# Approximation algorithms for the test cover problem

TU/e technische universiteit eindhoven

SPOR-Report 2002-10

# 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

SPOR-Report
Reports in Statistics, Probability and Operations Research

Eindhoven, August 2002
The Netherlands

/department of mathematics and computing science

# 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§**

August 8, 2002

**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 show 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, and derive performance guarantees for greedy algorithms and for a series of local improvement heuristics.

*Institute of Information and Computing Sciences, Utrecht University, The Netherlands; koendb@cs.uu.nl. 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.

‖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.

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 [11] [4] and minimum test set problem [16] [4]. It arises commonly in fault testing and diagnosis, pattern recognition, and biological identification [16].

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 [14]. 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] [13], the present paper gives a joint account of our research. A complementary paper [2] discusses optimization algorithms for the test cover problems.

The TCP is NP-hard in the strong sense [4]. Moret & Shapiro [16] 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 prove a performance ratio of 11/8 for the natural greedy algorithm. We then relate the TCP2 to the problem of packing paths of length 2 in a graph, which immediately 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. Our analysis of these heuristics adds to the growing body of literature on performance guarantees for local search.

Some of the more technical proofs are given in Appendices A and B.

# 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] [15]. It directly gives a greedy algorithm for the TCP, always choosing a test covering the largest number of uncovered pairs, with performance ratio $1+2\ln m$ [16] [11].

Moret & Shapiro [16] showed, conversely, how to reduce the SCP to the TCP. They observe that this alternative strong NP-hardness proof for the TCP 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. 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 $\sigma$ 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 unequal index values. Other tests only serve to cover pairs $(\mathbf{f}_i, \mathbf{m}_i)$. Since these tests only contain female items, a collection of such tests covers those pairs 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, if we can eliminate the influence of the term $\lceil \log_2 M \rceil$ [16]. Given an instance of the SCP, we make $k = \log_2^2 M$ disjoint copies of it and perform the above reduction to an instance of the TCP. The original SCP instance has a solution of size at most

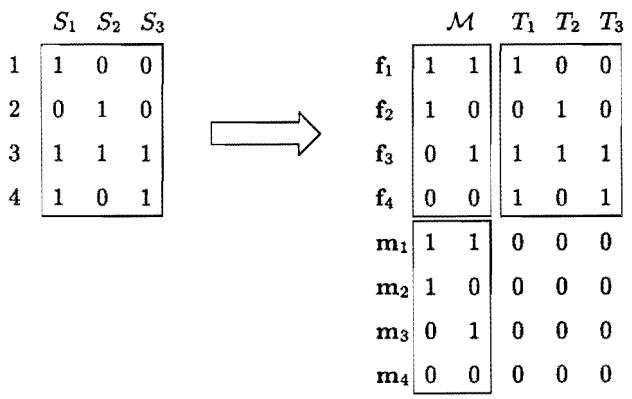


*Figure 1: Reduction from SCP to TCP*

$\sigma$ if and only if the multiplied instance has a solution of size at most $k\sigma$, and hence if and only if the resulting TCP instance has a solution of size at most $k\sigma + \lceil \log_2 kM \rceil = k\sigma(1+\delta)$, with $0 \leq \delta \leq 2/\log_2 M$. (Note that $\delta = \lceil \log_2 kM \rceil / (k\sigma) \leq \log_2 M^2/k = 2/\log_2 M$.)

Now, if we were able to approximate the TCP optimum within a factor of $\rho$, then we could apply our method to the instance constructed above, divide the result by $\log_2^2 M$, 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 P = NP [1]. And no such algorithm can have a performance ratio $(1-\epsilon)\ln M$, for any $\epsilon > 0$, unless NP $\subset$ DTIME($M^{\log\log M}$) [3].

**Theorem 2.1** *The* TCP *has no polynomial-time algorithm with performance bound* $o(\log m)$, *unless* P = NP, *and no polynomial-time algorithm with performance bound* $(1-\epsilon)\ln m$, *for any* $\epsilon > 0$, *unless* NP$\subset$*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 TCP$k$. 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 $\sigma^*$ and $\tau^*$ 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] [15]. 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 the largest set of uncovered item pairs has size at most $k(k-1)/2$, so that the greedy test cover found in phase 2 has size $\tau^G \leq (1+\ln(k(k-1)/2))\tau^*$ [8] [15]. The overall test cover has size $\sigma^G + \tau^G = O(\log k)\tau^*$.

**Theorem 3.1** *The two-phase greedy algorithm for* TCP$k$ *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. □

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.*

We state the following characterization of minimal test covers without proof.

**Lemma 4.4** *In a graph $G = (V, E)$, a subset $E' \subset E$ is a minimal test cover if and only if at most one of the components of $G' = (V, E')$ is an isolated vertex and each other component is a tree with diameter at least 2 and at most 4, containing at most one vertex with degree larger than 2.*

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. The seemingly artificial condition that a vertex must be left isolated is introduced for the sake of the relation to the TCP2. The PPP2 is NP-hard, because the problem of partitioning a graph into paths of length 2 is NP-complete [12] [4].

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

**Lemma 4.5** *If a graph $G = (V, E)$ has a minimal test cover of size $\tau$, 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 4.6** *If a graph $G = (V, E)$ has a maximal path packing of size $\pi$, 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 4.5 and 4.6 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 $\pi^*$ of a maximum path packing and the size $\tau^*$ 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 $\rho$, 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 $\pi^* \leq \tau^*/2$, we have $\tau^{A'} \leq \tau^* + (1 - \rho)\tau^*/2 = (3/2 - \rho/2)\tau^*$. □

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 no more than three vertices, it terminates with a maximal path packing. 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.

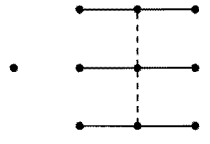

*Figure 2: Worst-case instance for the greedy algorithm for the PPP2*

**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. 3DM3 is known to be APX-hard [10].

**Lemma 4.7** *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 wil 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 $\mu^*$ triples, and let an optimal path packing $\mathcal{P}$ consist of $\pi^*$ paths. $\mathcal{P}$ contains *element paths* of type $(\bar{\gamma}, \gamma, c^\gamma)$ and

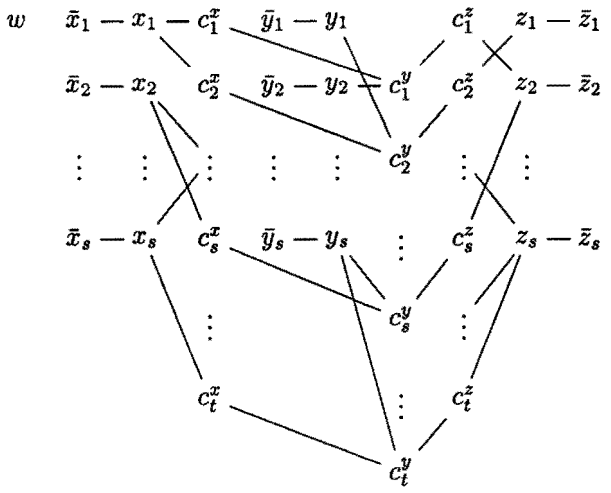


*Figure 3: Reduction of 3DM3 to PPP2*

*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 $\pi^*$ in terms of $\mu^*$. 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 $t_3 \leq \mu^*$.

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 $\epsilon = \epsilon' s/(4s + 2t)$. □

Lemma 4.7 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 $k$th 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.

The performance ratio of $H_{k+1}$ will be no worse than that of $H_k$. For fixed $k$, $H_k$ runs in polynomial time, but the running time of $H_k$ is not known to be polynomial in $k$.

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 $\phi_k$ on the performance ratio of their $k$th heuristic, and prove that it is tight if the subgraph is a clique. In particular, for $t = 3$,

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

Let $\rho_k$ be the performance ratio of heuristic $H_k$, for $k \geq 0$. Since a path of length 2 is a subgraph on three vertices, we know that $\rho_k \geq \phi_k$. Theorem 4.4 states that $\rho_0 = 1/3$. We will determine $\rho_1$, $\rho_2$, $\rho_3$, and $\rho_4$.

Table 1 lists the values of $\phi_k$ ($k \geq 0$) for the problem of packing triangles, $\rho_k$ ($k = 0, \ldots, 4$) for the PPP2, and the corresponding ratios for the TCP2 that are implied by Theorem 4.3. Note that $H_4$ has a performance ratio that can only be achieved in the limit when one has to pack triangles instead of paths. 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.

**Theorem 5.1** *The local improvement algorithms $H_1$, $H_2$, $H_3$, and $H_4$ for the* PPP2 *have the performance ratios given in Table 1. These bounds are tight.*

The proof is given in Appendix B. We give worst-case instances for $H_1$, $H_2$, $H_3$, and $H_4$ in Figures 7, 8, 10, and 13, respectively, where we have omitted the

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

mandatory isolated vertex. The performance upper bounds provided by these examples match the lower bounds $\phi_k$ for $k = 1$ and $k = 2$.

The proofs for $k = 3$ and $k = 4$ are based on linear programming formulations. Unfortunately, the difference in structure between paths of length 2 and triangles prohibits the use of the relatively clean analysis of Hurkens & Schrijver. A direct analysis as for $H_0$ in the proof of Theorem 4.4 may be extended to $H_1$, but it becomes cumbersome for $H_2$ and we do not see how to use it for $H_3$. The LP argument at least provides an analysis for $H_3$ and $H_4$. It may be extended to handle $H_5$ and $H_6$, but we have not attempted to do so.

# References


[1] S. Arora, M. Sudan (1997). Improved low degree testing and its applications. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, 485–495.

[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 (ed.). *Algorithms—ESA 2002*, LNCS, Springer, Berlin.

[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 (1991). Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters 37*, 27–35.

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

[12] 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.

[13] 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.

[14] 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.

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

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

# A Analysis of the greedy algorithm for TCP2

We consider the greedy algorithm for the TCP2 defined on the 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 there are at least four isolated vertices, the greedy algorithm will select isolated edges. In phase 1, the greedy algorithm constructs a maximal matching, provided that at least two vertices remain unmatched. 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 isolated vertices are left. 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 subgraph 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. The components of the graph $G'_4 = (V, E'_1 \cup E'_2 \cup E'_4)$ are paths of length 3 and 2, isolated edges, and isolated vertices. We denote their number by $c_4$, $c_3$, $c_2$, and $c_1$, respectively, where the index indicates the number of vertices in the components. In phase 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 resulting greedy test cover is

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

Theorem 4.2 together with $\pi^* \leq (m-1)/3$ implies that $\tau^* \geq 2(m-1)/3$. Since $m - 1 = 4c_4 + 3c_3 + 2c_2 + c_1 - 1$, we have

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

To obtain another lower bound on $\tau^*$, we consider the graph $G'_4$ again. Each of its isolated edges and each of its isolated vertices except one needs an adjacent edge. 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. Since the instances that we consider are feasible, we may assume that there is no solution in which the isolated edges and isolated vertices of $G'_4$ belong to the same path of length 2. 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 bound is asymptotically tight, consider the graph given in Figure 4. It consists of $c$ equal components on twelve vertices each and one isolated vertex. The numbers displayed at the edges indicate the phases in which

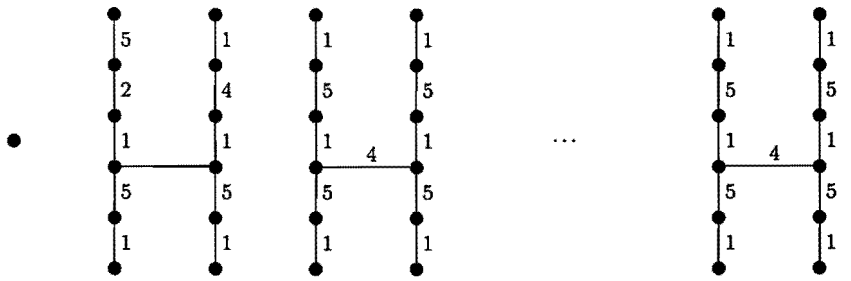


*Figure 4: Worst-case instance for the greedy algorithm for the TCP2*

the greedy algorithm selects the edges. The greedy test cover has size $\tau^G = 11c - 1$. Since each of the large components can be partitioned into four paths of length 2, we have $\tau^* = 8c$. Thus, $\lim_{c\to\infty} \tau^G/\tau^* = 11/8$.

# B Analysis of local improvement for PPP2

We denote the vertex set of a graph $G$ by $V(G)$. Consider the graph $G = (\{u\} \cup V(\mathcal{H}) \cup V(\mathcal{O}), E(\mathcal{H}) \cup E(\mathcal{O}))$, with $u$ an isolated vertex that we will disregard from now on, $\mathcal{H}$ a collection of paths of length 2 that cannot be improved within $G$ by the heuristic $H_k$, and $\mathcal{O}$ another collection of paths of length 2.

From now on we will use the word *path* for a path of length 2. We distinguish $\mathcal{H}$-paths and $\mathcal{O}$-paths. Clearly, heuristic $H_k$ achieves its worst-case performance on a graph $G$ with $|\mathcal{O}|/|\mathcal{H}|$ as large as possible. To determine upper bounds on this ratio for $H_0, \ldots, H_4$, we formulate five linear programming problems.

We give each vertex in $V(\mathcal{H})$ a label; vertices in $V(\mathcal{O})\backslash V(\mathcal{H})$ will remain unlabeled. The labeling is illustrated in Figure 5. Here and in all following figures we represent $\mathcal{H}$-paths by dashed lines and $\mathcal{O}$-paths by solid lines. All vertices $v \in V(\mathcal{H})\backslash V(\mathcal{O})$ receive label 0. All other vertices are both on an $\mathcal{O}$-path and on an $\mathcal{H}$-path, and their label depends on how the $\mathcal{O}$-path intersects the collection of $\mathcal{H}$-paths.

Let $p$ be an $\mathcal{O}$-path. If $p$ intersects three $\mathcal{H}$-paths, then each $v \in p$ receives label 4.

If $p$ intersects two $\mathcal{H}$-paths, then one of the following two cases occurs. One $\mathcal{H}$-path contains two vertices of $p$, both getting label 5, whereas the remaining vertex of $p$ gets label 3. In the other case both $\mathcal{H}$-paths contain only one vertex of $p$; the remaining vertex is unlabeled. If one of the border vertices is unlabeled, then the middle vertex of $p$ gets label 2 and the border vertex of $p$ gets label 3. The case in which the middle vertex is unlabeled is dominated by the previous case, which we illustrate by an exchange argument in Figure 6.

If $p$ intersects only one $\mathcal{H}$-path, then $v \in p$ receives label 1 if $|V(\mathcal{H}) \cap V(p)| = 1$. If $|V(\mathcal{H}) \cap V(p)| = 2$, then $v$ receives label 6, and if $|V(\mathcal{H}) \cap V(p)| = 3$, then $v$ receives label 7. However, the last case will not occur in a worst-case graph. A configuration with a label 6 vertex is dominated by a configuration with a label 1 vertex; see Figure 6. We disregard labels 6 and 7 from now on.

We denote the set of vertices with label $i$ by $V_i$, for $i = 0, \ldots, 5$, and express

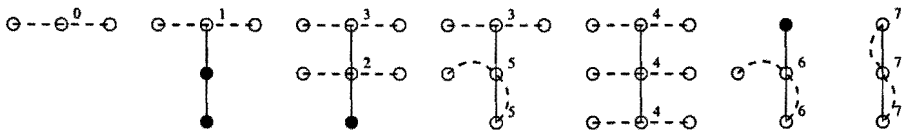

*Figure 5: Labeling of vertices; black dots are unlabeled*

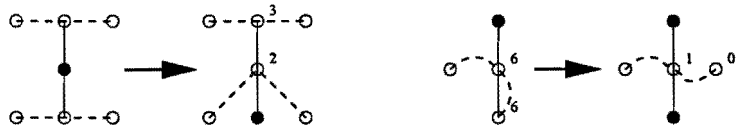


*Figure 6: Configurations replaced by dominating ones*

the number of $\mathcal{O}$-paths as

$$|V_1| + \frac{1}{2}|V_2| + \frac{1}{2}|V_3| + \frac{1}{3}|V_4| + \frac{1}{4}|V_5|. \tag{4}$$

Let an $\alpha\beta\gamma$-path be an $\mathcal{H}$-path with the middle vertex labeled $\beta$ and the border vertices labeled $\alpha$ and $\gamma$, with $\alpha \leq \gamma$, $0 \leq \alpha, \beta, \gamma \leq 5$. If $f(\alpha\beta\gamma)$ is the fraction of $\alpha\beta\gamma$-paths among the $\mathcal{H}$-paths, then, for $i = 0, \ldots, 5$,

$$|V_i| = \sum_{(\alpha\beta\gamma)} |\mathcal{H}| f(\alpha\beta\gamma)(1_i(\alpha) + 1_i(\beta) + 1_i(\gamma)), \tag{5}$$

where $1_i(x) = 1$ if $x = i$ and 0 otherwise. Substituting (5) in (4) and dividing the result by $|\mathcal{H}|$ yields a formulation of the objective of finding a graph with a highest possible ratio $|\mathcal{O}|/|\mathcal{H}|$.

The objective function is to be maximized under the restriction that all fractions are non-negative and add up to 1:

$$\sum_{(\alpha\beta\gamma)} f(\alpha\beta\gamma) = 1. \tag{6}$$

Another restriction follows from the definition of the labels 2, 3, and 5. For any vertex with label 3 there must be either a vertex with label 2 or two vertices with label 5. We therefore add the equality

$$|V_2| + \frac{1}{2}|V_5| = |V_3|. \tag{7}$$

The solution to this first LP problem, which we denote by $LP_0$, gives an upper bound on the ratio $1/\rho_0$ for $H_0$. The solution is 3, with $f(111) = 1$ and all other fractions equal to 0. The bound is matched by the example given in Figure 2. This result was already proved in Theorem 4.4.

Since a 111-path can be improved by any $H_k$, $k \geq 1$, such paths should get a 0-fraction in the LP solution for $k \geq 1$. However, there are more path types to be excluded, even for $H_1$; think e.g. of a 132-path. In order to facilitate the definition of the proper restriction, we define the notion of a *black vertex*.

Define $N_{\mathcal{H}}(v)$ as the pair of vertices on the same $\mathcal{H}$-path as $v$, and $N_{\mathcal{O}}(v)$ as the pair of vertices on the same $\mathcal{O}$-path as $v$. A vertex $v$ is black if $v \in V(\mathcal{H})$ and the subgraph induced by the vertex set $V(\mathcal{O}) \backslash V(\mathcal{H}) \cup (N_{\mathcal{H}}(v) \backslash N_{\mathcal{O}}(v))$ contains a path. A vertex being black depends only on the type of $\mathcal{H}$-path

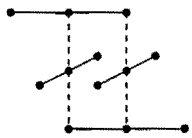

*Figure 7: Worst-case instance for $H_1$*

containing it. For instance, a 111-path has three black vertices, of a 212-path the two vertices labeled 2 are black but the vertex labeled 1 is not, a 333-paths does not contain any black vertices. A black vertex threatens to create a possibility for improvement. Specifically, if a black vertex exists with label 1, then $H_1$ can improve $\mathcal{H}$, which must be excluded at this point. Let $BL(\mathcal{H})$ be the set of black vertices. Thus, to create $LP_1$, we add to $LP_0$ the restriction that fractions of path types that contain black vertices with label 1 are 0. We give the description of this constraint comprehensively in terms of vertex sets, since the direct description in terms of fractions would be rather lengthy, and descriptions would become even lengthier in the next LP formulations.

$$|BL(\mathcal{H}) \cap V_1| = 0. \tag{8}$$

The optimal solution value of $LP_1$ is 2, which is matched by the example given in Figure 7. We emphasize that we did not aim at finding the LP formulation that excludes all configurations on which an $H_1$-improvement is possible.

To find an upper bound on $1/\rho_2$, we observe that any $\mathcal{O}$-path containing a black vertex labeled 3 and a black vertex labeled 2 or two black vertices labeled 5 (black vertices labeled 5 always come in pairs) gives rise to an $H_2$-improvement. Thus, for any $\mathcal{H}$-path $p$ with a black vertex labeled 2 or two vertices labeled 5, there must exist an $\mathcal{H}$-path $q$ with a non-black vertex labeled 3 that is on the same $\mathcal{O}$-path as the black vertex. The other way round should also hold. Therefore, a constraint that enforces these situations does not only depend on a pair of path-types but also on how they are related through an $\mathcal{O}$-path. To define such a constraint, we distinguish $\mathcal{H}$-paths of type 033, 303, 333, 032, 233, 334, 343, 234, 344, 434, 255, 355, 535, or 525 with a $^+$ or a $^-$ label. A path $q$ of any of these types gets a $^+$ label if there is a black vertex $v \in V_2 \cup V_3 \cup V_5 \setminus V(q)$ with $N_{\mathcal{O}}(v) \cap V(q) \neq \emptyset$, and a $^-$ label otherwise. For these types of paths we also distinguish two variables $f(\alpha\beta\gamma^+)$ and $f(\alpha\beta\gamma^-)$ in our LP formulations. The non-black vertices $v \in V_2 \cup V_3 \cup V_5$ contained in an $\mathcal{H}$-path with a $^-$ label are white. As a consequence, an $\mathcal{O}$-path cannot contain a black and a white vertex. All vertices in $V(\mathcal{H})$ that are not white or black are gray.

We define $BL(\mathcal{H})$, $GR(\mathcal{H})$, and $WH(\mathcal{H})$ as the sets of black, gray, and white vertices, respectively. We obtain the formulation $LP_2$ for bounding $1/\rho_2$ by adding the following constraints to $LP_1$:

$$|BL(\mathcal{H}) \cap V_2| + \frac{1}{2}|BL(\mathcal{H}) \cap V_5| \leq |GR(\mathcal{H}) \cap V_3|, \tag{9}$$

$$|BL(\mathcal{H}) \cap V_3| \leq |GR(\mathcal{H}) \cap V_2| + \frac{1}{2}|GR(\mathcal{H}) \cap V_5|. \tag{10}$$

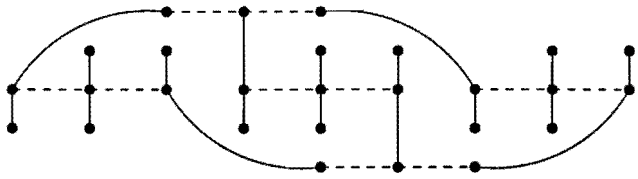

*Figure 8: Worst-case instance for $H_2$*

The optimal solution value of $LP_2$ is $9/5$, which is matched by the example given in Figure 8.

In order to find an upper bound on $1/\rho_3$, we define $LP_3$ by adding three constraints to $LP_2$. The first and the second constraint are based on the following observation. If two gray vertices $v$ and $w$ labeled 2, 3, or 5 are on the same $\mathcal{H}$-path but not on the same $\mathcal{O}$-path, then $N_{\mathcal{O}}(v)$ and $N_{\mathcal{O}}(w)$ cannot both contain a black vertex, unless these two vertices are contained in one $\mathcal{H}$-path $q$ of type 202, 222, 232, 242, 323, or 223. If in this case $q$ is of type 222, 232, 323, or 223, there cannot be an $\mathcal{O}$-path containing the gray vertex contained in $q$ and a black vertex. Figure 9 illustrates these situations. The relative sizes of configurations that are excluded in the above observations lead to the following two constraints, in which $|\alpha\beta\gamma|$ denotes the number of paths of type $\alpha\beta\gamma$:

$$\begin{aligned} &|BL(\mathcal{H}) \cap V_2| + \tfrac{1}{2}|BL(\mathcal{H}) \cap V_5| - |202| - |222| - |242| \\ &\leq |\mathcal{H}\text{-paths containing a gray vertex labeled 3}|; \end{aligned} \tag{11}$$

$$\begin{aligned} &|BL(\mathcal{H}) \cap (V_2 \cup V_3)| + \tfrac{1}{2}|BL(\mathcal{H}) \cap V_5| - |202| - |242| \\ &\leq |\mathcal{H}\text{-paths containing a gray vertex labeled 2, 3, or 5}|. \end{aligned} \tag{12}$$

The third constraint is based on the observation that an $\mathcal{O}$-path cannot contain three black vertices labeled 4:

$$|BL(\mathcal{H}) \cap V_4| \leq 2 \cdot |GR(\mathcal{H}) \cap V_4|. \tag{13}$$

The optimal solution value of $LP_3$ is $11/7$, matched by the example given in Figure 10.

The upper bound for $1/\rho_4$ is obtained by adding constraints based on two observations. First, consider an $\mathcal{H}$-path containing at least two gray vertices labeled 2, 3, or 5 that are not contained in the same $\mathcal{O}$-path. By definition such a path has a $^+$ label and therefore contains a gray vertex $v$ for which

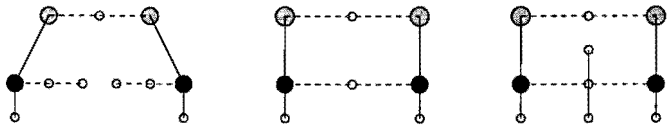

*Figure 9: $H_3$ improves the graphs on the left and on the right, but not the middle one*

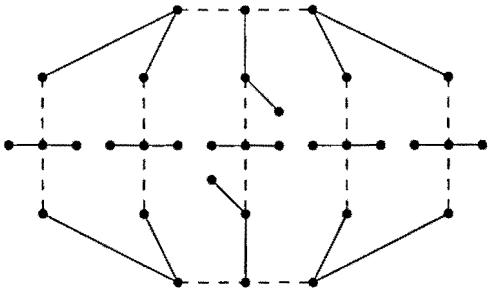

*Figure 10: Worst-case instance for $H_3$*

$N_{\mathcal{O}}(v) \cap BL(\mathcal{H}) \cap (V_2 \cup V_3 \cup V_5) \neq \emptyset$. Now observe that for each vertex $u \in (N_{\mathcal{H}}(v) \setminus N_{\mathcal{O}}(v)) \cap GR(\mathcal{H}) \cap (V_2 \cup V_3 \cup V_5)$ there can only be an $\mathcal{H}$-path $p$ with a $^+$ label for which $V(p) \cap N_{\mathcal{O}}(u) \neq \emptyset$ if there is an $\mathcal{H}$-path $q$ of type 202, 222, 232, 242, 323 or 223 containing a black vertex in $N_{\mathcal{O}}(v)$ and a black vertex $w$ for which $V(p) \cap N_{\mathcal{O}}(w) \neq \emptyset$. If in this case $q$ is of type 222, 232, 323, or 223, there cannot be an $\mathcal{O}$-path containing the gray vertex contained in $q$ and a black vertex.

If $N_{\mathcal{O}}(u)$ contains at least one black vertex, then an $\mathcal{H}$-path $r$ of type 202, 222, 232, 242, 323 or 223 can also contain the black vertices in $N_{\mathcal{O}}(v)$ and $N_{\mathcal{O}}(u)$. If in this case $r$ is of type 222, 232, 323 or 223, then the gray vertex contained in $r$ cannot be contained in the same $\mathcal{O}$-path as a black vertex or a gray vertex contained in a path with a $^+$ label. In all other cases the gray vertices of type 2, 3, or 5 incident to an $\mathcal{H}$-path without a label can be contained in the same $\mathcal{O}$-path as vertex $u$. Vertex $u$ can also be incident to an $\mathcal{O}$-path containing a white vertex; see Figure 11. We therefore add the following constraint to $LP_3$:

$$\begin{aligned} &2|BL(\mathcal{H}) \cap (V_2 \cup V_3)| + |BL(\mathcal{H}) \cap V_5| - |222| - |232| - |323| - |223| \\ &-3|242| - 3|202| \leq |\mathcal{H}\text{-paths with a } ^+ \text{ label}| - |333^+| + |434^+| + |344^+| \\ &+2|\mathcal{H}\text{-paths without a label containing a gray vertex labeled 2, 3, or 5}| \\ &+|WH(\mathcal{H}) \cap (V_2 \cup V_3)| + \tfrac{1}{2}|WH(\mathcal{H}) \cap V_5|. \end{aligned} \tag{14}$$

Second, note that for a gray vertex $v$ labeled 4 with $N_{\mathcal{O}}(v) \cap BL(\mathcal{H}) = 2$ there cannot exist a vertex $w \in N_{\mathcal{H}}(v) \cap V_3 \cap GR(\mathcal{H})$ with $N_{\mathcal{O}}(w) \cap (V_2 \cup V_5) \cap BL(\mathcal{H}) \neq \emptyset$; see Figure 12. The only exception is that there is an $\mathcal{H}$-path of type 224 that

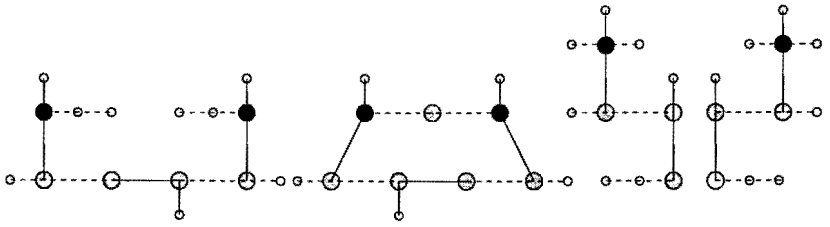

*Figure 11: $H_4$ improves the graph on the left, but not the other ones*

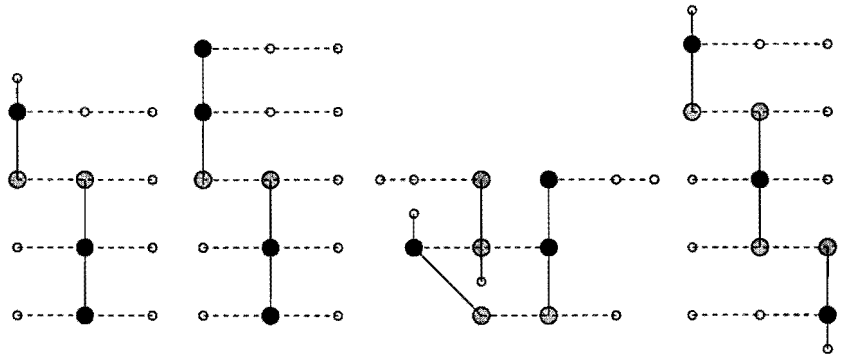

*Figure 12: $H_4$ improves the graph on the left, but not the other ones*

is incident to a vertex in $N_{\mathcal{O}}(v)$ and with a vertex in $N_{\mathcal{O}}(w)$. Note that for a vertex $v$ labeled 4 incident to an $\mathcal{H}$-path of type $344^+$, $434^+$, $334^+$, or $343^+$ we have $|N_{\mathcal{O}}(v) \cap BL(\mathcal{H})| \leq 1$, and as a consequence $|N_{\mathcal{O}}(v) \cap GR(\mathcal{H})| \geq 1$. We complete $LP_4$ by adding the following constraint:

$$\begin{aligned} &2 \cdot |BL(\mathcal{H}) \cap V_4| + 2 \cdot |BL(\mathcal{H}) \cap V_2| + |BL(\mathcal{H}) \cap V_5| - 2 \cdot |202| \\ &-2 \cdot |222| - 2 \cdot |232| - 2 \cdot |242| - 2 \cdot |224| \leq 4 \cdot |GR(\mathcal{H}) \cap V_4| \\ &+2 \cdot |\mathcal{H}\text{-paths incident to at least one gray vertex labeled 3 and no} \\ &\text{gray vertex labeled 4}| - 4 \cdot |344^+| - 4 \cdot |434^+| - |334^+| - |343^+| \end{aligned} \tag{15}$$

The optimal solution value of $LP_4$ is 3/2, which is matched by the example given in Figure 13.

$LP_4$ could be extended to an LP formulation for bounding $1/\rho_5$, but the formulations become rather complicated. Also, the analysis thus far did not shed much light on patterns that could be used in analyzing local order improvement heuristics of a higher order. We therefore decided to stop here.

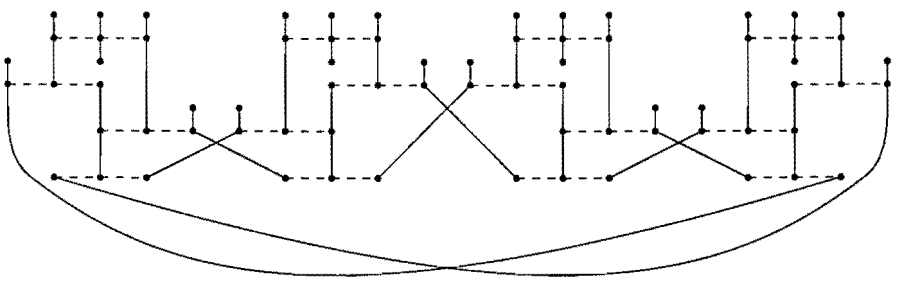

*Figure 13: Worst-case instance for $H_4$*