Maximum Coverage Problem with Group Budget Constraints and Applications

Chandra Chekuri[1] and Amit Kumar[2]

[1] Bell Labs, 600 Mountain Avenue
Murray Hill, NJ 07974
Email : chekuri@research.bell-labs.com
[2] Department of Computer Science and Engineering
IIT Delhi, INDIA - 110016
Email : amitk@cse.iitd.ernet.in

Abstract. We study a variant of the maximum coverage problem which we label the maximum coverage problem with group budget constraints (MCG). We are given a collection of sets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ where each set S_i is a subset of a given ground set X. In the maximum coverage problem the goal is to pick k sets from $\mathcal{S}$ to maximize the cardinality of their union. In the MCG problem $\mathcal{S}$ is partitioned into *groups* $G_1, G_2, \ldots, G_\ell$. The goal is to pick k sets from $\mathcal{S}$ to maximize the cardinality of their union but with the additional restriction that at most *one* set be picked from each group. We motivate the study of MCG by pointing out a variety of applications. We show that the greedy algorithm gives a 2-approximation algorithm for this problem which is tight in the *oracle* model. We also obtain a constant factor approximation algorithm for the cost version of the problem. We then use MCG to obtain the first constant factor approximation algorithms for the following problems: (i) multiple depot k-traveling repairmen problem with covering constraints and (ii) orienteering problem with time windows when the number of time windows is a constant.

1 Introduction

In this paper we are interested in a variant of the *set cover* problem and its maximization version, the *maximum coverage* problem. The set cover problem is the following: we are given a ground set X of n elements and a set of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ such that for $1 \leq i \leq m$, $S_i \subseteq X$. The objective is to find a minimum number of sets from $\mathcal{S}$ such that their union is X. In the cost version of the set cover problem each set S_i has a cost $c(S_i)$ and we seek a minimum cost collection of sets to cover all the elements of X. The maximum coverage problem is a close relative of the set cover problem. We are given a ground set X and $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of subsets of X. In addition we are given an integer k and the goal is to pick at most k sets from $\mathcal{S}$ such that the size of their union is maximized. In the profit version of the maximum coverage problem the items in X have profits and the goal is to maximize the profit of items picked.

Set cover and maximum coverage problems are fundamental algorithmic problems that arise frequently in a variety of settings. Their importance is partly due to the fact that many *covering* problems can be reduced to these problems. The *greedy* algorithm that iteratively picks the set that covers the maximum number of uncovered elements is a $(\ln n + 1)$ approximation for the set cover problem and an $\frac{e}{e-1}$ approximation [3] for the maximum coverage problem [15]. Feige [12] showed that these ratios are optimal unless NP is contained in quasi-polynomial time. In a number of applications the set system is implicitly defined and the number of sets is exponential in the number of elements. However, the greedy algorithm can still be applied if a polynomial time *oracle* that returns a set with good properties is available.

In this paper, motivated by several applications, we study a variant of the maximum coverage problem that we call the maximum coverage problem with group budget constraints (MCG). We start with a simple motivating example, the multiple knapsack problem (MKP), and the analysis of the greedy algorithm for that problem in [9]. MKP is a generalization of the classical knapsack problem to several knapsacks: we are given n items, where item i has a size s_i and a profit p_i; we are also given m knapsacks potentially of different capacities $b_1, b_2, \ldots, b_m$. The objective is to find a maximum profit subset of items that can be feasibly packed into the given set of knapsacks. We first consider the case where all the knapsacks have the same capacity b. For simplicity, in the following discussion, we assume that we have an exact algorithm to solve the single knapsack problem. We can apply the following greedy algorithm: pick an unused knapsack and use the single knapsack algorithm to pack it optimally with a subset of items from the remaining items. It is easy to show, via standard set cover style arguments, that this algorithm gives an $\frac{e}{e-1}$ approximation [9]. That the greedy algorithm gives this ratio can also be seen via a reduction to the maximum coverage problem as follows. Let $\mathcal{S}$ be the set of all distinct subsets of the items that can be feasibly packed into a knapsack of size b. The MKP problem can be rephrased as a maximum coverage problem on this implicit exponential sized set system and we are required to pick m sets. The greedy algorithm that we described for the MKP can be seen to be the greedy algorithm for maximum coverage problem on the implicit set system above where the oracle is the optimal single knapsack algorithm.

Now we consider instances of MKP in which the knapsack sizes could be different. In this case too we can define a greedy algorithm, however since the knapsacks are not identical, it is not a priori clear in which order to consider them. In [9] it is shown that *irrespective* of the ordering, the greedy algorithm results in a 2-approximation. Once again it is instructive to understand the implicit set system. Let $\mathcal{S}_i$ be set of distinct subsets of items that can be feasibly packed into a knapsack of size b_i and let $\mathcal{S} = \cup_i \mathcal{S}_i$. It is clear that if the knapsacks are not identical, we no longer have a maximum coverage problem although the greedy algorithm still gives a constant factor approximation algorithm. The

[3] In this paper approximation ratios for both minimization and maximization problems will be greater than or equal to 1.

problem we have is the following: we are required to choose at most *one* set from each of the $\mathcal{S}_i$ and cover as many elements as possible by the union of sets picked.

Motivated by MKP and other problems that we consider later in this paper, we define a variant of the maximum coverage problem that provides an abstraction for covering problems that have *groups* of sets.

Maximum Coverage with Group Budgets (MCG): We are given subsets $S_1, S_2, \ldots, S_m$ of a ground set X. We are also given sets $G_1, \ldots, G_\ell$, each G_i being a subset of $\{S_1, \ldots, S_m\}$. We call G_i a *group*. By making copies of sets, if necessary, we can assume that the groups G_i are disjoint from each other. We define two versions of the problem, the cardinality version and the cost version.

In the cardinality version, we are given an integer k, and an integer bound k_i for each group G_i. A solution is a subset $H \subseteq \{S_1, \ldots, S_m\}$ such that $|H| \leq k$ and $|H \cap G_i| \leq k_i$ for $1 \leq i \leq \ell$. The objective is to find a solution such that the number of elements of X covered by the sets in H is maximized. In fact, we can assume without loss of generality that all k_i are equal to 1. Otherwise, we can make k_i copies of each group G_i.

In the cost version, we associate a cost $c(S_j)$ with each set S_j. Further, we are given a budget B_i for group G_i, $1 \leq i \leq \ell$, and an overall budget B. A solution is a subset $H \subseteq \{S_1, S_2, \ldots, S_m\}$ such that the total cost of the sets in H is at most B. Further for any group G_i, the total cost of the sets in $H \cap G_i$ can be at most B_i. The objective is to find such a subset H to maximize the size of the union of sets in H.

In many applications, m, the number of given subsets of X, is exponential in n and the sets are defined implicitly. In such cases we require a polynomial time *oracle* with some desirable properties. We now make this more precise. In the cardinality version, we assume there exists an oracle $\mathcal{A}$ that takes as input, a subset X' of X, and an index i. $\mathcal{A}(X', i)$ outputs a set $S_j \in G_i$ such that $|S_j \cap X'|$ is maximized over all sets in G_i. We also work with approximate oracles. $\mathcal{A}$ is an α-approximate oracle if $\mathcal{A}(X', i)$ outputs a set $S_j \in G_i$ such that $|S_j \cap X'| \geq \frac{1}{\alpha} \max_{D \in G_i} |D \cap X'|$. For the cost version, we shall assume we are given an oracle $\mathcal{B}$ that takes as input a subset X' of X and a group index i. $\mathcal{B}(X', i)$ outputs a set $S_j \in G_i$ such that $\frac{|S_j \cap X'|}{c(S_j)}$ is maximized – we shall assume that all sets in G_i have cost at most B_i. As with the cardinality case we also work with approximate oracles.

We note that the maximum coverage problem is a special case of the cardinality case of MCG and the budgeted maximum coverage problem [17] is a special case of the cost version of MCG.

In this paper we show that a simple greedy algorithm gives a constant factor approximation for both the cardinality and the cost versions of MCG. The analysis differs from the usual analysis for the maximum coverage problem and is based on the analysis for MKP in [9]. For the cardinality version we show that greedy gives a 2-approximation and our analysis is tight. For the cost version we obtain a 12-approximation. The greedy algorithm works in the oracle model as

well and in fact this is the main thrust of the paper and leads to the applications that we mention below.

We note that for the cardinality version, an $\frac{e}{e-1}$-approximation is achievable if the input is given in an explicit form. This ratio is achieved by rounding an LP relaxation either by the pipage rounding technique of Ageev and Sviridenko [1] or the probabilistic rounding technique of Srinivasan [18].

Set Cover with Group Budgets (SCG): We define the set cover problem with group constraints. We only consider the cardinality version in this paper, it is easy to extend it to the cost case. We are given a ground set X of n items and a set $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of subsets of X. The set $\mathcal{S}$ is partitioned into groups $G_1, G_2, \ldots, G_\ell$. The objective is to find a subset H of S such that all elements of X are covered by sets in H and $\max_{i=1}^{\ell} |H \cap G_i|$ is minimized. Note that if we have a single group containing all sets then the problem is the same as the set cover problem. Elkin and Kortsarz [10] seem to be the first to consider this problem for its applications and they call it the multiple set cover problem. They present an $O(\log n)$ approximation using a randomized rounding of a natural linear programming relaxation. Kortsarz [16] asked if there is a combinatorial algorithm for this problem. From the 2-approximation bound on Greedy we obtain a simple combinatorial $(\log n + 1)$ algorithm for SCG. It also has the advantage of working in the oracle model.

We consider two problems for which we design the first constant factor approximation algorithms by using reductions to instances of MCG in the oracle model. We describe them next.

Multiple Depot k-Traveling Repairmen Problem: The k-traveling repairmen problem was considered by Fakcharoenphol, Harrelson, and Rao [11]. Their problem is the following. We are given a finite metric space on a set of nodes V induced by an edge weighted undirected graph G, and k not necessarily distinct vertices $s_1, s_2, \ldots, s_k$ from V. A feasible solution to the problem is a set of k tours $T_1, \ldots, T_k$, with tour T_i starting at s_i such that every vertex in V is visited by one of the tours. Given a feasible solution, we define the *latency* of a vertex as the time at which it gets visited by one of these tours. The objective is to minimize the sum of the latencies of the vertices in V. The problem models the case where k-repairmen are available at the k locations (depots) and we need to visit all the sites that have repairs. The goal is to minimize the average waiting time of the sites. If $k = 1$, the problem is the same as the the minimum latency problem for which a constant factor approximation was first given by Blum et al. [5]. The current best approximation ratio for the minimum latency problem is 3.59 [8]. In [11] a constant factor approximation algorithm is presented for the k-traveling repairmen problem when all the repairmen start at the same vertex s, that is $s_1 = s_2 = \ldots = s_k = s$. In the same paper [11], the generalization of the k-traveling repairmen problem to the case with multiple sources (depots) is left as an open problem. We obtain a constant factor approximation for this problem. We also obtain constant factor approximation algorithms for several generalizations as well.

Orienteering (or TSP) with Time Windows: The orienteering problem is defined as follows. We are given a metric space on a set of vertices V, a starting vertex s and a budget B. A feasible solution is a tour starting at s and having length at most B. The objective is to maximize the number of vertices covered by this tour. Blum et al. [7] gave the first constant factor approximation algorithm for this problem in general graphs, they obtained a ratio of 5 which has recently been improved to 3 in [6]. Previously, a $2+\epsilon$-approximation was known for the case when the metric was induced by points in the Euclidean plane [2].

In this paper we consider the more general problem where we associate a window $[r_v, d_v]$ with each vertex v. A vertex v can be visited only in the time interval $[r_v, d_v]$. We shall say that r_v is the *release time* of v and d_v is the *deadline* of v. The objective, again, is to find a path that starts at s and maximizes the number of vertices visited, however we can only visit a vertex within its time window. This problem models a variety of situations when technicians or robots have to visit a number of different locations in a time period. The problem is referred to by different names in the literature including *prize collecting traveling salesman problem with time windows* and *TSP with deadlines*. Tsitsikilis [19] showed that this problem is strongly NP-complete even when the metric space is a line. Bar-Yehuda et al. [4] gave an $O(\log n)$ approximation when the vertices lie on a line. Recently Bansal et al. [6] gave an $O(\log^2 n)$ for the general problem. In this paper we consider the case when the number of *distinct* time windows, k, is a fixed constant independent of the input. We give a constant factor approximation algorithm for this problem using a reduction to MCG and using the algorithm in [7, 6] as an oracle.

The thrust of this paper is the definition of MCG and its applications. In this extended abstract we have not attempted to optimize the constants that can be achieved for the problems we consider. We defer this to the final version of the paper.

2 Greedy Algorithm for MCG

In this section we show that simple greedy algorithms give constant factor approximation ratios for MCG. First we consider the cardinality version of the problem.

2.1 Cardinality Version

We can assume without loss of generality that the number of groups $\ell \geq k$. We work in the oracle model and assume that we have an α-approximate oracle. The greedy algorithm we consider is a natural generalization of the greedy algorithm for the maximum coverage problem. It iteratively picks sets that cover the maximum number of uncovered elements, however it considers sets only from those groups that have not already had a set picked from them. The precise algorithm is stated below.

Algorithm **Greedy**
$H \leftarrow \emptyset$, $X' \leftarrow X$.
For $j = 1, 2, \ldots, k$ do
  For $i = 1, \ldots, \ell$ do
    If a set from G_i has not been added to H then $A_i \leftarrow \mathcal{A}(G_i, X')$
    Else $A_i \leftarrow \emptyset$
  EndFor
  $r \leftarrow \text{argmax}_i |A_i|$
  $H \leftarrow H \cup \{A_r\}$, $X' \leftarrow X' - A_r$
EndFor
Output H.

By renumbering the groups, we can assume that Greedy picks a set from group G_j in the jth iteration. Let OPT denote some fixed optimal solution and let $i_1 < i_2 < \ldots < i_k$ be the indices of the groups that OPT picks sets from. We set up a bijection π from $\{1, 2, \ldots, k\}$ to $\{i_1, i_2, \ldots, i_k\}$ as follows. For $1 \leq h \leq k$, if $h \in \{i_1, i_2, \ldots, i_k\}$ then we require that $\pi(h) = h$. We choose π to be any bijection that respects this constraint.

Let C_j be the set that Greedy picks from G_j, and let O_j be the set that OPT picks from $G_{\pi(j)}$. We let $A'_j = A_j - \cup_{h=1}^{j-1} A_h$ denote the set of new elements that Greedy adds in the jth iteration. Let $C = \cup_j A_j$ and $O = \cup_j O_j$ denote the number of elements that Greedy and OPT pick.

Lemma 1. *For $1 \leq j \leq k$, $|A'_j| \geq \frac{1}{\alpha}|O_j - C|$.*

Proof. If $O_j - C = \emptyset$ there is nothing to prove. When Greedy picked A_j, the set O_j was available to be picked. Greedy did not pick O_j because $|A'_j|$ was at least $\frac{1}{\alpha}|O_j - \cup_{h=1}^{j-1} A_h|$. Since $\cup_{h=1}^{j-1} A_h \subseteq C$, the lemma follows. □

Theorem 1. *Greedy is an $(\alpha + 1)$-approximation algorithm for the cardinality MCG with an α-approximate oracle.*

Proof. From Lemma 1, we have that

$$|C| = \sum_j |A'_j| \geq \sum_j \frac{1}{\alpha}|O_j - C| \geq \frac{1}{\alpha}(|\cup_j O_j| - |C|) \geq \frac{1}{\alpha}(|O| - |C|).$$

Hence $|C| \geq \frac{1}{\alpha+1}|O|$. □

Corollary 1. *If $k = \ell$, Greedy is an $(\alpha + 1)$-approximation algorithm even if it is forced to pick sets from an adversarially chosen ordering of the groups.*

Proof. If $k = \ell$, the permutation π is the identity permutation. In this case Lemma 1 holds again. □

Easy examples show that our analysis of the Greedy algorithm is tight. In fact, in the oracle model, the ratio of 2 cannot be improved. When the set system

is available as part of the input, the problem is hard to approximate to within a factor of $\frac{e}{e-1}$ via a reduction from the maximum coverage problem. As we mentioned earlier, a matching ratio can be obtained via linear programming [1, 18].

A $\log n + 1$ approximation for SCG: We observe that the 2-approximation bound for MCG can be used to obtain a $\log n + 1$ approximation for SCG as follows. We simply guess the optimal value λ^*, that is there is an optimal cover H^* such that $\max_i |H^* \cap G_i| \leq \lambda^*$. We then create an instance of MCG by having a budget of λ^* on each group G_i. Greedy covers at least $1/2$ the elements in X. Iterating Greedy $\log n + 1$ times results in a solution that covers all elements. In each iteration the number of sets added from any given group is upper bounded by λ^*. Hence, when all elements are covered, the number of sets added from any group is at most $(\log n + 1)\lambda^*$.

2.2 Cost Version

We now consider the cost version of MCG. We give a greedy algorithm for this problem which is similar in spirit to the one for the cardinality case but differs in some technical details. The algorithm that we describe below may violate the cost bounds for the groups or the overall cost bound B. We will later show how to modify the output to respect these bounds. We work in the oracle model again. Recall that the oracle $\mathcal{A}$, given a set of elements X' and an index i returns a set $S \in G_i$ that approximately minimizes the ratio $\max_{D \in G_i} \frac{|D \cap X'|}{c(D)}$. The algorithm is described in more detail below. We assume without loss of generality that $B \leq \sum_{i=1}^{\ell} B_i$.

Algorithm **CostGreedy**
$H \leftarrow \emptyset$, $X' \leftarrow X$.
Repeat
  For $i = 1, 2, \ldots, n$ do
    If $c(H \cap G_i) < B_i$ then $A_i \leftarrow \mathcal{A}(X', G_i)$.
    Else $A_i \leftarrow \emptyset$.
  EndFor
  $r \leftarrow \text{argmax}_i \frac{|A_i|}{c(A_i)}$
  $H \leftarrow H \cup \{A_r\}$, $X' \leftarrow X' - A_r$.
Until $(c(H) \geq B)$.
Output H.

Note that H need not obey the budget requirements. Define $H_i = H \cap G_i$. For a set S chosen by the algorithm, define $X(S)$ as the extra set of elements in X that are covered at the time S is added to H. Define $X(H_i) = \cup_{S \in H_i} X(S)$. Similarly, $X(H)$ is the set of elements covered by the algorithm. Let OPT be some fixed optimal solution to the given problem instance. Let O be the set of sets chosen by OPT. Define $Y_i = O \cap G_i$. We call an index i good if $c(H \cap G_i) \leq B_i$, that is the algorithm did not exceed the budget for G_i. Otherwise we call i bad. We omit proofs of the following lemmas in this version.

Lemma 2. *If i is bad, then* $|X(H_i)| \geq \frac{1}{\alpha} \sum_{A \in Y_i} |A - X(H)|$.

Corollary 2. $|X(H)| \geq \frac{1}{\alpha+1} |\cup_{i:i \text{ bad}} \cup_{A \in Y_i} A|$.

Lemma 3. $|X(H)| \geq \frac{1}{\alpha+1} |\cup_{i:\ i \text{ good}} \cup_{A \in Y_i} A|$.

From Corollary 2 and Lemma 3, it follows that $X(H) \geq \frac{1}{2(\alpha+1)}\text{OPT}$. But H does not respect all the budget constraints. We partition H into three subsets H_1, H_2, H_3. H_3 is the last set picked by our algorithm. H_2 contains those sets S which when added to H caused the budget of some group G_i to be violated – however we do not include the set in H_3. H_1 contains all the remaining sets in H. It is easy to see that H_1, H_2 and H_3 do not violate the budget constraints. Further, one of these three sets must be covering at least 1/3 the number of elements covered by H. Thus we get the following theorem.

Theorem 2. *The algorithm* **CostGreedy** *is a $6(\alpha + 1)$-approximation algorithm for the cost version of MCG.*

3 Applications of MCG

3.1 The *k*-Traveling Repairmen Problem

Recall from Section 1 that in the $k-$traveling repairmen problem we are given a metric space on a set of nodes V induced by an edge weighted undirected graph G. We are given a set of k *source* vertices in V, call them $s_1, s_2, \ldots, s_k$. A feasible solution to the problem is a set of k tours, one tour starting at each source s_i, such that every vertex in V is visited by one of the tours. Given a feasible solution, we define the *latency* of a vertex as the time at which it gets visited by one of these tours. The objective is to minimize the sum of the latencies of the vertices in V.

We give the first constant factor approximation algorithm for the multiple depot k-traveling repairmen problem. We define a related problem, which we call the *budgeted cover* problem, as follows. The input to the problem is a subset V' of V and a positive integer B. A solution is a set of k tours, one tour starting at each source s_i, such that no tour has length more than B. The objective is to maximize the number of vertices of V' covered by these tours. We can view this problem in the framework of MCG as follows. The ground set is the set of vertices in V'. For each source s_i, we have a collection of sets $S^i_1, \ldots, S^i_{\ell_i}$: each set corresponds to a distinct tour of length at most B beginning at s_i. There are k groups, one group for each source vertex s_i. The group G_i corresponding to s_i is $\{S^i_1, \ldots, S^i_{\ell_i}\}$. Clearly, the cardinality version of MCG for this set system is the same as the budgeted cover problem for the original graph. From Section 2 we will get a constant factor approximation algorithm for the budgeted cover problem provided we have the following oracle $\mathcal{A}$.

The oracle $\mathcal{A}$ should be able to solve the *budget-MST* problem. In this problem, we are given a graph $G = (V, E)$, a source $s \in V$ and a budget B. A solution

is a tour starting at s and having cost at most B. The objective is to maximize the number of vertices covered by the tour. Unfortunately the *budget-MST* problem is NP-hard. However, by using the algorithm for the i-MST problem [13, 3], we can obtain a polynomial time algorithm, $\mathcal{A}$, which covers OPT vertices and costs at most $\beta \cdot B$. Here $\beta > 1$ is the approximation ratio for the i-MST problem. Hence, if we are willing to violate the budget constraint by a factor of β we can cover as many vertices as the optimal solution.

It follows from Theorem 1 that we can get a polynomial time algorithm for the budgeted cover problem which covers at least half as many vertices as the optimum, and constructs k tours, each tour of length at most βB. We call this algorithm $\mathcal{C}$.

We can now describe our algorithm for the traveling k-traveling repairmen problem. Our algorithm works in phases. We assume without loss of generality that all distances are at least 1. Let V_j be the set of *uncovered* vertices at the beginning of phase j (so $V_0 = V$). In phase j, we cover as many vertices as possible so that the budget of each tour is about 2^j. More precisely, we do the following

Algorithm **Visit**(j) :
For $p = 1, 2$ do
  Run $\mathcal{C}$ on the budget cover problem instance with inputs V_j and 2^j.
  Remove from V_j the covered vertices.

This describes phase j. We invoke the subroutine **Visit**(j) with increasing values of j until all vertices are covered. Given a source s_i, we have constructed several tours starting from s_i. We just stitch them together starting in the order these tours were found by the algorithm. Clearly, our algorithm produces a feasible solution. It remains to prove that it is a constant factor approximation algorithm.

We begin with some notation. Fix an optimal solution OPT. Let O_j denote the set of nodes in OPT's solution which have latency at most 2^j. Let C_j be the set of nodes visited by our algorithm by the end of phase j.

Lemma 4. **Visit**(j) *covers at least* $\frac{3}{4}|O_j - C_{j-1}|$ *vertices.*

Proof. Let R_j denote $O_j - C_{j-1}$. Let A_j be the set of nodes covered by **Visit**(j) when $p = 1$. Theorem 1 implies that $|A_j| \geq \frac{1}{2}|R_j|$. One more application of this theorem when $p = 2$ gives the desired result. □

The rest of the proof goes along the same lines as in [11]. Let n_j be the set of nodes in OPT whose latency is more than 2^j. Let n'_j be the set of nodes in our tour which do not get visited by the end of phase j.

Lemma 5. $n'_j \leq \frac{1}{4}n'_{j-1} + \frac{3}{4}n_j$.

Proof. From Lemma 4 it is easy to see that $n'_j \leq n'_{j-1} - 3/4|O_j - V_{j-1}|$. Clearly, $|O_j| = n - n_j$ and $|V_{j-1}| = n - n'_{j-1}$. Combining these proves the lemma. □

The total latency of the tours obtained by our algorithm is upper bounded by $\sum_j 4\beta 2^j n'_j$ and that produced by the tours in the optimal solution is lower bounded by $\sum_j 2^{j-1} n_j$. From Lemma 5 we obtain that

$$\sum_j 2^j n'_j \;\leq\; \frac{1}{2}\sum_j 2^{j-1} n'_{j-1} + \frac{3}{4}\sum_j 2^j n_j$$

which implies that

$$\sum_j 2^j n'_j \;\leq\; 3\sum_j 2^{j-1} n_j.$$

This proves that our algorithm yields a 12β approximation. We can improve the ratio by using ideas from [14, 11, 8]; we defer the details to the final version. We can also obtain constant factor approximations for each of the following generalizations: (i) each vertex v can be serviced only by a given subset S_v of repairmen, (ii) each vertex v has a service time p_v that the repairmen needs to spend at the vertex, and (iii) each vertex v has a weight w_v and the objective is to minimize the sum of the weighted latencies.

3.2 The Orienteering Problem with Time Windows

We now consider the orienteering problem with time windows. We assume that the number of *distinct* time windows is some fixed constant k. We use, as a subroutine, the algorithm of Bansal et al. [6] which provides a 3-approximation for the case when there is a single time window $[0, D]$ for all vertices and the tour is required to start at a vertex s and end at a vertex t. In the rest of the section we use β to denote the approximation ratio for the single deadline case. All our ratios will be expressed as functions of β. Let Δ be the maximum distance in the metric space. We begin by describing approximation algorithms for two special cases: (1) when all release times are zero, and (2) when all deadlines are the same.

Release Times are Zero: We consider the special case when $r_v = 0$ for all nodes v. Let $d_1 < d_2 < \ldots < d_k$ be the k distinct deadlines. Let V_i denote the set of vertices whose deadline is d_i. Let P^* be the tour constructed by some optimal solution. Define v_0^* as the source vertex s. For $1 \leq i \leq k$, let v_i^* as the last vertex in the tour P^* which is visited by the deadline d_i. It is possible that $v_i^* = v_{i'}^*$ for two distinct indices i and i'. Suppose v_i^* is visited at time t_i^*, then it follows that that $t_i^* \leq d_i$.

Our algorithm first *guesses* the vertices $v_1^*, v_2^*, \ldots, v_k^*$ and the time instances $t_1^*, t_2^*, \ldots, t_k^*$. Note that $t_i^* \leq n\Delta$. Hence the total number of guesses is $O(n^{2k}\Delta^k)$. Since Δ need not be polynomially bounded, the number of guesses is not poly-bounded. We omit details on how to use a polynomial number of guesses. Now, we define k groups $G_1, \ldots, G_k$ as follows. G_i is the set of all paths on the vertex set $V_i \cup V_{i+1} \cup \cdots \cup V_k$ which originate at v_{i-1}^* and end at v_i^* with the additional constraint that the length of the path is at most $t_i^* - t_{i-1}^*$.

Lemma 6. *Consider the instance of the MCG with groups defined as above where we need to pick exactly one set from each group. A γ-approximation algorithm for this problem instance implies the same for the corresponding orienteering problem.*

Proof. Suppose we are given a solution to the MCG which picks paths $P_1, \ldots, P_k$ from the corresponding groups. If we stitch these tours sequentially, it is easy to see that we get a path which satisfies the deadlines of the vertices visited by the individual paths. Therefore the number of vertices covered by this tour is $|P_1 \cup P_2 \cup \cdots \cup P_k|$. Further, if we consider the tour P^*, we can get a solution to the MCG which covers $|P^*|$ vertices. This proves the lemma. □

Thus, it is enough to approximate the MCG induced by the guess of $v_1^*, \ldots, v_k^*$ and $t_1^*, \ldots, t_k^*$. The oracle needed for this instance is an algorithm for the orienteering problem where the time windows for all the nodes are of the form $[0, D]$, hence we can use the algorithm of Blum et al. [7] or Bansal et al. [6]. From Corollary 1 we obtain a a $(\beta + 1)$-approximation algorithm for the case of k deadlines. The running time of the algorithm is $O(n^{2k}\Delta^k T)$ where T is the running time of the approximation algorithm for the single deadline case.

Single Deadline: We now consider the special case when all deadlines are the same, say D but the release dates can be different. Consider any feasible tour P which starts at s and ends at a vertex u. Let the length of P be $\ell(P)$. Suppose we reverse the tour, i.e., we view the tour as a path P^r starting at u and ending at s. If P visits a vertex v at time t, then P^r visits v at time $\ell(P) - t$. So $r_v \leq t \leq \ell(P)$ implies that $0 \leq \ell(P) - t \leq \ell(P) - r_v$. Thus, we can view this tour as one in which the release time of a vertex is 0 and the deadline is $\ell(P) - r_v$. Therefore, if we could guess the length of the optimal path P^* and the last vertex u^* in this path, then we could just use the algorithm mentioned in the previous section. Thus, we can get a $(\beta + 1)$-approximation algorithm for this problem as well. The running time of the algorithm increases by a factor of Δ from the algorithm for single release date.

k Time Windows: Now we address the case where there are both release times and deadlines. Let $r_1 < r_2 < \ldots < r_k$ be the k distinct release time and let $d_1 < d_2 < \ldots < d_k$ be the k distinct deadlines. Let P^* be the tour constructed by the optimal solution. As before let v_i^* be the last vertex in P^* to be visited before d_i and let t_i^* be the time at which v_i^* is visited. Recall that V_i is the set of vertices with deadline d_i. We define group G_i to be the set of tours that start at v_{i-1}^* at t_{i-1}^* and end at v_i^* by t_i^*. The vertices that a tour in G_i can visit are constrained to be in $V_i \cup V_{i+1} \cup \ldots \cup V_k$. Lemma 6 trivially generalizes to this setting to yield a γ approximation provided we have an oracle for the MCG instance. Consider a group G_i. The vertices all have a deadline at least as large as t_i^*, hence we have a single deadline. The vertices might have different release times, however there are at most k distinct release times. Hence the oracle needed for G_i can be obtained from the algorithm described above for this case that has an approximation ratio of $\beta + 1$. Thus, once again applying Theorem 1 we can obtain a $(\beta + 2)$-approximation algorithm for the case of k time windows.

Theorem 3. *Given a β-approximation algorithm for the orienteering problem with a single time window for all vertices, there is a $(\beta + 2)$-approximation algorithm for the orienteering problem with at most k distinct time windows that runs in time polynomial in $(n\Delta)^k$.*

Acknowledgments: We thank Moses Charikar for suggesting that we write a paper about MCG. We thank Chris Harrelson for pointing out a mistake.

References

1. A. Ageev and M. Sviridenko. Pipage Rounding: a New Method of Constructing Algorithms with Proven Performance Guarantee. To appear in *J. of Combinatorial Optimization*.
2. E. Arkin, J. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proceedings of SoCG*, 1998.
3. S. Arora and G. Karakostas. A $2 + \epsilon$ approximation for the k-MST problem. In *Proceedings of SODA*, 2000.
4. R. Bar-Yehuda, G. Even, and S. Sahar. On Approximating a Geometric Prize-Collecting Traveling Salesman Problem with Time Windows. *Proc. of ESA*, 2003.
5. A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proceedings of STOC*, 1994.
6. N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation Algorithms for Deadline-TSP and Vehicle Routing with Time-Windows. *Proc. of STOC*, 2004.
7. A. Blum, S. Chawla, D. Karger, T. Lane, A. Meyerson, and Maria Minkoff. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *Proc. of FOCS*, 2003.
8. K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees and minimum latency tours. *Proc. of FOCS*, 2003.
9. C. Chekuri and S. Khanna. A PTAS for the Multiple Knapsack Problem. *Proc. of SODA*, 2000.
10. M. Elkin and G. Kortsarz. Approximation Algorithm for the Directed Telephone Multicast Problem. *Proc. of ICALP*, 2003.
11. J. Fakcharoenphol, C. Harrelson, and S. Rao. The k-Traveling Repairmen Problem. In *Proceedings of SODA*, 2003.
12. U. Feige. A Threshold of ln n for Approximating Set Cover. *Journal of the ACM*, 45(4), 634–652, July 1998.
13. N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of FOCS*, 1996.
14. M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. In *Proceedings of SODA*, 1996.
15. Approximation Algorithms for NP-Hard Problems. Edited by D. Hochbaum. PWS Publishing Company, Boston, 1996.
16. G. Kortsarz. Personal communication, July 2003.
17. S. Khuller, A. Moss, and J. Naor. The Budgeted Maximum Coverage Problem. *Information Processing Letters*, Vol. 70(1), pp. 39–45, (1999)
18. A. Srinivasan. Distributions on level-sets with Applications to Approximation Algorithms. *Proc. of FOCS*, 2001.
19. J. Tsitsikilis. Special Cases of Traveling Salesman Problem and Repairmen Problems with Time Windows. *Networks*, 22:263-28, 1992.