Approximation Algorithms for the k-Clique Covering Problem

Olivier Goldschmidt* Dorit S. Hochbaum† Cor Hurkens‡
Gang Yu§

May 10, 1995

Abstract

The problem of covering edges and vertices in a graph (or in a hypergraph) was motivated by a problem arising in the context of component assembly problem. The problem is, given a graph and a clique size k, find the minimum number of k-cliques such that all edges and vertices of the graph are covered by (included in) the cliques. This paper provides a collection of approximation algorithms for various clique sizes with proven worst-case bounds. The problem has a natural extension to hypergraphs, for which we consider one particular class. The k-clique covering problem can be formulated as a Set Covering problem. It is shown that the algorithms we design, that exploit the structure of this special Set Covering problem, have better performance than those derived from direct applications of general purpose algorithms for the Set Covering. In particular, these special classes of Set Covering problems can be solved with better worst-case bounds and/or complexity than if treated as general Set Covering problems.

Key words: Approximation Algorithms; Clique Covering; Set Covering; Worst Case Analysis.

Suggested running title: Approximation Algorithms for k-Clique Covering.

*Department of Mechanical Engineering, The University of Texas at Austin

†I.E.& O.R. Department, University of California at Berkeley. This research has been supported in part by ONR grant N00014-91-J-1241.

‡Department of Mathematics and Computing Science, Eindhoven University of Technology, the Netherlands

§M.S.I.S. Department, The University of Texas at Austin. This research has been supported in part by ONR grant N00014-91-J-1241.

1 Introduction

The problem of *k-clique covering* (CC_k) is defined on a graph (or a hypergraph) and a given clique size k. The aim is to use the least number of cliques - or subsets of k vertices - so that each vertex and each edge is contained in at least one such clique. When all edges are covered then also all vertices incident to these edges are covered, so the issue of covering vertices in addition to edges is of interest only in graphs with isolated vertices.

The k-clique covering problem's objective value differs from the *clique covering number* of a graph (frequently used in literature related to perfect graphs) in that the clique covering number of a graph is a partitioning of the edges of the graph into a minimum number of complete subgraphs. Hence for each clique used in the clique covering number problem, all edges of the clique are present in the graph. As such, each edge must belong to exactly one clique in the cover. In our problem, each clique covers all edges contained in it. Contrary to the clique covering number, not every edge of the clique must be present in the graph. Furthermore, an edge may be covered by more than one clique.

One problem addressed in the literature that is related to the 3-clique covering problem is the partitioning of a graph into triangles. The feasibility decision problem - whether the edges of a graph can be partitioned into triangles - was proved NP-complete by Holyer [12]. Therefore finding the minimum number of triangles to cover the edges of a graph is NP-hard.

A general definition of the (CC_k) problem for hypergraphs is as follows. A hypergraph $H = (V, F)$ is defined by a vertex set $V = \{1, ..., n\}$ and a hyperedge set $F \subseteq 2^V$. A clique of size k in a hypergraph H is a subset of vertices $K \subseteq V$ such that the hyperedges representing all subsets of K exist in H and $|K| = k$. In the case $|f| = 2$, for all $f \in F$, each hyperedge is an edge containing a pair of vertices, and the hypergraph is a graph. The k-Clique Covering (CC_k) problem is to find the minimum number of cliques of size no more than k such that all hyperedges of H are covered by (included in) the cliques.

The k-clique covering problem can be viewed as a special case of the Set Covering problem. In this context, approximation algorithms that apply to the Set Covering problem are also applicable to the k-clique covering problem. The drawback of using the set covering problem is that the input to the Set Covering problem includes all possible subsets of size k and hence is exponential in k. We propose an approach that exploits the special structure of the clique covering problem and delivers better worst case bounds than the ones using the set covering.

The main results in this paper are approximation algorithms for the problem (CC_k) on graphs and on a class of hypergraphs. We present approximation algorithms based on the formulation of the problem as a Set Covering problem and algorithms specifically derived for the clique cover problem. The latter algorithms have better worst-case performance. Consequently, these classes of Set Covering problems can be solved with better worst-case bounds and/or complexity than if treated as general Set Covering problems.

The (CC_k) problem has many practical applications in flexible manufacturing systems and component assembly in the semiconductor industry. In [9], Goldschmidt et al. provide a detailed description of some related applications.

In related literature, Tang and Denardo [22] developed a branch-and-bound procedure for solving the (CC_k) problem. The lower bound is generated by a so called sweeping procedure and it can be arbitrarily bad. Several heuristic procedures have been proposed and tested for generating feasible solutions. These heuristics are similar to bin packing heuristics in that they select a starting seed hyperedge for a new bin (clique) and sequentially fill it according to some precedence rule. Tang and Denardo [22], and Whitney and Gaul [23] propose different rules for selecting next hyperedge to add to the current clique. In these papers, no analysis and worst-case bounds have been provided for the heuristic solutions.

Let z^H and z^* be respectively the number of cliques derived by an approximation algorithm and the minimum number of cliques of a (CC_k) instance. An algorithm is a δ-approximation algorithm if, for any family of instances of the problem, $z^H \leq \delta z^* + o(z^*)$. We call δ the *worst case bound* of the algorithm, and the algorithm a *δ- approximation algorithm.*

Our results include approximation algorithms for the following cases. For the triangle covering problem, the 3-clique covering, we present two approximation algorithms with worst case bounds of 1.4 and 1.5 respectively. For the 4-clique covering our algorithm is a $2\frac{1}{3}$- approximation algorithm. For the general k-clique covering we describe an algorithm that is a $(\frac{k}{2} + \frac{k}{k-1})$- approximation algorithm. For this case the *greedy heuristic* for Set Covering gives a bound of $\mathcal{H}(\binom{k}{2}) \approx 2\log k$ with running time $O(n^k)$. $\mathcal{H}(d)$ is a Harmonic series defined as $\mathcal{H}(d) = \sum_{j=1}^{d}(1/j)$. The Harmonic series is asymptotically equal to the natural logarithm of d (plus the Euler constant). For the problem on hypergraphs, when each hyperedge to be covered is of size $k-1$, we describe two approximation algorithms and demonstrate that they compare favorably with algorithms derived from setting the problem as a Set Covering problem.

In Section 2, we discuss the reduction of the k-Clique Covering problem to the Set Covering problem. Once reduced to the Set Covering problem, the greedy heuristic [4] for Set Covering becomes applicable. Throughout this paper, we refer to the greedy heuristic as *greedy.* For a Set Covering problem on a hypergraph with n vertices and with m elements to be covered, elements that are edges and vertices in a graph or hyperedges in a hypergraph, the *greedy* is a $\min\{k \log 2, \log m\}$-approximation algorithm of complexity $O(\binom{n}{k}\min(k!, m))$.

Section 3.1 includes the approximation algorithms for graphs with clique sizes 3 or 4. Section 3.2 contains two approximation algorithms for solving the problem on hypergraphs in which each hyperedge has exactly $(k-1)$ vertices.

Table 1 summarizes the results of section 3. *Case* (k, q) denotes an instance for which each hyperedge has cardinality q and *case* $(k, \leq q)$ an instance for which a

Algorithm	Case	Running time	δ
H_1	$(3, \leq 2)$	$O(m^{2.5})$	1.4
H_2		$O(m)$	1.5
H_3	$(4, \leq 2)$	$O(m)$	7/3
H_4	$(k, \leq 2)$	$O(m)$	$\frac{k}{2} + \frac{k}{k-1}$
H_5	$(k, k-1)$	$O(m^3 k)$	$\mathcal{H}(k) - \frac{1}{6}$
H_6		$O(m^{2.5})$	$\left\lceil \frac{k}{2} \right\rceil$

Table 1: δ-approximation algorithms

hyperedge consists of at most q vertices.

In section 4 we conclude with some future research directions.

2 Reduction to the Set Covering Problem

The *Set Covering* problem (SC) is defined for a universal set of m elements, $I = \{1 \dots m\}$, and a collection of p sets $S_i \subset I$, $i = 1 \dots p$. The problem is to find a smallest cardinality collection of sets, the union of which is I.

Not only is the Set Covering problem NP-hard, but it was recently established that an approximation algorithm for the problem with better than logarithmic bound is impossible, unless all NP problems are solvable in subexponential time ([18]) – a fairly unlikely prospect.

The Set Covering problem may be generalized to a problem of covering sets with sets, as opposed to covering elements with sets. A set S is said to be covered by a set $\bar{S}$ if $S \subseteq \bar{S}$. The (CC_k) is a class of instances of covering sets with sets when the covering sets S_i are cliques of size k and the elements are vertices and edges (hyperedges) of a graph (hypergraph). As such it is a special case of the Set Covering problem with all sets of fixed size k. As noted before, this Set Covering problem is NP-hard since covering edges with minimum number of triangles is a special case.

The (CC_k) has shorter input representation than the Set Covering. Let the k-clique covering problem be defined on a hypergraph with n vertices. Since any subset of k vertices must be considered as a potential clique, $p = \binom{n}{k}$ sets are listed as part of the input. On the other hand, for the input of (CC_k) only the size limit, k, is specified in addition to the graph (or hypergraph).

In order to reduce (CC_k) to (SC), we consider the union of the hyperedges (that are sets of vertices) as the universal set. Any subset of vertices of size k can be potentially used in a cover. So all such subsets are enumerated and for each one we list all the hyperedges and vertices that are contained in such a subset. This creates a Set Covering problem with the number of sets p equal to $\binom{n}{k}$, where n is the number of vertices in the graph, and the number of elements equal to the number of hyperedges and isolated vertices, m.

There are two known approximation algorithms available for (SC). One bounds the worst-case error by the maximum number of sets that an element belongs to, [10]. The other is the *greedy* with worst-case error bound equal to the Harmonic series $\mathcal{H}(d)$, where d is the largest set size [16, 17, 4]. The running time of the *greedy* is $O(n \log n)$, where n is the number of sets. With the recent result of Lund and Yanakakis, [18], the *greedy* is, up to a constant factor, a best possible approximation algorithm for the Set Covering problem, as it cannot be improved unless all NP problems are in DTIME$[n^{poly\ (\log n)}]$. The *greedy* is also ideally suited to the input of (CC_k) as the largest set size is the maximum number of hyperedges covered by a k-clique, 2^k. Consequently, this reduction makes a $\mathcal{H}(2^k)$-approximation algorithm readily available. There are $\binom{n}{k}$ potential covering sets. It takes $O(mk)$ to identify the hyperedges covered by a k-clique. At each iteration, we scan the list of cliques for the one that covers the most hyperedges. At each iteration, at least one hyperedge gets covered and thus removed. So the *greedy* iterates at most m times. Therefore, the running time of the *greedy* on an instance of (CC_k) is $O(\binom{n}{k} m^2 k)$.

In a companion paper, [9], we demonstrated an improvement to the *greedy* called the *modified greedy* heuristic or simply the *modified greedy*. The *modified greedy* is a $\mathcal{H}(d) - \frac{1}{6}$-approximation algorithm. Its running time is $O(n \log n + m^{2.5})$ on a general set covering problem with n sets and m elements and it follows that its running time on an instance of (CC_k) is $O(\binom{n}{k} m + m^{2.5})$. This improvement is achieved by solving, in addition to the greedy procedure, also a matching problem in a graph. The *modified greedy* is useful for instances of Set Covering with largest set size being bounded by a small value, such as the clique covering problem.

The other approximation algorithms described here have either faster running times or better bounds, or both, for various special cases.

3 Approximation Algorithms

In this section, we present approximation algorithms for the k-clique covering problem. We first examine the (CC_k) problem on graphs for the cases $k = 3$ and $k = 4$, and then for arbitrary k. We then describe two approximation algorithms for covering a hypergraph with hyperedges all of size $(k-1)$ with k-cliques.

For the clique covering problem on graphs, all vertices that are not isolated get covered when all edges are covered. Only isolated vertices may require additional cliques to be covered. For simplicity sake, we are going to present the algorithms for covering edges only, and comment separately about the adjustments required to cover isolated vertices as well. In all cases this adjustment does not modify the worst-case bound.

We use the common notation for a graph $G = (V, E)$, $|V| = n$ and $|E| = m$. In case G has isolated vertices, m denotes the number of edges and isolated vertices.

3.1 Covering Graphs with k-Cliques

The 2-clique covering problem on a graph is trivially solvable: Each edge is covered by a separate clique and the isolated vertices are covered two per clique. For $k = 3$, the problem is NP-hard. To see this, notice that the number of triangles required to cover the edges of the graph is at least $\lceil \frac{m}{3} \rceil$. When m is a multiple of 3, this is achievable only if the edges of G can be partitioned into triangles. Recognizing whether a graph has a partition into triangles was shown to be NP-complete by Holyer [12].

The rest of this section is organized as follows. We first analyze the cases $k = 3$ and $k = 4$. For the case $k = 3$, we provide two algorithms, a 1.4-approximation algorithm called (H_1) and a 1.5-approximation algorithm called (H_2), with running times $O(m^{3/2})$ and $O(m)$ respectively. While (H_1) has better worst-case bound, (H_2) has faster running time. These are the first known approximation algorithms for covering a graph with triangles. For the case of $k = 4$, we present a 7/3-approximation algorithm, (H_3), with running time $O(m)$. In the last subsection, we describe approximation algorithms for arbitrary k.

3.1.1 The Triangle Covering Problem

Both algorithms presented here are adaptations of the greedy approach.

Algorithm (H_1) runs in two phases. In the first phase, a maximal number of edge-disjoint triangles of G are covered. To find a maximal number of edge-disjoint triangles, we use the procedure by Itai and Rodeh [15] for determining whether a graph contains a triangle. The complexity of this procedure is $O(m^{3/2})$. We adapt it to find a maximal number of edge-disjoint triangles without increasing the complexity. The edges of these edge-disjoint triangles are covered and deleted from the graph.

In phase 2 there is no remaining triangle in the graph. It is possible to solve the 3-clique covering problem in a triangle-free graph in linear time. In a graph that contains no triangles, each covering triangle covers either a 2-chain (two adjacent edges) or a single edge. It is known ([19]) that if the number of edges m of a connected graph is even, then one can cover the edges with exactly $m/2$ 2-chains. Masuyama and Ibaraki [19] provide a linear time algorithm for solving this problem optimally. If a connected component of the triangle-free graph has an odd number of edges, then exactly one triangle is necessary to cover a single edge of the component while all others cover 2-chains. We call a triangle covering a single edge a *1-triangle.*

Note that if the input graph G does not contain triangles or if the triangles are edge disjoint, then the problem can be solved optimally in $O(m^{3/2})$ time.

The algorithm is given below:

Algorithm (H_1):

Input: Graph $G = (V, E)$.

Phase 1: Find a maximal collection of edge-disjoint triangles T. Set $T^H = T$. Let E_T be the edges of T, $E \leftarrow E \setminus E_T$.

Phase 2: While $G = (V, E)$ contains a non trivial component $G_i = (V_i, E_i)$, **do**

Cover E_i with a set T_i of $\left\lfloor \frac{|E_i|}{2} \right\rfloor$ 2-chains and possibly a 1-triangle (if $|E_i|$ is odd). Set $T^H \leftarrow T^H \cup T_i$, and $E \leftarrow E \setminus E_i$.

end

Output T^H.

End of (H_1)**.**

In case graph G contains a set of isolated vertices to be covered, the following adjustment is applied to (H_1). At the termination of phase 2, we assign isolated vertices, one for each 1-triangle. In phase 3, the remaining isolated vertices are covered three per triangle, with possibly one or two vertices in the last triangle used. T^H includes these additional triangles covering the isolated vertices.

Lemma 1 *The complexity of algorithm* (H_1) *is* $O(m^{3/2})$*.*

Proof: Phase 1 of the algorithm requires a maximal collection of edge-disjoint triangles. The algorithm of Itai and Rodeh finds a triangle, if one exists, by constructing a breadth-first-search tree and inspecting the non-tree edges. That procedure runs in $O(m^{3/2})$. We use this procedure and whenever a triangle is identified, its edges are removed from the graph and the data structure is updated in constant time. The procedure is then continued with no backtracking required. It is therefore possible to find a maximal collection of triangles with the same complexity as for finding a single triangle.

In phase 2, we find a maximum packing of 2-chains in a triangle-free graph G. This can be done in linear time by using the following procedure, which is an adaptation of the one by Masuyama and Ibaraki [19]. The procedure is presented for one connected component of G, $G_i = (V_i, E_i)$. Consider any spanning tree in that component. All non-tree edges are appended to the tree, each with a new vertex assigned to one of its endpoints. This creates a tree on $|E_i|$ edges and $|E_i| + 1$ nodes. The tree is suspended from any node, say 1, called the root node. While the tree contains more than one edge, consider any pair of leaf nodes that share the same parent node and cover the two edges incident to these leaf nodes with a 2-chain. Remove these two edges from the tree. If no such pair exists, then there is a node of degree 2 adjacent to a leaf node. The two edges incident to that node are packed in a 2-chain and removed from the tree. This operation is repeated until the tree contains at most one edge. The collection of 2-chains thus created forms a maximum packing.

If a single edge remains in the tree it is covered in phase 2 by a 1-triangle. The complexity of identifying a maximum packing is linear as can be shown by a straightforward inductive argument. The total number of triangles required to cover each component G_i is $\lceil |E_i|/2 \rceil$.

The complexity of algorithm (H_1) is therefore dominated by phase 1 and is $O(m^{3/2})$ as stated.

■

We claim that algorithm (H_1) is a 7/5-approximation algorithm.

Theorem 1 *The number of triangles delivered by (H_1), $|T^H|$, is at most 7/5 times the smallest number of triangles covering the graph.*

Proof: Let z^H denote the number of triangles found by the heuristic. Let T^* denote a triangle-covering of G of minimum cardinality z^*. Note that in our context, a triangle is defined as a graph $\Delta = (V_\Delta, E_\Delta)$, with at most three vertices and at most three edges. A triangle with three edges is called a *proper* triangle. Without loss of generality we may assume that for each edge $e \in E$ there is exactly one triangle $\Delta \in T^*$ for which $e \in E_\Delta$.

Let the graph remaining at the end of phase 1 be $G' = (V', E')$. Let s denote the number of components in G' with an odd number of edges. Let W denote the set of isolated vertices in the original graph G.

Case 1 : Assume that $s \leq |W|$.

Then $z^H = \frac{|E|-|E'|}{3} + \frac{|E'|+s}{2} + \left\lceil \frac{|W|-s}{3} \right\rceil$, where, for $i = 1,2,3$, the i-th term results from phase i of the heuristic. Obviously we have

$$z^* \geq \left\lceil \frac{|E| + |W|}{3} \right\rceil,$$

as any triangle in T^* contains at most three edges from E, at most three vertices from W, or one edge from E and one vertex from W. Similarly,

$$z^* \geq \frac{|E'| + s}{2} + \left\lceil \frac{|W| - s}{3} \right\rceil,$$

as any triangle contains at most two edges from E', one edge from E' and one vertex from W, or at most three vertices from W. At least s triangles contain one edge from E'. Combining the bounds we find:

$$\begin{aligned}
z^H &= \frac{|E| - |E'|}{3} + \frac{|E'| + s}{2} + \left\lceil \frac{|W| - s}{3} \right\rceil \\
&= \left\lceil \frac{|E| + |W|}{3} \right\rceil + \frac{|E'| + s}{6} + \frac{1}{3} \left\lceil \frac{|W| - s}{3} \right\rceil + \\
&\quad +\left(\frac{|E| + |W|}{3} - \left\lceil \frac{|E| + |W|}{3} \right\rceil\right) + \left(\frac{2}{3} \left\lceil \frac{|W| - s}{3} \right\rceil - \frac{|W| - s}{3}\right) \\
&\leq \frac{4z^* + 1}{3}
\end{aligned}$$

Note that equality holds only in the special case that $|W| - s = 1$, $|E|$ and $|W|$ are multiples of 3, and both bounds on z^* are binding. For $z^* \leq 4$ it is easily verified that $z^H \leq \frac{4}{3}z^*$. For $z^* \geq 5$ we have $z^H \leq \frac{4z^*+1}{3} \leq \frac{7}{5}z^*$.
Case 2 : Assume that $s > |W|$.
Then $6z^H = 2|E| + |E'| + 3s$.
In order to give an upper bound on this expression we derive three inequalities. Let s_i denote the number of components in E' with exactly i edges. Then $s - s_1 - s_3 - s_5 - s_7$ is the number of components with an odd number of at least 9 edges. Hence $9(s - s_1 - s_3 - s_5 - s_7) \leq |E'| - s_1 - 3s_3 - 5s_5 - 7s_7$ which is rearranged to

$$9s - 8s_1 - 6s_3 - 4s_5 - 2s_7 - |E'| \leq 0 \tag{1}$$

The following two inequalities represent different lower bounds on the minimum number of triangles in a triangle cover. The first one is by counting degrees. A vertex v of degree $\delta(v)$ must occur in at least $\left\lceil \frac{1}{2}\delta(v) \right\rceil$ triangles. Define $\bar{V} \subseteq V$ to be the set of vertices v of even degree for which there exists at least one triangle $\Delta \in T^*$ such that $v \in V_\Delta$ and $\#(e \in E_\Delta, e \ni v) \leq 1$. Such a vertex $v \in \bar{V}$ appears in at least $\frac{1}{2}\delta(v) + 1$ triangles. Note that $W \subseteq \bar{V}$. These considerations lead to the following:

$$\begin{aligned} 6z^* &\geq \sum_{\Delta \in T^*} \sum_{v \in V_\Delta} 2 = \sum_{v \in V} \sum_{\Delta \in T^*,\ V_\Delta \ni v} 2 \\ &\geq 2|W| + \sum_{v \in V,\ \delta(v) \text{odd}} (\delta(v) + 1) + \sum_{v \in V \setminus \bar{V},\ \delta(v) \text{even}} \delta(v) + \sum_{v \in \bar{V} \setminus W} (\delta(v) + 2) \end{aligned}$$

Here the first inequality uses the fact that a triangle contains at most three vertices, and the second one exploits the degrees of vertices. The result is reformulated as

$$6z^* \geq 2|W| + 2|E| + \sum_{v\ :\ \delta(v)\text{odd}} 1 + \sum_{v \in \bar{V} \setminus W} 2 \tag{2}$$

In phase 1 of the heuristic the parity of the degrees of the vertices does not change as only proper triangles are deleted from the graph. So odd degree vertices in G are odd degree vertices in G'. Note that in G' each component of size 1 or 3 contains at least two vertices of odd degree. Hence these components give a total contribution of $2s_1 + 2s_3$ in the third term of inequality (2). A component of size 5 or 7 that contains an odd-degree vertex, must contain at least two of these, and so it also gives a contribution of 2 in the third term of (2). If a component has only vertices of even degree, and at least one of these is in $\bar{E}$, then such a component contributes at least 2 in the fourth term in (2). It follows that components of size 5 or 7 that do not contribute 2 to (2) can only have vertices of even degree, and none of these vertices can belong to $\bar{V}$.

The second lower bound on z^* is derived by counting triangles in T^* around each component in G'. Let, for $i = 0, 1, 2$, T_i^* denote the set of triangles in T^* that cover

i edges of E'. For a component C of G', with vertex set $V(C)$ and edge set $E(C)$, and for $i = 1, 2$, let $T_i^*(C)$ denote the set of triangles $\Delta \in T^*$, with $|E(C) \cap E_\Delta| = i$. Note that $|E(C)| = |T_1^*(C)| + 2|T_2^*(C)|$. Furthermore note that a triangle in T^* can share edges with at most one component of G', as these components are pairwise vertex disjoint.

A triangle $\Delta \in T_0^*$ with $V_\Delta = \{a, b, c\}$, is assigned to components C_a, C_b, C_c, for one third each, where C_x denotes the component containing x.

A triangle $\Delta \in T_2^*$ is assigned completely to the component it shares two edges with.

A triangle $\Delta \in T_1^*$ sharing one edge with component C is completely assigned to C, if $|T_1^*(C)| = 1$. On the other hand, if $|T_1^*(C)| > 1$, then Δ is assigned to C for a fraction of $\frac{29}{36}$, and to C' for the remaining $\frac{7}{36}$, where C' is the component containing the third vertex of Δ. (Note that C' may be equal to C, and it is also possible that C' contains an isolated vertex, or that C' does not exist. In the latter cases, the fraction $\frac{7}{36}$ is considered lost.)

Let $z^*(C)$ denote the total number of triangles assigned to C. Then $z^* \geq \sum z^*(C)$, where the sum is taken over all components C in G'. We estimate the value of $z^*(C)$ by distinguishing between the following cases.

[A] If $|T_1^*(C)| = 0$, then $|E(C)|$ is even and $z^*(C) \geq |T_2^*(C)| = \frac{1}{2}|E(C)|$.

[B] If $|T_1^*(C)| = 1$, then $|E(C)|$ is odd and $z^*(C) \geq 1 + |T_2^*(C)| = \frac{1}{2}(|E(C)| + 1)$.

[C] If $|T_1^*(C)| > 1$, and $|E(C)|$ is even, then $z^*(C) \geq |T_2^*(C)| + \frac{29}{36}|T_1^*(C)| = \frac{1}{2}(|E(C)| - |T_1^*(C)|) + \frac{29}{36}|T_1^*(C)| > \frac{1}{2}|E(C)|$.

[D] If $|T_1^*(C)| > 1$, and $|E(C)|$ is odd, then $z^*(C) \geq |T_2^*(C)| + \frac{29}{36}|T_1^*(C)| = \frac{1}{2}(|E(C)| - |T_1^*(C)|) + \frac{29}{36}|T_1^*(C)| > \frac{1}{2}(|E(C)| + 1) + \frac{1}{3}$, as $|T_1^*(C)| \geq 3$.

Let $D5$ and $D7$ denote the set of components C of size 5 and 7, respectively, for which the premise of [D] applies, that is, for which $|T_1^*(C)| > 1$ and $|E(C)|$ is odd.

Let $B5$ denote the set of components C of size 5, for which $|T_1^*(C)| = 1$, $V(C) \cap \bar{V} = \emptyset$, and $\delta(v)$ is even, for all $v \in V(C)$. Such a component C forms a 5-cycle, and each vertex in C has degree two in each triangle of T^* in which it appears. Moreover, $E(T_2^*(C)) - E(C) = \{a, b\}$ and $E(T_1^*(C)) - E(C) = \{c, d\}$ with 4 distinct edges a, b, c, d which form a 4-cycle. Let T_a denote the proper triangle containing edge a found by the heuristic in phase 1. Let the other two edges in T_a be $\{a', a''\}$. By symmetry between a and b we may assume w.l.o.g. that a' and a'' are not in $E(T^*(C))$. If a' or a'' is in $E(T_0^*)$, then we have $z^*(C) \geq 3 + \frac{1}{3}$. If not, then they must both belong to $E(T_1^*(C'))$ for some other component C'. But then $|T_1^*(C')| > 1$, and so C has been assigned at least $2 \times \frac{7}{36} > \frac{1}{3}$ of a triangle. Again we find $z^*(C) \geq 3 + \frac{1}{3}$. We conclude that $z^*(C) \geq 3 + \frac{1}{3} = \frac{1}{2}(|E(C)| + 1) + \frac{1}{3}$, for each $C \in B5 \cup D5$.

Let $B7$ denote the set of components C of size 7, for which $|T_1^*(C)| = 1$, $V(C) \cap \bar{V} = \emptyset$, and $\delta(v)$ is even, for all $v \in V(C)$. Such a component C must be a 7-cycle. Let a be any edge from $E(T_2^*(C)) \setminus E(C)$, and let T_a denote the proper

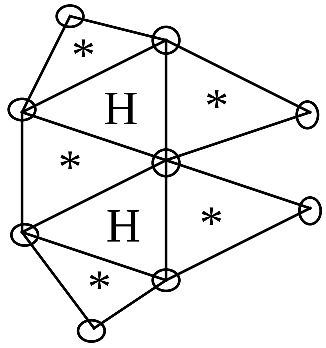

Figure 1: Illustration of the worst-case bound for Algorithm (H_1).

triangle containing edge a found by the heuristic in phase 1. Let the other two edges in T_a be $\{a', a''\}$. If a' and a'' have both ends in $V(C)$, then one of them is in $E(T_0^*)$. If one of a', a'' is in $E(T_0^*)$, then C has been assigned at least $\frac{1}{3}$ of a T_0^*-triangle, so $z^*(C) \geq 4 + \frac{1}{3}$. It is easily verified that if the above does not apply, then $E(T_2^*(C)) \setminus E(C)$ contains an edge a, for which both edges a' and a'' are not in $E(T_0^*) \cup E(T^*(C))$. Hence they both belong to $E(T_1^*(C'))$ for some other component C'. We procede as in the case for $B5$ and find that $z^*(C) \geq 4 + \frac{1}{3} = \frac{1}{2}(|E(C)|+1) + \frac{1}{3}$, for $C \in B7 \cup D7$.

Combining these results we find

$$2z^* \geq \sum_C 2z^*(C) \geq |E'| + s + \sum_{C \in B5 \cup D5} \frac{2}{3} + \sum_{C \in B7 \cup D7} \frac{2}{3} \tag{3}$$

Taking a linear combination of the three inequalities, with weight $1/5$ for inequality (1), weight 1 for inequality (2), and weight $6/5$ for inequality (3) we find:

$$\begin{aligned} 6z^H &= 2|E| + |E'| + 3s \\ &\leq 0.2 \times (9s - 8s_1 - 6s_3 - 4s_5 - 2s_7 - |E'|) + \\ &\quad 1.0 \times (2|W| + 2|E| + \sum_{v:\delta(v) \text{ odd}} 1 + \sum_{v \in \bar{V} \setminus W} 2) + \\ &\quad 1.2 \times (|E'| + s + \sum_{C \in B5 \cup D5} \frac{2}{3} + \sum_{C \in B7 \cup D7} \frac{2}{3}) \\ &\leq 0.2 \times 0 + 1.0 \times 6z^* + 1.2 \times 2z^* = 8.4z^* \end{aligned}$$

which concludes our proof.

∎

Figure 1 illustrates the tightness of the bound of algorithm (H_1): Stars within triangles indicate an optimal covering, H's within triangles indicate the triangles selected in phase 1. Five more triangles are needed in phase 2 to cover the peripheral edges.

We now propose an alternative algorithm (H_2), that runs in linear time, and delivers a solution that is at most 1.5 times the optimum. (H_2) should be selected when running time is an important consideration.

Algorithm (H_2):

Input: Graph $G = (V, E)$. $T^H = \emptyset$.

Phase 1: For each non trivial component $G_i = (V_i, E_i)$ of $G = (V, E)$ with $|E_i| \equiv 3$ modulo 6, **do**

Select a vertex $v \in V_i$, and find a vertex $w \in V_i$ at maximum distance from v. If there is a proper triangle Δ containing w, set $T^H \leftarrow T^H \cup \Delta$ and $E \leftarrow E \setminus E_\Delta$.

end

Phase 2: While $G = (V, E)$ contains a non trivial component $G_i = (V_i, E_i)$, **do**

cover E_i with a set T_i of $\left\lfloor \frac{|E_i|}{2} \right\rfloor$ 2-chains and possibly a 1-triangle (if $|E_i|$ is odd). Set $T^H \leftarrow T^H \cup T_i$ and $E \leftarrow E \setminus E_i$.

end

Output T^H.

End of (H_2).

In case the input graph contains a set of isolated vertices to be covered, the same adjustment applied to (H_1) is applied to (H_2).

Theorem 2 *(H_2) is a linear time 1.5-approximation algorithm.*

Proof: Phase 1 of algorithm (H_2) is linear in m since it requires for each component one breadth-first search starting from vertex v and one starting from w. Phase 2 is identical to phase 2 of algorithm (H_1) and its running time is likewise linear (see the proof of lemma 1).

In case there are no isolated vertices it suffices to prove the bound of 1.5 for each non-trivial component of G. The reason is that no triangle in any optimal cover can cover edges from different components of G. Hence, assume without loss of generality, that the input graph G is connected. Let $z^H = |T^H|$ and z^* be respectively the number of cliques used by algorithm (H_2) and the optimal number of cliques used to cover the edges and vertices of G. Then

$$z^H = \left\lfloor \frac{|E|+1}{2} \right\rfloor \text{ and } z^* \geq \left\lceil \frac{|E|}{3} \right\rceil .$$

It follows that

$$z^H \leq \frac{|E|+1}{2} = \frac{3}{2}\frac{|E|}{3} + \frac{1}{2} \leq \frac{3}{2}\left\lceil \frac{|E|}{3} \right\rceil + \frac{1}{2} \leq \frac{3}{2}z^* + \frac{1}{2}$$

Note that if $|E| \neq 6k+3$, then equality can not hold throughout, and so we find $z^H \leq 1.5z^*$. In case $|E| = 6k+3$, phase 1 of the algorithm tries to detect a triangle containing a vertex w. If such a triangle is not found, then we know that $z^* \geq \frac{|E|}{3}+1$, and again equality does not hold throughout. In the case that a triangle is detected, its edges are deleted from E. Note that the resulting graph is still connected. Hence the heuristic will find in phase 2 a number of triangles equal to $\frac{|E|-3}{2}$. As a result $z^H = 1 + \frac{3}{2}\frac{|E|-3|}{3} \leq \frac{3}{2}\frac{|E|}{3}$, which settles this case.

Next we consider the case that there are isolated vertices to be covered. Let W be the set of isolated vertices, and let s be the number of odd components in graph G, at the start of phase 2. Obviously, if $|W| \leq s$, then the heuristic finds the same number of triangles as without the isolated vertices. It follows immediately that again $z^H \leq 1.5z^*$. We only need consider the case that $|W| \geq s+1$. Let t and u denote the number of components in graph G for which the number of edges is 1 modulo 3, and 2 modulo 3, respectively. Let E denote the set of edges of G at start of phase 1, and E' the set of edges at start of phase 2. Then the bound for z^* can be sharpened to

$$z^* \geq \frac{|E|-|E'|}{3} + \frac{|E'|+2t+u}{3} + \frac{|W|-t}{3}$$

Similar to the proof for Theorem 1 we derive

$$\begin{aligned}
z^H &= \frac{|E|-|E'|}{3} + \frac{|E'|+s}{2} + \left\lceil \frac{|W|-s}{3} \right\rceil \\
&\leq \frac{|E|-|E'|}{2} + \frac{|E'|+s}{2} + \left\lceil \frac{|W|-s}{3} \right\rceil \\
&\leq \frac{|E|-|E'|}{2} + \frac{|E'|+s}{2} + \frac{|W|-s}{2} + \frac{1}{2} \\
&= \frac{|E|-|E'|}{2} + \frac{|E'|+|W|+t+u}{2} + \frac{1-t-u}{2} \\
&\leq \frac{3}{2}z^* + \frac{1-t-u}{2}
\end{aligned}$$

For $t+u \geq 1$ this settles the proof. For $t+u=0$ equality can hold throughout only when $|E| = |E'|$ and $|W|$ are multiples of 3, $z^* = (|E|+|W|)/3$ and $|W|-s=1$. Hence each component can be partitioned into triangles. But then a contradiction follows from the fact that $s \neq 0$, so there must be a component of odd size $6k+3$, for which the algorithm fails to find a triangle in phase 1.

■

The bound is tight as is shown by taking as input a graph the components of which are the union of an even number of triangles. See figure 2 for an example.

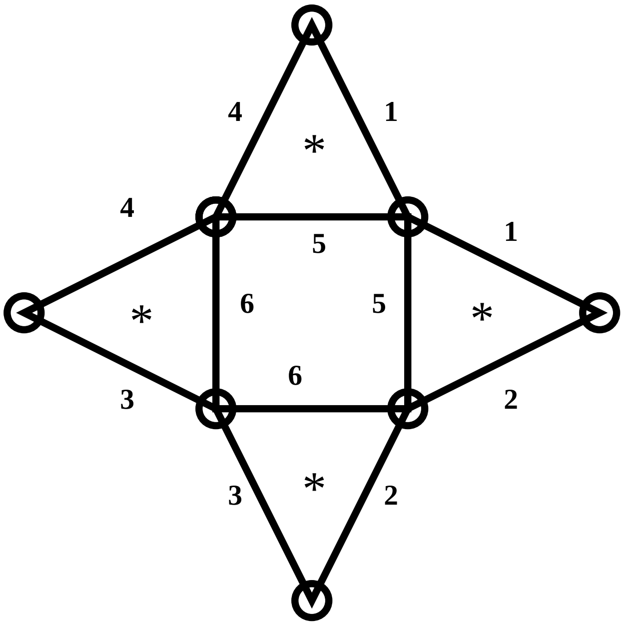

Figure 2: Illustration of the worst-case bound for Algorithm (H_2).

One might think that the following post-processing of phase 2 of algorithm (H_2) could improve the outcome: consider the 2-chains of phase 2, one by one. If there is an edge which forms a triangle with the two edges of the 2-chain under consideration, one covers this edge together with the 2-chain and deletes it from its own 2-chain. At the end of this post-processing, one eliminates the 2-chains which are now empty, i.e. the ones the two edges of which have been deleted. The example in figure 2 shows that the bound is not improved by applying this post-processing procedure.

3.1.2 The 4-clique Covering

Here we consider the problem of covering the edges and isolated vertices of a graph with 4-cliques. We present a linear time 7/3-approximation algorithm, algorithm (H_3).

An alternative approximation algorithm is the *modified greedy* applied to the problem presented as Set Covering problem. The worst-case bound of the *modified greedy* in this case is $\frac{23}{12}$ (see [9]) which is better than $\frac{7}{3}$. The running time of the *modified greedy* is $O(n^4 \log n)$, where n is the number of vertices of the graph, which is worse than the running time of algorithm (H_3), $O(m)$.

Algorithm (H_3) runs in two phases. In the first phase we cover a maximal number of edges with 4-cliques such that each 4-clique covers at least 3 edges. In the second phase, the remaining edges and the isolated vertices $W \subseteq V$, are covered optimally with a minimum number of 4-cliques.

An iteration of phase 1 consists of covering a connected subgraph induced on 4 vertices by a 4-clique and then of deleting the covered edges. Phase 1 proceeds until no connected component of G has more than 3 vertices. At the end of phase 1, all isolated vertices in $V \setminus W$ are deleted.

At the beginning of phase 2, the connected components of G, $G_1, G_2, \ldots, G_s$, are either triangles, isolated vertices of W, single edges, or 2-chains (two adjacent edges). The single edges are covered two per 4-clique (if the number of isolated edges is odd, then the extra edge is covered alone) and each 2-chain or triangle is covered by its own 4-clique.

The algorithm is given below:

Algorithm (H_3):

Input: Graph $G = (V, E)$.

Set $C^H = \emptyset$.

Phase 1:

Do until no connected component of $G = (V, E)$ has more than three vertices:

Find a connected subgraph on 4 vertices, C.

Set $C^H \leftarrow C^H \cup C$. Let E_C be the edge set of C. Set $E \leftarrow E \setminus E_C$.

enddo

Phase 2: Let T be the set of triangles and 2-chains of G. Set $C^H \leftarrow C^H \cup T$. Let C_F be the set of 4-cliques required to cover the set F of isolated edges of G (two per 4-clique). Set $C^H \leftarrow C^H \cup C_F$.

Output C^H.

End of (H_3).

In the case the input graph contains a set of isolated vertices to be covered, the following adjustment is applied to (H_3). At the termination of phase 2, we assign isolated vertices, one for each triangle or 2-chain in T, two for a single-edge clique. The remaining isolated vertices are covered 4 per clique with possibly 1, 2 or 3 vertices in the last clique used. We append these additional cliques to the set C^H.

The complexity and worst-case performance of algorithm (H_3) are given by the next theorem.

Theorem 3 (H_3) *is a linear time* $7/3$*-approximation algorithm.*

Proof: Phase 1 can be implemented using a linear time depth-first-search technique. Phase 2 scans through the remaining edges once. The assignment of isolated vertices will take linear time as well. Thus, the complexity of the algorithm is linear in the number of edges and isolated vertices, $O(m)$.

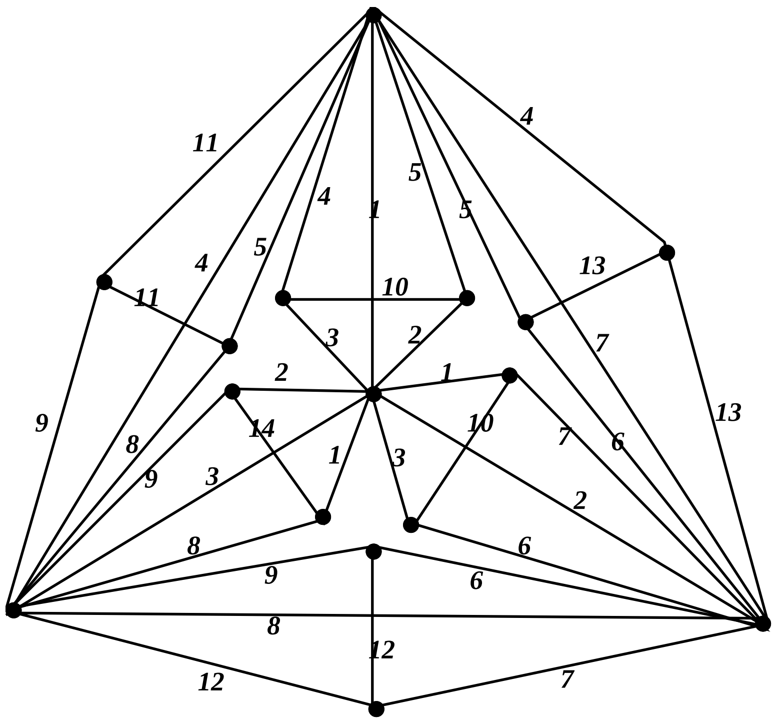

Figure 3: Illustration of the worst-case bound for the clique size 4.

Let z^* be the optimal number of 4-cliques and αz^* be the total number of 4-cliques used during phase 1. Because at least 3 edges are covered by each 4-clique used in phase 1, the total number of edges covered during phase 1 is at least $3\alpha z^*$. Also, no more than 6 edges can be covered by a 4-clique. Therefore, the number of elements which remain to be covered during phase 2 is at most $(6-3\alpha)z^*$. The number of 4-cliques used during phase 2 is at most $\min\{\left\lceil\frac{(6-3\alpha)z^*}{2}\right\rceil, z^*\}$. The first term follows from the fact that a phase 2 4-clique covers at least two edges. The second term is because the remaining edges and isolated vertices are covered optimally and therefore must use at most z^* 4-cliques. Hence, the total number of 4-cliques used by the approximation algorithm is at most

$$\begin{aligned}
z^H &\leq \alpha z^* + \min\{\left\lceil\frac{(6-3\alpha)z^*}{2}\right\rceil, z^*\} \\
&\leq \min\{\frac{(6-\alpha)z^*}{2} + \frac{1}{2}, (1+\alpha)z^*\} \\
&\leq \frac{7}{3}z^* + \frac{1}{3}.
\end{aligned}$$

The last inequality is derived by setting $\frac{(6-\alpha)z^*}{2} + \frac{1}{2} = (1+\alpha)z^*$.

∎

An example in figure 3 illustrates the tightness of the bound. In this example, there are 24 edges forming 6 cliques of size 4. The optimal solution is $z^* = 6$. By applying algorithm (H_3), edges are covered in the order as labeled in the graph. Numbers associated with each edge indicate which clique the corresponding edge belongs to. Algorithm (H_3) uses 14 cliques with the first 9 cliques containing 3 edges each, the next 4 cliques containing 2 edges each and the last clique containing only one edge. Thus the ratio $\frac{z^H}{z^*}$ is $\frac{14}{6} = \frac{7}{3}$.

3.1.3 k-Clique Covering in a Graph

We extend our discussion to the general clique size, k, for the graph $G = (V, E)$. For this case the *greedy* for Set Covering gives a bound of $\mathcal{H}(\binom{k}{2}) \approx 2\log k$ with running time $O(n^k)$, and the *modified greedy* [9] improves the bound to $\mathcal{H}(\binom{k}{2}) - \frac{1}{6}$. In this section, we present a linear time $(\frac{1}{2} + \frac{1}{k-1})k$-approximation algorithm (H_4) for the k-clique covering problem. Although the quality of bound is worse than that derived from the Set Covering representation of the problem, the running time is significantly reduced.

Algorithm (H_4) finds a cover with star-shaped components. Phase 1 of (H_4) finds a sequence of vertices and edges, such that each edge and each vertex occur once, and such that all edges in the sequence between consecutive vertices v_i and v_{i+1} are incident with v_i. The isolated vertices are appended at the end of this sequence. In phase 2 the sequence is broken into batches of size at most $k-1$. Such a batch of $k-1$ edges and vertices induces a graph on at most k vertices, and forms a clique. The algorithm is stated as follows.

Algorithm (H_4):

Input: Graph $G = (V, E)$.

Phase 1: Set $SEQ = \emptyset$, $\bar{V} = V$, $\bar{E} = E$;

while $\bar{V} \neq \emptyset$ **do begin**

  Select a vertex $v \in \bar{V}$;

  Set $SEQ = (SEQ, \{v\})$;

  repeat

    Find a vertex $u \in V \setminus \bar{V}$ such that $(u, v) \in \bar{E}$;

    Set $SEQ = (SEQ, \{u, v\})$;

    $\bar{E} \leftarrow \bar{E} \setminus \{(u, v)\}$;

  until no such u can be found;

  $\bar{V} \leftarrow \bar{V} \setminus \{v\}$;

end;

Let $SEQ = S_1, S_2, \ldots, S_N$, where $N = |E| + |V|$.

Phase 2: Set $C^H = \emptyset$, $\bar{k} = 0$;

while $\bar{k} < N$ **do begin**

Let C denote the component with vertex set $V(C) = S_{\bar{k}+1} \cup \ldots \cup S_{\bar{k}+k-1}$ and edge set $E(C) = \{S_{\bar{k}+1}, \ldots, S_{\bar{k}+k-1}\} \cap E$;

$C^H \leftarrow C^H \cup C$;

$\bar{k} = \bar{k} + k - 1$;

end;

Output $z_H = |C^H|$ as the number of cliques used by the algorithm;

End of (H_4)**.**

The complexity and a bound on the worst-case performance of algorithm (H_4) are given in the following theorem.

Theorem 4 (H_4) *is a linear time* $(\frac{k}{2} + \frac{k}{k-1})$*-approximation algorithm.*

Proof: Phase 1 can be realized in linear time by a simple breadth-first-search. Phase 2 is obviously linear in the length of the sequence, $O(m)$.

Let z^H be the number of cliques used for covering edges by heuristic (H_4) and let z^* be the number used by an optimal clique covering. Let w denote the number of isolated vertices, let V denote the set of non-isolated vertices and E the set of edges. Then

$$z^H = \left\lceil \frac{|E| + |V| + w}{k-1} \right\rceil$$

and

$$kz^* \geq w + \sum_{v \in V} \left\lceil \frac{\delta(v)}{k-1} \right\rceil,$$

as each vertex v of degree $\delta(v)$ occurs in at least $\left\lceil \frac{\delta(v)}{k-1} \right\rceil$ cliques. The latter number is bounded from below by $w + \frac{2|E|}{k-1}$, and also by $w + |V|$. It follows that

$$\begin{aligned} z^H &\leq \frac{|E| + |V| + w}{k-1} + 1 \\ &\leq \frac{1}{2}kz^* + \frac{1}{k-1}kz^* + 1 \end{aligned}$$

Note that for $k > 4$, this gives $z^H/z^* \leq \frac{3}{4}k$, and for increasing k, the bound goes to $\frac{1}{2}k$.

∎

The bound is tight as can be seen by taking as input a graph consisting of N k-cliques. Each clique has $\frac{k(k-1)}{2}$ edges and k vertices. The algorithm partitions these edges and vertices in $\left\lceil N(\frac{k}{2} + \frac{k}{k-1}) \right\rceil$ batches of size $k-1$, whereas the optimal cover uses N cliques.

3.2 k-Clique Covering of Hyperedges of Size $k-1$

In this section, we consider the k-clique covering problem for hypergraphs that have all hyperedges of size $k-1$. Following the notation introduced earlier, this is the case $(k, k-1)$. This problem is a Set Covering problem with all sets of size $\leq k$. This is because at most k hyperedges of size $k-1$ each fit in a clique on k vertices.

The complexity of applying the *greedy* or the *modified greedy* to this Set Covering problem is $\Omega(n^k)$ since the number of sets to be considered is $\binom{n}{k}$. The purpose of algorithm (H_5) is to select a small collection of $O(m^2)$ sets to be considered.

Let the hyperedges to be covered be $\{E_1, ..., E_m\}$. Two hyperedges (elements) E_i, E_j are called a *spanning pair* if $|E_i \cup E_j| = k$. Any k-clique containing more than one hyperedge must have a spanning pair.

Algorithm (H_5):

Input: $V; E = \{E_1, ..., E_m\}$.

Phase 1: { Generate a collection of spanning pairs }

Set $S = E$; { Include all singletons in the covering sets }

for all pairs $E_i, E_j \in E$ **do**

  if $|E_i \cup E_j| = k$ **then**

    $S \leftarrow S \cup \{E_i \cup E_j\}$; {Include all spanning pairs}

Phase 2:

Apply *modified greedy* to the Set Covering problem with a collection of sets S and a collection of elements $\{E_1, ..., E_m\}$;

End of (H_5).

Theorem 5 *(H_5) is an $(\mathcal{H}(k) - \frac{1}{6})$-approximation algorithm with running time $O(m^2k + m^{2.5})$.*

Proof: Algorithm (H_5) reduces the computational complexity of the *greedy* Set Covering heuristic by limiting the number of covering sets. The Set Covering matrix developed in phase 1 has at most $\binom{m}{2} + m$ columns and m rows. To check if a hyperedge is covered by a set, we need to scan the hyperedge's vertex set which takes $O(k)$ time. Since the *modified greedy* applied in phase 2 takes only $O(m^{2.5})$ [9], the overall running time of (H_5) is $O((\binom{m}{2} + m)k + m^{2.5}) = O(m^2k + m^{2.5})$.

As observed earlier, any clique containing more than one hyperedge must have a spanning pair. The algorithm (H_5) has included all spanning pairs together with singleton sets $E_i, i = 1, ..., m$ in the collection of sets S. Thus all possible covering sets are accounted for in solving the set covering problem. Therefore, using the *modified greedy* for this reduced problem is the same as for the original problem, the worst-case bound is then $\mathcal{H}(k) - \frac{1}{6}$.

∎

Now we provide another heuristic (H_6) based on graph matching. This algorithm has better complexity than (H_5) if $k > \sqrt{m}$, but its bound quality is not as good.

Algorithm (H_6):

Step 1: Construct the following undirected graph $G = (V, E)$: Each vertex represents a hyperedge; Two vertices i and j are connected by an edge if and only if the corresponding hyperedges i and j form a spanning pair.

Step 2: Find a maximum cardinality matching in G.

Step 3: Put each pair of hyperedges corresponding to the end vertices of a matching edge obtained in step 2 in a single clique. Put the remaining unmatched vertices (hyperedges) in separate cliques. Output the collection of cliques C^H.

End of (H_6).

The following theorem gives the worst-case bound of algorithm (H_6).

Theorem 6 *Algorithm* (H_6) *is a* $\left\lceil \frac{k}{2} \right\rceil$*-approximation algorithm with running time* $O(m^{2.5})$.

Proof: Since for two hyperedges to be packed together, their corresponding vertices must be connected by an edge in G. It suffices to prove the bound for any connected component of G, so assume that G is connected.

Let t_i be the number of hyperedges packed in the ith clique of the optimal solution. These t_i hyperedges must form a t_i-clique in G. There exists a feasible matching with $\left\lfloor \frac{t_i}{2} \right\rfloor$ edges in each of these t_i-cliques. If t_i is odd, then the number of sets required to cover these nodes is less than or equal to $\left\lceil \frac{t_i}{2} \right\rceil$. Hence step 3 results in a collection of $\sum_{i=1}^{z^*} \left\lceil \frac{t_i}{2} \right\rceil$ cliques. Let $z^H = |C^H|$ and z^* be the optimal number of cliques.

Since $t_i \leq k$, $z^H \leq \sum_{i=1}^{z^*} \left\lceil \frac{t_i}{2} \right\rceil \leq z^* \left\lceil \frac{k}{2} \right\rceil$.

The complexity of the algorithm (H_6) is dominated by step 2 for finding a maximum matching. The complexity of step 2 is $O(m^{2.5})$[20]. Therefore the overall running time is $O(m^{2.5})$.

∎

Figure 4 illustrates the tightness of the bound of theorem 6. The four hyperedges on the left side can be packed in one clique of size 4. The same holds for the three hyperedges on the right side. Therefore the optimal number of cliques is 2. The maximum matching obtained by the heuristic are the three bold edges of the figure.

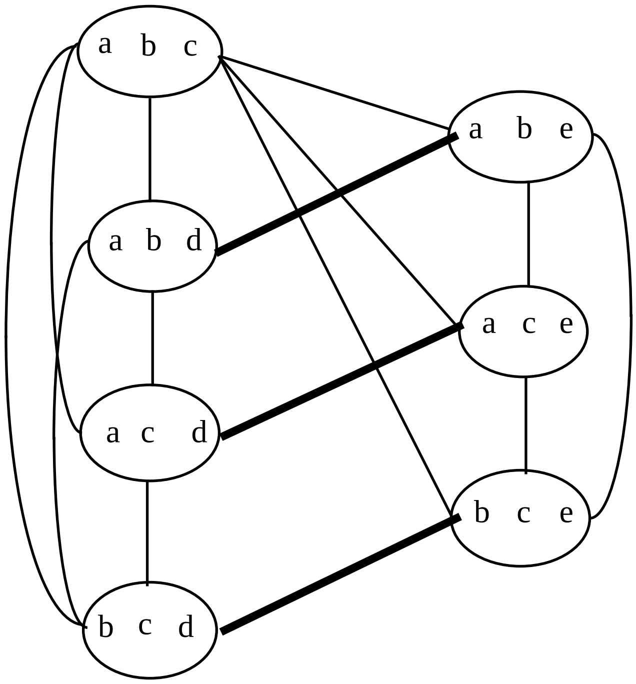

Figure 4: Illustration of the worst-case bound for $(k-1)$-hyperedges and cliques of size k

The last hyperedge $(\{a,b,c\})$ has to be packed in a clique by itself. It follows that the number of cliques used by the heuristic is 4, which is $k/2$ times the optimal number of cliques.

4 Concluding Remarks

In this paper, we introduced the problem of covering the edges of a hypergraph by k-cliques (CC_k). The problem is shown to be NP-hard and we describe a number of approximation algorithms. We identify the link between the (CC_k) problem and the Set Covering problem. The approximation algorithms described here are the first such algorithms for the k-clique covering problem. We describe a range of approximation algorithms applicable to various subclasses of the problem, with several algorithms for the same subclass offering a trade-off between algorithm's running time and quality of approximate solution.

One natural generalization of this problem is the case where each clique has a different weight. If the occurrence of such weighted case is practical, then there is a need for extending the results to the weighted case. Indeed, the Set Covering heuristic presented is immediately extendible, but such is not the case for every heuristic presented and analyzed.

Our study still leaves open the challenging task of coming up with optimal solutions to the (CC_k) problem. We believe that our approximation approach will prove instrumental and useful in algorithms that derive such optimal solutions.

References

[1] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, "Proof Verification and Intractability of Approximation Problems," In *Proc. of the 33rd IEEE Symp. on Foundations of Computer Science* (1992), Pittsburgh, PA, USA, 24-27 Oct. 1992), 14-23.

[2] M. L. Balinski, "On a Selection Problem," *Management Science*, **17:3**, (November 1970), 230-231.

[3] J. F. Bard, "A Heuristic for Minimizing the Number of Tool Switches on a Flexible Machine," *IIE Transactions* **20:4**, (1988).

[4] V. Chvátal, "A Greedy Heuristic for the Set-Covering Problem," *Math. of Oper. Res.* **4:3**, 233-235 (1979).

[5] D. A. Collier, "The Measurement and Operating Benefits of Component Part Commonality," *Decision Science*, **12:1**, (1981), 85-96.

[6] Y. Crama and A. Oerlemans, "A Column Generations Approach to Job Grouping for Flexible Manufacturing Systems," *European Journal of Operational Research*, in press.

[7] M. Daskin, P. C. Jones, and T. J. Lowe, "Rationalizing Tool Selection in a Flexible Manufacturing System for Sheet-Metal Products," *Operations Research*, **38:6**, (1990), 1104-1115.

[8] G. Gallo, M. D. Grigoriadis and R. E. Tarjan, "A Fast Parametric Maximum Flow Algorithm," *SIAM J. Comput.*, **18**, 30-55, (1989).

[9] O. Goldschmidt, D. S. Hochbaum, and G. Yu, "A Modified Greedy Heuristic for the Set Covering Problem with Improved Worst-case Bound," *Information Processing Letters*, **48**, 305–310, 1993.

[10] D. S. Hochbaum " Approximation Algorithms for the Weighted Set Covering and Node Cover Problems," GSIA Working Paper No. 64-79-80, April 1980; also, *SIAM J. Comput.*, **11:3**, 555-556, (1982).

[11] R. Hirabayashi, H. Suzuki, and N. Tsuchiya, "Optimal Tool Module Design Problem for NC Machine Tools," *Journal of the Operations Research Society of Japan*, **23:3**, (1984), 205-228.

[12] J. Holyer, "The NP-Completeness of Some Edge - Partition Problems," *SIAM J. Comput.*, **10:4**, 713-717, Nov. (1981).

[13] S. Hwang, "A Constraint-Directed Method to Solve the Part Selection Problem in Flexible Manufacturing Systems Planning Stage," Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems (1986), 297-309.

[14] S. S. Hwang and A. W. Shogan, "Modelling and Solution of an FMS Part Selection Problem," *International J. of Production Research*, **27**, 1349-1366, Aug. (1989).

[15] A. Itai and M. Rodeh, "Finding a Minimum Circuit in a Graph". *SIAM J. Computing*, **7:4**, 413–423, Aug. (1978).

[16] D. S. Johnson "Approximation Algorithms for Combinatorial Problems Problems," *Journal of Computer and System Sciences* , **9**, 256-278 (1974).

[17] L. Lovász, "On the Ratio of Optimal Integral and Fractional Covers," *Discrete Math.*, **13**, 383-390 (1975).

[18] C. Lund and M. Yanakakis, "On the Hardness of Approximating Minimization Problems," *Proc., 25th Annual ACM Symp. on Theory of Computing*, 286–293 (1993).

[19] S. Masuyama and T. Ibaraki, "Chain packing in graphs," Technical Report 87014, Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto 606, Japan. (1987).

[20] S. Micali and V.V. Vazirani, "An $O(|V|^{1/2}|E|)$ Algorithm for Finding Maximum Matching in General Graphs," *Proc. Twenty-first Annual Symposium on Foundations of Computer Science*, Long Beach, California (1980) 17–27.

[21] J. M. W. Rhys, "A Selection Problem of Shared Fixed Costs and Network Flows," *Management Science*, **17:3**, (November 1970), 200-207.

[22] C. S. Tang and E. V. Denardo, "Models Arising From a Flexible Manufacturing Machine, Part II: Minimizing the Number of Switching Instants," *Operations Research*, **36:5**, (1988), 778-784.

[23] C. Whitney, and T. Gaul, "Sequential Decision Problems for Batching and Balancing in FMSs," *Annals of Operations Research*, **3**, (1985), 301-316.

[24] G. Yu, D. Nehme and N. Nayak, "Designing and Managing Multiple Product Setups for Electronic Circuit Board Assembly Process ," IBM Internal Report, (1992).