BMC Bioinformatics

Methodology article

Approximation properties of haplotype tagging

Staal A Vinterbo*[1,2,3], Stephan Dreiseitl[4] and Lucila Ohno-Machado[1,2,3]

Address: [1]Decision Systems Group, Brigham and Women's Hospital, 75 Francis Street, Boston, MA 02115, USA, [2]Harvard Medical School, Boston, MA, USA, [3]Harvard-MIT, Division of Health Sciences and Technology Boston, MA, USA and [4]Dept. of Software Engineering, Upper Austria University of Applied Sciences, Hagenberg, Austria

Email: Staal A Vinterbo* - staal@dsg.harvard.edu; Stephan Dreiseitl - Stephan.Dreiseitl@fh-hagenberg.at; Lucila Ohno-Machado - machado@dsg.harvard.edu

* Corresponding author

Published: 09 January 2006

BMC Bioinformatics 2006, **7**:8 doi:10.1186/1471-2105-7-8

Received: 04 August 2005
Accepted: 09 January 2006

This article is available from: http://www.biomedcentral.com/1471-2105/7/8

Abstract

Background: Single nucleotide polymorphisms (SNPs) are locations at which the genomic sequences of population members differ. Since these differences are known to follow patterns, disease association studies are facilitated by identifying SNPs that allow the unique identification of such patterns. This process, known as haplotype tagging, is formulated as a combinatorial optimization problem and analyzed in terms of complexity and approximation properties.

Results: It is shown that the tagging problem is NP-hard but approximable within $1 + \ln((n^2 - n)/2)$ for n haplotypes but not approximable within $(1 - \varepsilon) \ln(n/2)$ for any $\varepsilon > 0$ unless NP $\subset$ DTIME$(n^{\log \log n})$.

A simple, very easily implementable algorithm that exhibits the above upper bound on solution quality is presented. This algorithm has running time $O(\frac{np}{2}(2m - p + 1)) \leq O(m(n^2 - n)/2)$ where $p \leq \min(n, m)$ for n haplotypes of size m. As we show that the approximation bound is asymptotically tight, the algorithm presented is optimal with respect to this asymptotic bound.

Conclusion: The haplotype tagging problem is hard, but approachable with a fast, practical, and surprisingly simple algorithm that cannot be significantly improved upon on a single processor machine. Hence, significant improvement in computatational efforts expended can only be expected if the computational effort is distributed and done in parallel.

Background

Much of the population-wide variation of the human genome can be attributed to single nucleotide polymorphisms (SNPs), which are changes in single base pairs within the genome. SNPs are of specific interest because they allow disease association studies; this means that the involvement of genes in particular diseases can be studied by the analysis of SNP alleles within these genes [1]. For the study of population genomics, SNPs can be used to measure *linkage disequilibrium*, an indication of how much more (or less) likely, compared to chance, certain combinations of neighboring SNP alleles are [2,3].

After the completion of the Human Genome Project emphasized the importance of SNPs to study the location of disease genes, the SNP Consortium project produced a genome-wide map of more than 1.4 million SNPs [4]. Due to linkage disequilibrium, the distribution of possible alleles at SNPs is not uniformly random, and some combinations of neighboring alleles occur more often than others. Such a combination of SNP alleles is called a *haplotype*, and a given set of SNPs can give rise to a wide variety of haplotypes.

It is an important problem to identify a subset of SNPs within a haplotype that allows the unique identification of all possible allele variations within a haplotype. If such a subset can be found, a haplotype can be uniquely identified by knowing only the allele values at a few SNP positions. SNPs that satisfy this requirement are called *haplotype tagging SNPs* (*htSNPs*).

The problem of identifying a minimal set of htSNPs is known to be NP-hard. NP-hardness means that there currently does not exist a way of solving the problem optimally with reasonable computing resources. Even though it is strongly believed among theoretical computer scientists that this state of affairs will prevail, NP-hardness does not exclude the possibility of finding an adequate solution with a reasonable effort. The validity of this last statement hinges on the definitions of the *adequacy* of a solution and what constitutes *reasonable effort*. We will define reasonable effort to be polynomial time computability, while adequacy will be achieved by an approximation algorithm, the solution of which is provably no worse than a certain factor r times the optimal.

In recent years, a number of algorithms for calculating htSNPs were developed. Due to the high computational complexity of the problem, these algorithms were either stochastic or, when deterministic, only applicable to haplotypes with a few hundreds to thousands of SNPs. At present, this is not a limitation, since there are few samples that contain these many SNPs. Given the speed of progress and innovation in the biosciences it will, however, be only a matter of time until brute-force approaches will no longer be feasible. Stochastic algorithms do not have this limitation; due to their nature they can give only probabilistic bounds on the results achieved. We believe that approximation algorithms are a viable alternative to both stochastic and brute-force approaches. The advantage over the former is that the bounds on the results are guaranteed to hold; the advantage over the latter is that approximation algorithms can be executed in polynomial time.

The theoretical question of how many SNPs are required to tag a given number of haplotypes was investigated, using Boolean algebra, by Wiuf et al. [5]. The number of htSNPs required for association studies was investigated by Thompson et al. [6]. Previous work on exact algorithms for identifying htSNPs were based on search strategies [7,8]; a stochastic algorithm is given by Johnson et al. [9]. Wiuf et al. [5] and Bafna et al. [10] both present polynomial time algorithms that find a minimum cardinality set of SNPs for the special case that Wiuf et al. call pairwise compatible and Bafna et al. call complete LD, i.e., the four gamete test by Hudson and Kaplan [11] fails for any pair of SNPs. We will in this exposition focus on the general unrestricted case.

Results

The main results are summarized as follows.

Theorem 1 *The haplotype tagging problem is NP-hard and approximable within* $1+\ln(\frac{n^2-n}{2})$, *but NP-hard to approximate within* $c\ln(n/2)$ *for* $c > 0$ *and not approximable within* $(1-\varepsilon)\ln(n/2)$ *for any* $\varepsilon > 0$ *unless* $\mathrm{NP} \subset \mathrm{DTIME}(n^{\log\log n})$. *If we bound n from above by a constant q, the problem is APX-complete.*

We have that $c\ln(n/2) = \Omega(\ln n)$ and $1 + \ln((n^2 - n)/2) = O(\ln n)$, by which we can see that the approximation bound is asymptotically tight. We present an algorithm with running time $O(\frac{np}{2}(2m - p + 1))$ where $p = \min(n, m)$, for the MHT problem that exhibits the above approximation bound. This algorithm is easily implemented, the detailed pseudo code listing presented in this paper contains only 22 lines.

A second algorithm is presented that can be seen as an instance of a schema that lets us approach a family of haplotype tagging problems including the problem of tagging the haplotypes using a set of SNPs with minimal diameter. This latter result addresses the problem of selecting practical "window" size bounds in approaches where such are needed (Bafna et al. [10]).

Discussion

For 250 million samples (approximately the current population of the USA) the size of the set of tags returned would be at most 39 times larger than the optimum. If we assume that our computer is capable of doing 35 trillion operations per second (the world's currently fastest parallel supercomputer, NEC's Earth Simulator, is capable of this), and we wish to search among 1 million SNPs, the computation using the algorithm presented in this paper would only take about 41 days. If we happen to know that an optimal solution uses at most 1000 of these SNPs (28 is an absolute lower bound for 250 million samples), we would be done in about 2 hours. Hence, to approach truly large scale experiments, the authors believe that implementations would likely come from the field of parallel algorithms.

Our analysis of the minimum haplotype tagging problem (MHT) relies on a relationship we establish to the minimum set cover problem (MSC). Exploiting this relationship, we can relate variants of the MSC problem to MHT. This we can do as we describe an algorithm that transforms a MHT instance into a MSC instance, solves this instance, and transforms this solution into a solution of the MHT problem. We can form variants of this algorithm by substituting particular variants of the MSC algorithm in this process. Below we list MSC variants, their known approximation properties, and if substituted into our algorithm schema, the corresponding resulting MHT problem that it solves.

• Minimum Exact Cover. This problem is approximable within $1 + \ln m$ [12] and the associated tagging problem is to find a minimum set of SNPs such that the sets of haplotype pairs each of the SNPs distinguishes are as disjoint as possible.

• Maximum Coverage by at most k sets. This problem is approximable within $e/(e - 1)$ [13] and not approximable within $e/(e - 1) - O(1)$ [14] for $e > 1$. The associated tagging problem is the problem of discerning a maximum number of haplotypes with at most k SNPs.

• Minimum k-redundant Coverage is the problem of creating a minimum cover such that each element is covered at least k times. This problem is approximable within $b - k + 1$ [15] for a constant b. The associated tagging problem is the problem of requiring each SNP in a minimum tagging set to discern between at least k pairs of haplotypes.

• Minimum Diameter Coverage is the problem of finding a cover where the greatest distance between two elements in the cover is minimal. This problem is in general not approximable within a constant, but approximable within 2 if the distance measure observes the triangle inequality and no better approximation is possible [16]. The associated tagging problem is to find a tagging set of SNPs such that the maximal pairwise distance is minimal.

• Minimum Cluster Diameter Coverage is the problem where we assume that the cover can be partitioned into k clusters and the objective is to minimize the greatest distance between two elements in the same cluster. This problem is approximable within 2 for any fixed k and upper bound of cluster size l if the distance measure satisfies the triangle inequality [16]. The associated tagging problem is to find k disjoint sets of maximally l SNPs such that the greatest distance between two SNPs in the same set is minimized.

• Maximum Dispersion Cover is the problem of finding a cover that maximizes the min imum pairwise distance between elements in the cover. This problem is NP-hard and assuming $P \neq NP$, no polynomial time approximation with bounded error guarantee exists [16]. The corresponding tagging problem is to find a set of SNPs that lets us discern between all haplotypes such that the minimum distance between two of these SNPs is maximized. We see that this problem is the most difficult to solve of all the problems considered in this exposition.

All these variations can be implemented by substituting the corresponding cover algorithm for the greedy set cover algorithm in the MHT algorithm schema. Approximation bounds and non-approximability results for the above variations can be used to establish the same for the resulting variation of the haplotype tagging problem analogously to what we do for the minimum set cover problem.

Conclusion

Although the haplotype tagging problem is hard, it can be approached with a simple, fast and practical algorithm. The contribution of this work is not only yet another fast and simple algorithm for the tagging problem, but also the proof that this algorithm delivers a solution with the asymptotically best error bound possible.

Furthermore, the algorithm schema we present via the connection to the minimum set cover problem is applicable to not only the original problem as it is defined in our analysis, but is applicable to a family of related problems, which address problems presented in published research.

As the algorithms presented are asymptotically optimal with respect to approximation bounds, meaning that solution quality cannot be significantly improved upon in polynomial time, efforts in tackling truly large scale problem instances should concentrate on distributing the computational efforts in parallel.

Methods

Let $H = \{h_1, h_2,..., h_n\}$ be a set of haplotypes with associated SNP markers at positions $S = \{1, 2,..., m\}$, and let S' be a subset of S. We define $h_i[S']$ to be the string consisting of marker values of haplotype h_i found at the positions in S'. We can view H as an $n \times m$ matrix over the set of possible SNP alleles values. Assuming that they are bi-allelic, H becomes a Boolean matrix. The problem of discerning a set of haplotypes by a minimum cardinality set of SNPs, alternatively the problem of predicting SNPs for a set of haplotypes using a minimal set of SNPs, which we will call the minimum haplotype tagging problem (MHT), can be formulated as follows.

Problem 1 *(MHT) Let H, S and h_i be as above. Find a minimum cardinality set $S' \subseteq S$ such that $h_i[S'] = h_j[S']$ implies $i = j$.*

We now define some formal concepts and present results that we will use in the analyses that follow.

Formal concepts

We formally define a minimization problem $\mathcal{P}$ as a 3-tuple (p, s, m) where

• p is the polynomial time computable characteristic function of the set of problem instances $I_{\mathcal{P}}$,

• $s(x)$ is the set of feasible solutions for instance x and its characteristic function is computable in polynomial time for any y such that $|y| \leq O(|x|^q)$ for some $q \in \mathbf{N}$,

• m is the polynomial time computable natural number measure we wish to minimize.

In other words, the problem $\mathcal{P}$ is: given instance x such that $p(x) = 1$, find $y \in s(x)$ such that $m(y) \leq m(z)$ for any $z \in s(x)$. We then let $m^*(x) = m(y)$ denote the optimum value for instance x.

The problem of deciding whether a given tuple (x, k) is in $L(\mathcal{P}) = \{(x, k)|p(x) = 1 \wedge m^*(x) \leq k\}$ we will call the decision problem associated with $\mathcal{P}$, and, overloading notation slightly, denote by $L(\mathcal{P})$.

Let f and g be two functions, and let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two problems. If for all $x \in I_{\mathcal{P}_1}$ we have that $f(x) \in I_{\mathcal{P}_2}$ and for all $y \in s_{\mathcal{P}_2}(f(x))$ we have that $g(x, y) \in s_{\mathcal{P}_1}(x)$ we say that the tuple (f, g) is a *reduction* from $\mathcal{P}_1$ to $\mathcal{P}_2$. If both f and g are computable in polynomial time, we write $\mathcal{P}_1 \leq_p \mathcal{P}_2$ and call (f, g) a polynomial time reduction. Note that g is the identity on the solution for a reduction between decision problems.

In the following, we will rely on the result that an optimization problem $\mathcal{P}$ as defined above is NP-hard if the associated decision problem is NP-complete [[17], Theorem 1.2], and this can be established by finding a polynomial time reduction from a known NP-complete problem.

We now turn to the approximation properties of minimization problems. For instance x and solution y of minimization problem $\mathcal{P}$, we define the performance ratio of y and the optimal to be

$$R_{\mathcal{P}}(x, y) = \frac{m(y)}{m^*(x)}$$

and say that $y \in s(x)$ is an $R_{\mathcal{P}}(x, y)$-approximate solution for instance x. Let $\mathcal{A}$ be an algorithm that computes $\mathcal{A}(x) \in s(x)$ for instance x of $\mathcal{P}$. We say that $\mathcal{A}$ is an approximation algorithm and $\mathcal{A}$ is an $r(|x|)$-approximation algorithm if $R_{\mathcal{P}}(x, \mathcal{A}(x)) \leq r(|x|)$ for all instances x such that $s(x) \neq \emptyset$. The class of optimization problems for which there exists a polynomial time r-approximation algorithm where r is a constant is called APX.

Definition 1 *[[17], Definition 8.3] Let $\mathcal{P}_1$ to $\mathcal{P}_2$ be two optimization problems as defined above, and let f and g be two functions. Let x be any instance $x \in I_{\mathcal{P}_1}$ and y any $y \in s_{\mathcal{P}_2}(f(x))$ and enumerate requirements on f and g as follows.*

1. $f(x) \in I_{\mathcal{P}_2}$ and $g(x, y) \in s_{\mathcal{P}_1}(x)$ are computable in polynomial time,

2. $s_{\mathcal{P}_1}(x) \neq \emptyset \Rightarrow s_{\mathcal{P}_2}(f(x)) \neq \emptyset$

3. $R_{\mathcal{P}_2}(f(x), y) \leq r \Rightarrow R_{\mathcal{P}_1}(x, g(x, y)) \leq 1 + \alpha(r - 1)$ for constant $\alpha \geq 1$ and constant rational $r > 1$.

If there exist functions f and g that fulfill requirements 1, 2, and 3 then the tuple (f, g, α) is an AP-reduction from $\mathcal{P}_1$ to $\mathcal{P}_2$ and we write $\mathcal{P}_1 \leq_{\mathrm{AP}} \mathcal{P}_2$.

Using AP-reductions, we can define completeness and hardness for classes of approximation problems analogously to how we do for problem complexity classes. In particular, a problem $\mathcal{P}$ is said to be APX-hard if $\mathcal{P}' \leq_{\mathrm{AP}} \mathcal{P}$ for all $\mathcal{P}'$ in APX. If $\mathcal{P}$ itself is in APX, we say that $\mathcal{P}$ is APX-complete. By transitivity of AP-reductions, given $\mathcal{P}_1 \leq_{\mathrm{AP}} \mathcal{P}_2$, we have that $\mathcal{P}_2$ is APX-hard if $\mathcal{P}_1$ is APX-complete. We will rely on this result in the analyses that follow.

Analysis roadmap

We will proceed in the analysis with two goals in mind.

• Proving NP-hardness, and

• proving approximation bounds.

Our strategy is to "bracket" the problem of interest in a sequence of problems. As we choose the sequence end-points such that we know the desired properties of these, we use reductions to propagate these results to the problems of interest.

We will prove NP-hardness of Problem 1 by reduction from the Minimum Set Cover problem (MSC). In fact, $L(P_{MSC}) \leq_p L(P_{MHT}) \leq_p L(P_{MSC})$. In order to prove the approximation properties we show that the reductions used in the NP-hardness proofs can be used to construct AP-reductions. We show that $L(P_{MSC}) \leq_{\mathrm{AP}} P_{MHT} \leq_{\mathrm{AP}} P_{MSC}$ with $\alpha = 1$. We further show that the existence of a bounded version of MSC that is in APX naturally leads to the existence of APX version of MHT. Since the bounded MSC problem is known to be APX-complete, we have that the bounded version of MHT is APX-complete. Requirement 3 in Definition 1 lets us conclude that a lower approximation bound for MSC can be used via the AP reductions to produce a lower approximation bound for MHT as well.

The optimization problems

The two problem formulations below are such that all instances can be seen as an $n \times m$ binary matrix B and the measure m for a feasible solution I is defined as $m(I) = |I|$. Hence, in the following, the problem definitions only consist of the specification of the function s for the instance B.

We can reformulate the "Minimum Haplotype Tagging" Problem 1 as follows.

Problem 2 ***(MHT)*** $s(B) = \{I \subseteq \{1, 2,..., m\} | b_i \neq b_j \Rightarrow b_i[I] \neq b_j[I]\}$

The following problem is the "Minimum Hitting Set" problem which we will use as a known starting point in our analyses.

Problem 3 ***(MSC)*** $s(B) = \{I \subseteq \{1, 2,..., n\} | \forall_j \in \{1, 2,..., m\} \sum_i \in {}_I b_{ij} > 0\}$.

Johnson [12] presents a greedy 1 + ln m-approximation algorithm for MSC(B). Furthermore, if we bound m from above by a constant K, the MSC problem becomes APX-complete and approximable by $\sum_{i=1}^{K} \frac{1}{i} - \frac{1}{2}$ [18].

NP-hardness

Lemma 1 $L(\mathrm{MSC}) \leq_p L(\mathrm{MHT}) \leq_p L(\mathrm{MSC})$.

Proof: (sketch) We will in the following represent the reduction from $L(\mathcal{P})$ to $L(\mathcal{P}')$ by two polynomial time computable functions f and g such that

$$(x, g(x,k')) \in L(\mathcal{P}) \Leftrightarrow (f(x),k') \in L(\mathcal{P}').$$

$L(\mathrm{MSC}) \leq_p L(\mathrm{MHT})$: Let the Boolean $n \times m$ matrix B be the transposed of an instance of the MSC problem such that $s(B^t) \neq \varnothing$. Note that we can represent the integers between 0 and n - 1 using a bit-string of minimal length $[\log_2 n]$. Let $b(i, j)$ denote the jth bit of this representation of the integer i, $0 \leq i < n$. Let $B' = \left\{b'_{ij}\right\}_{i=1,j=1}^{2n,m+\lceil \log_2 n \rceil}$ be the $(2n) \times (m + [\log_2 n])$ matrix constructed as follows.

$$b'_{ij} \begin{cases} 0 & \text{for } 1 \leq i \leq n, 1 \leq j \leq m \\ \mathrm{b}(i \bmod n, j) & \text{for } m < j \leq m + \lceil \log_2 n \rceil \\ b_{(i-n)j} & \text{for } n < i \leq 2n, 1 \leq j \leq m \end{cases}$$

Then $f_{MHT}(B) = B'$ is an $2n \times m + [\log_2 n]$ instance of the MHT problem. We now have to show the existence of g_{MHT}. Define $d(i,j) = \left\{k \middle| b'_{ik} \neq b'_{jk}\right\}$. Note that any solution S' of the MHT problem has to contain an element of $d(i, j)$ for any pair $i \neq j$. From the construction of B' we can see that any solution S' has to contain $S'' = \{i | m < i < m + [\log_2 n]\}$ and that S'' is sufficient for any (i, j) such that i mod $n \neq j$ mod n. Disregarding the cases of $i = j$, we are left with the cases $1 \leq i \leq n$ and $j = 2i$. For these cases we have that $d(i, 2i) = \{k | b_{ik} = 1\}$. We then have that S' has to contain S'' for (i, j) such that i mod $n \neq j$ mod n, and S' has to intersect $d(i, j) = \{k | b_{ik} = 1\}$ for $1 \leq i \leq n$ and $j = 2i$, the latter meaning S' has to contain a solution of the MSC problem instance B^t. Noting that the $d(i, 2i) \cap S'' = \varnothing$, we see that as $|S'| = |S''| + k$ we have that $g_{MHT}(x, k) = k - [\log_2 n]$ is the g_{MHT} we are looking for. As both f_{MHT} and g_{MHT} are computable in polynomial time, we have the sought reduction.

$L(\mathrm{MHT}) \leq_p L(\mathrm{MSC})$: Let the $n \times m$ matrix B be an instance of the MHT problem. Let $d(i, j) = \{k | b_{ik} \neq b_{jk}\}$. Let C be the $n' \times m$ Boolean matrix representation of $C = \{d(i, j) \neq \varnothing | 1 \leq i < j \leq n\}$, where $n' < (n^2 - n)/2$. Further, let $f_{MSC}(B) = C^t$, and let $g_{MSC}(x, k) = k$. It is again not hard to convince oneself that $s_{MHT}(B) = s_{MSC}(f_{MSC}(B))$. Then (f_{MSC}, g_{MSC}) is the sought reduction.

As we know that L(MSC) is NP-complete we have by Lemma 1 that L(MHT) is NP-hard. We also have that any solution of MHT is of size at most the size of the instance it is a solution of. This lets us conclude that there exists a polynomial time verifier for L(MHT) languages, meaning that L(MHT) is in NP and hence NP-complete by the above lemma. This in turn lets us state the following theorem.

Theorem 2 L(MHT) *is NP-complete and MHT is NP-hard.*

Approximability properties

Lemma 2 *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two minimization problems and let f and g be two functions that satisfy requirements 1. and 2. from Definition 1. Let $h(x) \geq 0$ for $x \in I_{\mathcal{P}_1}$. If $m_{\mathcal{P}_1}(g(x, y)) = m_{\mathcal{P}_2}(y) - h(x)$ for all $x \in I_{\mathcal{P}_1}$ and for all $y \in s_{\mathcal{P}_2}(f(x))$, then $(f, g, 1)$ is an AP-reduction from $\mathcal{P}_1$ to $\mathcal{P}_2$.*

Proof: We only need to check the requirement

$$R_{\mathcal{P}_2}(f(x), y) \leq r \Rightarrow R_{\mathcal{P}_1}(x, g(x, y)) \leq r.$$

We have that both $\mathcal{P}_1$ and $\mathcal{P}_2$ are minimization problems. Then,

$$\frac{m_{\mathcal{P}_2}(y)}{m^*_{\mathcal{P}_2}(f(x))} = \frac{m_{\mathcal{P}_1}(g(x,y)) + h(x)}{m^*_{\mathcal{P}_1}(x) + h(x)} = \frac{m_{\mathcal{P}_1}(g(x,y))(1 + \frac{h(x)}{m_{\mathcal{P}_1}(g(x,y))})}{m^*_{\mathcal{P}_1}(x)(1 + \frac{h(x)}{m^*_{\mathcal{P}_1}(x)})}.$$

Since $m^*_{\mathcal{P}_1}(x) \leq m_{\mathcal{P}_1}(g(x, y)$ we have that $\frac{h(x)}{m^*_{\mathcal{P}_1}(x)} \geq \frac{h(x)}{m_{\mathcal{P}_1}(g(x,y))}$ which in turn means that

$$\frac{1 + \frac{h(x)}{m_{\mathcal{P}_1}(g(x,y))}}{1 + \frac{h(x)}{m^*_{\mathcal{P}_1}(x)}} \leq 1.$$

This in turn means that

$$\frac{m_{\mathcal{P}_1}(g(x,y))}{m^*_{\mathcal{P}_1}(x)} \leq \frac{m_{\mathcal{P}_1}(g(x,y))(1 + \frac{h(x)}{m_{\mathcal{P}_1}(g(x,y))})}{m^*_{\mathcal{P}_1}(x)(1 + \frac{h(x)}{m^*_{\mathcal{P}_1}(x)})} = \frac{m_{\mathcal{P}_2}(y)}{m^*_{\mathcal{P}_2}(f(x))}.$$

We can conclude

$$\frac{m_{\mathcal{P}_2}(y)}{m^*_{\mathcal{P}_2}(f(x))} \leq r \Rightarrow \frac{m_{\mathcal{P}_1}(g(x,y))}{m^*_{\mathcal{P}_1}(x)} \leq r.$$

A similar argument can be used to prove the general case. i.e., not only for minimization problems.

Lemma 3 MSC $\leq_{AP}$ MHT $\leq_{AP}$ MSC

Proof: Recall f_{MHT} and f_{MSC} constructed in the proof of Lemma 1. We see from their construction that requirements 1. and 2. in Definition 1 all are met by the pairs (f_{MHT}, g_{MHT}) and (f_{MSC}, g_{MSC}) if we let $g_{MHT}(x, y) = y - S''$, where S" is defined as in the discussion of the reduction from $L(MSC)$ to $L(MHT)$ in the proof of Lemma 1, and $g_{MSC}(x, y) = y$.

Let $h_{MHT}(x) = [\log_2 n]$ and let $h_{MSC}(x) = 0$. We then see that both reductions together with the corresponding h match the conditions of Lemma 2, which completes our proof.

The good news

Theorem 3 *MHT is approximable within* $1 + \ln(\frac{n^2 - n}{2})$.

Proof: The theorem follows from that if B is an $n \times m$ instance of MHT, then $f_{MSC}(B)$ is the $m \times n'$, $n' \leq (n^2 - n)/2$ instance of MSC. MSC is known to be approximable within $1 + \ln m$ for $n \times m$ instances, and since we know from Lemma 3 that we have an AP-reduction from MHT to MSC with $\alpha = 1$ we have that MHT is approximable within $1 + \ln(\frac{n^2 - n}{2})$.

The bad news

Let MHT' be the version of MHT where we bound n from above by a constant.

Theorem 4 *MHT' is APX-complete.*

Proof: From Lemma 3 it follows that MSC' $\leq_{AP}$ MHT' $\leq_{AP}$ MSC', where MSC' is the version of MSC where we bound

m by a constant for $n \times m$ instances. From Theorem 3 it follows that MHT' is in APX, and as MSC' is known to be APX-complete [18], the theorem follows.

Theorem 5 *It is NP-hard to approximate MHT within* $c\ln(n/2)$ *for any constant* $c > 0$ *and MHT is not approximable within* $(1 - \varepsilon)\ln(n/2)$ *for any* $\varepsilon > 0$ *unless* NP $\subset$ DTIME$(n^{\log\log n})$.

Proof: We know [19] that it is NP-hard to approximate MSC within $c\ln m$ for any constant $c > 0$. We also know [14] that MSC is not approximable within $(1 - \varepsilon)\ln(n)$ for any constant $\varepsilon > 0$ unless NP $\subset$ DTIME$(n^{\log\log n})$. From the proof of Lemma 3, we know that MSC $\leq_{\text{AP}}$ MHT with $\alpha = 1$, meaning that

$$\frac{m_{\text{MHT}}(y)}{m^*_{\text{MHT}}(f(x))} \leq r \Rightarrow \frac{m_{\text{MHS}}(g(x,y))}{m^*_{\text{MHS}}(x)} \leq r.$$

If x is an $m \times n'$ instance of MSC, then $f_{MHT}(x)$ as defined in the proof of Lemma 3 is a $2n' \times (m + [\log_2 n])$ instance of MHT. If we have a $r(n)$-approximation algorithm for MHT, we have a $r(2n')$-approximation algorithm for MSC, and the theorem follows.

Algorithms

We present here two polynomial time algorithms for solving the MHT problem that exhibit the approximation bound found above. The first algorithm is based on the transformation to the minimum set cover problem. This allows us to guarantee the performance bounds. The second algorithm does not compute a transformation explicitly, but accomplishes the same effects directly. The second algorithm is much simpler to implement, uses significantly less space and has the same asymptotic running time as an optimal implementation of the first.

The advantage of the algorithm with the explicit transformation to the minimum set cover problem is that it lets us produce algorithms for a family of haplotype tagging problems as presented in Section.

Let B be an $n \times m$ instance of the MHT problem such that each row is unique. The first algorithm is essentially applying an implementation of f_{MSC} followed by the application of an algorithm for the set cover problem. To understand how this works, consider that each set in the collection represented in matrix form by $f_{MSC}(B)$ is associated with one column (representing a SNP) in B. We can form $(n^2 - n)/2$ pairs of row indices such that their respective rows in B are different. Let each of these pairs be associated with a unique identifier. Each set in $f_{MSC}(B)$ contains the unique identifiers of the row pairs that differ in the associated column in B. The objective of the MHT problem is to find a minimum number of columns (equivalently sets in $f_{MSC}(B)$) such that they together discern between all pairs of rows (cover all the unique row pair identifiers).

Johnson's $1 + \ln m$ greedy approximation algorithm [12] iteratively selects the set that covers the most uncovered elements and adds it to the initially empty solution. This is done until all elements are covered. It is well known that a greedy algorithm for the unweighted set cover problem on the collection C can be implemented to run in $O(\sum_{C\in C}|C| \leq O(m(n^2 - n)/2)$ time [[20], Exercise 37.3.3]. We state without proof that it is also not hard to implement f_{msc} to run within the same bounds. Hence the promised first algorithm consisting of running the implementation of f_{MSC} and applying the greedy set cover algorithm has running time $O(\sum_{C\in C}|C| \leq O(m(n^2 - n)/2)$ and exhibits the promised approximation bound. We will in the following refer to this algorithm as MHT.

The second algorithm is based on the following observation. Selecting the set that covers the most uncovered elements in the algorithm above is equivalent to selecting the column that discerns between the most pairs of previously undiscerned rows. Hence we can achieve the same effect as the first algorithm by recursively partitioning the set of rows in B by at each iteration selecting the column that refines the partition the most. This more direct algorithm we will call D-MHT. Pseudo code for it is given below.

The function delta(i, L) computes the number of pairs of rows in B indexed by elements in L that can be discerned by using column i in B, i.e., delta$(i, L) = s(l - s)$ where $s = \sum_{j\in L} b_{ji}$ and $l = |L|$. The function part(b, L) splits L into two lists, one that contains the indices of rows in B that have a 1 in column b and one that contains the indices of rows that have a 0 in column b. The function append$(LL", LL')$ appends $LL"$ to the list LL'. Note that delta(i, L) and part(b, L) run in $O(|L|)$ time, and that append() runs in $O(1)$ time. We note that these running times can also be achieved for the general case of B being a matrix D-MHT(B)

```
(1) $P \leftarrow \varnothing$
(2) $LL \leftarrow$ NULL
(3) $U \leftarrow \{1, 2,..., m\}$
(4) insert($(1, 2,..., n)$, $LL$)
(5) while $|LL| < n$
(6)     $a \leftarrow 0$
```

```
(7)  $b \leftarrow 0$
(8)  foreach $i \in U$
(9)    $s \leftarrow 0$
(10)     foreach $L \in LL$
(11)       $s \leftarrow s$ + delta($i$, $L$)
(12)     if $s > a$
(13)       $a \leftarrow s$
(14)       $b \leftarrow i$
(15)   $P \leftarrow P \cdots \{b\}$
(16)   $U \leftarrow U - \{b\}$
(17)   $LL' \leftarrow$ NULL
(18)   foreach $L \in LL$
(19)     $LL'' \leftarrow$ part($b$, $L$)
(20)     append($LL''$, $LL'$)
(21)   $LL \leftarrow LL'$
(22) return ($P$)
```

containing natural numbers, with the size of the range of values in each row being within n, making the D-MHT algorithm a general partitioning algorithm useful for among others pattern discovery.

The loop 5–21 in D-MHT is performed $p = |P|$ times. As one element gets added to P for each iteration of this loop, the loop 8–14 gets executed $\sum_{i=1}^{p}(m-i+1)$ = $p(2m - p + 1)/2$ times. As delta(i, L) is $O(|L|)$, and LL is a partition of n numbers, we have that loop 10–11 is $\sum_{L \in LL} O(|L|) = O(n)$. Hence line11 is $O(np(2m - p + 1)/2)$. Similarly, 20 is $O(np(2m - p + 1)/2)$ and 21 is $O(n)$. It follows that the running time of the algorithm is

$$O(\sum_{i=1}^{p}(m-i+1)n) = O(\frac{np}{2}(2m-p+1))$$

where $p = |P|$. We see that $p \le \min(n, m)$ by the fact that each column chosen splits at least one element in L and that for n elements, the maximal number of times just one element in L can be split is n. Also, we cannot choose more than m columns. Also, $O(\frac{np}{2}(2m - p + 1))$ is monotonously increasing with $p \le m$. If $m \le n$ we have that

$$\begin{aligned} O(\frac{np}{2}(2m-p+1)) &\le O(\frac{nm}{2}(2m-m+1)) \\ &= O(n(m^2+m)/2). \end{aligned}$$

If on the other hand $m > n$, then we have that

$$O(\frac{np}{2}(2m-p+1)) \le O(\frac{n^2}{2}(2m-n+1)).$$

Now assume that $m > n > 1$, and look at the ratio of the running times of D-MHT and MHT respectively. This ratio is given as

$$\frac{n^2(2m-n+1)}{m(n^2-n)} = \frac{2n(nm)-n^2(n-1)}{(mn)(n-1)} = \frac{2n}{(n-1)} - \frac{n}{m}.$$

Again using that $m > n$, we see that

$$\frac{2n}{(n-1)} - \frac{n}{m} \le \frac{2n}{(n-1)} \le 4$$

for $n > 1$. Indeed, as $n \to \infty$ the above approaches 2.

Now assume that $1 < m \le n$. Then we have that the same ratio is

$$\frac{n(m^2+m)}{m(n^2-n)} = \frac{(nm)(m+1)}{(mn)(n-1)} = \frac{(m+1)}{(n-1)} \le 3.$$

The ratio above is maximal when $m = n$, 1 when $m = n - 2$, and we see a linear increase of the reciprocal of the ratio with the quantity $n - m$. Knowing that $p \le \min(n, m)$ yields

$$\frac{np}{2}(2m-p+1) \le 4\frac{m(n^2-n)}{2}.$$

We hence conclude that the direct algorithm has the same worst case asymptotic running time as the one based on the optimal implementation of the minimum set cover problem. The direct algorithm is simpler to implement, might run faster in short solution cases, and has significantly smaller space requirements as we do not need to construct the set cover instance. Also, it might be of interest to weight SNPs. While including weights is trivial in the D-MHT algorithm, implementing an $O(mn)$ version of the minimum weighted set cover problem can prove to be a difficult task. This is because the unweighted algorithm depends on a data structure that allows remove-max, remove, and insert in constant time but depends on keys to lie in a known, finite, and small set. However, a $O((mn) \ln(mn))$ implementation is readily available.

A third algorithm can be constructed by applying the best suited of the two algorithms presented above. The analysis suggests that one could choose MHT over D-MHT whenever $m > n - 2$.

Authors' contributions

SAV analyzed the approximation properties of the MHT problem and suggested the D-MHT algorithm. SD participated in the analysis of the running time complexities and the placement of the paper in context. LOM related the work to the context it is presented in. All authors wrote parts of the manuscript.

Acknowledgements

This work was supported by grant R01-LM07273 from the National Library of Medicine.

References

1. Carlson C, Eberle M, Kruglyak L, Nickerson D: **Mapping complex disease loci in whole-genome association studies.** *Nature* 2004, **429(6990):**446-452.
2. Reich D, Cargill M, Bolk S, Ireland J, Sabeti P, Richter D, Lavery T, Kouyoumjian R, Farhadian S, Ward R, Lander E: **Linkage disequilibrium in the human genome.** *Nature* 2001, **411(6834):**199-204.
3. Goldstein D, Weale M: **Population genomics: Linkage disequilibrium holds the key.** *Current Biology* 2001, **11:**R576-R579.
4. International SNP Map Working Group: **A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms.** *Nature* 2001, **409(6822):**928-933.
5. Wiuf C, Laidlaw Z, Stumpf MPH: **Some notes on the combinatorial properties of haplotype tagging.** *Mathematical Biosciences* 2003, **185:**205-216.
6. Thompson D, Stram D, Goldgar D, Witte J: **Haplotype tagging single nucleotide polymorphisms and association studies.** *Hum Hered* 2003, **56(1–3):**48-55.
7. Ke X, Cardon L: **Efficient selective screening of haplotype tag SNPs.** *Bioinformatics* 2003, **19(2):**287-288.
8. Sebastiani P, Lazarus R, Weiss S, Kunkel L, Kohane I, Ramoni M: **Minimal haplotype tagging.** *Proc Natl Acad Sci USA* 2003, **100(17):**9900-9905.
9. Johnson G, Esposito L, Barratt B, Smith A, Heward J, Genova GD, Ueda H, Cordell H, Eaves I, Dudbridge F, Twells R, Payne F, Hughes W, Nutland S, Stevens H, Carr P, Tuomilehto-Wolf E, Tuomilehto J, Gough S, Clayton D, Todd J: **Haplotype tagging for the identification of common disease genes.** *Nature Genet* 2001, **29(2):**233-237.
10. Bafna V, Halldorsson BV, Schwartz R, Clark AG, Istrail S: **Haplotypes and informa tive SNP selection algorithms: don't block out information.** In *Proceedings of the seventh annual international conference on Computational molecular biology* ACM Press; 2003:19-27.
11. Hudson RR, Kaplan N: **Statistical properties of the number of recombination events in the history of a sample of DNA sequences.** *Genetics* 1985, **111:**147-164.
12. Johnson DS: **Approximation Algorithms for Combinatorial Problems.** *Journal of Computer and System Sciences* 1974, **9:**256-278.
13. Hochbaum D, Pathria A: **Analysis of the Greedy Approach in Covering Problems.** *Naval Research Quarterly* 1998, **45:**615-627.
14. Feige U: **A threshold of ln *n* for approximating set cover.** *J ACM* 1998, **45:**634-652.
15. Peleg D, Schechtman G, Wool A: **Approximating bounded 0–1 integer linear programs.** In *Proc. 2nd Israel Symp. on Theory of Computing and Systems* IEEE Computer Society, IEEE Computer Society; 1993:69-77.
16. EM Arkin RH: **Minimum Diameter Covering Problems.** 1994 [Http://www.ams.sunysb.edu/~estie/publications.html].
17. Ausiello G, Crescenzi P, Gambosi G, Kann V, Marchetti-Spaccamela A, Protasi M: *Complexity and approximation: combinatorial optimization problems and their approximability properties* Springer-Verlag; 1999.
18. Duh R, Fürer M: **Proc. 29th Ann. ACM Symp. on Theory of Comp.** In *Complexity and approximation: combinatorial optimization problems and their approximability properties* ACM; 1997:256-265.
19. Raz R, Safra S: **A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP.** In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* ACM Press; 1997:475-484.
20. Cormen TH, Leiserson CE, Rivest RL: *Introduction to Algorithms* MIT Press/McGraw-Hill; 1990.