Approximation algorithms for the test cover problem

K.M.J. De Bontridder* B.V. Halldórsson†
M.M. Halldórsson‡ C.A.J. Hurkens§ J.K. Lenstra§¶
R. Ravi|| L. Stougie§**

January 31, 2003

Abstract

In the test cover problem a set of m items is given together with a collection of subsets, called tests. A smallest subcollection of tests is to be selected such that for each pair of items there is a test in the selection that contains exactly one of the two items. It is known that the problem is NP-hard and that the greedy algorithm has a performance ratio $O(\log m)$. We observe that, unless P = NP, no polynomial-time algorithm can do essentially better. For the case that each test contains at most k items, we give an $O(\log k)$-approximation algorithm.

We pay special attention to the case that each test contains at most two items. A strong relation with a problem of packing paths in a graph is established, which implies that even this special case is NP-hard. We prove APX-hardness of both problems, derive performance guarantees for greedy algorithms, and discuss the performance of a series of local improvement heuristics.

*Siemens VDO Automotive, Eindhoven, The Netherlands; koen.debontridder@siemens.com. Partially supported by the Future and Emerging Technologies Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

†Celera Genomics, Rockville, MD, U.S.A.; bjarni.halldorsson@celera.com. Partially supported by a Merck Computational Biology and Chemistry Program Graduate Fellowship from the Merck Company Foundation.

‡Department of Computer Science, University of Iceland, Iceland; mmh@hi.is. Also Iceland Genomics Corporation, mmh@uvs.is.

§Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, The Netherlands; {wscor,jkl,leen}@win.tue.nl.

¶School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, U.S.A.; jkl@isye.gatech.edu.

||Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA, U.S.A.; ravi@cmu.edu. Partially supported by subcontract No. 16082-RFP-00-2C in the area of "Combinatorial Optimization in Biology (XAXE)," Los Alamos National Laboratories, and NSF grant CCR-0105548.

**CWI, Amsterdam, The Netherlands; stougie@cwi.nl.

1 Introduction

The input of the *test cover problem* (TCP) consists of a set of *items*, $\{1, \ldots, m\}$, and a collection of *tests*, $T_1, \ldots, T_n \subset \{1, \ldots, m\}$. A test T_j *covers* or differentiates the item pair $\{h, i\}$ if either $h \in T_j$ or $i \in T_j$, i.e., if $|T_j \cap \{h, i\}| = 1$. A subcollection $\mathcal{T} \subset \{T_1, \ldots, T_n\}$ of tests is a *test cover* if each of the $m(m-1)/2$ item pairs is covered by at least one test in $\mathcal{T}$. The objective is to find a test cover of minimum cardinality, if one exists.

The test cover problem arises naturally in identification problems. Given a set of individuals and a set of binary attributes that may or may not occur in each individual, the goal is to find a minimum-cardinality subset of attributes – an optimal test cover – that identifies each individual uniquely. That is, the incidence vector of each individual with the test cover is a unique binary signature, distinguishing him or her from any other individual. The problem is also known in the literature as the minimum test collection problem [10] [4] and minimum test set problem [15] [4]. It arises commonly in fault testing and diagnosis, pattern recognition, and biological identification [15].

This paper is the work of two independent groups of researchers. The first group was motivated, over twenty years ago, by a request from the Agricultural University in Wageningen, the Netherlands, concerning the identification of potato diseases [13]. Each potato variety is vulnerable to a number of diseases. In order to diagnose diseases efficiently, one wished to have a minimum selection of varieties that discriminates between all diseases. This application involved 28 diseases (items) and 63 varieties (tests).

The problem came to the attention of the second group of researchers in a project on protein identification by epitope recognition [6]. It proposed a new approach of using a set of antibodies that recognize and bind specifically to short peptide sequences, called epitopes. Such an epitope can distinguish proteins that contain it from those that do not. The epitopes are fluorescently tagged, so that the binding of antibodies to an unidentified protein can be detected. Thus the output is a binary vector of dimension equal to the number of antibodies, indicating to which of the antibodies the protein is bound. The idea is to generate a set of antibodies with three properties: they recognize epitopes that are shared by many proteins, the epitopes together cover all possible proteins in the organism's proteome, and each protein is recognized by a unique subset of antibodies. This leads to a test cover problem, with proteins as items and antibodies as tests. The cited application involved about 6,000 proteins. The eventual goal is to handle much larger catalogues and, in particular, the human organism, which has between 40,000 and 100,000 proteins.

Both problems were successfully attacked by a combination of greedy and local improvement algorithms. For the Dutch problem, optimality of the resulting solution was proved by a simple branch-and-bound algorithm, using a lower bound based on the observation that, for distinguishing m items, one needs at least $\lceil \log_2 m \rceil$ tests, and a branching scheme preferring tests of size close to $m/2$ to smaller or larger ones. This work inspired research into the performance of greedy and local improvement algorithms for the problem and into its complexity and approximability. After two earlier reports [5] [12], the present paper gives a joint account of our research. A complementary paper [2] discusses optimization algorithms for the test cover problem.

The TCP is NP-hard in the strong sense [4]. Moret & Shapiro [15] established a strong relation between the TCP and the well-known set covering problem, and used it to prove that the greedy algorithm for the TCP has a worst-case performance ratio to the optimum of $\Theta(\log m)$. In Section 2 we recall these results, and we show that no polynomial-time algorithm for the TCP is likely to have a lower-order performance ratio.

In Section 3 we consider the case that each test contains at most k items, where k is part of the input. This is a common restriction for the TCP. For the above protein identification problem the novelty of the approach is the utilization of antibodies that bind to many proteins. However, most known antibodies bind specifically to protein fragments, which justifies interest in the TCP with small tests. We give an $O(\log k)$-approximation algorithm for the TCP with no more than k items per test.

In Section 4 we turn to the special case that each test contains at most two items, denoted by TCP2. We formulate it as an optimization problem on a graph and derive a performance ratio of 11/8 for the natural greedy algorithm; the proof is given in Appendix A. We then relate the TCP2 to the problem of packing paths of length 2 in a graph, which implies its NP-hardness. (The TCP2 has been stated to be solvable in polynomial time [4], a claim that was withdrawn due to our work [9].) The relation between the two problems carries over to approximation bounds. In fact, the greedy algorithm for the path packing problem gives an algorithm for the TCP2 with performance ratio 4/3, which is better than 11/8. We prove that both problems are APX-hard and hence do not have a polynomial-time approximation scheme unless P = NP.

Finally, in Section 5 we present a series of local improvement heuristics for the path packing problem and the TCP2. Each next heuristic in the series searches over a larger neighborhood. An analysis of these heuristics is given in a companion paper [1], which adds to the growing body of literature on performance guarantees for local search.

2 The general TCP

The TCP has a natural reformulation as a *cut covering problem* on a complete graph. Items correspond to vertices and item pairs to edges. Each test defines a cut, consisting of the item pairs covered by the test. The objective is to find a minimum-size subcollection of those cuts whose union is the complete edge set. The cut covering problem can in turn be formulated as a *set covering problem* (SCP). In the SCP, given a set of M elements and a collection of N subsets, one wishes to find a minimum-size subcollection of subsets whose union is the entire set. Obviously, edges correspond to elements and cuts to subsets. Starting with a TCP instance with m items and n tests, one obtains an equivalent SCP instance with $M = m(m-1)/2$ elements and $N = n$ subsets.

As a consequence, algorithms for the SCP also apply to the TCP. The greedy algorithm for the SCP, which iteratively selects a subset covering the largest number of yet uncovered elements, has a performance ratio $1 + \ln M$ [8] [14]. It directly gives a greedy algorithm for the TCP, always choosing a test covering the largest number of uncovered pairs, with performance ratio $1+2\ln m$ [15] [10].

Moret & Shapiro [15] showed, conversely, how to reduce the SCP to the TCP. They observe that this alternative strong NP-hardness proof precludes the existence of a fully polynomial-time approximation scheme, unless P = NP, and also use the reduction to show that the performance ratio of the greedy algorithm is tight up to a constant factor. We repeat their reduction here.

Consider an SCP instance with elements $\{1, \ldots, M\}$ and subsets $S_1, \ldots, S_N$. Construct a TCP instance with $m = 2M$ items and $N + \lceil \log_2 M \rceil$ tests, as follows. For each element i create a female item $\mathbf{f}_i$ and a male item $\mathbf{m}_i$. For each subset S_j define a test $T_j = \{\mathbf{f}_i : i \in S_j\}$. In addition, introduce a minimum-size collection $\mathcal{M}$ of tests that covers all pairs of male items; note that $\lceil \log_2 M \rceil$ tests are necessary and sufficient for this purpose. Finally, if a test in $\mathcal{M}$ contains an item $\mathbf{m}_i$, put its partner $\mathbf{f}_i$ in the test as well. See Figure 1.

We claim that there is a set cover of size at most σ if and only if there is a test cover of size at most $\sigma + \lceil \log_2 M \rceil$. Any test cover must include $\mathcal{M}$, as there is no other way to cover the male pairs. $\mathcal{M}$ also covers the female pairs and the mixed pairs with nonequal index values. Any other tests are of type T_j and only serve to cover pairs of type $(\mathbf{f}_i, \mathbf{m}_i)$. Since the tests T_j only contain female items, a collection of such tests covers all pairs $(\mathbf{f}_i, \mathbf{m}_i)$ $(i = 1, \ldots, M)$ if and only if the corresponding subsets form a set cover. That is, $\mathcal{S}$ is a set cover if and only if $\mathcal{M} \cup \{T_j | S_j \in \mathcal{S}\}$ is a test cover.

This argument not only shows that the TCP is NP-hard. Also inapproximability results for the SCP carry over to the TCP. However, if we apply the above reduction to the class of bad SCP instances due to Johnson [8], on which the greedy algorithm achieves a logarithmic performance ratio, then we obtain a

$$
\begin{array}{c|ccc|}
 & S_1 & S_2 & S_3 \\
\hline
1 & 1 & 0 & 0 \\
2 & 0 & 1 & 0 \\
3 & 1 & 1 & 1 \\
4 & 1 & 0 & 1 \\
\hline
\end{array}
\quad \Longrightarrow \quad
\begin{array}{c|cc|ccc|}
 & \multicolumn{2}{c}{\mathcal{M}} & T_1 & T_2 & T_3 \\
\hline
\mathbf{f}_1 & 1 & 1 & 1 & 0 & 0 \\
\mathbf{f}_2 & 1 & 0 & 0 & 1 & 0 \\
\mathbf{f}_3 & 0 & 1 & 1 & 1 & 1 \\
\mathbf{f}_4 & 0 & 0 & 1 & 0 & 1 \\
\hline
\mathbf{m}_1 & 1 & 1 & 0 & 0 & 0 \\
\mathbf{m}_2 & 1 & 0 & 0 & 0 & 0 \\
\mathbf{m}_3 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{m}_4 & 0 & 0 & 0 & 0 & 0 \\
\hline
\end{array}
$$

Figure 1: Reduction from SCP to TCP

class of TCP instances on which the greedy solution is within a constant factor of the optimum, due to the presence of the tests in $\mathcal{M}$. Following Moret & Shapiro [15], given an SCP instance with M elements and N subsets, we make $k = \lceil \log_2^2 M \rceil$ disjoint copies of it so as to obtain a multiplied SCP instance with kM elements and kN subsets. We then construct a TCP instance with $(k+1)M$ items and $kN + \lceil \log_2 M \rceil$ tests, with kM female items corresponding to the elements, M additional male items, kN tests corresponding to the subsets, and $\lceil \log_2 M \rceil$ "even splitting" tests. The original SCP instance has a solution of size at most σ if and only if the multiplied instance has a solution of size at most $k\sigma$, and hence if and only if the TCP instance has a solution of size at most $k\sigma + \lceil \log_2 M \rceil \leq k\sigma(1+O(1/\log M))$.

Now, if we were able to approximate the TCP optimum within a factor of ρ, then we could apply our method to the instance constructed above, divide the result by $\lceil \log_2^2 M \rceil$, and obtain an algorithm for the SCP with performance ratio $\rho(1 + O(1/\log M))$. We cite two inapproximability results for the SCP: No polynomial-time algorithm can have a performance ratio $o(\log M)$ unless $\mathrm{P} = \mathrm{NP}$ [17]. And no such algorithm can have a performance ratio $(1-\epsilon)\ln M$, for any $\epsilon > 0$, unless $\mathrm{NP} \subset \mathrm{DTIME}(M^{\log\log M})$ [3].

Theorem 2.1 *The* TCP *has no polynomial-time algorithm with performance bound* $o(\log m)$, *unless* $\mathrm{P} = \mathrm{NP}$, *and no polynomial-time algorithm with performance bound* $(1-\epsilon)\ln m$, *for any* $\epsilon > 0$, *unless* $\mathrm{NP} \subset \mathit{DTIME}(m^{\log\log m})$.

3 The TCP with tests of size at most k

We now consider the TCP in which each test contains at most k items, denoted by TCPk. We propose an algorithm with performance ratio $O(\log k)$.

First note that a partial test cover defines an equivalence relation on the set of items, where two items are equivalent if there is no test in the partial cover that differentiates them. The equivalence classes are the subsets of pairwise equivalent items.

Our *two-phase greedy* algorithm proceeds as follows. In phase 1, given a TCP instance, view it as an SCP instance with items as elements and tests as subsets, and apply the greedy algorithm for the SCP to find a set cover $\mathcal{S}^G$. If $\mathcal{S}^G$ is a test cover, then stop. Otherwise, in phase 2 apply the greedy algorithm for the TCP to extend the partial test cover $\mathcal{S}^G$ to a complete test cover.

Let σ^* and τ^* denote the size of an optimum set cover and an optimum test cover for the item set, respectively. The greedy set cover $\mathcal{S}^G$ found in phase 1 has size $\sigma^G \leq (1+\ln k)\sigma^*$ [8] [14]. Since any test cover is a set cover of all but at most one of the items, we have $\sigma^* \leq \tau^*+1$ and hence $\sigma^G = O(\log k)\tau^*$.

At the start of phase 2, each equivalence class contains at most k items, because each item is in some test of $\mathcal{S}^G$ and thereby differentiated from at least $m-k$ other items. It follows that any test covers at most $k(k-1)$ more item pairs, so that the greedy test cover found in phase 2 has size $\tau^G \leq (1+\ln(k(k-1)))\tau^*$ [8] [14]. The overall test cover has size $\sigma^G + \tau^G = O(\log k)\tau^*$.

Theorem 3.1 *The two-phase greedy algorithm for* TCPk *has a performance ratio* $O(\log k)$.

4 The TCP with tests of size at most 2

4.1 A problem on graphs

The rest of this paper is concerned with the special case that each test contains at most two items, denoted by TCP2. We first argue that we may assume that each test contains exactly two items.

Lemma 4.1 *Any instance of the* TCP *with tests of size at most* 2 *can be transformed into an instance of the* TCP *with tests of size exactly* 2.

PROOF. Let $\mathbf{T} = \{T_1, \ldots, T_n\}$, and let $\mathcal{T} \subset \mathbf{T}$ be a minimum test cover. Suppose that we have u items not contained in any test in $\mathbf{T}$ with $u \in \{0, 1\}$, v items $g_1, \ldots, g_v$ with g_t only contained in the test $\{g_t\} \in \mathbf{T}$, for $t = 1, \ldots, v$, and w item pairs $\{h_1, i_1\}, \ldots, \{h_w, i_w\}$ with the property that, for $t = 1, \ldots, w$, $\{h_t, i_t\} \in \mathbf{T}$, $\{h_t\} \in \mathbf{T}$, possibly $\{i_t\} \in \mathbf{T}$, and no other test contains h_t or i_t. If $u+v+w > 0$, then $\mathcal{T}$ contains, without loss of generality, the first $u + v + 2w - 1$ tests from $\{g_1\}, \ldots, \{g_v\}, \{h_1\}, \{h_1, i_1\}, \ldots, \{h_w\}, \{h_w, i_w\}$, leaving one item isolated.

Each item h not among those $u + v + 2w$ ones has the properties that (a) there exists an item i such that $\{h, i\} \in \mathbf{T}$, and (b) for all such $\{h, i\}$ there exists an $\{h', i'\} \in \mathbf{T}$ such that $|\{h, i\} \cap \{h', i'\}| = 1$.

We may assume without loss of generality that $\mathcal{T}$ does not contain singleton tests except the ones mentioned above. For suppose $\mathcal{T}$ contains another singleton test $\{h\}$. As $\mathcal{T}$ is minimum, it does not contain two tests $\{h, i\}$ and $\{h, i'\}$. If $\mathcal{T}$ contains no test $\{h, \cdot\}$, replace $\{h\}$ by any test $\{h, i\} \in \mathbf{T}$, which exists by (a). If by this action h and i become indistinguishable (i was apparently left isolated), or if $\mathcal{T}$ already contains a test $\{h, i\}$, replace $\{h\}$ by the corresponding test $\{h', i'\} \in \mathbf{T}$, see (b).

By eliminating all $u + v + 2w$ items involved, the tests that contain them, and all other singleton tests, and adding one isolated item if $u + v + w > 0$, we obtain an equivalent instance of the TCP2 with tests of size 2 only. $\square$

From now on we will restrict our attention to the TCP2 with tests of size exactly 2. This TCP2 can be formulated as an optimization problem on a graph, in which the m items correspond to vertices and the n tests to edges. We obtain the following characterization of test covers.

Lemma 4.2 *In a graph $G = (V, E)$, a subset $E' \subset E$ is a test cover if and only if the graph $G' = (V, E')$ has no isolated edges and at most one isolated vertex.*

PROOF. If E' is a test cover, then $G' = (V, E')$ has at most one isolated vertex (an item with an all-zero signature) and no isolated edges (since otherwise its vertices would not be differentiated). Conversely, a graph with these properties satisfies the condition that, for any two vertices, there is an edge incident to exactly one of them. □

Note that this lemma also characterizes feasible instances of the TCP2. We will assume from now on that the instances that we consider are feasible.

A test cover is *minimal* if no edge can be deleted from it without causing infeasibility. In addition to having the properties stated in Lemma 4.2, a minimal test cover is obviously acyclic. This implies the following.

Lemma 4.3 *In a graph $G = (V, E)$, if $E' \subset E$ is a minimal test cover, then at most one of the components of $G' = (V, E')$ is an isolated vertex and each other component is a tree of at least two edges.*

The greedy algorithm for the TCP2 iteratively selects an edge that covers the largest number of yet uncovered vertex pairs. In Appendix A we prove the following performance bound for the greedy algorithm.

Theorem 4.1 *The greedy algorithm for the* TCP2 *has performance ratio* 11/8. *This bound is asymptotically tight.*

4.2 Packing paths of length 2

We will now examine the relation of the TCP2 to another optimization problem on a graph. In the *problem of packing paths of length* 2 (PPP2), we are given a graph on m vertices, and we wish to find a maximum number of vertex-disjoint paths of length 2, leaving at least one vertex isolated. We will often use the term *path packing* to indicate a feasible solution to the PPP2. Since the problem of partitioning a graph into paths of length 2 is NP-complete [11] [4], the PPP2 is NP-hard.

The seemingly artificial condition that any solution to the PPP2 has *at least* one isolated vertex is matched by the property that any solution to the TCP2 has *at most* one isolated vertex. It is introduced for the sake of a duality relation between the PPP2 and the TCP2, as elaborated below.

Given a test cover, we can easily find a path packing.

Lemma 4.4 *If a graph $G = (V, E)$ has a minimal test cover of size τ, then it has a path packing of size $\pi = m - 1 - \tau$.*

PROOF. Let $E' \subset E$ be the minimal test cover. Suppose that the graph $G' = (V, E')$ has k components. By Lemma 4.3, G' is a forest, and hence $\tau = |E'| = m - k$. By the same lemma, we can select a path of length 2 from each but one of the components, and obtain a path packing of size $\pi = k - 1 = m - 1 - \tau$. □

A converse relation holds as well. A path packing is *maximal* if no path can be added to it.

Lemma 4.5 *If a graph $G = (V, E)$ has a maximal path packing of size π, then it has a test cover of size $\tau = m - 1 - \pi$.*

Proof. The graph induced by the path packing contains $m - 3\pi$ isolated vertices. We distinguish two cases.

(1) The path packing has a path in each component of G. We extend it to a test cover by successively connecting all but one of the isolated vertices to one of the paths, and obtain a test cover of size $\tau = 2\pi + m - 3\pi - 1 = m - 1 - \pi$.

(2) The path packing has a path in each but one component of G. (Since G is feasible, the component without a path has one or three vertices.) We extend the path packing to a test cover by spanning a tree in the component without a path and connecting each of the remaining isolated vertices to one of the paths, and thus obtain a test cover of size $\tau = 2\pi + m - 3\pi - 1 = m - 1 - \pi$. □

Given any algorithm that produces a maximal path packing, its *extension to the* TCP2 constructs a test cover by the procedure in the above proof.

Lemmas 4.4 and 4.5 together imply a relation between optimal solution values to the TCP2 and the PPP2, and also allow us to relate the performance of approximation algorithms.

Theorem 4.2 *In a graph $G = (V, E)$, the size π^* of a maximum path packing and the size τ^* of a minimum test cover satisfy $\pi^* + \tau^* = m - 1$.*

Since the PPP2 is NP-hard, it follows that the TCP2 is NP-hard too.

Theorem 4.3 *If the* PPP2 *has an algorithm with performance ratio ρ, then the* TCP2 *has an algorithm with performance ratio $3/2 - \rho/2$.*

Proof. Suppose algorithm A for the PPP2 satisfies $\pi^A \geq \rho\pi^*$. Consider its extension A' to the TCP2. We know that $\tau^{A'} + \pi^A = m - 1 = \tau^* + \pi^*$. Hence, $\tau^{A'} = \tau^* + \pi^* - \pi^A \leq \tau^* + (1 - \rho)\pi^*$. Since $3\pi^* \leq m - 1 = \pi^* + \tau^*$, we have $\pi^* \leq \tau^*/2$ and thereby $\tau^{A'} \leq \tau^* + (1 - \rho)\tau^*/2 = (3/2 - \rho/2)\tau^*$. □

The greedy algorithm for the PPP2 iteratively selects a path of length 2 from the graph and deletes its vertices and adjacent edges. When the graph contains no path of length 2 or when it has at most three vertices, the algorithm has obtained a maximal path packing and terminates. A bad example is given by the graph in Figure 2. The greedy algorithm may select only one path of length 2, whereas three is optimal. We show that this is the worst case.

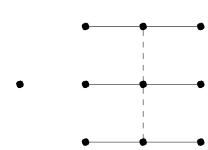

Figure 2: Worst-case instance for the greedy algorithm for the PPP2

Theorem 4.4 *The greedy algorithm for the* PPP2 *has performance ratio* 1/3. *Its extension to the* TCP2 *has performance ratio* 4/3. *These bounds are tight.*

Proof. Any path of length 2 in the greedy solution intersects at most three paths of length 2 in the optimal solution. Since the greedy solution is maximal, either each path in the optimal solution intersects a greedy path, which implies the desired performance bound, or the greedy solution leaves exactly three vertices isolated that form a path of length 2, in which case the greedy solution is optimal. Theorem 4.3 implies the bound for the extension to the TCP2. □

Theorems 4.1 and 4.4 tell us that, for the TCP2, picking paths of length 2 at random gives a better guarantee than choosing most distinctive single edges.

4.3 APX-hardness

We will show that the PPP2 and thereby also the TCP2 is APX-hard. Our result will follow through a reduction from 3-*dimensional matching with at most three occurrences per element* (3DM3): Given disjont sets X, Y, Z containing s elements each, and a set C of t triples in $X \times Y \times Z$, such that each element of $X \cup Y \cup Z$ occurs in at most three triples of C, find a maximum-cardinality matching $C' \subset C$, i.e., a subset of triples such that no element of $X \cup Y \cup Z$ occurs in more than one triple. For 3DM3, it is NP-hard to decide whether a maximum matching is perfect or misses a constant fraction of the elements [16].

Lemma 4.6 *There exists a constant* $\epsilon > 0$ *such that it is* NP-*hard to determine whether an instance of the* PPP2 *has a path packing of size* $(m-1)/3$ *or of size at most* $(1-\epsilon)(m-1)/3$.

Proof. Given an instance of 3DM3, we create a graph G with $m = 6s + 3t + 1$ vertices

- $\bar{x}_g, x_g$ for each $x_g \in X$, $\bar{y}_h, y_h$ for each $y_h \in Y$, $\bar{z}_i, z_i$ for each $z_i \in Z$,
- c_j^x, c_j^y, c_j^z for each $c_j \in C$,
- w, a vertex that will remain isolated,

and $n = 3s + 5t$ edges

- $\{\bar{x}_g, x_g\}$ for each $x_g \in X$, $\{\bar{y}_h, y_h\}$ for each $y_h \in Y$, $\{\bar{z}_i, z_i\}$ for each $z_i \in Z$,
- $\{x_g, c_j^x\}, \{y_h, c_j^y\}, \{z_i, c_j^z\}$ for each triple $c_j = \{x_g, y_h, z_i\} \in C$,
- $\{c_j^x, c_j^y\}, \{c_j^y, c_j^z\}$ for each $c_j \in C$.

We claim that G contains $2s + t$ vertex-disjoint paths of length 2 if and only if there exists a matching of size s. The reduction is illustrated in Figure 3.

If the instance of 3DM3 has a matching C' of size s, then G contains paths $(\bar{x}_g, x_g, c_j^x)$, $(\bar{y}_h, y_h, c_j^y)$, $(\bar{z}_i, z_i, c_j^z)$ for each triple $c_j = \{x_g, y_h, z_i\} \in C'$ and a path (c_j^x, c_j^y, c_j^z) for each triple $c_j \in C \backslash C'$, giving a total number of $3s + (t - s) = 2s + t$ paths.

Now, let a maximum matching consist of μ^* triples, and let an optimal path packing $\mathcal{P}$ consist of π^* paths. $\mathcal{P}$ contains *element paths* of type $(\bar{\gamma}, \gamma, c^\gamma)$ and

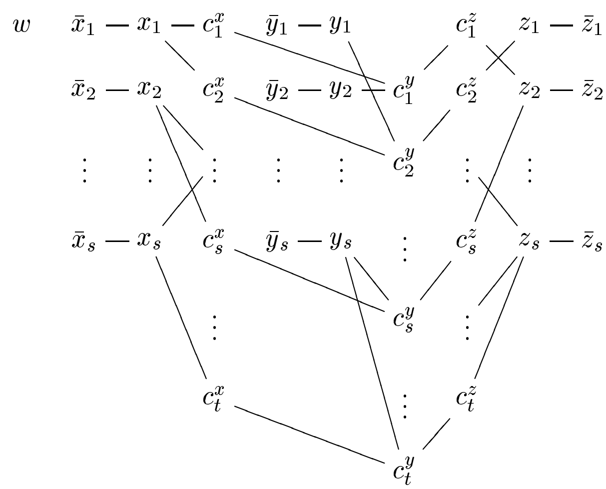

Figure 3: Reduction of 3DM3 to PPP2

triple paths of type (c_j^x, c_j^y, c_j^z); it is easy to see that other types of paths in any path packing can be replaced by element paths. We will bound π^* in terms of μ^*. Let t_0, t_1, t_2, t_3 be the number of triples in C intersecting $0, 1, 2, 3$ element paths in $\mathcal{P}$, respectively. Then,

$$\pi^* \leq t_0 + t_1 + 2t_2 + 3t_3 = t + t_2 + 2t_3 = t + \frac{1}{2}(2t_2 + 3t_3) + \frac{1}{2}t_3 \leq t + \frac{3}{2}s + \frac{1}{2}\mu^*.$$

The first equality holds because $t = t_0 + t_1 + t_2 + t_3$. The second inequality follows from $t_1 + 2t_2 + 3t_3 \leq 3s$ ($\mathcal{P}$ contains at most $3s$ element paths) and $t_3 \leq \mu^*$. Hence, if $\pi^* = 2s + t$, then $\mu^* = s$.

Let $\epsilon' > 0$ be such that it is NP-hard to decide whether $\mu^* = s$ or $\mu^* \leq (1 - \epsilon')s$. Hence, it is NP-hard to decide whether $\pi^* = 2s + t = (m-1)/3$ or $\pi^* \leq 2s + t - \epsilon' s/2 = (1-\epsilon)(m-1)/3$, if we choose $\epsilon = \epsilon' s/(4s + 2t)$. For 3DM3, we have $t \leq 3s$, so that $\epsilon \geq \epsilon'/10$. This completes the proof. $\square$

Lemma 4.6 and Theorem 4.2 imply the following.

Theorem 4.5 *The* PPP2 *and the* TCP2 *are both* APX*-hard.*

5 Local improvement for PPP2 and TCP2

In this final section we propose a series of local improvement algorithms for the PPP2. Each next algorithm in the series starts from a maximal path packing, searches over a larger neighborhood, and requires more time. Its extension to the

TCP2, as described in Section 4.2, transforms the locally optimal path packing into a test cover.

The basic heuristic, denoted H_0, applies the greedy algorithm to obtain a maximal path packing. For $k \geq 1$, the kth heuristic in the series, denoted H_k, starts from a maximal path packing, and attempts to improve it by replacing any k paths of length 2 by $k+1$ paths of length 2. This involves a complete search over all sets of k paths and, for each such set, over all possibilities for improvement. When no further improvements are found, H_k terminates. For fixed k, H_k runs in polynomial time, but the running time of H_k is not known to be polynomial in k.

Let ρ_k be the performance ratio of heuristic H_k, for $k \geq 0$. Obviously, ρ_k is nondecreasing in k. Theorem 4.4 states that $\rho_0 = 1/3$. Here we will discuss ρ_1, ρ_2, ρ_3, and ρ_4.

Theorem 5.1 *The local improvement algorithms H_1, H_2, H_3, and H_4 for the* PPP2 *have performance ratios $\rho_1 = 1/2$, $\rho_2 = 5/9$, $\rho_3 = 7/11$, and $\rho_4 = 2/3$. These bounds are tight.*

Hurkens & Schrijver [7] consider a series of analogous local improvement algorithms for the more general problem of packing vertex-disjoint subgraphs on t vertices in a given graph. Their work was, in fact, inspired by questions about the performance of our heuristics H_k. They derive a lower bound ϕ_k on the performance ratio of their kth heuristic, and prove that it is tight if the subgraph is a clique. In particular, for $t = 3$,

$$\phi_k = \begin{cases} \frac{2\cdot 2^{(k+2)/2}-3}{3\cdot 2^{(k+2)/2}-3} & \text{if } k \text{ is even,} \\ \frac{2\cdot 2^{(k+1)/2}-2}{3\cdot 2^{(k+1)/2}-2} & \text{if } k \text{ is odd.} \end{cases}$$

Since a path of length 2 is a subgraph on three vertices, we know that $\rho_k \geq \phi_k$.

Table 1 lists the values of ϕ_k ($k \geq 0$) for the problem of packing triangles, ρ_k ($k = 0, \ldots, 4$) for the PPP2, and the corresponding ratios for the TCP2 that are implied by Theorem 4.3. Note that $\rho_4 = \lim_{k\to\infty} \phi_k$. The asymptotic value $\lim_{k\to\infty} \rho_k$ remains open, but it is likely to be strictly smaller than 1, in view of Theorem 4.5.

Instances for which H_1, H_2, H_3, and H_4 meet their claimed performance ratios are given in Figures 4, 5, 6, and 7, respectively. In each case the dashed

problem	k	0	1	2	3	4	5	6	7	8	$\cdots$	∞
triangle packing	ϕ_k	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{5}{9}$	$\frac{3}{5}$	$\frac{13}{21}$	$\frac{14}{22}$	$\frac{29}{45}$	$\frac{30}{46}$	$\frac{61}{93}$	$\cdots$	$\frac{2}{3}$
PPP2	ρ_k	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{5}{9}$	$\frac{7}{11}$	$\frac{2}{3}$						
TCP2	$\frac{3}{2} - \frac{\rho_k}{2}$	$\frac{4}{3}$	$\frac{5}{4}$	$\frac{11}{9}$	$\frac{13}{11}$	$\frac{7}{6}$						

Table 1: Performance ratios for local improvement heuristics

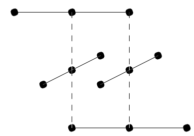

Figure 4: Worst-case instance for H_1

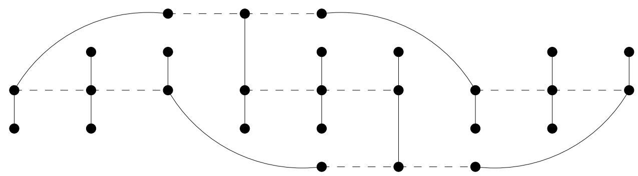

Figure 5: Worst-case instance for H_2

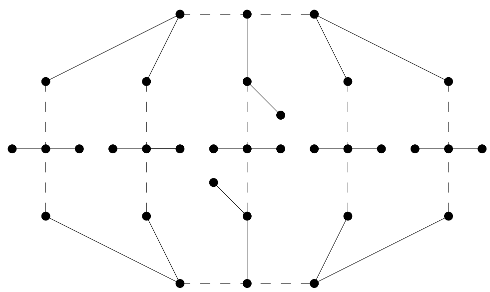

Figure 6: Worst-case instance for H_3

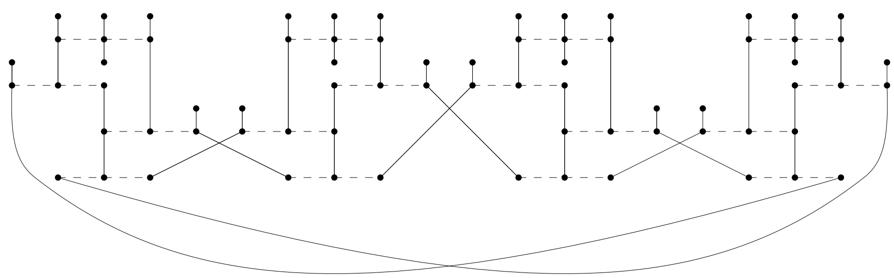

Figure 7: Worst-case instance for H_4

lines indicate a locally optimal path packing, and the solid lines indicate a larger packing. Note that we have omitted the mandatory isolated vertex and that here, as well as in Figure 2, we can provide an infinite family of worst-case instances by creating multiple copies of the graph.

The upper bounds on ρ_k provided by these examples match the lower bounds ϕ_k for $k = 1$ and $k = 2$, which proves part of Theorem 5.1. The proof for $k = 3$ and $k = 4$ is more involved. We outline the general idea here, and refer to a companion paper [1] for details. The argument may be extended to handle H_5 and H_6, but we have not attempted to do so.

Our approach to obtain lower bounds on ρ_k is based on linear programming. Consider a graph G with a locally optimal path packing $\mathcal{P}$ found by H_k and any other path packing $\mathcal{Q}$. In order to show that $|\mathcal{P}|/|\mathcal{Q}| \geq \rho_k$, we may make the following assumptions:

– G does not contain other edges than those appearing in $\mathcal{P}$ and $\mathcal{Q}$;
– $|\mathcal{P}| < |\mathcal{Q}|$;
– each path in $\mathcal{P}$ intersects at least one path in $\mathcal{Q}$;
– each path in $\mathcal{Q}$ intersects at least one path in $\mathcal{P}$;
– no set of three vertices is covered by a $\mathcal{P}$-path and by a $\mathcal{Q}$-path;
– each middle vertex of a $\mathcal{P}$-path is covered by some $\mathcal{Q}$-path.

For every vertex that is both on a $\mathcal{P}$-path and on a $\mathcal{Q}$-path, we define a label, which expresses the interaction of its $\mathcal{Q}$-path with the $\mathcal{P}$-paths. Based on this labeling we distinguish several types of $\mathcal{P}$-paths. This leads to eight vertex labels and 96 path types, 40 of which can be excluded due to the above assumptions. For each remaining path type we introduce a variable, denoting the fraction of $\mathcal{P}$-paths of that type in $\mathcal{P}$. The variables add up to 1. Furthermore, the ratio $|\mathcal{Q}|/|\mathcal{P}|$ can be written as a linear combination of these variables.

By carefully analyzing configurations that can or cannot be improved by H_k, we are able to formulate restrictions on certain combinations of the variables. For instance, consider a $\mathcal{Q}$-path that intersects exactly one $\mathcal{P}$-path, in exactly one vertex. Such a vertex is labeled 1. It is immediate from the definition of H_1 that no path in $\mathcal{P}$ contains two or three vertices labeled 1. This observation sets sixteen variables to 0.

When describing the conditions corresponding to configurations that are not improved by H_1, H_2, H_3, or H_4, we end up with three, five, eight, or ten linear constraints, respectively. Maximizing the ratio under these constraints proves Theorem 5.1, and yields fractions that are in agreement with the instances given in Figures 4, 5, 6, and 7.

Acknowledgment

We are grateful to the referees, whose comments helped us to improve the paper.

References

[1] K.M.J. De Bontridder, B.V. Halldórsson, M.M. Halldórsson, C.A.J. Hurkens, J.K. Lenstra, R. Ravi, L. Stougie (2003). Local improvement algorithms for a path packing problem: a performance analysis based on linear programming. In preparation.

[2] K.M.J. De Bontridder, B.J. Lageweg, J.K. Lenstra, J.B. Orlin, L. Stougie (2002). Branch-and-bound algorithms for the test cover problem. R.H.

Möhring, R. Raman (eds.). *Algorithms—ESA 2002*, LNCS, Springer, Berlin, 223–233.

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

[4] M.R. Garey, D.S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco.

[5] B.V. Halldórsson, M.M. Halldórsson, R. Ravi (2001). On the approximability of the minimum test collection problem. F. Meyer auf der Heide (ed.). *Algorithms—ESA 2001*, LNCS 2161, Springer, Berlin, 158–169.

[6] B.V. Halldórsson, J.S. Minden, R. Ravi (2001). PIER: Protein identification by epitope recognition. N. El-Mabrouk, T. Lengauer, D. Sankoff (eds.). *Currents in Computational Molecular Biology 2001*, 109–110.

[7] C.A.J. Hurkens, A. Schrijver (1989). On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics 2*, 68–72.

[8] D.S. Johnson (1972). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences 9*, 256–278.

[9] D.S. Johnson (1981). The NP-completeness column: an ongoing guide. *Journal of Algorithms 4*, 393–405.

[10] V. Kann (1992). *On the approximability of NP-complete optimization problems*, PhD thesis, Royal Institute of Technology, Stockholm, Sweden.

[11] D.G. Kirkpatrick, P. Hell (1978). On the complexity of a generalized matching problem. *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, 240–245.

[12] A.W.J. Kolen, J.K. Lenstra (1995). Combinatorics in operations research. R. Graham, M. Grötschel, L. Lovász (eds.). *Handbook of Combinatorics*, Elsevier Science, Amsterdam, 1875–1910.

[13] B.J. Lageweg, J.K. Lenstra, A.H.G. Rinnooy Kan (1980). Uit de practijk van de besliskunde. A.K. Lenstra, H.W. Lenstra, J.K. Lenstra (eds.), *Tamelijk briljant; Opstellen aangeboden aan Dr. T.J. Wansbeek*, Amsterdam.

[14] L. Lovász (1975). On the ratio of optimal integral and fractional covers. *Discrete Mathematics 13*, 383–390.

[15] B.M.E. Moret, H.D. Shapiro (1985). On minimizing a set of tests. *SIAM Journal on Scientific and Statistical Computing 6*, 983–1003.

[16] E. Petrank (1994). The hardness of approximations: gap location. *Computational Complexity 4*, 133-157.

[17] R. Raz, S. Safra (1997). A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 475–484.

A Analysis of the greedy algorithm for TCP2

We consider the greedy algorithm for the TCP2 defined on a graph $G = (V, E)$ with m vertices (items) and n edges (tests). The greedy algorithm iteratively selects an edge that covers the largest number of yet uncovered vertex pairs.

To examine the options, consider a partial test cover $E' \subset E$. Let V_k denote the set of vertices that lie in a component of $G' = (V, E')$ of size k. By adding an edge connecting $h, i \in V_1$ we cover $2(|V_1|-2)$ more vertex pairs. An edge between $h \in V_1$ and $i \in V_2$ covers $|V_1|$ more vertex pairs, whereas an edge between $h \in V_1$ and $i \notin V_1 \cup V_2$ covers $|V_1| - 1$ more vertex pairs. An edge between $h, i \in V_2$ connects two isolated edges and hence covers two more vertex pairs. Finally, an edge between $h \in V_2$ and $i \notin V_1 \cup V_2$ covers one more vertex pair.

It follows that, as long as at least four vertices are isolated, the greedy algorithm will select isolated edges. In phase 1 it constructs a maximal matching, leaving at least two vertices isolated. (If it would continue adding edges to the matching until just one vertex remains isolated, then the latest edge covered two more pairs, while connecting one of the three isolated vertices to the matching would have covered three more pairs.) Let E'_1 be the set of edges in the matching.

In phase 2 the greedy algorithm selects edges that are incident to only one edge in E'_1, thus creating paths of length 2 in the graph, until this is no longer possible, or until only one vertex is left isolated. Let E'_2 be the set of edges selected in this phase. After phase 2, the graph $G_2 = (V, E'_1 \cup E'_2)$ consists of paths of length 2, isolated edges, and isolated vertices.

In phase 3 edges are selected that connect isolated vertices to a path in G_2, until at most two vertices are left isolated. Let E'_3 be the set of edges selected in this phase. The graph $G_3 = (V, E'_1 \cup E'_2 \cup E'_3)$ consists of trees on three or more vertices, isolated edges, and at most two isolated vertices.

In phase 4 edges are selected that connect two isolated edges in G_3, constituting the set E'_4. The resulting graph is G_4.

Finally, in phase 5 edges are selected that connect the remaining isolated edges and at most one isolated vertex to trees in G_4, constituting the set E'_5.

We are now ready to prove Theorem 4.1.

The edges that are isolated at the start of phase 4 were already isolated at the end of phase 2. Thus, reversing phases 3 and 4 does not change the outcome of the greedy algorithm. After phases 1, 2, and 4, the components of the graph $G'_4 = (V, E'_1 \cup E'_2 \cup E'_4)$ are paths of length 3 or 2, isolated edges, and isolated vertices. We denote their number by c_4, c_3, c_2, and c_1, respectively, where the index denotes the number of vertices in the components. In phases 3 and 5, all isolated edges and all but one of the isolated vertices in G'_4 are connected to one of the paths in G'_4. Therefore, the size of the greedy test cover is

$$\tau^G = 3c_4 + 2c_3 + c_2 + (c_2 + c_1 - 1) = 3c_4 + 2c_3 + 2c_2 + c_1 - 1. \quad (1)$$

Theorem 4.2 together with $\pi^* \leq (m-1)/3$ implies that $\tau^* \geq 2(m-1)/3$.

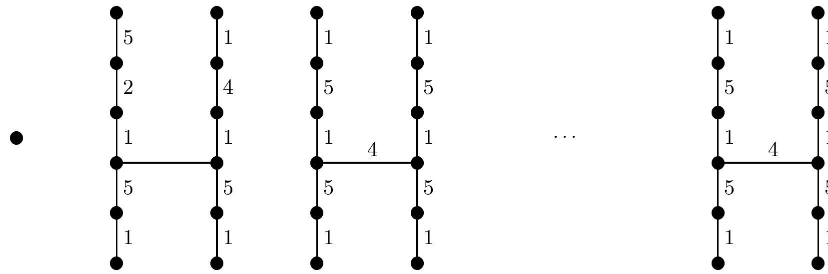

Figure 8: Worst-case instance for the greedy algorithm for the TCP2

Since $m = 4c_4 + 3c_3 + 2c_2 + c_1$, we have

$$\tau^* \geq \frac{2}{3}(4c_4 + 3c_3 + 2c_2 + c_1 - 1). \tag{2}$$

To obtain another lower bound on τ^*, we consider the graph G'_4 again. Each of its isolated edges and each of its isolated vertices except one needs an adjacent edge in any test cover. Moreover, no pair of isolated edges or vertices can be combined by an extra edge into a path of length 2 or 3, as otherwise this would have been done in phase 2 or phase 4. Hence,

$$\tau^* \geq 2c_2 + c_1 - 1. \tag{3}$$

Adding 9/8 times (2) and 2/8 times (3) and applying (1) yields

$$\frac{11}{8}\tau^* \geq 3c_4 + \frac{9}{4}c_3 + 2c_2 + c_1 - 1 \geq \tau^G.$$

To show that the ratio is asymptotically tight, consider the graph given in Figure 8. It consists of one isolated vertex and c isomorphic components on twelve vertices each. The number displayed at an edge indicates the phase in which the edge is selected by the greedy algorithm. The greedy test cover has size $\tau^G = 11c - 1$. Since each of the large components can be covered by four paths of length 2, we have $\tau^* = 8c$. Thus, $\lim_{c\to\infty} \tau^G/\tau^* = 11/8$.