



Title	Dynamic programming based algorithms for set multicover and multiset multicover problems
Author(s)	Hua, QS; Wang, Y; Yu, D; Lau, FCM
Citation	Theoretical Computer Science, 2010, v. 411 n. 26-28, p. 2467-2474
Issued Date	2010
URL	http://hdl.handle.net/10722/65464
Rights	Creative Commons: Attribution 3.0 Hong Kong License

Dynamic Programming Based Algorithms for Set Multicover and Multiset Multicover Problems*

Qiang-Sheng Hua[†] Yuexuan Wang[‡] Dongxiao Yu[§] Francis C.M. Lau[¶]

Abstract

Given a universe N containing n elements and a collection of multisets or sets over N , the multiset multicover (MSMC) or the set multicover (SMC) problem is to cover all elements at least a number of times as specified in their coverage requirements with the minimum number of multisets or sets. In this paper, we give various exact algorithms for these two problems with or without constraints on the number of times a multiset or set may be chosen. First, we show that the MSMC without multiplicity constraints problem can be solved in $O^*((b+1)^n|F|)$ time and polynomial space where b is the maximum coverage requirement and $|F|$ denotes the total number of given multisets over N . (The O^* notation suppresses a factor polynomial in n .) To our knowledge, this is the first known exact algorithm for the MSMC without multiplicity constraints problem. Second, by combining dynamic programming and the inclusion-exclusion principle, we can exactly solve the SMC without multiplicity constraints problem in $O((b+2)^n)$ time. Compared with two recent results, in [Hua et al., Set multi-covering via inclusion-exclusion, Theoretical Computer Science, 410(38-40):3882-3892 (2009)] and [Nederlof, J.: Inclusion Exclusion for hard problems. Master Thesis. Utrecht University, The Netherlands (2008)] respectively, ours is the fastest exact algorithm for the SMC without multiplicity constraints problem. Finally, by directly using dynamic programming, we give the first known exact algorithm for the MSMC or the SMC with multiplicity constraints problem in $O((b+1)^n|F|)$ time and $O^*((b+1)^n)$ space. This algorithm can also be easily adapted as a constructive algorithm for the MSMC without multiplicity constraints problem.

*This is an extended version with improved results of a preliminary version which appears in Proceedings of ISAAC 2009.

[†]Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong, China, qshua@cs.hku.hk. Part of the work was done when the author was a visitor in the Institute for Theoretical Computer Science, Tsinghua University.

[‡](Corresponding Author) Institute for Theoretical Computer Science, Tsinghua University, Beijing, 100084, China, wangyuexuan@tsinghua.edu.cn

[§]Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong, China, dxyu@cs.hku.hk

[¶]Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong, China, fcmlau@cs.hku.hk

1 Introduction

In this paper, we consider exact algorithms for the MultiSet MultiCover (MSMC) and the Set MultiCover (SMC) problems [27, 23, 14]. The MSMC problem is a generalization of set cover in which we have multisets instead of sets and an element may be covered multiple times. Formally, we are given a universe $N = \{1, \dots, n\}$, a positive integral coverage requirement vector $B = (b_i)$ for the elements $i \in N$ and a collection of multisets F_{ms} over N . Any one of these multisets, $S \in F_{ms}$, may contain a number of copies of each element. The objective of the MSMC problem is to determine the minimum number of multisets such that each element i is covered at least b_i times. It is similar for the SMC problem which deals with a collection of sets F_{mc} instead of multisets F_{ms} . We use F_{sc} for the collection of sets in the set cover (SC) problem. For both the MSMC and the SMC problems, in order to satisfy the minimization requirement, each multiset or set can be chosen multiple times. Mathematically, both problems can be posed as special cases of the covering integer programs problem [23, 25, 24, 18, 19] as defined by Definition 1.

Definition 1 (CIP: Covering Integer Programs [25]). *Let Z_+ denote the nonnegative integers. The CIP problem $P = (A, B, D, E)$ seeks to minimize $E^T \cdot x$ subject to $Ax \geq B$, $x \in Z_+^R$ and $x \leq D$ where A_{ij} , $b_i \in B$ and $e_j \in E$ for $i = 1, \dots, n$, $j = 1, \dots, R$ are all nonnegative integers. $Ax \geq B$ and $x \leq D$, where $d_j \in D$ is some positive integer, are called the covering constraints and the multiplicity constraints, respectively.*

Now if we take the n rows and the R columns of the matrix A as the n elements in the universe N and the R sets or multisets over $N = \{1, \dots, n\}$ respectively, then the set or multiset j would contain A_{ij} copies of element i . In the CIP problem, if we set $A_{ij} \in [0, c]$, $e_j = 1$, $b_i \in [1, b]$ and remove the multiplicity constraints, we have the Multiset Multicover problem (**MSMC**). Here the constant values **c** and **b** mean the maximum number of times any element appears in any multiset and the maximum coverage requirement, respectively. For the MSMC problem, if we keep the multiplicity constraints $x_j \leq d_j$ where $\max_{1 \leq j \leq R}(d_j) = \mathbf{d}$, we have the **MSMC with multiplicity constraints problem**. Similarly, if we further require $A_{ij} \in \{0, 1\}$ and based on whether we remove the multiplicity constraints or not, we have the Set Multicover problem (**SMC**) and the **SMC with multiplicity constraint problem**.

1.1 Approximation Algorithms for Multicovering Problems and their Application

For the MSMC and SMC with multiplicity constraints problems, the multiplicity constraints limit the number of copies of each multiset or set. These constraints in reality can be capacity limits, security goals, or due to fault tolerance reasons [18, 25]. Chuzhoy et al. [7] and Gandhi et al. [9] have studied the capacitated vertex cover with hard capacities problem, i.e., each vertex can cover only a limited number of edges and each vertex can only be used a restricted number of times. Besides the multiplicity constraint, we also need to emphasize that the MSMC and SMC problems are different from the fractional covering problem [22] which can be formulated by requiring x_j in the CIP definition to be nonnegative real values instead of integer values. Thus the fractional covering problem can be solved exactly using standard LP solvers whereas solving the MSMC and SMC problems optimally is NP-hard [27]. There has been a long line of research in designing heuristic and approximation algorithms for these problems since the 80s [10, 27]. There are also many (parallel) approximation algorithms for CIP with or without multiplicity constraints [19, 18, 24, 25, 23] which may also be applied to the SMC and MSMC problems. Besides the general CIP problem, Hochbaum and Levin [11] have studied the cyclical scheduling and multi-shift scheduling problems which are special cases of the SMC with multiplicity constraints problem. For more related work on these various covering problems, please refer to [12].

Many problems in the real world can be cast as either an SMC or an MSMC problem. For example, the minimum length wireless link scheduling problem [16] with non-unit traffic demands can be interpreted as an SMC problem. Many other problems in different areas can be formulated as an MSMC problem, such as the total late orders minimization problem [20], the traffic grooming problem in WDM networks [3], the memory reduction problem in general paging scenarios [1], a special case of the proposed deal splitting with packages problem [26] and the minimum cost cell planning problem in 4G cellular networks [2].

1.2 Exact Algorithms for Various Covering and Multicovering Problems

Besides approximate algorithms, recently there have been some efforts in understanding how fast we can exactly solve these covering or multicovering problems. By using the inclusion-exclusion principle and a fast zeta transform technique, Björklund et al. [5] have shown that the set cover problem can be exactly solved in $O^*(2^n)$ time using $O^*(2^n)$ space. Here, the O^* notation omits a polynomial factor in n . Based on this observation, they also proposed a family of exact algorithms for set partitioning problems such as graph coloring, which outperform all the previous algorithms. Subsequently they showed that similar faster algorithms can also be obtained by using the so called fast subset convolution [6]. To our knowledge, the first exact algorithms for the SMC problem were given independently by Hua et al. [14] and Nederlof [21]. As shown in [14], the set multicover problem can be exactly solved in $O^*((2b)^n)$ time with $O^*((b+1)^n)$ space or in $O^*(2^{O(bn^2)})$ time with polynomial space. In [21], based on a novel counting formulation, the set multicover problem can be solved in $O^*((b+1)^n |F_{mc}|)$ time with polynomial space, where $|F_{mc}|$ is the total number of the given sets. Although this result outperforms the polynomial space exact algorithm given by Hua et al. in [14], as discussed in [15], Hua et al.'s algorithm can also exactly count the number of set multicovers that satisfy the coverage requirements. We are not aware of any known exact algorithms for the MSMC problem, the MSMC with multiplicity constraints problem and the SMC with multiplicity constraints problem.

1.3 Our Results

In this paper, we give (1) the first known polynomial space exact algorithm for the MSMC problem (Section 3); (2) the fastest exact algorithm for the SMC problem (Section 4); and (3) the first known exact algorithms for the SMC and MSMC with multiplicity constraints problems (Section 5). A compendium of previous and this paper's results is given in Table 1.

2 Preliminaries

2.1 The Inclusion-Exclusion Principle

This basic principle of combinatorics has been frequently used in recent literatures [5, 6, 14, 21]. Let B be a finite set with subsets $A_1, A_2, \dots, A_n \subseteq B$, and with the convention that $\bigcap_{i \in \emptyset} A_i = B$, then we know the number of elements in B which lie in none of the A_i is

$$\left| \bigcap_{i=1}^n \overline{A_i} \right| = \sum_{X \subseteq N} (-1)^{|X|} \cdot \left| \bigcap_{i \in X} A_i \right| \quad (1)$$

Table 1: Summary of exact algorithms for covering and various multicovering problems

Problem	Time Complexity	Space Complexity	Reference
Set Cover (SC)	$O^*(2^n F_{sc})^1$	Polynomial	[5]
Set Cover (SC)	$O^*(2^n)$	$O^*(2^n)$	[5]
Set Multicover (SMC)	$O^*((2b)^n)$	$O^*((b+1)^n)$	[14]
Set Multicover (SMC)	$O^*(2^{O(bn^2)})$	Polynomial	[14]
Set Multicover (SMC)	$O^*((b+1)^n F_{mc})^1$	Polynomial	[21]
Set Multicover (SMC)	$O((b+2)^n)$	See note ²	This paper
Multiset Multicover (MSMC)	$O^*((b+1)^n F_{ms})^1$	Polynomial	This paper
Multiset Multicover (MSMC)	$O((b+1)^n F_{ms})$	$O^*((b+1)^n)$	This paper ³
SMC with Multiplicity Constraints	$O((b+1)^n F_{mc})$	$O^*((b+1)^n)$	This paper ³
MSMC with Multiplicity Constraints	$O((b+1)^n F_{ms})$	$O^*((b+1)^n)$	This paper ³

¹ It is easy to see that $|F_{sc}| = |F_{mc}| \leq 2^n$, $|F_{ms}| \leq (c+1)^n = \sum_{m=0}^n \binom{n}{m} c^m$.

² $\max\{\binom{n}{m_1}(b+1)^{n-m_1}, \binom{n}{m_2}(b+1)^{n-m_2}\}$, where $m_1 = \lfloor \frac{n+1}{b+2} \rfloor$ and $m_2 = \lceil \frac{n+1}{b+2} \rceil$.

³ This improves the corresponding result in the preliminary version of this paper [13].

2.2 Solving the Set Cover Problem via Counting Set Covers

We define $c_k(F_{sc})$ as the number of k -tuples $\langle s_1, \dots, s_k \rangle$ over F_{sc} such that the union of the sets $\bigcup_{i=1}^k s_i$ without removing duplicate elements satisfies the coverage requirements. Given this definition, in order to find the minimum number of sets that satisfy the coverage requirements, we just need to find the minimum k value that satisfies $c_k(F_{sc}) > 0$ using binary search. This is a standard technique which has been used in [5, 14] for exactly solving the set cover and set multicover problems. By using the inclusion-exclusion principle (Equation 1), Björklund et al. [5] prove that the number $c_k(F_{sc})$ of set covers can be computed by Equation 2. Here $a_{sc}(X)$ denotes the number of sets in F_{sc} that avoid (do not cover) any element in the set $X \subseteq N$.

$$c_k(F_{sc}) = \sum_{X \subseteq N} (-1)^{|X|} a_{sc}(X)^k \quad (2)$$

Based on Equations 1 and 2, we summarize the complexity results for counting k -set covers in the following lemma 2. For their detailed proofs please refer to [5].

Lemma 2 ([5]). (1) $c_k(F_{sc})$ can be immediately solved with $O^*(2^n |F_{sc}|)$ time and polynomial space by using Equation 2; (2) By using fast zeta transform [5], $c_k(F_{sc})$ can be solved with $O^*(2^n)$ time and $O^*(2^n)$ space.

3 A Polynomial Space Exact Algorithm for the MSMC Problem

Similar to what is done in [21], we will not directly count the number of multiset multicovers. Instead, we will first transform the multiset multicover problem into the multiset cover problem.

3.1 Transforming Counting Multicovers into Counting Covers

We transform the set or multiset multicover problem into the corresponding set or multiset cover problem, as follows. For each element $i \in N$ with b_i coverage requirement, we replace this element with b_i copies. This

augments the universe N with n elements to give a new universe N' with at most bn elements. Accordingly, the collection of (multi)sets F_{mc} or F_{ms} will be expanded into a new collection of (multi)sets F'_{mc} or F'_{ms} respectively. As an example, if $b_i = 2$ and $b_j = 1$, the element i will be replaced by elements i_1 and i_2 ; similarly, the element j will be replaced by element j_1 or we may just say that it remains unchanged. Then the set $\{i, j\}$ will be replaced by two sets $\{i_1, j_1\}$ and $\{i_2, j_1\}$. Accordingly, the multiset $\{i, i, j\}$ will be replaced by three multisets $\{i_1, i_1, j_1\}$, $\{i_1, i_2, j_1\}$ and $\{i_2, i_2, j_1\}$. Then we can count the number $c_k(F'_{mc})$ of set covers for the set multicover problem and can count number $c_k(F'_{ms})$ of multiset covers for the multiset multicover problem.

A straightforward formulation for $c_k(F'_{mc})$ or $c_k(F'_{ms})$ is to directly apply the $c_k(F_{sc})$ formula given in Equation 2. By using $a_{mc}(X)$ or $a_{ms}(X)$ to denote the number of sets or multisets in F'_{mc} or F'_{ms} that do not cover any element in X respectively, we can give similar formulations for counting the transformed (multi)set covers, by Equations 3 and 4.

$$c_k(F'_{mc}) = \sum_{X \subseteq N'} (-1)^{|X|} a_{mc}(X)^k \quad (3)$$

$$c_k(F'_{ms}) = \sum_{X \subseteq N'} (-1)^{|X|} a_{ms}(X)^k \quad (4)$$

It is easy to see that, however, the straightforward formulations for calculating $c_k(F'_{mc})$ and $c_k(F'_{ms})$ are very inefficient in terms of time. For instance, based on Lemma 2, $c_k(F'_{mc})$ (Equation 3) is computed in either $O^*(|F'_{mc}|2^{bn})$ time and polynomial space or $O^*(2^{bn})$ time and $O^*(2^{bn})$ space. It will be even worse when coming to calculating $c_k(F'_{ms})$ since $|F'_{ms}|$ could be much larger than $|F'_{mc}|$ (see Inequalities 9 and 10 in Section 3.2 for the maximum $|F'_{ms}|$ and $|F'_{mc}|$ values). In the next section, we will briefly explain a more efficient formulation for counting $c_k(F'_{mc})$ which was first given by Nederlof in [21] and then extend it to the multiset multicover problem.

3.2 A New Formulation for Computing $c_k(F'_{ms})$

We introduce some necessary notations in Table 2.

Table 2: Some notations for counting the transformed (multi)set covers

Notations	Definitions
$Y(X) = (Y(1), \dots, Y(i))$ ¹	Y is a nonnegative integer function on the set $X \subseteq N$.
$Y(X) \preceq B(X)$	For each $i \in X$, we have $Y(i) \leq b_i$.
F_{mc}^Y (F_{ms}^Y) ²	A new collection of (multi)sets constructed on F_{mc} (F_{ms}) where each element $i \in N$ is replaced by $Y(i)$ elements. If $Y(i) = 0$ then any (multi)set $S \in F_{mc}$ (F_{ms}) which covers element i will be deleted.

¹ We use Y instead of $Y(X)$ when it is clear from the context.

² For example, if we set $Y = B = (b_1, \dots, b_n)$, then $F_{mc}^Y = F_{mc}^B = F'_{mc}$ and $F_{ms}^Y = F_{ms}^B = F'_{ms}$.

The new formulation for calculating $c_k(F'_{mc})$ is given in Equation 5 [21].

$$c_k(F'_{mc}) = \sum_{Y \preceq B} (-1)^{\sum_{1 \leq i \leq n} Y(i)} \left(\prod_{1 \leq i \leq n} \binom{b_i}{Y(i)} \right) (|F_{mc}^{B-Y}|)^k \quad (5)$$

This new formulation is obtained by taking advantage of the symmetry information behind Equation 3. By analyzing all the subsets X used in this equation, since the augmented universe N' is composed of many

replicated elements for each single element with non-unit coverage requirement, we can see that there are many symmetric subsets $X \subseteq N'$ in the sense that this family of subsets $\{X\}$ have the same $a_{mc}(X)$ values. As a result, in order to lower the time complexity, it is not necessary to calculate the $a_{mc}(X)$ value anew for each subset $X \subseteq N'$ (see Equation 3). Instead, we can just calculate the $a_{mc}(X)$ value once for all symmetric subsets $X \subseteq N'$. As shown in Equation 5, for each $Y(N) \preceq B(N)$, these symmetric subsets are represented by F_{mc}^{B-Y} , which is constructed on F_{mc} where each element $i \in N$ is replaced by $b_i - Y(i)$ elements.

This new formulation for calculating $c_k(F'_{mc})$ can be easily extended for computing $c_k(F'_{ms})$, as in Equation 6.

$$c_k(F'_{ms}) = \sum_{Y \preceq B} (-1)^{\sum_{1 \leq i \leq n} Y(i)} \left(\prod_{1 \leq i \leq n} \binom{b_i}{Y(i)} \right) (|F_{ms}^{B-Y}|)^k \quad (6)$$

Now the remaining question is: given F_{ms} , how to calculate $|F_{ms}^Y|$ for each $Y(N) \preceq B(N)$? We need to first give a helping lemma. (For an application of this lemma, please refer to the first paragraph of Section 3.1.)

Lemma 3. *If an element a is replaced by r copies, the multiset which contains s number of elements a will be expanded into $\binom{r+s-1}{r-1}$ number of new multisets.*

Proof. Let x_i ($1 \leq i \leq r$) denote the number of times that the new element i appears in this specific multiset; then the problem is equivalent to finding the number of different combinations of the set $\{x_i\}$ which satisfy $x_i \geq 0$ and $\sum_{i=1}^r x_i = s$. If we set $y_i = x_i + 1$, then the problem is changed to finding the number of different combinations of the set $\{y_i\}$ that satisfy $y_i \geq 1$ and $\sum_{i=1}^r y_i = s + r$. Consider a line formed by $s + r$ elements; each possible combination of the set $\{y_i\}$ then corresponds to the $r - 1$ cuts among the $s + r - 1$ possible cutting places. From this observation, the total number of different combinations of the set $\{y_i\}$ is equal to $\binom{r+s-1}{r-1}$. \square

With Lemma 3, we can give the formula for calculating $|F_{ms}^Y|$ in Equation 7, where S denotes a multiset belonging to F_{ms} and $t(S)$ is a set composed by distinct elements in the set S ; c_j denotes the number of times that the element j appears in a multiset S .

$$|F_{ms}^{B-Y}| = \sum_{S \in F_{ms}} \prod_{j \in t(S)} \binom{c_j + b_j - Y(j) - 1}{b_j - Y(j) - 1} \quad (7)$$

If for all $j \in t(S)$ we set $c_j = 1$, we can obtain the same formula given in [21] for calculating $|F_{mc}^Y|$ in Equation 8.

$$|F_{mc}^{B-Y}| = \sum_{S' \in F_{mc}} \prod_{j \in S'} (b_j - Y(j)) \quad (8)$$

Since $|F'_{ms}| = |F_{ms}^B|$ and each element $i \in N$ can appear at most c times in each multiset, according to Equation 7, we can give an upper bound for $|F'_{ms}|$ in the following Inequality 9.

$$|F'_{ms}| \leq \sum_{m=0}^n \binom{n}{m} \left(\sum_{t_1=1}^c \cdots \sum_{t_m=1}^c \prod_{i=1}^m \binom{t_i + b_i - 1}{b_i - 1} \right) \quad (9)$$

If we set $c = 1$, we can obtain an upper bound for $|F'_{mc}|$ in Inequality 10.

$$|F'_{mc}| \leq \sum_{m=0}^n \binom{n}{m} b^m = (b+1)^n \quad (10)$$

Based on Equations 6 and 7, by directly computing $|F_{ms}^Y|$ for each $Y \preceq B$, we have Theorem 4.

Theorem 4. *The multiset multicover problem can be solved in $O^*((b+1)^n |F_{ms}|)$ time using polynomial space.*

Proof. First, according to Equation 7, since for all $j \in t(S)$, both c_j and b_j are constant values, $\binom{c_j+b_j-Y(j)-1}{b_j-Y(j)-1}$ can be calculated in $O(1)$ time. Then for all $i \in N$, since $Y(i)$ varies from 0 to at most b , we obtain the claimed time complexity. We only used polynomial space. \square

As mentioned in the last paragraph of Section 3.1, calculating $c_k(F'_{ms})$ using Equation 4 takes $O^*(2^{bn} |F'_{ms}|)$ time and polynomial space. Comparing with Theorem 4, using our new $c_k(F'_{ms})$ formulation (Equation 6) is much better for the MSMC problem.

4 A Dynamic Programming Based Algorithm For the SMC Problem

According to Equation 5 which is for computing $c_k(F'_{mc})$, since directly calculating the $|F_{mc}^Y|$ values for each $Y \preceq B$ would necessitate checking each set in F_{mc} , in this section, we give a dynamic programming based algorithm that avoids this problem. Some necessary notations are given in Table 3. We can then compute all $|F_{mc}^Y|$ values using Algorithm 1.

Table 3: Some notations for calculating $|F_{mc}^Y|$

Notations	Definitions
X^Y	Replace each element $i \in X$ with $Y(i)$ copies.
$X^{Y(i)-1}$	The same as X^Y except that the element i is replaced by $Y(i) - 1$ copies.
$c(X^1, (N \setminus X)^Y)$	The number of sets in $F_{mc}^{Y'}$ that include all the elements in X . Here the new collection of sets $F_{mc}^{Y'}$ is constructed on F_{mc} as follows: each element $i \in X$ remains unchanged and each element $i \in N \setminus X$ is replaced by $Y(i)$ copies.

The time and space complexities of Algorithm 1 are given in Lemma 5.

Lemma 5. *For all $Y \preceq B$, the $|F_{mc}^Y|$ values can be calculated in $O((b+2)^n)$ time using $\max\{\binom{n}{m_1}(b+1)^{n-m_1}, \binom{n}{m_2}(b+1)^{n-m_2}\}$ space where $m_1 = \lfloor \frac{n+1}{b+2} \rfloor$ and $m_2 = \lceil \frac{n+1}{b+2} \rceil$.*

Proof. First, both the time and space used from Step 1 to Step 4 equal $O(2^n)$. The total time used for Steps 5–12 can be calculated through the formula $\sum_{m=0}^n \binom{n}{m} (b+1)^{n-m} = (b+2)^n$. Due to Step 11, the total space used for these steps is $\max_{0 \leq i \leq n} \{\binom{n}{i} (b+1)^{n-i}\}$. By observing $\frac{\binom{n}{i} (b+1)^{n-i}}{\binom{n}{i-1} (b+1)^{n-i+1}} \geq 1$, we obtain the space complexity. Summing up the time and space used for these two parts, we have the result. \square

When all the $|F_{mc}^Y|$ values have been stored into a table, according to Equation 5 and Lemma 5, we can see that the time and space complexities for calculating $c_k(F'_{mc})$ are dominated by Algorithm 1. Thus we have Theorem 6.

Theorem 6. *The set multicover problem can be solved in $O((b+2)^n)$ time using $\max\{\binom{n}{m_1}(b+1)^{n-m_1}, \binom{n}{m_2}(b+1)^{n-m_2}\}$ space where $m_1 = \lfloor \frac{n+1}{b+2} \rfloor$ and $m_2 = \lceil \frac{n+1}{b+2} \rceil$.*

Since our former exponential space exact algorithm can solve the set multicover problem in $O^*((2b)^n)$ time [14], and since directly using Equation 5 immediately yields a $O^*((b+1)^n |F_{mc}|)$ time polynomial

Algorithm 1 Calculating $|F_{mc}^Y|$ using Dynamic Programming

Input: - F_{mc} and the coverage requirement vector B

Output: The $|F_{mc}^Y|$ values for all $Y \preceq B$

- 1: **For** each $X \subseteq N$ **do**
 - 2: If X is not empty, we set $c(X^1, (N \setminus X)^0) = F_{mc}(X)$ where $F_{mc}(X)$ is the indicator function which equals 1 if $X \in F_{mc}$ and 0 otherwise; if X is an empty set, we set $c(\emptyset^1, N^0) = 0$.
 - 3: Store the $c(X^1, (N \setminus X)^0)$ value into a look-up table.
 - 4: **End For**
 - 5: **For** t from n downto 0 **do**
 - 6: **For** each $X \subseteq N$ and $|X| = t$ **do**
 - 7: **For** each $Y(N \setminus X) \preceq B(N \setminus X)$ where $Y(N \setminus X)$ is from $(0, \dots, 0)^{|N \setminus X|}$ to $(b_1, \dots, b_{|N \setminus X|})$ (using lexicographic order) **do**
 - 8: for some $i \in N \setminus X$ where $Y(i) \neq 0$, we calculate $c(X^1, (N \setminus X)^Y)$ using the recursion $c(X^1, (N \setminus X)^Y) = c(X^1, (N \setminus X)^{Y(i)-1}) + c((X \cup \{i\})^1, (N \setminus (X \cup \{i\}))^Y)$ and then store it into a table
 - 9: **End For**
 - 10: **End For**
 - 11: Remove all $c(Z^1, (N \setminus Z)^Y)$ values from the table where $|Z| = |X| + 1$
 - 12: **End For**
 - 13: Return all the $|F_{mc}^Y|$ values and for each $Y \preceq B$ we have $|F_{mc}^Y| = c(\emptyset^1, N^Y)$.
-

space exact algorithm, our dynamic programming based algorithm presented in this section gives the fastest exact algorithm so far for the set multicover problem.

In the next section, by using an idea similar to the Viterbi algorithm [8] which is to compute the maximum a posteriori probability in the hidden Markov model, we will design a dynamic programming based algorithm for the set or multiset multicover with multiplicity constraints problem. And this algorithm can be easily adapted for the multiset multicover without multiplicity constraints problem. Compared with the dynamic programming based exact algorithm for the multiset multicover without multiplicity constraints problem (Section 3 of [13]) and the shortest path based exact algorithm for the set or multiset multicover with multiplicity constraints problem (Section 4 of [13]), our algorithm can greatly reduce the space complexities without sacrificing time.

5 A Dynamic Programming Based Algorithm for Set or Multiset Multicover with Multiplicity Constraints Problem

In this section, we focus on the SMC or MSMC with multiplicity constraints problem (see Section 1). Some necessary notations and their definitions are given in Table 4.

Table 4: Some notations used in Algorithm 2 (*DMCM*)

Notations	Definitions
$D = (d_{S_i})$	The multiplicity constraints vector indicating the maximum number of times that each multiset (set) $S_i \in F_{ms} (F_{mc})$ can be chosen and $\mathbf{d} = \max_{S_i \in F_{ms} (F_{mc})} (d_{S_i})$.
$v = (S_{[i]}, y_1, \dots, y_n)$	A covering state vertex showing that, by only choosing some multisets (sets) from $\{S_j : 1 \leq j \leq i\}$, each element $i \in N$ has been covered at least y_i times.
$w(u, v)$	The weight of the edge between vertex $u = (S_{[i-1]}, z_1, \dots, z_n)$ and vertex $v = (S_{[i]}, y_1, \dots, y_n)$.
$H(v)$	The minimum number of multisets (sets) from $\{S_j : 1 \leq j \leq i\}$ that satisfy the coverage requirements in v , i.e., (y_1, \dots, y_n) .
$P(v)$	A vector that records the shortest path for obtaining $H(v)$, i.e., each chosen multiset (set) S_j ($1 \leq j \leq i$) together with its number of copies.

With these notations, we now give a dynamic programming based algorithm called *DMCM* (Algorithm 2) to exactly solve the SMC or MSMC with multiplicity constraints problem.

The correctness of the *DMCM* algorithm is obvious. Now we give the time and space complexities of the *DMCM* algorithm in Theorem 7.

Theorem 7. *The DMCM algorithm can solve the multiset multicover (set multicover) with multiplicity constraints problem with $O^*((b+1)^n |F_{ms}|)$ ($O^*((b+1)^n |F_{mc}|)$) time and $O^*((b+1)^n)$ ($O^*((b+1)^n)$) space.*

Proof. In each iteration, Steps 4–13 construct a directed bipartite graph on level $i-1$ and level i vertices. There are at most d outgoing edges for each vertex in this bipartite graph. According to Step 3, there are $|F_{ms}|$ ($|F_{mc}|$) iterations. So the claimed time complexity can be easily obtained. For the space complexity, since we only store a directed bipartite graph on level $i-1$ and level i vertices in each iteration and the occupied space will be reused in the subsequent iterations (Step 13), and since the length of the $P(v)$ vector for each vertex v equals $O(n)$, we have the space complexity equal to $O^*((b+1)^n)$. \square

Remark: Since the polynomial space exact solution for the MSMC without multiplicity constraints problem given in Section 3 is a non-constructive algorithm, the proposed *DMCM* algorithm can be easily adapted as a constructive algorithm for the SMC or the MSMC without multiplicity constraints problem. The reason: for the SMC or the MSMC without multiplicity constraints problem, each set or multiset in F_{mc} or F_{ms} will be used at most b times. The difference is that, without the multiplicity constraints, we can always find a multicover that satisfies the coverage requirements.

6 Future Work

Comparing with the dynamic programming based exact algorithm for the MSMC without multiplicity constraints problem in [13], although the *DMCM* algorithm (Algorithm 2) can greatly reduce the space complexity, the time complexity is still considerably high. It should be possible to devise a more efficient algorithm which uses $O((b+c+1)^n)$ time. It should also be interesting to design an exact multiset multicover algorithm with $O^*((b+1)^n)$ time—that is, the time is independent of the number of times that any element appears in a multiset.

Algorithm 2 *DMCM*: Dynamic Programming Based Algorithm for Set or Multiset Multicover with Multiplicity Constraints Problem

Input: F_{ms} or F_{mc} , the coverage requirement vector B and the multiplicity constraints vector (d_{S_i})

Output: The minimum number of (multi)sets that satisfy B and respect (d_{S_i})

- 1: Set an initial vertex v with label $(S_0, 0, \dots, 0)$ and we call this vertex as level 0 vertex.
 - 2: Initialize $H(v) = 0$ and $P(v) = \emptyset$.
 - 3: **For** $i = 1$ to $|F_{ms}|$ ($|F_{mc}|$) **do**
 - 4: For (multi)set $S_i \in F_{ms}$ (F_{mc}) we define $(b+1)^n$ vertices with labels from $(S_{[i]}, 0, \dots, 0)$ to $(S_{[i]}, b, \dots, b)$; we call all the vertices constructed on S_i as level i vertices.
 - 5: **For** $j = 0$ to d_{S_i} **do**
 - 6: For each vertex $(S_{[i-1]}, y_1, \dots, y_n)$ with non-empty incoming edges (except the level 0 vertex) we add the $(j+1)$ th directed edge with edge weight j to $(S_{[i]}, y_1 + j * q_1, \dots, y_n + j * q_n)$. Here q_i means (multi)set S_i contains q_i element i . Note that if $y_i + j * q_i \geq b$ then we just set $y_i + j * q_i = b$.
 - 7: **End For**
 - 8: **For each** level i vertex $v = (S_{[i]}, y_1, \dots, y_n)$ with non-empty incoming edges **do**
 - 9: First, denote by U all the $(S_{[i-1]}, z_1, \dots, z_n)$ vertices that have outgoing edges to vertex v . Then we use the following recurrence to calculate $H(v)$.
 - 10: $H(v) = \min_{u \in U} (H(u) + w(u, v))$.
 - 11: $P(v) = (P(\arg \min_u (H(u) + w(u, v))), (S_i, w(\arg \min_u (H(u) + w(u, v)), v)))$. This means that we append the (multi)set S_i together with its number of chosen copies in the $P(v)$ vector.
 - 12: **End For**
 - 13: Remove all the level $i - 1$ vertices and their incident edges together with the corresponding $H(v)$ and $P(v)$ values.
 - 14: **End For**
 - 15: Return $H(v)$ and $P(v)$ values for the vertex $v = (S_{[|F_{ms}|]}, b_1, \dots, b_n)$ or $v = (S_{[|F_{mc}|]}, b_1, \dots, b_n)$. Otherwise, we know that we can not find a multicover that satisfies both the coverage requirement $B = (b_1, \dots, b_n)$ and the multiplicity constraints (d_{S_i}) .
-

We must emphasize that counting the number of transformed set covers for multiset multicover, i.e., the $c_k(F'_{ms})$ value, is different from directly counting the number of multiset multicovers, i.e., the $c_k(F_{ms})$ value. Although there is now an exact algorithm for calculating $c_k(F_{ms})$ [15], it requires exponential space. So it is worthwhile to try to devise polynomial space efficient algorithms for computing $c_k(F_{ms})$.

It would be worthwhile also to apply our results to some practical scenarios. For example, the minimum length wireless link scheduling with non-unit traffic demands problem [16] can be cast as a set multicover problem. Similarly, the minimum cost cell planning problem in 4G cellular networks [2] can be interpreted as a multiset multicover problem.

Finally, besides the multicovering problems studied in this paper, similar to the set partition problem studied in [5], exact algorithms for the partition versions of the set or multiset multicover problems, i.e., we need to cover all elements **exactly** the number of times as specified in their coverage requirements with the minimum number of sets or multisets would be interesting to design. A natural application of this kind of partitioning algorithms is the graph multicoloring problem [4]. A straightforward method for solving the graph multicoloring problem is to first transform it into a graph coloring problem. This is realized by simply replacing each vertex v with $b(v)$ coloring demands as a clique of $b(v)$ copies of the vertex v , where each copy of v is adjacent to all the copies of the neighbors of v in the original graph. Then by applying the fastest

exact graph coloring algorithm proposed in [5], we can exactly solve the graph multicoloring problem in $O^*(2^{bn})$ time and $O^*(2^{bn})$ space since there are at most bn vertices in the transformed graph. This appears to be a tremendously time and space-consuming method. We are currently working on more efficient exact algorithms for the graph multicoloring problem, including the non-preemption based version, i.e., the colors assigned to each node need to be contiguous, as well as the preemption based version. In addition, we are also working on the exact algorithms for the sum multicoloring problem [4] and various machine scheduling problems [17] which can be cast as either a graph multicoloring problem or a sum multicoloring problem [4].

Acknowledgments

The first author would like to thank Simai He for helpful discussions. We also thank the anonymous reviewers (of the preliminary version) for their insightful comments. This research is supported in part by Hong Kong RGC–GRF grants (7136/07E and 714009E), and the National Basic Research Program of China Grant 2007CB807900, 2007CB807901.

References

- [1] Albers, S., Arora, S., Khanna, S.: Page Replacement for General Caching Problems. In: Proc. *SODA*, Baltimore, Maryland, US, pp. 31-40 (1999).
- [2] Amzallag, D., Engelberg, R., Naor, J., Raz, D.: Capacitated Cell Planning of 4G Cellular Networks (*Technical Report CS-2008-04*). Computer Science Department, Technion, Haifa 32000, Israel (2008).
- [3] Angel, E., Bampis, E., Pascual, F.: Traffic Grooming in a Passive Star WDM Network. In: Proc. *SIROCCO*, pp. 1-12 (2004).
- [4] Bar-Noy, A., Halldrsson, M.M., Kortsarz, G., Salman, R., Shachnai, H.: Sum multicoloring of graphs. *J. Algorithms* 37(2): 422-450 (2000).
- [5] Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via Inclusion-Exclusion. *SIAM Journal on Computing*, 39(2):546-563 (2009).
- [6] Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets Möbius: fast subset convolution. In: Proc. *39th Annual ACM Symposium on Theory of Computing (STOC)*, San Diego, CA, USA (2007).
- [7] Chuzhoy, J., Naor, J.: Covering problems with hard capacities. *SIAM J. Comput.* 36(2): 498–515 (2006).
- [8] Forney, G.D. JR.: The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268-278 (1973).
- [9] Gandhi, R., Halperin, E., Khuller, S., Kortsarz, G., Srinivasan, A.: An improved approximation algorithm for vertex cover with hard capacities. *J. Comput. Syst. Sci.* 72(1): 16-33 (2006).
- [10] Hall, N.G., Hochbaum, D.S.: A fast approximation algorithm for the multicovering problem. *Discrete Applied Mathematics*, 15(1):35-40 (1986).
- [11] Hochbaum, D.S., Levin, A.: Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optimization* 3(4): 327-340 (2006).

- [12] Hochbaum, D.S.: Approximating covering and packing problems: set cover, vertex cover, independent set, and related problems. In: Hochbaum, D.S. (Eds.) : *Approximation algorithms for NP-hard problems*, pp 94–143, PWS Publishing Co. Boston, MA, USA (1996).
- [13] Hua, Q.-S., Yu D., Lau, F.C.M., Wang Y.: Exact algorithms for set multicover and multiset multicover problems. In: *Proc. 20th International Symposium on Algorithms and Computation (ISAAC)*, Hawaii, USA (2009).
- [14] Hua, Q.-S., Wang, Y., Yu, D., Lau, F.C.M.: Set multi-covering via inclusion-exclusion. *Theoretical Computer Science*, 410(38-40):3882-3892 (2009).
- [15] Hua, Q.-S., Yu, D., Lau, F.C.M., Wang, Y.: Exact Algorithms for Counting k -Set Multicover and Counting k -Multiset Multicover Problems. *manuscript* (2009)
- [16] Hua, Q.-S., Lau, F.C.M.: Exact and approximate link scheduling algorithms under the physical interference model. In: *Proc. 5th SIGACT-SIGOPS International Workshop on Foundation of Mobile computing (DIALM-POMC)*, Toronto, Canada (2008).
- [17] Karger, D., Stein, C., Wein, J.: Scheduling algorithms. In *Algorithms and theory of computation handbook*. CRC, Boca Raton, Fla, (1999).
- [18] Kolliopoulos, S.G., Young, N.E.: Approximation algorithms for covering/packing integer programs. *J. Comput. Syst. Sci.* 71(4): 495-505 (2005).
- [19] Kolliopoulos, S.G.: Approximation Algorithms for Covering Integer Programs with Multiplicity Constraints. *Discrete Applied Mathematics*, (129), 461-473 (2003).
- [20] Leung, Joseph Y.-T., Li, H., Pinedo, M.: Scheduling orders for multiple product types with due date related objectives. *European Journal of Operational Research*, 168(2): 370-389 (2006).
- [21] Nederlof, J.: Inclusion Exclusion for hard problems. Master Thesis. Utrecht University, The Netherlands (2008)
- [22] Plotkin, S. A., Shmoys, D.B., Tardos, E.: Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257-301 (1995).
- [23] Rajagopalan, S., Vazirani, V.V.: Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, 28(2):525-540 (1998).
- [24] Srinivasan, A.: New approaches to covering and packing problems. In *Proc. 12th SODA*, Washington, DC, USA (2001).
- [25] Srinivasan, A.: Improved approximation guarantees for packing and covering integer programs. *SIAM J. Comput.* 29(2): 648-670 (1999).
- [26] Shachnai, H., Shmueli, O., Sayegh, R.: Approximation Schemes for Deal Splitting and Covering Integer Programs with Multiplicity Constraints. In: *Proc. WAOA*, pp. 111-125 (2004).
- [27] Vazirani, V.V.: *Approximation Algorithms*. Berlin, Springer (2003).