each of these subpaths. Then we recurse sequentially in a greedy fashion on each of these subpaths. Now the depth of the recursion is $O(\log k/\log h)$. We can prove that the approximation ratio is proportional to the depth which yields the desired result. Since we guess h nodes and their budgets at each level of the recursion, the running time goes up to $(nB)^{O(h \log k/\log h)}$. We improve this to $(n \log B)^{O(h \log k/\log h)}$ using the idea from Section 3.1.

4 Applications

We discuss some applications of the algorithms for SOP and SOP-TW. We omit some details in this version.

4.1 Orienteering with Multiple Disjoint Time Windows

Orienteering with time windows is a special case of SOP-TW. We demonstrate that a more general version can also be cast as a special case of SOP-TW. Consider the following problem. As in orienteering, we are given an arc weighted graph G and we are interested in an s-t walk of maximum profit. The profit function is defined as follows. Each node v has ℓ disjoint time windows $[R_1(v), D_1(v)]$, $[R_2(v), D_2(v)]$, …, $[R_\ell(v), D_\ell(v)]$ and corresponding profit values $p_1(v), p_2(v), \ldots, p_\ell(v)$. Visiting v in its ith time window yields a profit of $p_i(v)$ and if v is visited multiple times, only the most profitable visit is retained. We are assuming that all nodes have the same number of time windows for simplicity - the case where they have different numbers is easily captured by the uniform case by adding dummy time windows. It is relatively easy to show that this version of the orienteering problem with multiple disjoint time windows for each node is a special case of SOP-TW and hence we obtain a logarithmic approximation for this as well. By choosing the number of time windows appropriately, we can closely approximate any arbitrary time varying profit function for each node. Most common profit functions can be approximated by a polynomial number of time windows. The most general case is when the profit for visiting v at time t, $p(v, t)$ is given as an oracle. For this case we can obtain a running time quasi-polynomial in nB. Using a stronger oracle, that for a given a time interval $[t_1, t_2]$ and vertex v, returns the value $\max_{t \in [t_1, t_2]} p(v, t)$, we again obtain running time quasi-polynomial in $n \log B$.

4.2 Rooted k-TSP in Directed Graphs

In the rooted k-TSP problem, we are given an arc weighted graph $G = (V, A, \ell)$ and a specified root node r. The goal is to find a tour of minimum length starting at r that contains at least k nodes. Let OPT be the length of such a tour. Suppose we have an upper bound B for OPT. Using the algorithm for SOP with a budget of B, we can find a tour of length B that contains $\Omega(k/\log k)$ nodes. We can remove the nodes visited by the tour and run the algorithm again to cover more nodes. Using standard analysis for covering problems, after $O(\log^2 k)$ iterations, the algorithm will cover k nodes. We can stitch the tours together to obtain a tour of length $O(\log^2 k \cdot B)$ that contains k nodes. We can use binary search to guess B to within a constant factor of OPT. This yields an $O(\log^2 k)$ approximation for k-TSP in directed graphs.

4.3 Group Steiner and Covering Steiner Problems

We consider two network design problems in *undirected* graphs. The input to the group Steiner problem consists of an edge weighted graph $G = (V, E, \ell)$ and a k subsets of nodes $g_1, g_2, \ldots, g_k$ called *groups*. The goal is to find a minimum weight tree $T = (V(T), E(T))$ in G such that $V(T) \cap g_i \neq \emptyset$ for $1 \leq i \leq k$. We focus on the rooted version of the problem: we are given a special root node r and we require that $r \in V(T)$. The covering Steiner problem generalizes the group Steiner problem. In this problem each group g_i has a positive integer coverage requirement d_i. Now the goal is to find a minimum cost tree T such that $|V(T) \cap g_i| \geq d_i$.

In this section we obtain algorithm for the above two problems via an algorithm for SOP. We discuss the covering Steiner problem since it captures the group Steiner problem as a special case. From the given problem instance we define a monotone submodular set function f as follows: for each set $S \subseteq V(G)$, we define $f(S) \doteq \sum_{i=1}^{k} \min(d_i, |S \cap g_i|)$. The quantity $f(S)$ is the covering requirement that is satisfied by S. It is easy to verify that f is indeed a monotone submodular set function. Further, given S, $f(S)$ can be computed in polynomial time.

Given an instance of the covering Steiner problem, consider an optimum solution T^* of cost OPT. Since the graph is undirected, by taking an Euler tour of T^* we obtain a tour from r of length at most 2OPT that covers all groups. Suppose we have an upper bound B for OPT. By using SOP with function f defined as above we obtain a tour P of length at most $2B$ that covers $f(V(T^*))/\log f(V(T^*))$. Note that $f(V(T^*)) = \sum_i d_i$. We can remove the nodes in P, reduce the requirements of the groups that are already partially covered and repeat the above procedure. Using standard set cover style arguments, in $O(\log^2(\sum_i d_i)$ iterations, all the groups will be covered to their requisite amount. Putting together the tours yields tree of length $O(\log^2 \sum_i d_i)B$ that is a feasible solution. We can use binary search to find a B that is within a constant factor of OPT and hence we obtain an $O(\log^2 \sum_i d_i)$ approximation. When specialized to the group Steiner problem the ratio becomes $O(\log^2 k)$ where k is the number of groups.

The above discussion implies that an α-approximation for SOP in undirected graphs implies an $O(\alpha \log k)$ approximation for the group Steiner problem in undirected graphs. Halperin and Krauthgamer [20] have shown that the group Steiner problem is hard to approximate to within an $\Omega(\log^{2-\epsilon} k)$ factor unless NP has quasi-polynomial time Las-Vegas algorithms. Under the same assumption, they also establish an inapproximability bound of $\Omega(\log^{2-\epsilon} n)$ where n is the number of nodes in the group Steiner instance. Using this and the above lemma, we obtain the following.

Theorem 4.1 *The submodular orienteering problem (SOP) in undirected graphs is hard to approximate to within a factor of* $\Omega(\log^{1-\epsilon} \mathrm{OPT})$ *unless NP* $\subseteq$ *ZTIME*$(n^{polylog(n)})$.

Note that, using the algorithm in Section 3.3, we can obtain an $O(\log^{1-\delta} \mathrm{OPT})$ approximation for SOP in time $\exp(O(\log^2 n \cdot 2^{\log^\delta n}))$ which is sub-exponential for any fixed $\delta < 1$. Combined with Theorem 4.1, this might appear to imply that NP is contained in sub-exponential time. This is not the case since the hardness in Theorem 4.1 relies on a quasi-polynomial size reduction. However, it shows that the reduction size in [20] cannot be improved to polynomial, unless NP has sub-exponential time algorithms. Indeed, assuming NP is not contained in sub-exponential time, any reduction showing that the group Steiner problem is hard to approximate within $\Omega(\log^{2-\epsilon} n)$ for some $\epsilon < 1$, must necessarily have superpolynomial size. We formalize this in the theorem below.

Theorem 4.2 *For some* $0 < \delta < 1$, *suppose there is a reduction* g *that maps 3SAT instances of size* n *to group Steiner instances of size* $N = \exp(o(\log^{1/\delta} n))$ *such that an* $O(\log^{2-\delta} N)$ *approximation for the group Steiner problem would decide the satisfiability of 3SAT. In addition suppose* g *runs in sub-exponential time as a function of* n. *Then 3SAT has a sub-exponential time algorithm.*

For the covering Steiner problem, an approximation ratio of $O(\min\{N, \log n \log N \log k\})$ is known from prior work [18, 23, 11]. Here n is the number of nodes in the given graph and N is the maximum size of a group. Thus the ratio improves with decreasing N. We comment that the dependence on N stems from the LP relaxation on which the algorithms are based. The integrality gap of the LP improves as N decreases because the graph is *undirected* (when $N = 1$ the problem becomes a Steiner tree problem). In contrast, our algorithm yields a ratio of $O(\log^2 \sum_i d_i)$ irrespective of N since it does not distinguish between directed and undirected instances. We note that, on trees, the known approximation ratio for covering Steiner tree is $O(\log N \log k)$ and the additional $\log n$ factor for general graphs is the result of approximating a graph by a tree. It is an open problem whether this extra factor is necessary. The worst integrality gap for the natural relaxation is $\Omega(\log N \log k / \log \log N)$ and it applies to trees [19] and no worse gap is known for graphs. Our results indicate that an $O(\log N \log k)$ approximation ratio is possible even in graphs.

5 Conclusions

The most interesting open problem resulting from our work is to obtain polynomial time algorithms for SOP and SOP-TW that have a poly-logarithmic approximation ratio. The hardness of approximation for SOP is based on the hardness of the group Steiner problem. The hardness applies also to undirected graphs because we allow a submodular reward function - otherwise the basic orienteering problem has a constant factor approximation in undirected graphs [7]. An interesting question is whether the basic orienteering problem in directed graphs has a constant factor approximation. The approximability of the orienteering problem with time windows is open in both undirected and directed graphs. In fact the problem is open even on the line where the best approximation ratio known is $O(\log \mathrm{OPT})$ [6] while we know only NP-hardness [26]. For the group Steiner and covering Steiner problems, our results present good evidence that the extra $\log n$ factor lost in approximating a graph by a tree is perhaps not inherent and better rounding procedures might yield an $O(\log N \log k)$ approximation via the natural LP, matching the ratio achieved for trees.

Acknowledgments: We are grateful to Rajat Bhattacharjee and Amit Kumar for collaborations on related problems that led to this work. We thank Sanjeev Khanna for discussions and comments on quasi-polynomial time, reductions, and complexity, in particular for help in establishing Theorem 4.2. Thanks to Guy Even, Ashish Goel, and Guy Kortsarz for discussions on the group Steiner problem. We also thank the reviewers for suggestions that improved the presentation. Chandra Chekuri acknowledges support from an ONR grant MA14681000 to Lucent Bell Labs. Martin Pál was supported by an NSF grant EIA 02-05116 while at DIMACS.

References

[1] E. Arkin, J. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. *Proc. of ACM SoCG*, 307–316, 1998.

[2] S. Arora and G. Karakostas. A $2 + \epsilon$ approximation algorithm for the k-mst problem. *Proc. of ACM-SIAM SODA*, 754–759, 2000.

[3] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[4] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation Algorithms for Deadline-TSP and Vehicle Routing with Time-Windows. *Proc. of ACM STOC*, 166–174, 2004.

[5] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor and B. Schieber. A unified approach to approximating resource allocation and scheduling. *JACM*, 48(5):1069–1090, 2001.

[6] R. Bar-Yehuda, G. Even and S. Shahar. On Approximating a Geometric Prize-Collecting Traveling Salesman Problem with Time Windows. To appear in *J. of Algorithms*. Preliminary version in *Proc. of ESA*, 55–66, 2003.

[7] A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *Proc. of IEEE FOCS*, 46–55, 2003.

[8] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha and M. Li. Approximation Algorithms for Directed Steiner Problems. *J. of Algorithms*, 33, p. 73–91, 1999.

[9] C. Chekuri, G. Even, and G. Kortsarz. A greedy approximation algorithm for the group Steiner problem. To appear in *Discrete Applied Math*. Manuscript 2002.

[10] C. Chekuri and A. Kumar. Maximum Coverage Problem with Group Budget Constraints and Applications. *Proc. of APPROX-RANDOM*, LNCS, 72–83, 2004.

[11] G. Even, G. Kortsarz and W. Slany. On network design problems: fixed cost flows and the covering Steiner problem. *Proc. of SWAT*, LNCS, 318–327, 2002.

[12] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Proc. of ACM STOC*, 448-455, 2003.

[13] A. Frieze, G. Galbiati and M. Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks* 12, 23–39, 1992.

[14] N. Garg. Saving an ϵ: A 2-approximation for the k-MST problem in graphs. *Proc. of ACM STOC*, 396–402, 2005.

[15] N. Garg and G. Konjevod and R. Ravi, A polylogarithmic approximation algorithm for the Group Steiner tree problem. *J. of Algorithms*, 37, 66–84, 2000. Preliminary version in *Proc. of ACM-SIAM SODA*, 1998.

[16] M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. *SIAM J. on Computing*, 24:296–317, 1995.

[17] B. Golden, L. Levy and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.

[18] A. Gupta and A. Srinivasan. On the Covering Steiner Problem. *Proc. of FST&TCS*, LNCS, 244–251, 2003.

[19] E. Halperin, G. Kortsarz, R. Krauthgamer, A. Srinivasan and N. Wang. Integrality ratio for Group Steiner Trees and Directed Steiner Trees. *Proc. of ACM-SIAM SODA*, 275–284, 2003.

[20] E. Halperin and R. Krauthgamer. Polylogarithmic Inapproximability. *Proc. of ACM STOC*, 585–594, 2003.

[21] C. H. Helvig, G. Robins, and A. Zelikovsky. Improved approximation scheme for the group Steiner problem. *Networks*, 37(1):8–20, 2001.

[22] H. Kaplan, M. Lewenstein, N. Shafir and M. Sviridenko. Approximation Algorithms for Asymmetric TSP by Decomposing Directed Regular Multidigraphs. *Proc. of IEEE FOCS*, 56–67, 2003.

[23] G. Konjevod, R. Ravi and A. Srinivasan. Approximation Algorithms for the Covering Steiner Problem. *Random Structures & Algorithms*, vol 20, 465–482, 2002.

[24] Walter J. Savitch. Relationships Between Nondeterministic and Deterministic Tape Complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[25] The Vehicle Routing Problem. P. Toth, D. Vigo eds, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

[26] J. Tsitsiklis. Special Cases of Traveling Salesman and Repairman Problems with Time Windows. *Networks*, vol 22, 263–282, 1992.

[27] A. Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18: 99–110, 1997.