Algorithmica (2004) 40: 219–234
DOI: 10.1007/s00453-004-1110-5

Algorithmica
© 2004 Springer-Verlag New York, LLC

Approximating Min Sum Set Cover[1]

Uriel Feige,[2] László Lovász,[3] and Prasad Tetali [4]

Abstract. The input to the *min sum set cover* problem is a collection of n sets that jointly cover m elements. The output is a linear order on the sets, namely, in every time step from 1 to n exactly one set is chosen. For every element, this induces a first time step by which it is covered. The objective is to find a linear arrangement of the sets that minimizes the sum of these first time steps over all elements.

We show that a greedy algorithm approximates min sum set cover within a ratio of 4. This result was implicit in work of Bar-Noy, Bellare, Halldorsson, Shachnai, and Tamir (1998) on *chromatic sums*, but we present a simpler proof. We also show that for every $\varepsilon > 0$, achieving an approximation ratio of $4 - \varepsilon$ is NP-hard. For the min sum vertex cover version of the problem (which comes up as a heuristic for speeding up solvers of semidefinite programs) we show that it can be approximated within a ratio of 2, and is NP-hard to approximate within some constant $\rho > 1$.

Key Words. Greedy algorithm, Randomized rounding, NP-hardness, Threshhold.

1. Introduction.

The *min sum set cover* (**mssc**) problem is a problem related both to the classical *min set cover* problem and to the linear arrangement problems. Problems related to set cover are often expressed in terms of *sets* that cover *points*. Equivalently, they can be expressed as problem on hypergraphs, with *vertices* that cover *hyperedges*. We use the latter representation, which is also the more common representation for linear arrangement problems.

Notation. $H(V, E)$ denotes a hypergraph H with vertex set V and hyperedge set E, where each hyperedge is a set of vertices. A hypergraph is r*-uniform* if every hyperedge contains exactly r vertices. A 2-uniform hypergraph is simply a graph, and in this case we use the notation $G(V, E)$ and use the term *edge* rather than hyperedge. A hypergraph is d*-regular* if every vertex has degree d, namely, is contained in exactly d hyperedges.

Min sum set cover (**mssc**). For hypergraph $H(V, E)$, a linear ordering is a bijection f from V to $\{1, \ldots, |V|\}$. For a hyperedge e and linear ordering f, we define $f(e)$ as the minimum of $f(v)$ over all $v \in e$. The goal is to find a linear ordering that minimizes $\sum_e f(e)$.

We note that minimizing the sum of $f(e)$ is equivalent to minimizing the average of $f(e)$. So another way of viewing **mssc** is as that of seeking a linear

[1] A preliminary version of this paper appeared in the conference proceedings of APPROX 2002. The work of the first author was supported in part by a grant from the Israel Science Foundation. The research of the third author was in part supported by NSF Grant DMS-0100298.
[2] Department of Computer Science and Applied Mathematics, The Weizmann Institute, Rehovot 76100, Israel. uriel.feige@weizmann.ac.il.
[3] Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA. lovasz@microsoft.com.
[4] School of Mathematics and College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0160, USA. tetali@math.gatech.edu.

Received March 2003; revised January 2004. Communicated by S. Leonardi.
Online publication July 29, 2004.

arrangement of the vertices of a hypergraph that minimizes the average cover time for the hyperedges.

An important special case of **mssc** is the following.

Min sum vertex cover (**msvc**). The hypergraph is a graph $G(V, E)$. Hence one seeks a linear arrangement of the vertices of a graph that minimizes the average cover time of the edges. Linear arrangement problems on graphs often come up as heuristics for speeding up matrix computation. Indeed, **msvc** came up in (Section 4 of [4]) in the context of designing efficient algorithms for solving semidefinite programs, and was one of the motivations for our work.

Another problem that in a sense is a special case of **mssc** is the following.

Min sum coloring. The input to this problem is a graph. The output is linear ordering of its independent sets or, equivalently, a legal coloring of its vertices by natural numbers. The objective is to find such a coloring that minimizes the sum of color-numbers assigned to vertices. Given an input graph $G'(V', E')$, one can cast the min sum coloring problem as an **mssc** problem as follows. The vertices of the hypergraph H are the independent sets of G', and the hyperedges of the hypergraph H are the vertices V'. Note however that the size of the hypergraph H would typically be exponential in the size of the graph G'. Min sum coloring has been extensively studied in the past and many of the results carry over to **mssc**. We mention the results of [2] and [3] later.

All the above problems are NP-hard, and we study their approximability.

1.1. *Related Work*. We are not aware of previous work on the **mssc** problem. Regarding **msvc**, this problem was suggested to us by the authors of [4]. They use a greedy algorithm that repeatedly takes the vertex of largest degree in the remaining graph as a heuristic for **msvc**. The problem **msvc** itself is used as a heuristic for speeding up a solver for semidefinite programs.

Min sum coloring was studied extensively. It models the issue of minimizing average response time in distributed resource allocation problems. The vertices of the underlying graph (the so-called *conflict graph*) represent tasks that need to be performed, and an edge between two vertices represents a conflict—the corresponding tasks cannot be scheduled together. Part of the difficulty of the min sum coloring problem is that of identifying the independent sets in the conflict graph, which makes it more difficult than **mssc** (where the underlying hypergraph is given explicitly). In [2] it is observed that min sum coloring is hard to approximate within a ratio of $n^{1-\varepsilon}$ for every $\varepsilon > 0$, due to the hardness of distinguishing between graphs that have no independent sets of size n^{ε} and graphs that have a chromatic number below n^{ε} (which is shown in [8]). This hardness result does not apply to **mssc**.

In [2] it is shown that the greedy algorithm that iteratively picks (and removes) the largest independent set in the graph approximates min sum coloring within a factor of 4. This algorithm can be applied for certain families of graphs (such as perfect graphs), and also in the case of **mssc** (where of course we iteratively pick the vertex with largest degree in the remaining hypergraph). We observe that the proof in [2] of the factor 4 approximation applies also to **mssc** (and not just to the special case of min sum coloring). Hence **mssc** is approximable within a factor of 4.

In [3] examples are shown where the greedy algorithm does not approximate min sum coloring within ratios better than 4, showing the optimality of the analysis in [2].

We observe that the proof given there also applies to the use of the greedy algorithm for **msvc** (which is the algorithm used in [4]).

There are close connections between **mssc** and set cover. For problems related to set cover, tight approximation thresholds (up to low-order terms) are often known. Examples include $\ln n$ for min set cover and $(1 - 1/e)$ for max k-cover [6], $\ln n$ for the Domatic Number [7], roughly $\sqrt{n}$ for maximum disjoint packing of sets (a result published in the context of auction design). This is some indication that one may be able to find a tight approximation threshold for **mssc**. However, we point out a major difference between **mssc** and other problems related to set cover. Given an instance of **mssc** which is composed of two disjoint instances, the optimal solution is not necessarily a combination of the optimal solutions to each of the subinstances. (For example, consider a graph G_1 on nine vertices $u, v_1, w_1, \dots, v_4, w_4$ in which vertex u is connected as a star to vertices $v_1, \dots, v_4$, and for every $1 \leq i \leq 4$, v_i is connected to w_i. The optimal solution to **msvc** first uses u to cover four edges, and then covers the remaining edges one by one. However, if we consider a graph G that is the disjoint union of G_1 and G_2, where G_2 is a graph consisting of three isolated edges, the optimal solution for **msvc** becomes first to take $v_1, \dots, v_4$, and then to cover the three edges of G_2 one by one.) This makes it more difficult to design and analyze algorithms for **mssc**. In particular, we do not even know if there is a polynomial time algorithm for **msvc** when the underlying graph is a tree (whereas min vertex cover is polynomial time solvable on trees.) As we shall see later, the hardest instances for **mssc** (in terms of approximation ratio) have different properties than the hardest instances for min set cover. A major difference (already manifested in [3]) is that they are not regular.

1.2. *New Results*. The main result regarding the approximation of **mssc** is the following.

THEOREM 1.

1. *The greedy algorithm approximates* **mssc** *within a ratio no worse than* 4.
2. *For every* $\varepsilon > 0$, *it is NP-hard to approximate* **mssc** *within a ratio of* $4 - \varepsilon$.

As noted earlier, the first part of Theorem 1 was essentially already proved in [2]. In [10] we presented a simpler alternative proof (inspired by the primal–dual approach for approximation algorithms based on linear programming). The proof presented here is a further simplification of the proof from [10]. We also show that this proof in fact works for a related version of the **mssc** problem. This version is called the f-**mssc** problem and is considered after the proof of Theorem 4 in Section 2.

The second part of Theorem 1 is proved by modifying a reduction of [6], and combining it with ideas from [3].

For **msvc**, we observe that the results of [3] imply that the greedy algorithm does not approximate it within a ratio better than 4. We then show:

THEOREM 2.

1. *An approximation algorithm based on linear programming approximates* **msvc** *within a ratio of* 2.
2. *There exists a constant* $\rho > 1$ *such that* **msvc** *is NP-hard to approximate within a ratio better than* ρ.

The first part of Theorem 2 is proved by using a linear programming relaxation for **msvc**, and rounding it using a randomized rounding technique. We conjecture that the integrality ratio of the linear programming is in fact better than 2, and that our approximation ratio for **msvc** can be improved upon by using a more sophisticated rounding technique.

Our last set of results relate to the special case of **mssc** instances on r-uniform d-regular instances. We observe that on such instances **mssc** can be approximated within a ratio of $2r/(r+1)$. For large values of r, this approximation ratio tends to 2. For **msvc** (where $r = 2$), this approximation ratio is $\frac{4}{3}$. Our main extensions of these results are as follows:

THEOREM 3.

1. *For every $\varepsilon > 0$, there exist r, d such that it is NP-hard to approximate* **mssc** *within a ratio better that $2 - \varepsilon$ on r-uniform d-regular hypergraphs.*
2. *For some $\rho < \frac{4}{3}$ and every d,* **msvc** *can be approximated within a ratio of ρ on d-regular graphs.*

The first part of Theorem 3 is obtained as part of the proof of the second part of Theorem 1. The proof of the second part of Theorem 3 uses semidefinite programming.

2. The Greedy Algorithm.

Let $H(V, E)$ be a hypergraph on which we wish to approximate **mssc**. The greedy algorithm produces a sequence of vertices that cover all hyperedges as follows:

```
1. Initialize $i = 1$.
2. While hypergraph $H$ has an edge do
   (a) Take $v_i$ to be a vertex of maximum degree in $H$.
   (b) Update $H$ by removing $v_i$ and all hyperedges incident with it from $H$.
   (c) Increment $i$.
```

THEOREM 4. *The greedy algorithm approximates* **mssc** *within a ratio no worse than* 4.

PROOF. The proof of Theorem 4 is based on a sequence of equalities and inequalities. Most of them hold for every algorithm for **mssc** and not just for the greedy algorithm. The only place where we use properties of the greedy algorithm is toward the end of the proof of Proposition 6.

Let **opt** denote the optimal value of the **mssc** problem. Let **greedy** denote the value returned by the greedy algorithm.

For $i = 1, 2, \ldots, n$, let X_i denote the set of edges first covered in step i by the greedy algorithm. Let $R_i = E - \bigcup_{j=1}^{i-1} X_i$ be the set of edges not covered prior to step i. Note that **greedy** $= \sum_{i=1}^{n} i|X_i|$, and equivalently

$$\textbf{greedy} = \sum_{i=1}^{n} |R_i|. \tag{1}$$

Define, for every $1 \le i \le n$, $P_i = |R_i|/|X_i|$. For every edge $e \in X_i$, define its *price* as $p_e = P_i$. Now we define **price** $= \sum_e p_e$.

Summing the price over sets X_i,

$$\textbf{price} = \sum_i |X_i| P_i = \sum_i |X_i| \frac{|R_i|}{|X_i|} = \sum_i |R_i|. \tag{2}$$

PROPOSITION 5. *For the assignment of prices given above*, **price** = **greedy**.

PROOF. Follows by comparing (1) and (2). □

PROPOSITION 6. *For the assignment of prices given above*, **opt** $\ge$ **price**$/4$.

PROOF. Consider the following diagram (see Figure 1 for an illustration) corresponding to the optimal solution. There are $|E|$ columns, one for every edge, where the edges are ordered from left to right by the order in which they were covered by the optimal algorithm. The height of a column is the time step at which it was covered by the optimal algorithm. Hence we get a histogram with nondecreasing integer heights. The total area beneath this histogram is exactly **opt**.

Consider now another diagram corresponding to the greedy solution. Again there are $|E|$ columns, one for every edge, and, in analogy to the previous diagram, the edges are ordered by the order in which they were covered by the greedy solution. However, unlike the previous diagram, the height of a column is not the time step by which the

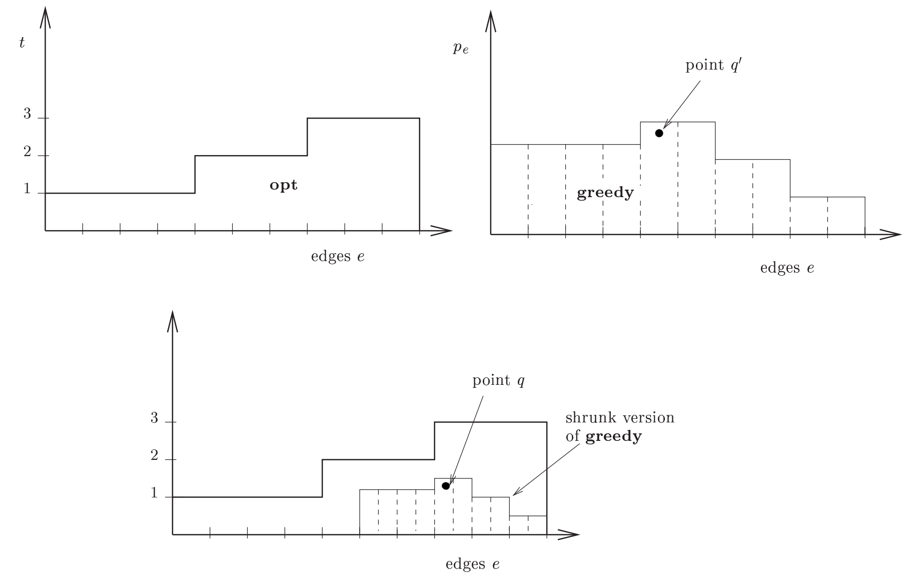

Fig. 1. Greedy at most four times **opt**.

edge was covered, but rather its price. Hence the heights need not be integer, and need not be monotone. The total area of the histogram is exactly **price**.

We want to show that the area of the second histogram is at most four times that of the first. To do this we shrink the second histogram by a factor of 4 as follows. We shrink the height of each column by a factor of 2. Hence column heights are $p_e/2$. We shrink the width of each column by a factor of 2. Hence the total width of the second histogram is now $|E|/2$. We align the second histogram to the right. Namely, it now occupies the space that was previously allocated to columns $|E|/2 + 1$ up to $|E|$ (assume for simplicity of notation and without loss of generality that $|E|$ is even). Now we claim that this shrunk version of the second histogram fits completely within the first histogram, implying that its total area is no more than that of the first histogram. This suffices in order to prove Proposition 6.

Consider an arbitrary point q' in the original second histogram, let e be the edge to which it corresponds, and let i denote the time step by which the greedy algorithm covered edge e. Then the height of q' is at most $p_e = |R_i|/|X_i|$, and the distance of q' from the right-hand side boundary is at most $|R_i|$. The shrinking of the second histogram maps q' to a new point q. We now show that q must lie within the first histogram. The height of q (which we denote by h) satisfies $h \leq |R_i|/2|X_i|$, and the distance of q from the right-hand side boundary (which we denote by r) satisfies $r \leq |R_i|/2$.

For this point q to lie within the first histogram, it suffices to show that by time step h (rounded down to the nearest integer), at least r edges (rounded up to the nearest integer) are still uncovered by the optimal algorithm. Consider now only the edges in the set R_i. No vertex whatsoever can cover more than $|X_i|$ edges from R_i. (This last assertion is the only place in our proof where we use the property of the greedy algorithm.) Hence in $\lfloor h \rfloor$ time steps the optimal could cover at most $\lfloor h \rfloor |X_i| \leq \lfloor |R_i|/2 \rfloor$ edges from R_i, leaving at least $\lceil |R_i|/2 \rceil \geq \lceil r \rceil$ edges of R_i uncovered. Hence the point q indeed lies within the first histogram. □

Summing up:

$$\textbf{opt} \geq \textbf{price}/4 = \textbf{greedy}/4,$$

where the inequalities follow from Propositions 6 and 5, respectively. This completes the proof of Theorem 4. □

In [3] it is shown that for every $\varepsilon > 0$, there are instances of **mssc** for which the approximation ratio of the greedy algorithm is no better than $4 - \varepsilon$. (Technically, this result is stated for min sum coloring, but it applies also to **mssc** which is a more general problem.) See also Proposition 8.

THE f-**mssc** PROBLEM. We observe that the above analysis can in fact be used to guarantee the same factor of approximation for the following related problem. For $0 < f \leq 1$, let f-**mssc** be the **mssc** problem in which we only charge for the first $f|E|$ edges that are covered, and we charge nothing for the remaining $(1 - f)|E|$ edges. (This problem is in turn related to a problem studied in [5]. However, in [5] the remaining $(1 - f)|E|$ edges are charged the same as the $f|E|$th covered edge.) It is easy to see that for every f the worst case approximation ratio of greedy on f-**mssc** cannot be better than that on **mssc**. (Given an instance of **mssc** with m edges, one can reduce it to an instance of f-**mssc** by adding $(1-f)m/f$ auxiliary edges that can only be covered one by one.) The following theorem shows that it is no worse.

THEOREM 7. *The greedy algorithm approximates f-**mssc** within a ratio no worse than* 4.

PROOF. The proof is similar to that of Theorem 4. We just point out the differences.

- The price of an edge e covered by set X_i is now
$$P_i = \frac{|R_i| - (1-f)|E|}{|X_i|}$$
if e is among the first $f|E|$ edges covered by the greedy algorithm, and 0 otherwise.
- The first histogram is 0 beyond column $f|E|$.
- The shrunk version of the second histogram is aligned with the first histogram so that (the end of) columns $f|E|$ coincide. Beyond this column, both histograms are 0.

As before, consider a point q in the nonzero part of the shrunk second histogram. The height of q is at most $t = (|R_i| - (1-f)|E|)/2|X_i|$. The distance of q from the column numbered $f|E|$ is at most $(|R_i| - (1-f)|E|)/2$, and the distance of q from the right-hand side boundary is $(1-f)|E|$ more, namely, at most $(|R_i| + (1-f)|E|)/2$. By time step t, the optimal solution covers at most $t|X_i| = (|R_i| - (1-f)|E|)/2$ edges from R_i. Hence the number of uncovered edges from R_i is at least $(|R_i| + (1-f)|E|)/2$. Hence q lies also inside the first histogram. □

3. Min Sum Vertex Cover.

As noted earlier, Bar-Noy et al. [3] provide an example showing the tightness of analysis of factor 4 for the greedy algorithm on min sum coloring. As shown below, the same construction can be used to describe a bipartite multigraph on which the greedy algorithm performs no better than a factor of $4-\varepsilon$ of the optimal algorithm to solve the **msvc** problem. Moreover, this multigraph can be further modified to give a simple graph.

PROPOSITION 8. *There exist simple bipartite graphs on which the greedy algorithm performs no better than a factor of* $4 - o(1)$ *of the optimal algorithm for the* **msvc** *problem.*

PROOF. First we comment on the construction of the bipartite multigraph, and then we describe how to convert it into a bipartite *simple* graph without altering (up to at least the first-order terms) the performance of either the greedy or the optimal algorithm.

Since the following construction and the algorithmic analysis involved are very much based on "the chopping procedure" of Bar-Noy et al. (see Sections 2 and 5 of [3]), we keep the discussion brief. The resulting bipartite multigraph $G = (U \cup V, E)$ will have the property that one of the greedy solutions ends up always picking vertices from U in the order of decreasing degrees to cover all the edges, while it is much better to cover the edges by always choosing the vertices from V in the order of decreasing degrees. For arbitrary $x > 1$, and arbitrary $n > 1$, a multigraph with the above feature can be constructed with the additional properties that the degree sequence of V is $(x, x, x/4, x/9, \cdots, x/(n-1)^2, x/n^2)$ and that of U is $(x, x/2, x/4, x/6, x/9, \cdots, x/(n-1)^2, x/(n-1)n, x/n^2, \cdots, x/n^2)$, where the degree x/n^2 appears $n+1$ times in U.

To describe the edge structure of G we start (as in [3]) with a sequence of $n+1$ columns. The columns are indexed by the elements of V. Each column will have a stack of tokens. Column 0 has x tokens, and, for $1 \leq i \leq n$, column i has x/i^2 tokens; the number of tokens in each column is equal to the degree of the corresponding vertex of V. We systematically remove tokens from these columns and assign them to the vertices of U. There are $|U| = 3n-1$ steps removing tokens, and at each step j the tokens that are removed are assigned to vertex j of U. The number of tokens removed in any step j is equal to the degree of vertex j of U. The choice of which tokens to remove will ensure that the number of tokens in the most loaded column at the beginning of any step j is not larger than the degree of vertex j of U (with equality holding for the first $2n-1$ steps). Specifically, the tokens to be removed in any step j are chosen one by one from the column currently containing the largest amount of tokens, breaking ties arbitrarily. Finally, the number n_{ij} of tokens picked from a particular column i (corresponding to vertex i of V) in step j (corresponding to vertex j of U) is precisely the multiplicity of edges between the corresponding vertices of V and U. As a toy example, the reader may check that for the parameters $x = 36$ and $n = 3$ (giving degree sequence (36, 36, 9, 4) for V and (36, 18, 9, 6, 4, 4, 4, 4) for U), the number of tokens in the most loaded column at the beginning of any step j is not larger than the degree of vertex j of U.

For arbitrary x and n, using the general degree sequences for V and U stated above, it can be verified that choosing the vertices of V in the order of nonincreasing degrees yields that

$$\mathbf{opt} \leq x + \sum_{i=1}^{n}(i+1)\frac{x}{i^2} < (H_n + 2.65)x,$$

minding the notation that $\sum_{i=1}^{n}(1/i) = H_n$ and the computation that $\sum_{i=1}^{n}(1/i^2) < 1.65$. On the other hand, it can also be checked that the greedy algorithm could indeed choose vertices of U (also in the order of nonincreasing degrees) and sustain a cost of

$$\begin{aligned}\mathbf{greedy} &= \sum_{i=1}^{n}\frac{2i-1}{i^2}x + \sum_{i=1}^{n-1}\frac{2i}{i(i+1)}x + \sum_{i=1}^{n}\frac{2n-1+i}{n^2}x \\ &> (4H_n - 1.65)x,\end{aligned}$$

establishing the tightness of factor 4.

To convert the above into a simple graph one may proceed as follows. Let k be the maximum multiplicity of any edge in the multigraph. The vertex set of the simple graph is obtained by replacing every vertex v of the multigraph by a cluster of k vertices $v_0, \ldots, v_{k-1}$ (regardless of the number of edges connected to v and their multiplicity). The edge set of the simple graph is as follows. Within a cluster there are no edges. For every edge (u, v) of the original graph, we put a complete matching between the respective clusters—we put the k edges (u_i, v_i) for $0 \leq i \leq k-1$. If edge (u, v) had a multiplicity of q, we put q edge-disjoint matchings between the clusters, namely, for every $0 \leq i \leq k-1$ and every $0 \leq j \leq q-1$ we put the edge (u_i, v_{i+j}) (where $i+j$ is computed modulo k). As $q \leq k$, all these edges are distinct, and moreover there are no parallel edges. This completes the description of the simple graph. Note that for each vertex v_i in the simple graph, its degree is equal to the degree (counting multiplicities) of its origin vertex v in the multigraph.

It is not hard to see that the greedy algorithm in the simple graph copies its actions on the multigraph, systematically covering clusters of U one by one. In analogy to the case of the multigraph, a better solution covers V cluster by cluster on the simple graph. The ratio between the value of the solutions remains unchanged up to low-order terms. (Observe that the ratio would have stayed *unchanged* had we defined the cost of an edge covered at step t as $t - \frac{1}{2}$ rather than as t. Then the cost of each solution simply multiplies by k^2. However, as we charge t for an edge covered at step t, this adds a small error term which is negligible for large enough t. As in the paper of Bar-Noy et al., only a small fraction of the edges are covered in the first few steps, making the overall deviation from the ratio of 4 negligible.) Hence the greedy algorithm does not approximate **msvc** within a ratio better than 4. $\square$

We now show a different algorithm that does approximate **msvc** within a ratio better than 4.

Consider the following integer program for **msvc**. The indices i and j run over all vertices. The index t runs over all time steps. The variable x_{it} is an indicator variable that indicates whether vertex i is chosen at step t. y_{ijt} is an indicator variable that indicates whether edge (i, j) is still uncovered before step t.

Minimize $\displaystyle\sum_{(i,j)\in E} \sum_t y_{ijt}$ **subject to**

1. $x_{it} \in \{0, 1\}$. (Integrality constraint.)
2. $y_{ijt} \in \{0, 1\}$. (Integrality constraint.)
3. $\sum_i x_{it} \leq 1$. (In every time step, at most one vertex is chosen.)
4. $y_{ijt} \geq 1 - \sum_{t'<t}(x_{it'} + x_{jt'})$. (An edge is uncovered at the beginning of time t unless one of its endpoints was covered at a previous time step.)

The integer program is relaxed to a linear program by relaxing the integrality constraints to $0 \leq x_{it} \leq 1$ and $0 \leq y_{ijt} \leq 1$. Clearly, the linear program (that is solvable in polynomial time) provides a lower bound for **msvc**.

We propose a procedure for rounding a fractional solution of the linear program. The procedure is randomized and produces an integer solution with expected value at most twice that of the linear program. We note that the rounding technique can be made deterministic using the method of conditional expectation.

The rounding technique works in two stages. The first stage is performed independently for each vertex. Consider vertex i and the fractional variables x_{it} for $t \geq 1$. Let t_i be that value of t' for which $\sum_{t<t'} x_{it} < \frac{1}{2}$ and $\sum_{t\leq t'} x_{it} \geq \frac{1}{2}$. (If no such t' exists, namely, $\sum_t x_{it} < \frac{1}{2}$, then let $t_i = \infty$.) Now introduce new variables z_{it}, where $z_{it} = 2x_{it}$

for $t < t_i$, $z_{it_i} = 1 - \sum_{t<t_i} z_{it}$, and $z_{it} = 0$ for $t > t_i$. Note that $\sum_t z_{it} \leq 1$ and that $z_{it} \leq 2x_{it}$ for every i, t. Now randomly choose at most one value of t, where value t is chosen with probability z_{it}. For the chosen t, x_{it} is rounded to 1, and for all other values of t, x_{it} is rounded to 0. Let $\bar{x}_{it}$ denote the rounded values obtained by this procedure. It is easy to check that for every optimal solution to the linear program and for every edge (i, j), either $\sum_t x_{it} \geq \frac{1}{2}$ or $\sum_t x_{jt} \geq \frac{1}{2}$ (or both). In any case, $\bar{x}$ is a cover.

The outcome of the first stage of the rounding technique satisfies the integrality constraints for the $\bar{x}_{it}$ (constraint 1) but may violate constraint 3. In the second stage of the rounding technique we scan the time steps one by one. For time step t, let $s_t = \sum_i \bar{x}_{it}$. Replace time step t by s_t time slots. Now allocate the vertices i for which $\bar{x}_{it} = 1$ to these time slots in a random order. (The value of t for which $\bar{x}_{it} = 1$ is shifted to the respective time slot.) Now constraint 3 is satisfied, because each time slot has exactly one vertex assigned to it.

Given values for $\bar{x}_{it}$ that satisfy constraints 1 and 3, a 0/1 assignment to the y_{ijt} is derived in a straightforward way (ignoring the assignment originally given to them by the fractional solution). This completes the description of the rounding procedure.

LEMMA 9. *The expected value of the rounded solution to the linear program is at most twice the fractional value of the linear program.*

PROOF. Consider an arbitrary edge (i, j) and an arbitrary time step t. The contribution of this to the fractional solution is $y_{ijt} \geq 1 - \sum_{t'<t}(x_{it'} + x_{jt'})$. We compare this to the expected contribution of edge (i, j) to time step t in the rounded solution. This contribution is a product of two factors:

1. The probability that edge (i, j) is not covered before time step t.
2. Conditioned on edge (i, j) not being covered before time step t, the expected number of time slots within time step t. (Note that we will be comparing time step t of the fractional solution with the time slots derived from it, rather than with time slot t.) Here there is subtlety involved. Without the conditioning, the expected number of time slots into which time step t is transformed (which is the expected value of s_t) would be at most two. However, the conditioning may cause the expectation to increase. To compensate for this, we use the fact that if in the rounded solution edge (i, j) was first covered in time step t, then the particular time slot within time step t in which (i, j) was covered is random, and later time slots need not be counted.

For the first factor, we compute the probability that edge (i, j) is not covered by the rounded solution before time t. This probability is

$$\left(1 - \sum_{t'<t} z_{it'}\right)\left(1 - \sum_{t'<t} z_{jt'}\right) \leq y_{ijt}, \tag{3}$$

where the inequality follows from the relation $\sum_{t'<t} z_{it'} = \min[1, 2\sum_{t'<t} x_{it'}]$, and from constraint 4, namely $y_{ijt} \geq 1 - \sum_{t'<t} x_{it'} - \sum_{t'<t} x_{it'}$. (Observe that (3) holds if the left-hand side is 0, because $y_{ijt} \geq 0$. Hence we may assume that $\sum_{t'<t} x_{it'} < \frac{1}{2}$ and $\sum_{t'<t} x_{jt'} < \frac{1}{2}$, implying $4(\sum_{t'<t} x_{it'})(\sum_{t'<t} x_{jt'}) < \sum_{t'<t} x_{it'} + \sum_{t'<t} x_{it'}$. If follows that $(1 - 2\sum_{t'<t} x_{it'})(1 - 2\sum_{t'<t} x_{it'}) \leq 1 - \sum_{t'<t} x_{it'} - \sum_{t'<t} x_{it'}$, implying (3).)

For the second factor (the value of s_t, and the random order within the time slots) we introduce two new random variables, r (for "rest") and w (for "waiting time"). r counts the number of vertices other than i and j that are rounded to 1 at time step t. w counts the number of relevant time slots within time step t (those at the beginning of which edge (i, j) is not yet covered). We are interested in the expectation of w. The value of this expectation can be expressed as a function of r, taking into acount also the random order of time slots within a time step. We obtain:

- If $\bar{x}_{it} = \bar{x}_{jt} = 0$, then $r = s_t$ and $w = r$.
- If $\bar{x}_{it} = 0$ and $\bar{x}_{jt} = 1$, or $\bar{x}_{it} = 1$ and $\bar{x}_{jt} = 0$, then $r = s_t - 1$ and $E[w] = 1 + E[r]/2$.
- If $\bar{x}_{it} = \bar{x}_{jt} = 1$, then $r = s_t - 2$ and $E[w] = 1 + E[r]/3$.

Now $E[r] = \sum_{k \neq i,j} E(\bar{x}_{kt}) = \sum_{k \neq i,j} z_{kt} \leq 2 \sum_{k \neq i,j} x_{kt} \leq 2$, due to constraint 3. It follows that in all cases $E[w] \leq 2$. Hence, altogether the contribution of y_{ijt} to the rounded solution is at most $2y_{ijt}$.

Using the linearity of expectation (over all y_{ijt}), the expected value of the rounded solution is at most twice that of the fractional solution. $\square$

The analysis of the rounding technique is essentially best possible. This can be verified by considering a graph composed of disjoint edges. The fractional solution can cover edge by edge by giving its two endpoints weight $\frac{1}{2}$. The rounded solution will then take both endpoints, paying twice as much. We conjecture that a different rounding techniques for the same linear program can give an approximation ratio better than 2. Moreover, we suspect that using semidefinite programming rather than linear programming can further improve the approximation ratio. This we can show for the special case of regular graphs. The integrality ratio of the linear program is $\frac{4}{3}$ (on a clique), whereas semidefinite programming gives a better approximation ratio (see Theorem 11).

4. Regular Hypergraphs.

Let H be an r-uniform, d-regular hypergraph. That is, each hyperedge contains exactly r vertices, and each vertex has degree d. Let n denote the number of vertices and m the number of hyperedges. (Clearly, $rm = dn$.)

For every such hypergraph, the optimal value of **mssc** is at least $m((n + r)/2r)$, because at most d hyperedges are covered per step, and at this rate it takes $m/d = n/r$ steps to cover all hyperedges. Hence the average number of steps until a hyperedge is covered is at least $(n + r)/2r$.

On the other hand, the optimal solution has value at most $m((n + 1)/(r + 1))$. This can be seen as follows. Consider a random permutation of all vertices. Then for every hyperedge, the expected step in which it is first covered is exactly $(n + 1)/(r + 1)$. (This last statement can be proven by considering a random cyclic permutation on $n + 1$ elements, of which $r + 1$ are special. Now select at random which of the $r + 1$ special elements marks the start of the permutation on the rest of the n elements, and the other r special elements are the vertices that compose the hyperedge. Then over the choice of where the permutation starts, the expected number of steps until another special element is reached is exactly $(n + 1)/(r + 1)$.) Hence there is some ordering of the vertices for which the average time to cover a hyperedge is at most $(n + 1)/(r + 1)$. Moreover, the greedy algorithm produces such an ordering. (One way of seeing this is that the method of conditional expectations produces the greedy algorithm as a derandomization of the randomized algorithm.)

The above proves the following theorem.

THEOREM 10. *For every r-uniform d-regular hypergraph, the approximation ratio of the greedy algorithm for* **mssc** *is no worse than $2r/(r+1)$. In particular, the approximation ratio of the greedy algorithm for* **msvc** *on regular graphs is no worse than $\frac{4}{3}$.*

As r gets larger, the approximation ratio of the greedy algorithm on r-uniform regular hypergraphs approaches 2. This cannot be significantly improved unless $P = NP$, as we shall see in Theorem 12. However, for the special case of $r = 2$ (regular graphs), we can improve over the greedy algorithm.

THEOREM 11. *There is some constant $1 < \rho < \frac{4}{3}$ such that* **msvc** *can be approximated within a ratio of ρ on regular graphs.*

PROOF. The central algorithmic tool used in our proof is semidefinite programming. The presentation of the algorithm is greatly simplified (perhaps at some loss in the approximation ratio) by using in a "blackbox" manner previous results regarding the use of semidefinite programming for the *max k-vertex cover* problem. For some fixed $\varepsilon > 0$, if the average step by which the optimal solution covers an edge is $(\frac{1}{4} + \varepsilon)n$, then the greedy algorithm achieves an approximation ratio of $\frac{4}{3} - \Omega(\varepsilon) < \frac{4}{3}$, as desired. Hence we can assume that the optimal solution covers at least $(1-\varepsilon)m$ edges in $n/2$ steps. This means that there is a set of $n/2$ vertices that covers $(1-\varepsilon)m$ edges.

Apply now an algorithm for the *max k-vertex cover* problem with $k = n/2$, which asks for a set of k vertices that covers as many edges as possible. As shown in [9] (and improved later by others), when $k = n/2$ this problem can be approximated (using semidefinite programming) within ratios strictly better than $\frac{4}{5}$, say, $\frac{4}{5} + \delta$. (Using [11] we can take $\delta = 0.0452$.) Hence we can find in polynomial time a set S of $n/2$ vertices that cover at least $(\frac{4}{5} + \delta - \varepsilon)m$ edges. We take $\varepsilon < \delta/2$, giving $(\frac{4}{5} + \delta/2)m$ edges.

Now cover all edges by first taking all vertices of S in a random order, and then taking the rest of the vertices in a random order. A $(\frac{4}{5} + \delta/2)$ fraction of the edges are covered on average by step $n/4$. The rest of the edges are covered on average at step $n/2 + \frac{1}{3}(n/2) = 2n/3$. Computing a weighted average, the average time to cover an edge is $(\frac{1}{3} - \Omega(\delta))n$. This gives an approximation ratio of $\frac{4}{3} - \Omega(\delta) < \frac{4}{3}$, as desired. □

5. Hardness of Approximation

THEOREM 12. *For every $\varepsilon > 0$, it is NP-hard to approximate* **mssc** *within a ratio of $2 - \varepsilon$ on uniform regular hypergraphs.*

Theorem 10 shows that Theorem 12 is essentially best possible. The proof of Theorem 12 is very similar to the proof given in [6] of the result that it is NP-hard to approximate the max k-coverage problem within a ratio better than $1 - 1/e + \varepsilon$. We do not wish to reproduce here the proof already given in [6]; instead, we provide a sketch of the proof, using the terminology of [6].

Sketch of Proof of Theorem 12. In [6] a reduction from max 3SAT-5 to max k-coverage is described. We note that the resulting instance of max k-coverage (which is a hypergraph) is *regular*—each set contains the same number of points (or, equivalently, each vertex appears in the same number of hyperedges)—but not *uniform*, since some points are covered by more sets than others (equivalently some hyperedges contain more vertices than others). To make the hypergraph also uniform, we change the starting point of the reduction. Rather than starting from a 3CNF formula in which each variable appears in exactly five clauses, we start from a 3CNF formula in which each *literal* appears in exactly three clauses (and each variable in six clauses). We call the satisfiability problem for such formulas 3SAT-6. We note that for some $\delta < 1$, it is NP-hard to distinguish between satisfiable 3SAT-6 formulas, and those in which at most a δ-fraction of the clauses are satisfiable. (This can be proven by reduction from 3SAT-5, in which each variable appears three times in positive form and twice negated. For a 3SAT-5 formula with n variables, join to it a satisfiable 2SAT-6 formula with n clauses on a fresh set of variables, and to each of the 2CNF clauses add one of the original variables negated.) The adaptation of the reduction of [6] now gives a regular uniform hypergraph; this is a consequence of the following symmetries:

1. Every clause in the CNF formula contains the same number of literals (three in our case).
2. Every literal appears in the same number of clauses (three in our case).
3. Every code word (in the proof system of Section 2 in [6]) has exactly the same Hamming weight ($\ell/2$ when the Hadamard code is used).
4. In the partition system (proof of Theorem 12 in [6]) each part has exactly the same size (m/k, using the notation of [6]).

Now there are two cases:

1. If the original 3SAT-6 formula is satisfiable, then the reduction has the property that there is a collection of disjoint sets (and necessarily of equal cardinality) that covers all points. We denote by t the number of sets used in such a cover. Hence for the **mssc**, a hyperedge is covered by step $t/2$, on average.
2. If the original 3SAT-6 formula was only δ-satisfiable (for $\delta < 1$), the reduction has the following property:

   For every choice of constants $c_0 > 0$ *and* $\varepsilon > 0$, *it is possible to choose the parameter* ℓ (*number of repetitions in the proof system*) *to be a large enough constant so that for every* $1 \leq x \leq c_0 t$, *at least a fraction of* $1 - (1 - 1/t)^x - \varepsilon$ *of the points remain uncovered by* x *sets*.

   The proof of this property is an extension of the proof of Theorem 12 in [6] and is omitted. Now for an arbitrarily small $\varepsilon > 0$, pick $c_0 \simeq -\ln \varepsilon$ and $\varepsilon \simeq \varepsilon/c_0$. We note that t may be assumed to be arbitrarily large (this is always the case in reductions to set cover, as checking whether there is a cover by t sets can be done in time roughly n^t, and having t constant would show that $P = NP$), implying that

$(1 - 1/t)$ is approximated arbitrarily well by $e^{-1/t}$. Hence the average time step by which a hyperedge is covered in the **mssc** instance is roughly at least

$$f(x) = \sum_{x=1}^{c_0 t} (e^{-x/t} - \varepsilon).$$

Approximating this sum by an integral and integrating we get $F(x) = (-te^{-x/t} - \varepsilon x)$. Substituting in F the range of the integration we get $F(c_0 t) = -te^{-c_0} - \varepsilon c_0 t \simeq -2\varepsilon t$, and $F(1) = -te^{-t} - \varepsilon \simeq -(t-1) - \varepsilon < -t + \varepsilon t$. Hence $F(c_0 t) - F(1) \geq (1 - 3\varepsilon)t$, implying that a hyperedge is covered by step $(1 - O(\varepsilon))t$, on average.

The gap between the two cases can be made arbitrarily close to a factor of 2. □

For nonregular instances of **mssc** we prove a stronger hardness of approximation result which matches the positive result of Theorem 4. The proof of the following theorem was inspired by [3].

THEOREM 13. *For every $\varepsilon > 0$, it is NP-hard to approximate* **mssc** *within a ratio of $4 - \varepsilon$ on uniform hypergraphs.*

PROOF. The proof goes via a reduction from the regular uniform case (Theorem 12). Consider an instance of uniform regular set cover with n points on which it is NP-hard to distinguish between the case in which all points can be covered by t disjoint sets, and the case in which every c sets cover at most a fraction of $1 - (1 - 1/t)^c + \varepsilon$ of the points (as in the proof of Theorem 12). For a large enough constant k, make k disjoint copies of this instance. Let $a = (k!)^2$. For each $1 \leq i \leq k$, duplicate a/i^2 times each point in copy i (all duplicates of a point e appear in exactly the same sets as the point e does). Note that the instance obtained by this process is still uniform (every point appears in the same number of sets) but not regular (sets in copy i are larger than those in copy j, for $i < j$).

We claim that if **mssc** can be approximated within a ratio better than 4 on this instance, then **mssc** can be approximated within a ratio better than 2 on the original instance.

Consider first the case that the original regular **mssc** instance can be covered by t disjoint sets. Then the optimal way of covering the new instance is to cover it copy by copy, starting with copy 1 and ending with copy k. The contribution $F(i)$ of copy i to the objective function is

$$F(i) = \left[(i-1)t + \frac{t}{2}\right] n \frac{a}{i^2}.$$

Asymptotically, for large i we have that $F(i) \simeq tna/i$. It follows that the cost of the whole solution is roughly

$$\sum_{i=1}^{k} F(i) \simeq \sum_{i=1}^{k} \frac{tna}{i} \simeq tna \ln k.$$

An important consequence of this is that the contribution of $F(i)$ to the total cost summed over all "small" values of i (e.g., for all $i < 100$, when k is a sufficiently large constant) is negligible compared with the total cost.

Consider now the case that for the original regular **mssc** instance, every c sets cover at most a fraction of $1 - (1 - 1/t)^c + \varepsilon$ of the points (for every $c \leq c_0 t$, where c_0 is a sufficiently large constant). In this case the best way to cover the new instance is to start with copy 1, when only $na/2^2$ points remain in copy 1 continue with copies 1 and 2 simultaneously, when only $na/3^2$ points remain in each of copies 1 and 2 continue with copies 1, 2, and 3 simultaneously, and so on. Any better way of covering the new instance gives a better way of covering the original instance, which is a contradiction. Let $I(i)$ denote the contribution of copy i to the objective function. This contribution is computed as the difference in the **mssc** cost of covering the first i copies compared with the **mssc** cost of covering the the first $i - 1$ copies. This difference occurs only at the time the first point of set i gets covered. This time step is $\sum_{j=1}^{i-1} t \ln(i^2/j^2)$. At this point, when we need to cover i copies we still have ina/i^2 elements left, and the expected additional time to cover each element is ti. When we need to cover $i - 1$ copies we still have $(i - 1)na/i^2$ elements left, and the expected additional time to cover each element is $t(i - 1)$. It follows that

$$I(i) = n\frac{a}{i^2}\left[i\left(\sum_{j=1}^{i-1} t \ln \frac{i^2}{j^2} + it\right) - (i-1)\left(\sum_{j=1}^{i-1} t \ln \frac{i^2}{j^2} + (i-1)t\right)\right].$$

Simplifying, one gets that $I(i) = n(a/i^2)[\sum_{j=1}^{i-1} t \ln(i^2/j^2) + (2i - 1)t]$. The term $\sum_{j=1}^{i-1} t \ln(i^2/j^2) = 2\sum_{j=1}^{i-1} t(\ln i - \ln j) \simeq 2ti$, which can be verified by approximating the sum by an integral, and noting that the integral of $\ln j$ is $j \ln j - j$. Hence we obtain that for large i,

$$I(i) \simeq n\frac{a}{i^2} \cdot 4ti \simeq 4F(i).$$

Summing up, the factor 2 gap in the regular **mssc** problem is amplified to a factor 4 gap in the irregular **mssc** problem. □

We now prove hardness of approximation for **msvc**.

THEOREM 14. *For some $\varepsilon' > 0$, it is NP-hard to approximate* **msvc** *within ratios better than $1 + \varepsilon'$.*

PROOF. For some universal constant $d \geq 3$, let G be a graph with n vertices, m edges, and degree at most d. It is known that for every $d \geq 3$, min vertex cover is hard to approximate on graphs of degree at most d (implicit in [1]). Moreover, for these NP-hard instances the minimum vertex cover contains at least $n/2$ vertices. This follows from the fact that whenever a graph has a vertex cover with less than $n/2$ vertices, a set of vertices not belonging to the minimum vertex cover can be found in polynomial time (see, for example, [12]), and then the input instance can be simplified. It follows that for every $d \geq 3$ it is NP-hard to approximate min vertex cover within an additive factor of εn, where $\varepsilon > 0$ may depend on d, but is independent of n. We reduce the problem of approximating the min vertex cover problem on bounded degree graphs to the problem of approximating **msvc**.

Let $k = d/2\varepsilon$. Construct a graph G' that is the disjoint union of G and kn additional isolated edges (i.e., the vertex disjoint union of G and a matching of size kn). On G' we wish to approximate **msvc**. The optimal solution to **msvc** on G' may be assumed without loss of generality first to cover all edges of G, and only then to cover the isolated edges, because the isolated edges can be covered at a rate of at most one at a time, and edges in G can be covered at a rate of at least one at a time.

Let us consider the case that G has a vertex cover with at most t vertices. Then G' has an **msvc** of value at most $m(t/2) + kn(t + kn/2)$.

Let us now consider the case that G has no vertex cover with less than $t + \varepsilon n$ vertices. Then it costs at least $kn(t + \varepsilon n + kn/2)$ to cover the kn isolated edges, and we use this as a lower bound on the value of **msvc** for G'.

The difference between the two cases is at least $\varepsilon kn^2 - m(t/2)$. Using $m \le dn/2$ and $t \le n$, this difference is at least $n^2(\varepsilon k - d/4)$. Using $k = d/2\varepsilon$, this difference is at least $(\varepsilon k/2)n^2$. The optimal solution to **msvc** on G' is of value at most $m(n/2) + kn(n + kn/2) \le k^2n^2$, where the last inequality follows from simple manipulations, using $d \ge 3$ and $\varepsilon \le \frac{1}{2}$ (which are true in our context). Setting $\varepsilon' = 1/2k = \varepsilon/d$, it follows that if we could approximate **msvc** in G' within a ratio better than $1 + \varepsilon'$, then we could approximate min vertex cover in G within a ratio better than δ. □

We have not made an effort to find the best possible value of ε' for Theorem 14.

Acknowledgements. We thank Uri Barenholz for useful comments on an earlier version of this manuscript.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[2] A. Bar-Noy, M. Bellare, M. Halldorsson, H. Shachnai, and T. Tamir. On chromatic sums and distributed resource allocation. *Information and Computation*, 140:183–202, 1998.

[3] A. Bar-Noy, M. Halldorsson, and G. Kortsarz. A matched approximation bound for the sum of a greedy coloring. *Information Processing Letters*, 71: 135-140, 1999.

[4] S. Burer and R. Monteiro. A projected gradient algorithm for solving the maxcut SDP relaxation. *Optimization Methods and Software*, 15: 175-200, 2001.

[5] E. Cohen, A. Fiat, and H. Kaplan. Efficient sequences of trails. In *Proceedings of SODA*, pp. 737–746, 2003.

[6] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4): 634–652, 1998.

[7] U. Feige, M. Halldorsson, G. Kortsarz, and A. Srinivasan. Approximating the domatic number. Preliminary version in STOC 2000.

[8] U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998.

[9] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms*, 41: 174–211, 2001.

[10] U. Feige, L. Lovasz, and P. Tetali. Approximating min sum set cover. In *Proceedings of APPROX*, pp. 94–107, 2002.

[11] E. Halperin and U. Zwick. A unified framework for obtaining improved approximation algorithms for maximum graph bisection problems. In *Proceedings of IPCO*, pp. 210–225, 2001.

[12] G. Nemhauser and L. Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975.