Better Approximations for the Minimum Common Integer Partition Problem

David P. Woodruff*

[1] MIT
dpwood@mit.edu
[2] Tsinghua University

Abstract. In the k-Minimum Common Integer Partition Problem, abbreviated k-MCIP, we are given k multisets $X_1, \ldots, X_k$ of positive integers, and the goal is to find an integer multiset T of minimal size for which for each i, we can partition each of the integers in X_i so that the disjoint union (multiset union) of their partitions equals T. This problem has many applications to computational molecular biology, including ortholog assignment and fingerprint assembly.

We prove better approximation ratios for k-MCIP by looking at what we call the *redundancy* of $X_1, \ldots, X_k$, which is a quantity capturing the frequency of integers across the different X_i. Namely, we show $.614k$-approximability, improving upon the previous best known $(k - 1/3)$-approximability for this problem. A key feature of our algorithm is that it can be implemented in almost *linear time*.

Keywords: minimum common integer partition problem, approximation algorithms, computational biology.

1 Introduction

In a recent work [2] a new combinatorial optimization problem called the *Minimum Common Integer Partition* problem was introduced. This problem is one of the many recent combinatorial problems with applications to computational molecular biology, including ortholog assignment [1, 3, 4, 5] and DNA fingerprint assembly [10]. The problem also poses interesting new algorithmic challenges.

Formally, the problem is as follows. Consider two multisets $X = \{x_1, \ldots, x_m\}$ and T of positive integers. If there is a partition of T into multisets T_i such that for each i the sum of integers in T_i equals x_i, then T is called an *integer partition* of X. We say that T is a *common integer partition* of multisets $X_1, \ldots, X_k$ if it is an integer partition of each X_i. The *k-Minimum Common Integer Partition Problem*, abbreviated k-MCIP$(X_1, \ldots, X_k)$, is to find a common integer partition T of minimum cardinality.

As an example, for a pair of multisets $X_1 = \{2, 2, 3\}$ and $X_2 = \{1, 1, 5\}$, the integer partition $T = \{1, 1, 2, 3\}$ is a minimum common integer partition of the

* The author would like to thank Andrew Yao and Tsinghua University for hospitality and support while performing this researh.

J. Diaz et al. (Eds.): APPROX and RANDOM 2006, LNCS 4110, pp. 248–259, 2006.

X_i. Indeed, to see that T is an integer partition of X_1, partition T into multisets $T_1 = \{1, 1\}, T_2 = \{2\}$, and $T_3 = \{3\}$. Then the sum of integers in T_1 is 2, the sum in T_2 is 2, and the sum in T_3 is 3. To see that T is an integer partition of X_2, partition it into $T_1 = \{1\}$, $T_2 = \{1\}$, and $T_3 = \{2, 3\}$. To see that T has minimal size, observe that any integer partition of either X_1 or X_2 must have size at least 3. Further, the only integer partition of X_1 of size 3 is X_1 itself. However, X_1 is not an integer partition of X_2. Thus every common partition has size at least 4, which is the size of T.

A common partition T exists if and only if the integers in each X_i have the same sum. As this property is easy to verify, we will assume it holds for the rest of the paper. Let $m = \sum_{i=1}^{k} |X_i|$. We will think that k is much smaller than m, as is the case in practice. Nevertheless, in our asymptotic notation we will write the dependence on both m and k.

In [2], it is shown that k-MCIP is NP-hard[1], and in fact APX-hard [9] for every $k \geq 2$. To show the former, the authors present a Cook-reduction from Set-Partition, while for the latter they present an L-reduction from Maximum-3-Dimensional-Matching with a bounded number of occurrences, which is known to be APX-hard [7]. The authors also give a 5/4-approximation algorithm when $k = 2$ and a $\frac{3k(k-1)}{3k-2}$-approximation algorithm for general k. Note that $\frac{3k(k-1)}{3k-2} \approx k - 1/3$. The former is based on an approximation algorithm for Set-Packing with small sets, and the latter is described below. Although their algorithm for $k = 2$ is polynomial-time, its running time[2] is $\Omega(m^9)$, which is likely to make it impractical. Indeed, as mentioned in the applications below, it is likely that $m \approx 2^{12}$, for which this running time is much too large to be of use. Their algorithm for general k is much more efficient, running in time $O(mk)$.

We note that an $O(m \log k)$-time k-approximation for k-MCIP is straightforward, though in [2] the authors only provide an $O(mk)$-time k-approximation. To see this, first suppose $k = 2$ and the multisets are X, Y. Repeatedly choose an element $x \in X$ and $y \in X$, and add $\min(x, y)$ to the common partition. Remove x from X and y from Y if $x = y$. Otherwise remove $\min(x, y)$ from the multiset it occurs in and replace $\max(x, y)$ with $\max(x, y) - \min(x, y)$ in the other multiset. This procedure produces at most $m - 1$ numbers in the common partition. Since the optimal solution has size at least $\max(|X|, |Y|) \geq m/2$, the algorithm provides a 2-approximation. It runs in $O(m)$ time. To solve k-MCIP, divide the input multisets into $\lfloor k/2 \rfloor$ pairs (plus one multiset if k is odd), run the above algorithm on each pair, and repeat the process on pairs of output multisets. The running time is now $O(m \log k)$ and the output size is again at most m, so we get an $m/(m/k) = k$ approximation. We refer to this algorithm as k-Greedy.

In fact, it is not hard to achieve ratio $\frac{5k}{8}$ for even k and $(\frac{5k}{8} + \frac{3}{8})$ for odd k. This was also missed in [2], and already improves the previous best known ratio for every $k \geq 3$. To see this, for simplicity suppose that k is even. Partition the k multisets into $k/2$ pairs (X_{2i-1}, X_{2i}). Run the algorithm of [2] for 2-MCIP on

[1] Lan Liu and the author have shown that k-MCIP is NP-hard in the strong sense.

[2] We assume the unit-cost RAM model on words of size $O(\log m)$ and that arithmetic operations on words can be done in constant time.

each pair. Let the output partition of the algorithm on inputs $X_{2i-1}X_{2i}$ be Y_i. Finally, output k-Greedy$(Y_1, \ldots, Y_{k/2})$.

Let opt_i denote the size of the minimum common partition of X_{2i-1} and X_{2i}. Then $|Y_i| \leq 5opt_i/4$. Moreover, if opt denotes the size of the minimum common partition of all of the X_i, then $opt_i \leq opt$ for all i. As the common partition output by k-Greedy never has size larger than its total input size, we get that

$$|\mathsf{k} - \mathsf{Greedy}(Y_1, \ldots, Y_{k/2})| \leq \sum_{i=1}^{k/2} |Y_i| \leq \sum_{i=1}^{k/2} \frac{5opt_i}{4} \leq \frac{5}{4} \sum_{i=1}^{k/2} opt = \frac{5k}{8} \cdot opt,$$

and the ratio of $5k/8$ follows.

The main problem with the above algorithm is that it invokes the algorithm for $k = 2$ given in [2], and thus its running time is also $\Omega(m^9)$. Thus the algorithm is likely to be very impractical.

In this paper we give a new approximation algorithm for k-MCIP which runs in almost *linear time*. More precisely, we have a randomized $O(m \log k)$ and a deterministic $O(m\text{poly}(k))$-time algorithm. Both running times are $O(m)$ for constant k. Moreover their ratios are bounded above by $.614k(1 + o(1))$. Since $.614 < 5/8$, we not only reduce the running time to linear, we even improve the approximation ratio of the natural (though inefficient) algorithm sketched above. Although the algorithm in [2] for general k is also efficient, it was only shown to achieve ratio $k - 1/3$. We improve the analysis of [2], and show their algorithm actually provides a $(k - 1/2)$-approximation. We also provide an instance to their algorithm for which this is best-possible, which turns out to be a bit non-trivial. Finally, for the special case when the multisets X_i are disjoint, we improve the analysis of our algorithm to show a ratio of $(k + 1)/2$.

Applications: Suppose we are given a collection of k genomes, one for each of k different species. We look at the following special case: each genome consists of the same number of copies of a single gene, but the copies are clustered into different substrings in the different genomes. Thus, we may view each genome i as a sequence of integer substring sizes $x_1^i, \ldots, x_r^i$, with the property that for all pairs of genomes i, j, $\sum_\ell x_\ell^i = \sum_{\ell'} x_{\ell'}^j$. The goal in this application is to partition the substrings into the same collection of strings, minimizing the number of strings in the common partition. This provides a measure of similarity between the different genomes, and has been proposed in practice. This is exactly the Minimum Common Integer Partition problem. For more detail, see [2, 3, 4, 5].

Actually, the main motivating example for $k > 2$ is *DNA fingerprint assembly*, as described in great detail on page 3 of [2]. This is a problem that has arisen in the ongoing *Oligonucleotide Fingerprinting Ribosomal Genes (OFRG)* project [10]. The goal of this project is to identify different microbial organisms using fingerprints obtained in the lab. Here k is a parameter determined by a specific measuring device, while m refers to a quantity known as the number of probe subsets of a fingerprint. We refer the reader to [2, 10] for the details, but we merely state that from [6] we have learned that a typical setting of MCIP parameters likely to occur in practice is $k = 28$ and $m = 2^{12}$.

2 Overview of the Algorithms

To illustrate our techniques, we first recall the algorithm CommonElements given in [2] which invokes the subroutine 2-Greedy described in the introduction. For a formal treatment of 2-Greedy, see [2] where it is shown to terminate with output partition size less than m (so the ratio is $m/(m/2) = 2$) in $O(m)$ time.

The algorithm CommonElements first adds the integers common to all of the X_i to a common partition, and then repeatedly invokes 2-Greedy. Let $X_1, \ldots, X_k$ be an instance of k-MCIP.

CommonElements($X_1, \ldots, X_k$):

1. $T \leftarrow \emptyset$.
2. While there is an x occurring in all of the X_i, choose such an x, add x to T, and remove one copy of x from each X_i.
3. Let $X'_1, \ldots, X'_k$ denote the resulting multisets.
4. $T' \leftarrow$ 2-Greedy(X'_1, X'_2).
5. For $i = 3, \ldots, k$,
   (a) $T' \leftarrow$ 2-Greedy(T', X'_i).
6. Output $T \cup T'$.

In [2], it is shown that this algorithm is a $(k - 1/3)$-approximation. We will later show that it is in fact a $(k - 1/2)$-approximation. However, let us first define our new algorithm to see how it contrasts with this one.

The structure of our algorithm for k-MCIP is as follows. Let $[k] = \{1, 2, \ldots, k\}$.

HighFrequency($X_1, \ldots, X_k$):

1. $T \leftarrow \emptyset$.
2. Choose a set-partition π of $[k]$ into pairs of integers, with one unpaired integer r if k is odd.
3. For each pair $(i, j) \in \pi$,
   (a) Compute $C_{i,j} \leftarrow$ CommonElements(X_i, X_j).
4. If there is only a single pair $(1, 2)$, output $C_{1,2}$, else
   - k even: output HighFrequency($\{C_{i,j} \mid (i, j) \in \pi\}$).
   - k odd: output HighFrequency($\{X_r\} \cup \{C_{i,j} \mid (i, j) \in \pi\}$).

We have not yet specified how to choose the partition π in step 2 of HighFrequency. We will try to choose π so that the output in step 4 has minimal size. For constant k, this is easy to do by an exhaustive enumeration of partitions. For larger k, we show a random π is a good choice, and in fact this choice can be efficiently derandomized. For now the choice is not essential, as we merely wish to compare the structure of HighFrequency with that of CommonElements.

At a high level, the main differences between HighFrequency and CommonElements are the following. In CommonElements, the multisets $X_1, \ldots, X_k$ (or more precisely, $X'_1, \ldots, X'_k$) are traversed sequentially, invoking 2-Greedy on each new

X_i, together with the current common partition of $X_1, \ldots, X_{i-1}$. In our algorithm, we traverse $X_1, \ldots, X_k$ in parallel, and we recurse. Moreover, the traversal order is not fixed, but rather determined by π. Also, instead of invoking 2-Greedy on each instance of 2-MCIP we encounter, we invoke CommonElements, which has a better approximation ratio and still can be implemented in linear time.

To get a feeling for the algorithms, consider the following example. Suppose $k = 4$ and the input multisets are $X_1 = \{2, 3\}$, $X_2 = \{1, 4\}$, $X_3 = \{2, 3\}$, and $X_4 = \{2, 3\}$. When we run CommonElements, step 2 has no effect since although items 2 and 3 occur many times, they do not occur in X_2. In step 4 we may assume that $T' = \{1, 1, 3\}$ (we are constructing a worst-case execution of 2-Greedy). Then after the first iteration of step 5a, we have $T' = \{1, 1, 1, 2\}$, and after the last iteration we obtain $T' = \{1, 1, 1, 1, 1\}$ (again, in the worst-case).

However, let $\pi = \{1, 2\}, \{3, 4\}$. Then $C_{1,2} = \{1, 1, 3\}$ or $C_{1,2} = \{1, 2, 2\}$, but $C_{3,4} = \{2, 3\}$, so that the output of step 4 is $\{3, 1, 1\}$ or $\{2, 2, 1\}$, which are of minimal size. Thus, our algorithm HighFrequency is able to exploit the high frequency of integers $2, 3$ in the input, even though CommonElements is not. This is the reason we've named our algorithm HighFrequency.

One of the main technical aspects of this paper is how to handle the case when there are not many integers occurring in multiple input mutisets X_i. In this case we show that even the optimal solution must be large, as intuitively if many integers have low frequency, then most of the integers in the X_i will have to be split into at least two new integers in any common partition. We show this by developing a framework for capturing the frequency of integers across the different input mutisets.

In the next section we prove a key lemma for *lower-bounding* the size of the optimal common partition, and in section 4 we use this lemma to analyze the performance of HighFrequency. We believe our lower bound can lead to future results. For example, in the next section we use this characterization to improve the analysis of the main algorithm of [2].

3 A Key Lemma and Two Quickies

Consider an instance S of k-MCIP consisting of k multisets of integers $S = \{X_1, \ldots, X_k\}$. We will define a certain quantity of S, called its *redundancy*, which captures the distribution of the number of occurrences, across the different X_i, of integers occurring in S.

At first glance it may seem that our definition is needlessly complicated. After presenting it, we explain the need for this complication.

Recall that the X_i are multisets, but may also be viewed as ordered lists. Thus, we may refer to the element in the jth position of X_i for $1 \leq j \leq |X_i|$.

Consider elements T of $[|X_1| + 1] \times [|X_2| + 1] \times \cdots \times [|X_k| + 1]$. T translates naturally into a multiset $\tilde{T}$ as follows: if its ith coordinate j does not equal $|X_i| + 1$, add the integer in the jth position of X_i to $\tilde{T}$. We say that T is *lonely* if the multiset $\tilde{T}$ has the form $\{t, t, \ldots, t\}$. In this case we use the notation $int(T)$ to denote the integer t. We say a set $\mathcal{C}$ of lonely elements of $[|X_1| + 1] \times [|X_2| +$

$1] \times \cdots \times [|X_k| + 1]$ is *consistent* if there are no two distinct elements $T, T' \in \mathcal{C}$ and an i for which $T_i = T'_i \neq |X_i| + 1$. That is, no two elements of $\mathcal{C}$ can agree on any coordinate i, unless they both have the value $|X_i| + 1$ on that coordinate.

We define the *weighted-size* of a set $\mathcal{C}$ of lonely elements T_j to be $\sum_{j=1}^{|\mathcal{C}|} |\tilde{T}_j|$.

Definition 1. *The* r-**redundancy** *of* S*, denoted* $\mathbf{Red(r, S)}$*, is the maximum, over all consistent sets* $\mathcal{C}$ *of at most* r *lonely elements, of the weighted-size of* $\mathcal{C}$*.*

We note that a simpler alternative, though incorrect, definition is the following: define the degree of a variable x as $deg(x, S) = |\{i \mid x \in X_i\}|$. Then define the redundancy $Red(r, S)$ to be $\max_{x_1,\ldots,x_r \text{ distinct}} \sum_{i=1}^{r} deg(x_i, S)$.

Although simpler, this definition fails to capture the following example: $X_1 = \{1, 1\}$, $X_2 = \{1, 1\}$. Here, $Red(2, S) = 4$. Indeed, consider $\mathcal{C} = \{(1, 1), (2, 2)\}$. Then the elements $(1, 1), (2, 2)$ are both lonely since their corresponding multisets have the form $\{1, 1\}$. Moreover, they are consistent. Finally, the weighted size of $\mathcal{C}$ is 4. However, the alternative definition would put $Red(2, s) = deg(1, S) = 2$. One could instead remove the word "distinct" from the definition, but this also does not solve the problem, since then for $X_1 = \{1, 3, 4\}$ and $X_2 = \{1, 2, 5\}$ it would return $Red(3, s) = 6$ since $x_1 = x_2 = x_3 = 1$, but our definition gives $Red(3, s) = 4$ with say $T_1 = (1, 1), T_2 = (2, 4)$, and $T_3 = (3, 4)$.

Define *opt*(S) to be the size of a minimum common partition of S. When S is clear from the context, we will often just write *opt*. Recall that $m = \sum_{i=1}^{k} |X_i|$. The following lemma lower bounds *opt* in terms of the redundancy of S.

Lemma 1. $opt \geq (2m - Red(opt, S))/k$.

Proof. Let T be a minimum common integer partition of $X_1, \ldots, X_k$. Define the bipartite graph with right partition T and left partition S (here S is the multiset union[3] of the X_i). Each $x \in S$ is incident exactly to those elements $t_i \in T$ which partition x. So, for instance, the sum over all neighbors of x is equal to x.

Then $Red(opt, S)$ is an upper bound on the number of degree-1 vertices in the left part. To see this, we construct a consistent set $\mathcal{C}$ of *opt* lonely elements whose weighted-size is exactly the number of degree-1 vertices in the left part. For each vertex v on the right, let $\tilde{S}(v)$ denote v's neighbors on the left with degree 1. As each such v is incident to exactly 1 element in each X_i, we can naturally associate $\tilde{S}(v)$ with an element $S(v)$ of $[|X_1| + 1] \times [|X_2| + 1] \times \cdots \times [|X_k| + 1]$, where $S(v)_j = |X_j| + 1$ iff v partitions a vertex in S_j with degree more than 1. Then $S(v)$ is lonely since each integer in $\tilde{S}(v)$ equals the integer corresponding to v. The set $\{S(v) \mid v \text{ on the right }\}$ is consistent since if $w = S(v)_j = S(v')_j$ for $v \neq v'$ and $j \leq |X_j|$, then w would have degree more than 1. Finally, $\{S(v) \mid v \text{ on the right }\}$ has exactly *opt* elements. Thus, its weighted size is at most $Red(opt, S)$. Since every degree-1 vertex on the left is counted exactly once in the weighted-size of $\{S(v) \mid v \text{ on the right }\}$, there are at most $Red(opt, S)$ such vertices.

[3] The multiset union of two multisets is defined by the following rule: if x occurs f_1 times in the first multiset and f_2 times in the second, then x occurs $f_1 + f_2$ times in the multiset union.

Resuming the proof of the lemma, there are at least $m - Red(opt, S)$ remaining vertices in the left part, and each has degree at least 2. Thus, there are at least $Red(opt, S) + 2(m - Red(opt, S)) = 2m - Red(opt, S)$ edges in the graph. On the other hand, every vertex on the right has degree exactly k. Thus, $2m - Red(opt, S) \leq k|T| = k \cdot opt$, and the lemma follows by dividing by k.

Corollary 1. *If for all $j \neq j'$, X_j and $X_{j'}$ are disjoint, then k-MCIP is $(k+1)/2$-approximable in $O(m \log k)$ time.*

Proof. In this case $Red(r, S) \leq r$ for any r, and the bound above gives $opt \geq 2m/(k+1)$. The claim follows by running k-Greedy whose output size is $\leq m$.

We now look at the approximation ratio of CommonElements. In [2], it is shown the ratio is $3k(k-1)/(3k-2) \leq k - 1/3$. On the other hand, $3k(k-1)/(3k-2) \geq k - 1/3 - \epsilon$ for any constant $\epsilon > 0$ and large enough k. We show,

Corollary 2. CommonElements *outputs a $(k - 1/2)$-approximation.*

Proof. Recall the notation of section 2. Suppose CommonElements adds ℓ integers to T in step 2. It follows that T' is of size at most $m - \ell k$. Thus, $|T \cup T'| \leq \ell + (m - \ell k) = m - \ell(k-1)$. On the other hand, there are at most ℓ elements with corresponding multisets of size k in any consistent set $\mathcal{C}$ of lonely elements. It follows that the weighted-size of $\mathcal{C}$, and thus $Red(opt, S)$, can be at most $\ell k + (opt - \ell)(k-1)$. Applying Lemma 1, $k \cdot opt \geq 2m - (\ell k + (opt - \ell)(k-1))$, which, after rearranging, shows $opt \geq (2m - \ell)/(2k-1)$. Using that $k \geq 2$ and $\ell \geq 0$, the corollary follows from the following bound on the approximation ratio,

$$(2k-1)\frac{m - \ell(k-1)}{2m - \ell} \leq (2k-1)\frac{m - \ell/2}{2m - \ell} = (2k-1)\frac{1}{2} = k - \frac{1}{2}.$$

Claim. The approximation ratio of CommonElements is at least $k - 1/2 - o(1)$.

Proof. Let r be a large positive integer, and consider $X_1 = X_2 = \cdots = X_{k-1} = \{1, 1, 3, 3, 5, 5, 7, 7, \ldots, 2r+1, 2r+1\}$, and $X_k = \{2, 6, 10, 14, \ldots, 4r+2\}$. Then $\sum_{x \in X_i} x = \sum_{x \in X_j} x = 2(r+1)^2$ for all $i \neq j$. Thus, $S = \{X_1, \ldots, X_k\}$ is an instance of k-MCIP. The optimal solution is X_1, which has size $opt = 2r + 2$.

The output of CommonElements on S is just the output of steps 3-6 on S (e.g., T') since no integer occurs in all of the X_i, and thus step 2 does not modify S. In 2-Greedy it is not specified how to choose the two integers x, y, and our strategy is to present a sequence of choices for which T' is of size at least $(2k-1)r - O(k^2)$. It will follow that the approximation ratio is at least

$$\frac{(2k-1)r - O(k^2)}{2r+2} = \frac{(2k-1)(r+1) - (2k-1) - O(k^2)}{2r+2} = (k - 1/2) - \frac{O(k^2)}{2r+2},$$

which can be made arbitrarily close to $k - 1/2$ by increasing r.

We show by induction, after i invocations of 2-Greedy, $0 \leq i \leq k-2$ (recall that there are $k-1$ invocations in total - we handle the last one separately), the common partition of $X_1, \ldots, X_{i+1}$ generated by CommonElements has the form:

$$\{1, 1, 3, 3, \ldots, 2(r-i)+1, 2(r-i)+1\} \cup 1^{(2i)} \cup 2^{(s)}, \quad (1)$$

where $a^{(b)}$ indicates b copies of a, and where $s = 2\sum_{j=0}^{i-1}(r-j)$.

Base Case: When $i = 0$, we have not yet invoked 2-Greedy, and so the multiset in expression 1 should be equal to X_1. Since $2i = 0$ and $s = 0$ in this case, this holds by definition of X_1.

Inductive Step: Suppose expression 1 is the common partition after $i \geq 1$ invocations, and consider the $(i+1)$st invocation, in which the common partition after i invocations is invoked together with $X_{i+2} = \{1, 1, 3, 3, \ldots, 2r+1, 2r+1\}$. We claim 2-Greedy may first repeatedly subtract 1s and 2s from X_{i+2} until the two multisets both have the form $\{1, 1, 3, 3, \ldots, 2r+1-2i, 2r+1-2i\}$. To see this, since each integer in X_{i+2} is odd, and there are $2i$ integers in X_{i+2} larger than $2r+1-2i$, 2-Greedy may subtract $2i$ different 1s so that X_{i+2} has the form

$$\{1, 1, \ldots, 2r+1-2i, 2r+1-2i, 2r+2-2i, 2r+2-2i, 2r+4-2i, 2r+4-2i, \ldots, 2r, 2r\}.$$

Next, observe that the sum of the last $2i$ terms of X_{i+2}, $2\sum_{j=1}^{i}(2r+2j-2i)$, is equal to s. Thus, 2-Greedy may subtract s different 2s so that the two multisets become $\{1, 1, 3, 3, \ldots, 2r+1-2i, 2r+1-2i\}$, as claimed, and the current partition is $1^{(2i)} \cup 2^{(s)}$.

Next 2-Greedy may choose pairs, $(1, 3), (1, 3), (3, 5), (3, 5), \ldots, (2r-1-2i, 2r+1-2i), (2r-1-2i, 2r+1-2i)$, where the first element in each pair is from the common partition after i invocations, and the second is from X_{i+2}. The first element in each pair is added to the new partition. The multisets now have the form $\{2r+1-2i, 2r+1-2i\}$ and $1^{(2)} \cup 2^{(2(r-i))}$. Finally, 2-Greedy may subtract 1 from the two different $2r+1-2i$, and then repeatedly subtract 2. Thus the common partition after $i+1$ invocations has the form

$$1^{(2i)} \cup 2^{(s)} \cup \{1, 1, 3, 3, \ldots, 2r-1-2i, 2r-1-2i\} \cup 1^{(2)} \cup 2^{(2(r-i))},$$

which is easily seen to satisfy the inductive hypothesis.

Last Invocation: By the inductive argument, the common partition of $X_1, \ldots, X_{k-1}$ has the form of expression 1 with $i = k-2$, namely, the form $\{1, 1, 3, 3, \ldots, 2r-2k+5, 2r-2k+5\} \cup 1^{(2k-4)} \cup 2^{(s)}$, where $s = 2\sum_{j=0}^{k-3}(r-j) = (2k-4)r - O(k^2)$. Recall $X_k = \{2, 6, 10, 14, \ldots, 4r+2\}$. First 2-Greedy may repeatedly subtracts 1s and 2s from the two mutisets, so that they become

$$\{1, 1, 3, 3, \ldots, 2r-2k+5, 2r-2k+5\} \text{ and } \{2, 6, \ldots, 4r-4k+10\},$$

and the common partition has the form $1^{(2k-4)} \cup 2^{(s)}$. Note that since every integer in X_k is even, this can be accomplished by first subtracting s different 2s from the largest integers of X_k, followed by $2k-4$ different 1s. Now 2-Greedy may choose pairs $(1, 2), (3, 6), (5, 10), \ldots, (2r-2k+5, 4r-4k+10)$, so that the multisets both have the form $\{1, 3, \ldots, 2r-2k+5\}$.

Then it chooses pairs $(1, 3), (3, 5), \ldots, (2r-2k+3, 2r-2k+5)$, so that the multisets have the form $\{2r-2k+5\}$ and $1 \cup 2^{(r-k+2)}$. Finally, 2-Greedy may add $1 \cup 2^{(r-k+2)}$ to the common partition, so the output of Common-Elements is

$$1^{(2k-3)} \cup 2^{(r-k+2+s)} \cup \{1, 3, 5, \ldots, 2r-2k+5\} \cup \{1, 3, 5, \ldots, 2r-2k+3\}.$$

Since $2k-3+r-k+2+s \geq (2k-3)r - O(k^2)$, and there are $2r - O(k)$ elements in the last two sets, the output partition has size $(2k-1)r - O(k^2)$, as needed.

4 Analysis of HighFrequency

In this section we prove our main theorem, Theorem 1. We use the probabilistic method to show that there are good set-partitions π that HighFrequency can choose in step 2. We quantify how well HighFrequency performs in terms of the average size f of a multiset from an optimal consistent set of lonely elements. On the other hand, we also use Lemma 1 to lower bound the size of the minimum common partition in terms of f. We then choose f so that the ratio between this upper and lower bound is maximized, which is a worst-case ratio.

In the following, we will use $O(), \Omega(), o(), \omega()$ to denote functions of k which are independent of m, e.g., $o(1)$ is a function which tends to 0 as $k \to \infty$.

Theorem 1. HighFrequency *outputs a* $.614k(1+o(1))$*-approximation.*

Proof. First observe that for two multisets X_i, X_j containing $c(X_i, X_j)$ elements in common [4], the output size of CommonElements(X_i, X_j) is at most

$$((|X_i| + |X_j| - 2c(X_i, X_j)) - 1) + c(X_i, X_j) < |X_i| + |X_j| - c(X_i, X_j).$$

In particular, its output size is always less than its input size.

Suppose in the ith invocation of HighFrequency, the algorithm is called with multisets $Y_1, \ldots, Y_r$. Then HighFrequency will partition these multisets into pairs (with one extra Y_j if r is odd) and invoke CommonElements on each pair. For any call to CommonElements in the ith invocation of HighFrequency, say CommonElements(Y_a, Y_b), $|\text{CommonElements}(Y_a, Y_b)| < |Y_a| + |Y_b| - c(Y_a, Y_b)$. Let c_i be the sum of $c(Y_a, Y_b)$ over all pairs (Y_a, Y_b) in the ith invocation.

Let m_i denote the output size of the ith invocation of HighFrequency, so for example, m_1 is the *input size* to the first recursive call (or the output size of HighFrequency if there are no recursive calls). Define $m_0 = m$. Let x be such that $2^x = o(\sqrt{k})$. Since $2^x \le k$ for large enough k, there are at least x invocations of HighFrequency (and in fact, there may be many more, though we will only need to consider the first x). Then for $1 \le i \le x$, $m_i < m_{i-1} - c_i$. Summing these inequalities up for all i and canceling common terms, $m_x < m - \sum_{i=1}^{x} c_i$. Since $|\text{HighFrequency}(X_1, \ldots, X_k)| \le m_x$, we have $|\text{HighFrequency}(X_1, \ldots, X_k)| < m - \sum_{i=1}^{x} c_i$. It follows that,

$$\mathbf{E}_{\pi_1,\ldots,\pi_x}[|\text{HighFrequency}(X_1, \ldots, X_k)|] < m - \mathbf{E}_{\pi_1,\ldots,\pi_x}\left[\sum_{i=1}^{x} c_i\right],$$

where π_i is a uniformly random set-partition chosen in the ith invocation of HighFrequency, and thus each of the integer pairs (a, b) in each invocation is (by itself) a uniformly random pair of integers. Indeed, by symmetry the first chosen pair and also all other pairs have the same probability distribution, namely they

[4] By elements in common, we mean we can find $c(X_i, X_j)$ *disjoint* pairs of elements, each pair containing one element from X_i and one element from X_j, such that the elements within each pair are equal as integers. So if $X_i = \{1, 1, 3, 4\}$ and $X_j = \{1, 1, 2, 5\}$, then $c(X_i, X_j) = 2$, even though 1 is the only integer value in common.

are uniformly drawn at random from all possible pairs. Thus, $\mathbf{E}[c(Y_a, Y_b)] = \mathbf{E}[c(Y_{a'}, Y_{b'})]$ for every two integer pairs $(a, b), (a', b')$ determined by π_i.

We may bound $\mathbf{E}[c(Y_a, Y_b)]$ as follows. Consider the largest (in terms of weighted size) consistent set of *opt* lonely elements of $[|X_1| + 1] \times \cdots \times [|X_k| + 1]$. Suppose the sizes of their corresponding multisets are $f_1, \ldots, f_{opt}$, and let $f = \lfloor \sum_j f_j/opt \rfloor$. Observe that f is a positive integer since each $f_j \geq 1$. Now from Lemma 1 we know that $opt \geq (2m - Red(opt, S))/k$. But $Red(opt, S) = \sum_{i=1}^{opt} f_i \leq (f + 1)opt$, and after rearranging, we have $opt \geq 2m/(k + f + 1)$.

Suppose $f < k/5$. Then, since $|\mathsf{HighFrequency}(\mathsf{X}_1, \ldots, \mathsf{X}_k)| < m$, we have

$$\frac{|\mathsf{HighFrequency}(X_1, \ldots, X_k)|}{opt} < \frac{m}{\frac{2m}{k+f+1}} = \frac{k + f + 1}{2} < \frac{6k}{10}(1+o(1)) = .6k(1+o(1)),$$

and the theorem is proven in this case.

Let us now handle the case when $f \geq k/5$. Consider two input multisets Y_a, Y_b in the ith invocation of HighFrequency. Each is formed by successively applying CommonElements on at most 2^{i-1} different input multisets X_i. Suppose a lonely element S with $int(S) = y$ intersects each of the (at most) 2^{i-1} input multisets corresponding to Y_a. Then y will occur in Y_a. This also holds for Y_b. Thus, if y occurs in the (at most) 2^i different input multisets corresponding to Y_a and Y_b, y will be common to Y_a and Y_b. By our choice of $\pi_1, \ldots, \pi_i$, the set of these (at most) 2^i input multisets is uniformly random amongst all such sets. Thus,

$$\mathbf{E}[c(Y_a, Y_b)] \geq \sum_j \frac{\binom{f_j}{2^i}}{\binom{k}{2^i}} = \sum_j \frac{f_j(f_j - 1)\cdots(f_j - (2^i - 1))}{k(k-1)\cdots(k - (2^i - 1))}.$$

We claim the above expession is minimized when all of the f_j are at least as large as $f = \lfloor \sum_j f_j/opt \rfloor$. To see this, suppose, w.l.o.g., that $f_1 \geq f_2 \geq \cdots \geq f_{opt}$. If this were not the case, then $f_1 \geq f + 1$ and $f_{opt} \leq f - 1$. Suppose we decrease f_1 by 1 and increase f_{opt} by 1. Then the average is the same and we still have $f_1 \geq f$. On the other hand, the expression changes by

$$\frac{2^i}{k \cdots (k - (2^i - 1))} \left((f_1 - 1) \cdots (f_1 - (2^i - 1)) - f_{opt} \cdots (f_{opt} - (2^i - 1) + 1) \right).$$

Now, $f_1 > f \geq k/5 > 2^i$ for large enough k (since $i \leq x$) and $f_1 - j > f_{opt} - j + 1$ for all j, so the above expression is non-negative. This substitution of variables did not cause the value of the sum to increase, so the sum is minimized when all the f_j are at least f. Moreover, since $f > 2^i$,

$$\mathbf{E}[c(Y_a, Y_b)] \geq \sum_j \frac{f(f-1)\cdots(f - (2^i - 1))}{k(k-1)\cdots(k - (2^i - 1))} \geq \sum_j \left(\frac{f - 2^i}{k - 2^i}\right)^{2^i} \quad \text{since } k \geq f > 2^i$$

$$\geq \sum_j c^{2^i} \left(\frac{1 - \frac{5 \cdot 2^i}{k}}{1 - \frac{2^i}{k}}\right)^{2^i} \quad \text{where } c = f/k, \text{ and } f \geq k/5.$$

To analyze this, observe that $\Theta(\frac{2^i}{k}) = 1/\omega(2^i)$ since $i \leq x$ and $2^x = o(\sqrt{k})$. We use the following inequality, which follows from Proposition B.3 of [8].

$$\left(1 - \frac{1}{\omega(2^i)}\right)^{2^i} / \left(1 - \frac{1}{\omega(2^i)}\right) \geq e^{-2^i/\omega(2^i)} \geq \left(1 - \frac{1}{\omega(2^i)}\right)^{2^i}.$$

Plugging these inequalities into our bound above, we have that, $\mathbf{E}[c(Y_a, Y_b)] \geq \sum_j c^{2^i}(1 - o(1)) = opt \cdot c^{2^i}(1 - o(1))$. In the ith invocation there are at least $\lfloor k/2^i \rfloor$ pairs. By linearity of expectation, $\mathbf{E}_{\pi_1,\ldots,\pi_x}[c_i] \geq \lfloor \frac{k}{2^i} \rfloor \mathbf{E}[c(Y_a, Y_b)]$, and so $\mathbf{E}_{\pi_1,\ldots,\pi_x}[c_i] \geq \lfloor \frac{k}{2^i} \rfloor \cdot opt \cdot c^{2^i}(1 - o(1))$. Thus,

$$\mathbf{E}_{\pi_1,\ldots,\pi_x}[|\mathsf{HighFrequency}(X_1, \ldots, X_k)|] < m - opt \sum_{i=1}^{x} \left\lfloor \frac{k}{2^i} \right\rfloor c^{2^i}(1 - o(1)).$$

Since $\mathsf{HighFrequency}$ chooses the optimal $\pi_1, \ldots, \pi_x$, it follows that

$$|\mathsf{HighFrequency}(X_1, \ldots, X_k)| < m - opt \sum_{i=1}^{x} \left\lfloor \frac{k}{2^i} \right\rfloor c^{2^i}(1 - o(1)).$$

The approximation ratio R of $\mathsf{HighFrequency}$ is $|\mathsf{HighFrequency}(X_1, \ldots, X_k)|/opt$. Dividing the expression above by opt gives $R < \frac{m}{opt} - \sum_{i=1}^{x} \lfloor \frac{k}{2^i} \rfloor c^{2^i}(1 - o(1))$. Now since $c \leq 1$ and $x = o(k)$, we can drop the floors,

$$R < \frac{m}{opt} - \sum_{i=1}^{x} \left(\frac{k}{2^i} - 1\right) c^{2^i}(1 - o(1)) < \frac{m}{opt} + o(k) - \sum_{i=1}^{x} \frac{k}{2^i} \cdot c^{2^i}.$$

Recall that we have shown $\frac{k+f+1}{2} \geq \frac{m}{opt}$. Using this and $f = ck$, we have

$$R < \frac{k}{2} + o(k) + k \max_c \left(\frac{c}{2} - \sum_{i=1}^{x} \frac{c^{2^i}}{2^i}\right).$$

We upper bound R by $\frac{k}{2} + o(k) + k \max_c \left(\frac{c}{2} - \frac{c^2}{2} - \frac{c^4}{4} - \frac{c^8}{8}\right)$, as looking at higher terms turns out to only negligibly reduce the approximation ratio further. Set $p(c) = \frac{c}{2} - \frac{c^2}{2} - \frac{c^4}{4} - \frac{c^8}{8}$. Then $p'(c) = \frac{1}{2} - c - c^3 - c^7$. We solve $p'(c^*) = 0$. By continuity, it is easy to see that there is exactly one positive real solution c^*. A MATLAB routine shows that this value c^* satisfies $.4222 < c^* < .4223$. Moreover, $p''(c)$ is non-positive for any c, and thus c^* is a local maximum. Again by computation, $p(c^*) < .11391$. At the extremes $p(1/5) \leq 1/10$ and $p(1) < 0$, and thus c^* is a global maximum. It follows that $R < \frac{k}{2} + o(k) + .114k = .614k(1 + o(1))$, and the proof is complete.

Remark 1. We claim that our analysis cannot show $R < k/2$. Indeed, one can construct S for which $|\mathsf{HighRedundancy}(S)| = m - (k - 1)$. Then, using Lemma 1, the best lower bound we can obtain for opt is $2m/k$. Thus, $R > k/2 - o(1)$.

Theorem 2. *k-MCIP is $.614k(1+o(1))$-approximable in $O(m \log k)$ probabilistic time and $O(m poly(k))$ deterministic time. Here, $o(1) \to 0$ as $k \to \infty$.*

Proof. It remains to establish the running time. The proof of Theorem 1 actually shows that only 3 invocations of HighFrequency are necessary to achieve the bound $.61391k(1 + o(1))$. So if we choose π_1, π_2, and π_3 judiciously, we may choose the π_i, $i \geq 4$, arbitrarily. By a Markov bound, the probability over the choices of π_1, π_2, and π_3, that the approximation ratio is less than $.614k(1+o(1))$ is $\Omega(1)$. To evaluate HighFrequency and all recursive calls on a given set of set-partitions π_i takes $O(m \log k)$ time since (1) there are $O(\log k)$ recursive calls, (2) CommonElements can be implemented in time proportional to its input size, and (3) the sum of input sizes across all calls to CommonElements in a given invocation of HighFrequency is at most m. By a Chernoff bound, we can output a $.614k(1+o(1))$-approximation in $O(m \log k)$ time with probability at least 99/100 by running HighFrequency on $O(1)$ different triples (pi_1, π_2, π_3) and outputting the smallest partition found. The choice of (π_1, π_2, π_3) can be derandomized in $m$poly(k) time with the method of conditional expectations. We omit the details.

Conclusions. We have given an $O(m \log k)$-time algorithm for k-MCIP with approximation ratio $.614k$, improving the previous bound of $k - 1/3$. The best lower bound is $\Omega(1)$. We believe it may be possible to slightly improve our aproximation ratio, but that significant progress will require a new approach.

Acknowledgment. We thank the referees and Lan Liu for helpful comments.

References

[1] X. Chen. *The minimum common partition revisited,* manuscript, 2005.
[2] X. Chen, L. Liu, Z. Liu, and T. Jiang. *On the minimum common integer partition problem*, CIAC 2006.
[3] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. *Computing the assignment of orthologous genes via genome rearrangement,* APBC, 2005.
[4] X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, and T. Jiang. Assignment of orthologous genes via genome rearrangement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 2-4, pp. 302-315, 2005
[5] Z. Fu, X. Chen, V. Vacic, P. Nan, Y. Zhong, and T. Jiang. *A parsimony approach to genome-wide ortholog assignment,* RECOMB, 2006.
[6] Tao Jiang. *Personal Communication.*
[7] V. Kann. *Maximum bounded 3-dimensional matching is MAX SNP-complete.* Information Processing Letters (IPL) 37: 27-35, 1991.
[8] R. Motwani and P. Raghavan. *Randomized Algorithms*, Cambridge University Press, 1995.
[9] C. H. Papadimitriou and M. Yannakakis. *Optimization, approximation, and complexity classes.* J. Computer and System Sciences (JCSS) 43: 425-440, 1991.
[10] L. Valinsky A. Schupham, G. D. Vedova, Z. Liu, A. Figueroa, K. Jampachaisri, B. Yin, E. Bent, R. Mancini-Jones, J. Press, T. Jiang, and J. Borneman. *Oligonucleotide fingerprinting of ribosomal RNA genes (OFRG)*, pp. 569-585. In *Molecular Microbial Ecology Manual* (2nd ed). Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.