Combinatorial Algorithms for the Maximum k-plex Problem

Benjamin McClosky, Illya V. Hicks
Department of Computational and Applied Mathematics, Rice University, 6100 Main St - MS 134, Houston, Texas 77005-1892, USA, {bjm4@rice.edu, ivhicks@rice.edu}

The maximum clique problem provides a classic framework for detecting cohesive subgraphs. However, this approach can fail to detect much of the cohesive structure in a graph. To address this issue, Seidman and Foster introduced k-plexes as a degree-based relaxation of graph completeness. More recently, Balasundaram et al. formulated the maximum k-plex problem as an integer program and designed a branch-and-cut algorithm. This paper derives a new upper bound on the cardinality of k-plexes and adapts combinatorial clique algorithms to find maximum k-plexes.

---

1. Introduction

All graphs in this paper are finite, simple, and undirected. A *complete* graph consists of pairwise adjacent vertices. A maximal complete subgraph defines a *clique*. The problem of finding maximum cardinality cliques is a classic NP-complete problem and is of fundamental importance in combinatorial optimization. The maximum clique problem has applications in ad hoc wireless networks (Chen et al. 2004), data mining (Washio and Motoda 2003), and biochemistry and genomics (Butenko and Wilhelm 2006).

Cliques also provide an intuitive approach for detecting cohesion in social networks (Festinger 1949, Luce and Perry 1949). A *social network* is a graph whose vertices consist of a set of actors and whose edges indicate relationships between actors. Cohesive subgroups are subsets of actors among whom there are relatively strong, direct, intense, frequent, or positive ties (Wasserman and Faust 1994). These subgroups are interesting because they facilitate the emergence of consensus among the actors (Friedkin 1984). In other words, members within a cohesive subgroup tend to exhibit homogeneity.

Despite the intuitive appeal of using cliques to model cohesion in social networks, cliques are in fact rarely appropriate for analysis of real data because the definition is too strict (Wasserman and Faust 1994). This motivates the study clique relaxations. Researchers

have relaxed a variety of clique properties including familiarity, reachability, and robustness (Balasundaram et al. 2007). In graph theoretic terms, these properties respectively correspond to vertex degree, path length, and connectivity. This paper focuses on a degree-based relaxation known as a k-plex (Seidman and Foster 1978).

In order to define k-plexes, consider a graph $G = (V, E)$ and vertex $v \in V$. Define $N_G(v) := \{u \in V \ : \ uv \in E\}$, $deg_G(v) := |N_G(v)|$, $\Delta(G) := max_{v \in V} \ deg_G(v)$, and $\delta(G) := min_{v \in V} \ deg_G(v)$. Let $G[K]$ denote the subgraph induced by $K \subseteq V$. Define $\bar{G} = (V, \bar{E})$ to be the complement of G, where $e \in \bar{E} \Leftrightarrow e \notin E$. In this paper, $k \geq 1$ is a positive integer.

Definition 1 (Seidman and Foster 1978)**.** *$K \subseteq V$ induces a k-plex if $\delta(G[K]) \geq |K| - k$.*

The term k-plex refers to both the set K and the subgraph $G[K]$. Notice that 1-plexes are complete subgraphs. Let $\omega_k(G)$ denote the cardinality of a largest k-plex in G.

Definition 2 (Seidman and Foster 1978)**.** *$C \subseteq V$ induces a co-k-plex if $\Delta(G[C]) \leq k - 1$.*

A co-k-plex in G is a k-plex in $\bar{G}$. Notice that co-1-plexes consist of pairwise nonadjacent vertices, or *stable sets*.

Seidman and Foster (1978) introduced k-plexes in the context of social networks. Balasundaram et al. (2006) provided an integer programming formulation for the maximum k-plex problem, derived inequalities for the k-plex polytope, and established the NP-completeness of the k-plex decision problem. McClosky and Hicks (2007) studied the co-k-plex polytope.

This paper develops combinatorial algorithms for finding maximum k-plexes. Section 2 describes heuristics for finding upper bounds on $\omega_k(G)$. Section 3 describes a heuristic for finding lower bounds on $\omega_k(G)$. Section 4 contains exact k-plex algorithms. Section 5 summarizes and discusses future research.

Computational Results. Sections 2-4 contain computational results obtained by conducting experiments on two classes of graphs. The first class of graphs consists entirely of DIMACS (2007) instances. The DIMACS graphs are widely considered the standard testbed for clique algorithms. The second class of graphs consists of real-life social networks. The COMP-GEOM-t instances are collaboration networks for computational geometers (Beebe 2002, Jones 2002). Here, two authors are adjacent if they have jointly published more than t articles. The ERDOS-x-y graphs are collaboration networks of all authors with an Erdös number at most y as of year x (Grossman et al. 1995). Data for both collaboration networks can be downloaded from the Pajek data website (Batagelj and Mrvar 2006).

Table 1: Test Instances

G	$\|V\|$	$\|E\|$	$\omega_2(G)$	$\omega_3(G)$
brock200-1	200	14834	[25,53]	–
brock200-2	200	9876	[13,24]	–
brock200-4	200	13089	[19,41]	–
brock400-2	400	59786	[27,133]	–
brock400-4	400	59765	[27,133]	–
brock800-2	800	208166	[23,253]	–
brock800-4	800	207643	[23,252]	–
c-fat200-1	200	1534	12	–
c-fat200-2	200	3235	24	–
c-fat200-5	200	8473	58	–
c-fat500-1	500	4459	14	–
c-fat500-2	500	9139	26	–
c-fat500-5	500	23191	64	–
c-fat500-10	500	46627	126	–
hamming6-2	64	1824	32	–
hamming6-4	64	704	6	–
hamming8-2	256	31616	128	–
hamming8-4	256	20864	16	–
hamming10-2	1024	518656	[512,530]	–
hamming10-4	1024	434176	[45,153]	–
johnson8-2-4	28	210	5	–
johnson8-4-4	70	1855	14	–
keller4	171	9435	15	–
MANN-a9	45	918	26	–
MANN-a27	378	70551	236	–
MANN-a45	1035	533115	[662,668]	–
p-hat300-1	300	10933	[9,66]	–
p-hat300-2	300	21928	[28,85]	–
p-hat300-3	300	33390	[43,108]	–
p-hat700-1	700	60999	[10,291]	–
p-hat700-2	700	121728	[50,298]	–
p-hat700-3	700	183010	[73,311]	–
san200-0.7-2	200	13930	–	–
san200-0.9-1	200	17910	–	–
COMP-GEOM-0	7343	11898	22	22
COMP-GEOM-1	7343	3939	10	11
COMP-GEOM-2	7343	1976	8	10
ERDOS-97-1	472	1314	8	9
ERDOS-98-1	485	1381	8	9
ERDOS-99-1	492	1417	8	9
ERDOS-97-2	5488	8972	8	9
ERDOS-98-2	5822	9505	8	9
ERDOS-99-2	6100	9939	8	9

Table 1 contains information corresponding to the testbed. The columns $\omega_2(G)$ and $\omega_3(G)$ show the largest known k-plex in G. This data is due to Balasundaram et al. (2007). All values not specified within a range are optimal. There appears to be no known values of $\omega_4(G)$ for these graphs. All implementations were tested using a 2.2 GHz Dual-Core AMD Opteron processor with 3 GB of memory.

2. Co-k-plex Coloring

This section develops an upper bound on $\omega_k(G)$ by generalizing the concept of graph coloring. A coloring of G is a function $c_m : V \mapsto \{1, ..., m\}$ such that $c_m(u) \neq c_m(v)$ whenever $uv \in E$. The *chromatic number*, $\chi(G)$, of G is the smallest integer m for which there exists a valid coloring c_m. Notice that $c_m(u) \neq c_m(v)$ for all $u, v \in K$ whenever $K \subseteq V$ induces a complete subgraph. It follows that the chromatic number is an upper bound for $\omega(G)$. That is, $\omega(G) \leq \chi(G)$.

The goal is to find a k-plex analogue of this bound. To that end, suppose the co-k-plexes $C_1, ..., C_m$ partition the vertex set, and let K be a maximum k-plex in G. The sets $C_1, ..., C_m$ define a *co-k-plex coloring* of G. Observe that

$$\omega_k(G) = |K| = \sum_{i=1}^{m} |K \cap C_i| \leq \sum_{i=1}^{m} \omega_k(G[C_i]), \tag{1}$$

where the inequality holds because k-plexes are closed under set inclusion (Seidman and Foster 1978). Let Π be the set of all co-k-plex colorings of G and define the graph invariant

$$\chi_k(G) := min\{\sum_{C \in \mathcal{C}} \omega_k(G[C]) \ : \ \mathcal{C} \in \Pi\}. \tag{2}$$

$\chi_k(G)$ is the *co-k-plex chromatic number* of G. Notice that (1) reduces to $\omega(G) \leq \chi(G)$ when $k = 1$ and $C_1, ..., C_m$ is an optimal coloring. It follows that $\chi_1(G) = \chi(G)$. Moreover, (1) and (2) together imply the bound

$$\omega_k(G) \leq \chi_k(G). \tag{3}$$

In practice, determining the exact value of $\chi_k(G)$ can be computationally prohibitive, so the remainder of this section is devoted to heuristics for approximating $\chi_k(G)$. These heuristics fall into two categories: integral and fractional. To see the distinction, let $\mathcal{I}$ be the set of all co-k-plexes in G, and let $\mathcal{I}_v$ denote the set of co-k-plexes containing v. Define $x(A) := \sum_{a \in A} x_a$. Consider the following dual pair of integer programs:

$$max\{x(V) \ : \ x \in \{0,1\}, \ x(C) \leq \omega_k(G[C]) \text{ for all } C \in \mathcal{I}\} \tag{4}$$

$$min\{\sum_{C \in \mathcal{I}} \omega_k(G[C])y_C \ : \ y \in \{0,1\}, \ y(\mathcal{I}_v) \geq 1 \text{ for all } v \in V\}. \tag{5}$$

The optimal objective values for (4) and (5) are $\omega_k(G)$ and $\chi_k(G)$, respectively. Moreover, by strong duality, the optimal objective values for the linear relaxations are equal. It follows that any feasible solution to the linear relaxation of (5) produces an upper bound on $\omega_k(G)$. Integer Co-k-plex Coloring Heuristics (ICCH) find feasible solutions to (5). Fractional Co-k-plex Coloring Heuristics (FCCH) find feasible solutions to the linear relaxation of (5). Both heuristics make use of the following lemmas which provide analytic bounds on $\omega_k(G)$.

Lemma 1. *Every graph G satisfies $\omega_k(G) \leq \Delta(G) + k$.*

Proof. If there exists a k-plex K in G such that $|K| > \Delta(G)+k$, then $deg_{G[K]}(v) \geq |K|-k > \Delta(G)$ for all $v \in K$, a contradiction. □

Lemma 2. *Given a graph G and an integer $m \geq 1$, let a_m denote the number of vertices $v \in V$ such that $deg_G(v) \geq m$. If $j := max\{m \ : \ a_m \geq k + m\}$, then $\omega_k(G) \leq k + j$.*

Proof. Suppose G contains a k-plex K such that $|K| \geq k + j + 1$. By definition of k-plex,

$$deg_{G[K]}(v) \geq |K| - k \geq j + 1 \quad \text{for all } v \in K.$$

In other words, K contains at least $k + j + 1$ vertices v such that $deg_{G[K]}(v) \geq j + 1$. It follows that $a_{j+1} \geq k + j + 1$, contradicting the definition of j. □

Lemma 3 (Balasundaram et al. 2006)**.** *If G is a co-k-plex, then $\omega_k(G) \leq 2k - 2 + k \ mod \ 2$.*

Figure 1 shows an ICCH. Line 4 stores the degree of every vertex in C_m. Line 7 uses Lemmas 1, 2, and 3 to bound $\omega_k(G[C_i])$. Lines 3, 4, 6, and 7 can each be accomplished in linear time using an adjacency matrix. It follows that this ICCH is an $\mathcal{O}(|V|^2)$ algorithm. Table 2 contains computational results obtained by running the function ICCH on the benchmark graphs with an arbitrary initial vertex ordering.

The FCCH adapts the fractional coloring procedure of Balas and Xue (1996). More precisely, the FCCH constructs a set of co-k-plexes $C_1, ..., C_h \subseteq V$ such that, after p iterations,

Table 2: ICCH Results

G	$\chi_2(G)$	seconds	$\chi_3(G)$	seconds	$\chi_4(G)$	seconds
brock200-1	93	0.0	151	0.0	169	0.0
brock200-2	55	0.0	95	0.0	118	0.0
brock200-4	78	0.0	131	0.0	151	0.0
brock400-2	172	0.1	285	0.1	332	0.1
brock400-4	168	0.1	286	0.1	330	0.1
brock800-2	248	0.3	442	0.3	570	0.3
brock800-4	247	0.3	440	0.3	557	0.3
c-fat200-1	18	0.0	20	0.0	21	0.0
c-fat200-2	34	0.0	37	0.0	38	0.0
c-fat200-5	82	0.0	89	0.0	90	0.0
c-fat500-1	19	0.0	23	0.0	24	0.0
c-fat500-2	36	0.0	41	0.0	42	0.0
c-fat500-5	85	0.1	98	0.0	99	0.0
c-fat500-10	172	0.1	191	0.1	192	0.1
hamming6-2	32*	0.0	48	0.0	56	0.0
hamming6-4	8	0.0	12	0.0	16	0.0
hamming8-2	128*	0.0	192	0.0	224	0.0
hamming8-4	32	0.0	48	0.0	64	0.0
hamming10-2	512*	0.7	768	0.7	896	0.7
hamming10-4	128	0.5	192	0.5	256	0.5
johnson8-2-4	12	0.0	16	0.0	19	0.0
johnson8-4-4	28	0.0	42	0.0	48	0.0
keller4	54	0.0	100	0.0	128	0.0
MANN-a9	38	0.0	44	0.0	45	0.0
MANN-a27	324	0.1	369	0.1	378	0.1
MANN-a45	833	1.0	1032	0.8	1035	0.8
p-hat300-1	39	0.0	69	0.0	90	0.0
p-hat300-2	76	0.0	135	0.0	173	0.0
p-hat300-3	129	0.0	210	0.0	245	0.0
p-hat700-1	76	0.1	135	0.1	184	0.1
p-hat700-2	165	0.2	289	0.1	375	0.2
p-hat700-3	267	0.3	451	0.2	556	0.2
san200-0.7-2	105	0.0	147	0.0	159	0.0
san200-0.9-1	133	0.0	184	0.0	193	0.0
COMP-GEOM-0	29	10	48	10	50	11
COMP-GEOM-1	22	9.8	37	10	37	11
COMP-GEOM-2	16	10	28	11	27	10
ERDOS971	20	0.1	32	0.1	31	0.1
ERDOS972	19	5.9	36	4.9	36	5.1
ERDOS981	18	0.1	33	0.1	32	0.1
ERDOS982	20	5.8	36	5.7	37	5.9
ERDOS991	18	0.0	34	0.1	34	0.1
ERDOS992	20	6.3	38	6.5	38	6.6

* optimal

every vertex belongs to exactly p distinct co-k-plex sets. These co-k-plexes combine to form a feasible solution to the linear relaxation of (5) as follows:

$$\bar{y} := \frac{1}{p} \sum_{i=1}^{h} y_{C_i}.$$

The feasibility of $\bar{y}$ implies that

$$\omega_k(G) \leq \frac{1}{p} \sum_{i=1}^{h} \omega_k(G[C_i]) \bar{y}_{C_i}.$$

Figure 2 shows the FCCH. The set $\mathcal{C}$ consists of the co-k-plexes $C_1, ..., C_h$. At every iteration, each vertex is either added to an existing $C_i \in \mathcal{C}$ in Line 7 or to a new partition set in Line 10. By Line 12, every vertex belongs to exactly p partition sets, so t_{new} is a valid upper bound on $\omega_k(G)$. The FCCH termination condition can be inefficient, so the number of iterations and the number partition sets in $\mathcal{C}$ are both required to be $\mathcal{O}(|V|)$.

Theorem 1. *If the number of iterations and the number of partition sets are $\mathcal{O}(|V|)$, then FCCH can be executed in $\mathcal{O}(|V|^4)$ time.*

Proof. At every iteration, for each vertex $v \in V$, the algorithm tests if $C_i \cup \{v\}$ is a co-k-plex. This can be done by counting the number of $u \in N_G(v) \cap C_i$, which requires $\mathcal{O}(|V|)$ time. Since there are $\mathcal{O}(|V|)$ partition sets, there can be $\mathcal{O}(|V|^2)$ possible pairs (C_i, v). Thus, after $\mathcal{O}(|V|)$ iterations, this step has complexity $\mathcal{O}(|V|^4)$. Lines 2 and 10 execute the $\mathcal{O}(|V|^2)$ ICCH algorithm. Since there are at most $\mathcal{O}(|V|)$ iterations, these steps have complexity $\mathcal{O}(|V|^3)$. All other operations contribute $\mathcal{O}(|V|^2)$ to the complexity. Therefore, the overall complexity of FCCH is $\mathcal{O}(|V|^4)$. □

function ICCH(V)
1. $C_i = \varnothing$ for $1 \leq i \leq |V|$
2. **for** all $u \in V$
3. $m = min\{i : C_i \cup \{u\}$ is a co-k-plex in $G\}$
4. $C_m = C_m \cup \{u\}$
5. **end**
6. Compute $j_i := max\{m \ : \ a_m \geq k + m\}$ for each C_i
7. $bound = \sum_{i=1}^{|V|} min\{\ 2k - 2 + k \ mod\ 2,\ k + j_i,\ \Delta(G[C_i]) + k,\ |C_i|\}$
8. **return** *bound*

Figure 1: An Integer Co-k-plex Coloring Heuristic

Table 3: FCCH Results

G	$\chi_2(G)$	seconds	$\chi_3(G)$	seconds	$\chi_4(G)$	seconds
brock200-1	82	0.0	139	0.1	164	0.0
brock200-2	48	0.0	89	0.0	118	0.0
brock200-4	68	0.0	122	0.0	151	0.0
brock400-2	151	0.2	267	0.2	320	0.2
brock400-4	151	0.2	265	0.3	320	0.1
brock800-2	221	1.4	401	1.7	535	1.3
brock800-4	223	2.4	410	0.8	537	1.2
c-fat200-1	16	0.0	20	0.0	21	0.0
c-fat200-2	30	0.0	37	0.0	38	0.0
c-fat200-5	76	0.0	89	0.0	90	0.0
c-fat500-1	18	0.1	23	0.0	24	0.0
c-fat500-2	33	0.1	41	0.0	42	0.0
c-fat500-5	82	0.2	98	0.1	99	0.1
c-fat500-10	166	0.4	191	0.1	192	0.2
hamming6-2	32*	0.0	48	0.0	56	0.0
hamming6-4	8	0.0	12	0.0	16	0.0
hamming8-2	128*	0.0	192	0.0	224	0.1
hamming8-4	32	0.0	48	0.0	64	0.0
hamming10-2	512*	1.3	768	1.3	896	2.0
hamming10-4	128	0.7	192	0.6	256	0.6
johnson8-2-4	10	0.0	16	0.0	18	0.0
johnson8-4-4	28	0.0	42	0.0	46	0.0
keller4	45	0.0	88	0.0	113	0.0
MANN-a9	36	0.0	44	0.0	45	0.0
MANN-a27	321	0.2	366	0.1	378	0.1
MANN-a45	803	5.9	1028	1.8	1035	1.1
p-hat300-1	34	0.0	62	0.0	85	0.0
p-hat300-2	71	0.1	129	0.0	161	0.1
p-hat300-3	115	0.2	201	0.1	240	0.1
p-hat700-1	68	0.3	123	0.3	168	0.3
p-hat700-2	146	0.7	272	0.3	349	0.5
p-hat700-3	243	1.4	428	0.8	532	0.6
san200-0.7-2	79	0.0	140	0.0	159	0.0
san200-0.9-1	127	0.0	177	0.1	191	0.1
COMP-GEOM-0	23	18	44	18	43	18
COMP-GEOM-1	15	17	28	17	30	16
COMP-GEOM-2	12	17	22	17	23	17
ERDOS971	14	0.0	25	0.0	29	0.1
ERDOS972	14	9.6	28	8.7	30	8.8
ERDOS981	12	0.1	25	0.1	28	0.1
ERDOS982	14	9.9	30	9.9	29	10
ERDOS991	12	0.1	25	0.0	28	0.1
ERDOS992	14	11	28	11	30	11

* optimal

function FCCH(V)
1. $t_{old} = \infty; p = 1$
2. t_{new} =**ICCH**(V); store the partition sets in $\mathcal{C}$
3. **while** $t_{new} < t_{old}$
4. $U = V;\ \ t_{old} = t_{new};\ \ p = p + 1$
5. **for** all $v \in U$
6. **if** $\exists\ C_i \in \mathcal{C}$ such that $v \notin C_i$ and $C_i \cup \{v\}$ is a co-k-plex
7. $C_i = C_i \cup \{v\};\ \ U = U \setminus \{v\}$
8. **end**
9. **end**
10. **ICCH**(U); append new partition sets in $\mathcal{C}$
11. Compute $j_i := max\{m\ :\ a_m \geq k + m\}$ for each $C_i \in \mathcal{C}$
12. $t_{new} = \frac{1}{p} \cdot \sum_{C_i \in \mathcal{C}} min\{\ 2k - 2 + k\ mod\ 2,\ k + j_i,\ \Delta(G[C_i]) + k,\ |C_i|\}$
13. **end**
14. **return** t_{old}

Figure 2: A Fractional Co-k-plex Coloring Heuristic

Table 3 contains computational results obtained by running the function FCCH on the benchmark graphs with an arbitrary initial vertex. The bound on the number of iterations was set to $5 \cdot |V|$.

3. A k-plex Heuristic

This section describes a heuristic for finding maximum k-plexes. Feasible k-plexes provide lower bounds on $\omega_k(G)$. The heuristic indirectly searches for cohesive subgraphs in G and extends them to maximal k-plexes.

There has been extensive research on heuristics for finding large complete subgraphs (Busygin et al. 2002, Feo and Resende 2005, Gendreau 1993, Marchiori 2002). A typical combinatorial heuristic systematically searches a set of neighborhoods in the feasible solution space for local optima (Hansen et al. 2004). When a local optimum is obtained, it is compared to the incumbent solution and stored if necessary. The heuristic then continues searching in other neighborhoods. Obviously, the solution quality heavily depends on both the choice of neighborhoods and the local search method.

Recall that if $\mathcal{I}_G$ denotes the set of all complete subgraphs in G, then $\mathcal{I}_G$ also denotes the set of all stable sets in $\bar{G}$. The remainder of this section focuses on finding stable sets in $\bar{G}$ which are extended to maximal k-plexes in G. This approach is valid because every element in $\mathcal{I}_G$ is extendible to a maximal k-plex in G. Without loss of generality, assume G

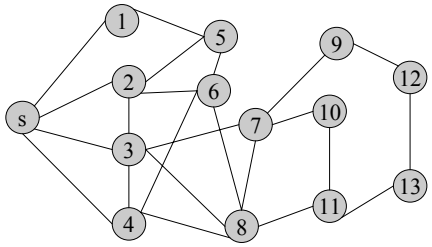

Figure 3: $\bar{H}$ with root s.

is connected. For if not, simply run the heuristic on each component.

For $u, v \in V$, let $d(u, v)$ be the length of a shortest path from u to v in G. The concept of neighborhood is based on the parity of shortest path lengths from some root node s. Given a root $s \in V$, define the following sets:

$$K_0 := \{v \in V \mid d(s, v) \text{ even}\} \quad \text{and} \quad K_1 := \{v \in V \mid d(s, v) \text{ odd}\}.$$

For example, consider the search for k-plexes in some graph H, and suppose that $\bar{H}$ is shown in Figure 3. The vertex set $V(H)$ partitions into the sets $K_0 = \{s, 5, 6, 7, 8, 12, 13\}$ and $K_1 = \{1, 2, 3, 4, 9, 10, 11\}$. For $i \in \{0, 1\}$, notice that $u, v \in K_i$ and $uv \in E(\bar{H})$ together imply $d(u, s) = d(v, s)$. Otherwise, $d(u, s)$ and $d(v, s)$ would have different parities. Therefore, for every $v \in K_i$,

$$N_{\bar{H}}(v) \cap \{u \in K_i \setminus \{v\} : d(u, s) \neq d(v, s)\} = \varnothing.$$

Hopefully, this property causes K_i to contain large stable sets.

Now $K_i \notin \mathcal{I}_H$ in general, but there will typically exist many subsets $K_i' \subseteq K_i$ such that $K_i' \in \mathcal{I}_H$. In order to examine a variety of these subsets, construct elements in $\mathcal{I}_H$ from K_i by removing one end of every edge in $\bar{H}[K_i]$. To determine which end of edge $uv \in E(\bar{H}[K_i])$ to remove, always apply exactly one of the following rules:

Rule 1. If $deg_{\bar{H}[K_i]}(v) \leq deg_{\bar{H}[K_i]}(u)$, remove u. Otherwise, remove v.

Rule 2. If $deg_{\bar{H}}(v) \leq deg_{\bar{H}}(u)$, remove u. Otherwise, remove v.

Rule 3. Always remove v.

Rule 4. Always remove u.

Let K_i^j be the subset obtained from K_i be applying Rule j to every edge in $E(\bar{H}[K_i])$. Rules 1 and 2 are greedy metrics. Rules 3 and 4 are included to diversify the search space.

Now extend each set K_i^j to a maximal k-plex in H. All k-plexes that can be constructed from a set K_i in this way constitute a neighborhood. Therefore, the search space is essentially

function lbound($\mathcal{R}$)
1. **for** all $s \in \mathcal{R}$
2. define K_0 and K_1 with respect to root s
3. construct sets $K_i^j \subseteq K_i$
4. extend sets K_i^j to maximal k-plexes in H
5. **for** all j and i
6. **kick**(K_i^j)
7. **end**
8. update incumbent I if necessary
9. **end**

function kick(K)
10. construct set $S := \{v \in V \setminus K \ : \ |N_{\bar{H}}(v) \cap K| \leq 1\}$
11. let $K = K \cup S$
12. construct sets $K^j \subseteq K$
13. extend sets K^j to maximal k-plexes in H
14. **end**

Figure 4: k-plex heuristic.

a function of the root nodes, and specifying a set of neighborhoods is equivalent to specifying a set of root nodes $\mathcal{R}$. The k-plex heuristic is shown in Figure 4. The incumbent solution I is initially empty and stored as a global variable.

To find a k-plex in H, arbitrarily choose a set of vertices to define $\mathcal{R}$. Line 2 builds a breadth-first-search tree in $\bar{H}$ rooted at s to determine $d(v, s)$ for all $v \in V$. The breadth-first-search tree is also used to define $deg_{\bar{H}[K_i]}(v)$ for all v. Line 3 applies Rules 1-4, and Line 4 uses a greedy heuristic. Line 6 passes the sets K_i^j to the new function **kick**. Its purpose is to help the heuristic escape local optima. Line 12 uses Rules 1-4 for each input set K. Figure 4 is a basic k-plex heuristic. Table 4 contains computational results obtained by running **lbound** on the benchmark graphs. **LB1** corresponds to choosing an arbitrary set of $\frac{|V|}{40}$ vertices to define $\mathcal{R}$. **LB2** corresponds to setting $\mathcal{R} = V$. The heuristics terminate after an hour time limit.

4. Exact Algorithms

This section describes exact algorithms for finding maximum k-plexes in a graph $G = (V, E)$. The first type is based on a standard clique algorithm (Applegate and Johnson 1993, Carraghan and Pardalos 1990). The second type adapts an algorithm of Östergård (2002).

Table 4: **lbound** Results

	LB1		**LB2**		**LB1**		**LB2**		**LB1**		**LB2**	
G	$\omega_2(G)$	sec.	$\omega_2(G)$	sec.	$\omega_3(G)$	sec.	$\omega_3(G)$	sec.	$\omega_4(G)$	sec.	$\omega_4(G)$	sec.
brock200-1	25	1	25	3	27	1	28	3	31	1	32	3
brock200-2	12	1	13*	5	15	1	15	6	17	1	17	5
brock200-4	18	1	19	4	22	1	23	4	24	1	25	4
brock400-2	27	2	28	23	31	2	32	23	35	2	36	23
brock400-4	33	2	33	23	33	2	33	23	36	2	37	23
brock800-2	22	15	23	299	26	15	26	298	29	15	30	299
brock800-4	22	15	23	301	25	15	26	301	29	15	30	301
c-fat200-1	12*	2	12*	10	12*	2	12*	10	12*	2	12*	10
c-fat200-2	24*	2	24*	9	24*	2	24*	9	24*	2	24*	10
c-fat200-5	58*	2	58*	7	58*	2	58*	7	58*	2	58*	7
c-fat500-1	14*	20	14*	225	14*	19	14*	226	14*	19	14*	226
c-fat500-2	26*	19	26*	210	26*	18	26*	210	26*	19	26*	212
c-fat500-5	64*	16	64*	191	64*	16	64*	185	64*	16	64*	194
c-fat500-10	126*	13	126*	146	126*	13	126*	150	126*	13	126*	152
hamming6-2	32*	0	32*	0	32*	0	32*	0	32	0	32	0
hamming6-4	4	0	4	0	8*	0	8*	0	8	0	8	0
hamming8-2	128*	0	128*	2	128*	0	128*	2	128	0	128	2
hamming8-4	16*	1	16*	8	16	1	16	8	16	1	16	8
hamming10-2	512*	3	512*	65	512	3	512	67	512	3	512	74
hamming10-4	43	12	43	281	64	12	64	288	63	12	64	297
johnson8-2-4	4	0	4	0	8*	0	8*	0	9*	0	9*	0
johnson8-4-4	14	0	14	0	14	0	14	0	14	0	14	0
keller4	15*	1	15*	3	18	1	18	2	20	1	20	2
MANN-a9	22	0	22	0	30	0	30	0	36*	0	36*	0
MANN-a27	218	2	218	14	258	3	260	29	250	2	257	17
MANN-a45	646	35	646	859	762	76	762	1748	756	21	756	540
p-hat300-1	9	4	9	28	11	4	11	28	12	4	13	28
p-hat300-2	30	3	30	20	34	3	34	19	39	3	39	20
p-hat300-3	42	1	43	10	49	1	49	10	53	2	55	11
p-hat700-1	10	33	12	537	13	33	14	555	16	32	16	529
p-hat700-2	50	19	51	316	58	19	58	320	65	19	66	321
p-hat700-3	70	8	71	140	82	9	84	140	92	9	95	141
san200-0.7-2	26	1	26	3	36	1	36	4	48	1	48	4
san200-0.9-1	90	0	90	2	125*	0	125*	2	125	0	125	2
COMP-GEOM-0	16	1439	22*	3600	16	1551	22*	3600	16	1545	22	3600
COMP-GEOM-1	8	1489	10*	3600	8	1597	11*	3600	8	1531	11	3600
COMP-GEOM-2	8	1529	8*	3600	8	1543	8	3600	8	1468	8	3600
ERDOS971	8*	18	8*	3600	9*	17	9*	3600	10	17	10	3600
ERDOS972	8*	2506	8*	3506	9*	2433	9*	3600	10	2483	10	3600
ERDOS981	8*	17	8*	3600	9*	17	9*	3600	10	17	10	3600
ERDOS982	8*	2473	8*	3600	9*	2698	9*	3600	10	2632	10	3600
ERDOS991	8*	18	8*	3600	9*	18	9*	3600	10	18	10	3600
ERDOS992	8*	2103	8*	3600	9*	2319	9*	3600	10	2746	10	3600

* optimal

function basicClique(U, K)
1. **while** $U \neq \varnothing$
2. **if** $|K| + |U| \leq \ max$
3. **return**
4. **end**
5. $U = U \setminus \{v\}$ for some $v \in U$
6. **basicClique**$(U \cap N_G(v),\ K \cup \{v\})$
7. **end**
8. **if** $|K| > max$
9. $max = |K|$
10. **end**
11. **return**

Figure 5: Basic Clique Algorithm

4.1. Algorithm Type 1

Consider the standard clique algorithm shown in Figure 5. At any point, the algorithm is constructing a complete graph K. The candidate set, $U \subseteq V \setminus K$, contains all vertices v such that $K \cup \{v\}$ is complete. In other words, $U := \bigcap_{v \in K} N_G(v)$. The global variable max stores the cardinality of the largest clique found. To find a maximum clique, initialize $max = 0$ and make the function call **basicClique**$(V, \varnothing)$.

This clique algorithm generalizes to find maximum k-plexes. The main difference is that the candidate set U for a k-plex K is no longer $\bigcap_{v \in K} N_G(v)$. It is now defined as

$$U := \{v \in V \setminus K \ : \ K \cup \{v\} \text{ is a } k\text{-plex}\}.$$

Figure 6 shows the basic k-plex algorithm. To find a maximum k-plex, initialize $max = 0$ and make the function call **basicPlex**$(V, \varnothing)$. Table 5 contains computational results obtained by running **basicPlex** on the benchmark graphs with a one hour time limit.

Without Lines 2-4, the clique algorithm examines every clique in G. Recall that G can contain an exponential, with respect to $|V|$, number of cliques (Moon and Moser 1965). Lines 2-4 attempt to avoid enumeration of an exponential number of subgraphs. This is known as pruning the search tree. Although there may exist graphs which require the enumeration of an exponential number of cliques, pruning can reduce the runtime.

The basic clique algorithm has many variants (Régin 2003, Sewell 1998, Tomita and Seki 2003, Wood 1997). Many researchers focus on improving the pruning strategy using the coloring bound. A coloring heuristic provides an upper bound on $\omega(G[U])$ and has the

Table 5: **basicPlex** Results

G	$\omega_2(G)$	seconds	BBN	$\omega_3(G)$	seconds	BBN	$\omega_4(G)$	seconds	BBN
brock200-1	25	≥3600	172822699	28	≥3600	182056437	31	≥3600	180633250
brock200-2	13*	166	9759381	15	≥3600	199178608	17	≥3600	192281927
brock200-4	20	≥3600	193074734	22	≥3600	199289654	25	≥3600	193677120
brock400-2	27	≥3600	169761253	31	≥3600	162264447	34	≥3600	155153487
brock400-4	27	≥3600	160979618	32	≥3600	146807899	36	≥3600	145283598
brock800-2	23	≥3600	134201916	25	≥3600	139748190	28	≥3600	124918468
brock800-4	23	≥3600	133857528	24	≥3600	144247348	27	≥3600	127834010
c-fat200-1	12*	0	975	12*	4	58324	12*	170	2025883
c-fat200-2	24*	0	7308	24*	5	115832	24*	226	3827925
c-fat200-5	58*	3112	86024721	58	≥3600	104935293	58	≥3600	108945815
c-fat500-1	14*	1	2712	14*	115	364617	14	≥3600	6098258
c-fat500-2	26*	1	31068	26*	126	818322	26	≥3600	14272751
c-fat500-5	64	≥3600	84968699	64	≥3600	94102915	64	≥3600	92359122
c-fat500-10	126	≥3600	39813170	126	≥3600	45937780	126	≥3600	45046647
hamming6-2	32*	506	26461612	32	≥3600	244753572	37	≥3600	261840105
hamming6-4	6*	0	4709	8*	1	71069	10*	9	849851
hamming8-2	128	≥3600	39716014	128	≥3600	43138327	128	≥3600	50079738
hamming8-4	16	≥3600	237558610	18	≥3600	222683938	22	≥3600	230542048
hamming10-2	512	≥3600	3595516	512	≥3600	3790553	512	≥3600	3877853
hamming10-4	32	≥3600	146893539	43	≥3600	132802297	64	≥3600	54961531
johnson8-2-4	5*	0	1666	8*	0	12837	9*	0	104984
johnson8-4-4	14*	110	11542436	18	≥3600	350491163	22	≥3600	342248079
keller4	15	≥3600	247583422	20	≥3600	207711375	22	≥3600	258859895
MANN-a9	26*	66	5585820	36*	2	106834	36*	278	25470013
MANN-a27	234	≥3600	79044110	351	≥3600	7146812	351	≥3600	10158168
MANN-a45	660	≥3600	19339018	990	≥3600	1022834	990	≥3600	1283088
p-hat300-1	10*	14	665249	12*	1111	40704167	14	≥3600	128042727
p-hat300-2	29	≥3600	167764775	35	≥3600	162883168	41	≥3600	154569677
p-hat300-3	42	≥3600	145501695	51	≥3600	145528614	57	≥3600	139965092
p-hat700-1	13*	1887	55769755	14	≥3600	110462323	16	≥3600	98797915
p-hat700-2	49	≥3600	116066244	58	≥3600	116785628	65	≥3600	117405227
p-hat700-3	69	≥3600	105454118	81	≥3600	105553105	93	≥3600	97553599
san200-0.7-2	24	≥3600	441219398	36	≥3600	395072520	48	≥3600	330336652
san200-0.9-1	90	≥3600	107493877	125	≥3600	35590748	125	≥3600	39843163
COMP-GEOM-0	21	≥3600	8346	22	≥3600	4672	22	≥ 3600	6269
COMP-GEOM-1	10	≥3600	754265	11	≥3600	313457	12	≥ 3600	21132
COMP-GEOM-2	8	≥3600	647635	10	≥3600	423436	11	≥ 3600	8674
ERDOS-97-1	8*	10	1390	9*	1122	198148	4	≥ 3600	339906
ERDOS-97-2	3	≥3600	737	3	≥3600	480	4	≥ 3600	470
ERDOS-98-1	8*	9	1424	9*	1250	204612	4	≥ 3600	303239
ERDOS-98-2	3	≥3600	650	3	≥3600	420	4	≥ 3600	410
ERDOS-99-1	8*	10	1463	9*	1323	211967	4	≥ 3600	292341
ERDOS-99-2	3	≥3600	591	3	≥ 3600	376	4	≥ 3600	369

* optimal

potential to prune a larger portion of the search tree because $\chi(G[U]) \leq |U|$. Figure 7 shows an algorithm which uses co-k-plex coloring to prune the search tree.

Let k-**plex1a** and k-**plex1b** denote the functions obtained by using ICCH and FCCH, respectively, to execute Line 2 of k-**plex1**. ICCH and FCCH are discussed in Section 2. To find a maximum k-plex in G, run **LB1** to find an initial value for max and make the function call to k-**plex1a**$(V, \varnothing)$ or k-**plex1b**$(V, \varnothing)$. Tables 6 and 7 contain computational results obtained by running these algorithms on the benchmark graphs with a one hour time limit.

4.2. Algorithm Type 2

The algorithm in this subsection is based on the following idea of Östergård (2002). Let $V = \{v_1, ..., v_n\}$ and $S_i = \{v_i, ..., v_n\}$. The algorithm in Figure 5 searches for the largest clique in S_1 containing v_1, the largest clique in S_2 containing v_2, and so on. Östergård suggests reversing this order. That is, search S_n for the largest clique containing v_n, S_{n-1} for the largest clique containing v_{n-1}, and so on. Let $c(i)$ be the size of the largest clique in S_i. Clearly, $c(n) = 1$ and $c(1) = \omega(G)$. Moreover, $c(i) \in \{c(i+1), c(i+1)+1\}$ for $i = 1, ..., n-1$.

Figure 8 shows Östergård's clique algorithm. The search order allows for the following pruning strategy. Given candidate set U, let $i = min\{j \ : \ v_j \in U\}$. Notice that $U \subseteq S_i$ and hence $\omega_k(G[U]) \leq c(i)$. This new bound is used in Line 10 in Figure 8.

Östergård's algorithm adapts to find maximum k-plexes with two modifications. First,

function basicPlex(U, K)
1. **while** $U \neq \varnothing$
2. **if** $|K| + |U| \leq \ max$
3. **return**
4. **end**
5. $K = K \cup \{v\}$; $U = U \setminus \{v\}$ for some $v \in U$
6. $U' := \{u \in U : K \cup \{u\}$ is a k-plex$\}$
7. **basicPlex**(U', K)
8. **end**
9. **if** $|K| > max$
10. $max = |K|$
11. **end**
12. **return**

Figure 6: Basic k-plex Algorithm

Table 6: k-**plex1a** Results

G	$\omega_2(G)$	seconds	BBN	$\omega_3(G)$	seconds	BBN	$\omega_4(G)$	seconds	BBN
brock200-1	25	≥3600	86030174	28	≥3600	95147581	31	≥3600	97664910
brock200-2	13*	289	8663613	15	≥3600	100542983	17	≥3600	90143057
brock200-4	19	≥3600	102282008	22	≥3600	106907189	25	≥3600	98966956
brock400-2	27	≥3600	95110220	31	≥3600	88619566	35	≥3600	81344425
brock400-4	33	≥3600	51472394	33	≥3600	74724985	36	≥3600	80938337
brock800-2	23	≥3600	94857693	26	≥3600	81559009	29	≥3600	71154628
brock800-4	23	≥3600	96369941	25	≥3600	90097273	29	≥3600	72735746
c-fat200-1	12*	2	873	12*	27	57730	12*	922	1960167
c-fat200-2	24*	3	5269	24*	28	109070	24*	909	3385277
c-fat200-5	58	≥3600	19298000	58	≥3600	24247513	58	≥3600	28799458
c-fat500-1	14*	32	2293	14*	1580	357420	14	≥3600	545917
c-fat500-2	26*	33	20601	26*	1617	787649	26	≥3600	1224327
c-fat500-5	64	≥3600	14631858	64	≥3600	21078075	64	≥3600	24930408
c-fat500-10	126	≥3600	5104395	126	≥3600	7643111	126	≥3600	9068983
hamming6-2	32*	0	0	32	≥3600	92535097	37	≥3600	119263227
hamming6-4	6*	0	3668	8*	1	59533	10*	15	701425
hamming8-2	128*	0	0	128	≥3600	14422543	128	≥3600	20007766
hamming8-4	16	≥3600	158903409	18	≥3600	149846604	22	≥3600	157055208
hamming10-2	512*	1	0	512	≥3600	434296	512	≥3600	456014
hamming10-4	43	≥3600	44661342	64	≥3600	21254123	64	≥3600	33084666
johnson8-2-4	5*	0	1585	8*	0	12378	9*	1	104804
johnson8-4-4	14*	138	7755953	18	≥3600	191111049	22	≥3600	172931195
keller4	15	≥3600	147002319	20	≥3600	128108327	22	≥3600	169102280
MANN-a9	26*	123	4111457	36*	6	102896	36*	739	25470013
MANN-a27	234	≥3600	78820556	351	≥3600	383569	351	≥3600	781334
MANN-a45	660	≥3600	18866263	990	≥3600	18029	990	≥3600	53068
p-hat300-1	10*	33	562727	12*	2827	39631513	14	≥3600	54618899
p-hat300-2	30	≥3600	77146967	35	≥3600	84162645	41	≥3600	84779804
p-hat300-3	42	≥3600	73906134	50	≥3600	70787196	57	≥3600	70408792
p-hat700-1	13*	3186	46290951	14	≥3600	56967864	16	≥3600	43437614
p-hat700-2	50	≥3600	51487369	58	≥3600	56760785	65	≥3600	61159187
p-hat700-3	70	≥3600	42921661	82	≥3600	46670755	92	≥3600	48056462
san200-0.7-2	26	≥3600	247380139	36	≥3600	368232192	48	≥3600	301494773
san200-0.9-1	90	≥3600	64534714	125	≥3600	5514998	125	≥3600	6899702
COMP-GEOM-0	22*	1218	3301	22	≥3600	18785	22	≥3600	173911
COMP-GEOM-1	10*	1205	2366	11	≥3600	110109	11	≥3600	122167
COMP-GEOM-2	8*	1220	1211	8	≥3600	14166	8	≥3600	6208
ERDOS-97-1	8*	313	954	9	≥ 3600	5742	10	≥ 3600	5571
ERDOS-97-2	8	≥ 3600	2698	9	≥ 3600	1494	10	≥ 3600	1685
ERDOS-98-1	8*	427	1016	9	≥ 3600	6008	10	≥ 3600	5876
ERDOS-98-2	8	≥ 3600	2359	9	≥ 3600	1086	10	≥ 3600	1771
ERDOS-99-1	8*	406	472	9	≥ 3600	6248	10	≥ 3600	5135
ERDOS-99-2	8	≥ 3600	2913	9	≥ 3600	943	10	≥ 3600	1544

* optimal

Table 7: k-**plex1b** Results

G	$\omega_2(G)$	seconds	BBN	$\omega_3(G)$	seconds	BBN	$\omega_4(G)$	seconds	BBN
brock200-1	25	≥3600	10054924	28	≥3600	9610060	31	≥3600	11595841
brock200-2	13*	1778	7722362	15	≥3600	15728716	17	≥3600	15883075
brock200-4	19	≥3600	12056663	22	≥3600	11842251	24	≥3600	13308843
brock400-2	27	≥3600	3298972	31	≥3600	2506819	35	≥3600	3084792
brock400-4	33	≥3600	2879018	33	≥3600	2769768	36	≥3600	3339363
brock800-2	22	≥3600	847946	26	≥3600	687075	29	≥3600	719475
brock800-4	22	≥3600	846155	25	≥3600	708185	29	≥3600	771036
c-fat200-1	12*	3	802	12*	41	57612	12*	1397	1958425
c-fat200-2	24*	3	2050	24*	60	96754	24	≥3600	1586893
c-fat200-5	58*	836	444241	58	≥3600	3960477	58	≥3600	5319030
c-fat500-1	14*	34	2060	14*	1796	356744	14	≥3600	427000
c-fat500-2	26*	39	10177	26*	2368	754350	26	≥3600	797974
c-fat500-5	64*	1579	557081	64	≥3600	2800459	64	≥3600	3453142
c-fat500-10	126	≥3600	327032	126	≥3600	974543	126	≥3600	1193712
hamming6-2	32*	0	0	32	≥3600	6160422	36	≥3600	44987310
hamming6-4	6*	0	3380	8*	3	58663	10*	33	693982
hamming8-2	128*	0	0	128	≥3600	636791	128	≥3600	2835717
hamming8-4	16	≥3600	9945892	18	≥3600	9748498	18	≥3600	10199713
hamming10-2	512*	1	0	512	≥3600	7508	512	≥3600	29602
hamming10-4	43	≥3600	323816	64	≥3600	370717	64	≥3600	236250
johnson8-2-4	5*	0	1585	8*	0	12337	9*	2	104679
johnson8-4-4	14*	475	6389736	18	≥3600	48544486	22	≥3600	52307238
keller4	15	≥3600	18263136	20	≥3600	17714412	22	≥3600	20281149
MANN-a9	26*	395	3240597	36*	15	100969	36*	1733	25470013
MANN-a27	234	≥3600	3053292	351	≥3600	78758	351	≥3600	253074
MANN-a45	660	≥3600	28446	990	≥3600	7035	990	≥3600	20406
p-hat300-1	10*	139	500766	12	≥3600	12816637	14	≥3600	12577276
p-hat300-2	30	≥3600	7335988	35	≥3600	6439794	40	≥3600	8083792
p-hat300-3	42	≥3600	5279160	50	≥3600	5693325	57	≥3600	5957872
p-hat700-1	12	≥3600	2617164	13	≥3600	2557821	16	≥3600	2516929
p-hat700-2	50	≥3600	1176955	58	≥3600	1407568	65	≥3600	1459008
p-hat700-3	70	≥3600	750652	82	≥3600	1007772	92	≥3600	1001573
san200-0.7-2	26	≥3600	12473403	36	≥3600	16047112	48	≥3600	16009322
san200-0.9-1	90	≥3600	9748246	125	≥3600	970390	125	≥3600	1705032
COMP-GEOM-0	22*	2202	2797	22	≥3600	15368	22	≥3600	71325
COMP-GEOM-1	10*	2075	2013	11	≥3600	10073	11	≥3600	20237
COMP-GEOM-2	8*	2731	1193	8	≥3600	4102	8	≥3600	6028
ERDOS-97-1	8*	370	869	9	≥ 3600	4928	10	≥ 3600	4961
ERDOS-97-2	8	≥ 3600	574	9	≥ 3600	702	10	≥ 3600	671
ERDOS-98-1	8*	414	953	9	≥ 3600	4497	10	≥ 3600	4506
ERDOS-98-2	8	≥ 3600	449	9	≥ 3600	492	10	≥ 3600	487
ERDOS-99-1	8*	438	989	9	≥ 3600	4379	10	≥ 3600	4376
ERDOS-99-2	8	≥ 3600	592	9	≥ 3600	633	10	≥ 3600	615

* optimal

function k-**plex1**(U, K)
1. **while** $U \neq \varnothing$
2. Compute $\tilde{\chi}_k(G[U]) \geq \chi_k(G[U])$
3. **if** $|K| + \tilde{\chi}_k(G[U]) \leq\ max$
4. **return**
5. **end**
6. $K = K \cup \{v\};\ U = U \setminus \{v\}$ for some $v \in U$
7. $U' := \{u \in U :\ K \cup \{u\}$ is a k-plex$\}$
8. k-**plex1**$(U',\ K)$
9. **end**
10. **if** $|K| > max$
11. $max = |K|$
12. **end**
13. **return**

Figure 7: k-plex Algorithm

define $c_k(i) = \omega_k(G[S_i])$. Second, the candidate set with respect to K is defined as

$$U := \{v \in V \setminus K\ :\ K \cup \{v\} \text{ is a } k\text{-plex}\}.$$

Figure 9 shows k-**plex2**. Table 8 contains computational results obtained by running k-**plex2** on the benchmark graphs with a one hour time limit.

5. Conclusions and Future Work

This paper describes combinatorial algorithms for finding maximum k-plexes in a graph. Section 2 focuses on co-k-plex coloring heuristics which are used as an upper bound on the k-plex number. Section 3 discusses a heuristic for finding maximum k-plexes. This heuristic provides a lower bound on the k-plex number. Section 4 describes exact algorithms for finding maximum k-plexes. Table 9 summarizes the number of instances solved to optimality by each exact algorithm.

The first three exact algorithms perform similarly within the hour time limit. Although this type of algorithm appears to benefit from the upper and lower bound heuristics, they solve a relatively small number of instances to optimality. This suggests that the coloring heuristics might not produce tight upper bounds, so an interesting avenue for future work would be to develop stronger coloring heuristics.

function OsterClique(U, K)
1. **if** $|U| = 0$
2. **if** $|K| > max$
3. $max = |K|$
4. found=true
5. **end**
6. **return**
7. **end**
8. **while** $U \neq \varnothing$
9. **if** $|K| + |U| \leq max$
10. **return**
11. **end**
12. $i = min\{j \ : \ v_j \in U\}$
13. **if** $|K| + c(i) \leq max$
14. **return**
15. **end**
16. $U = U \setminus \{v_i\}$
17. **OsterClique**$(U \cap N_G(v_i),\ K \cup \{v_i\})$
18. **if** found=true
19. **return**
20. **end**
21. **end**
22. **return**

function findClique
23. $max = 0$
24. **for** $i = n$ down to 1
25. $found = false$
26. **OsterClique**$(S_i \cap N_G(v_i), \{v_i\})$
27. **end**
28. $c(i) = max$
29. **return**

Figure 8: Östergård's Clique Algorithm

```
function OsterPlex(U, K)
1.   if |U| = 0
2.     if |K| > max
3.       max = |K|
4.       found=true
5.     end
6.     return
7.   end
8.   while U ≠ ∅
9.     if |K| + |U| ≤ max
10.      return
11.    end
12.    i = min{j : v_j ∈ U}
13.    if |K| + c_k(i) ≤ max
14.      return
15.    end
16.    K = K ∪ {v_i}; U = U \ {v_i}
17.    U' := {u ∈ U : K ∪ {u} is a k-plex}.
18.    OsterPlex(U', K)
19.    if found=true
20.      return
21.    end
22.  end
23.  return

function k-plex2
24.  max = 0
25.  for i = n down to 1
26.    found = false
27.    OsterPlex(S_i, {v_i})
28.  end
29.  c_k(i) = max
30.  return
```

Figure 9: Östergård's Algorithm Adapted for k-plexes

A natural approach for improving the co-k-plex coloring heuristics would be to generalize the DSATUR graph coloring heuristic (Brélaz's 1979). The DSATUR heuristic dynamically determines the order in which vertices are colored. More precisely, it maintains the number of distinct colors adjacent to each uncolored vertex and always colors the vertex with the highest number of adjacent color classes. The number of adjacent colors classes defines a vertex's *saturation degree*. This idea generalizes to co-k-plex coloring. The saturation degree of v is redefined as the number of distinct partition sets C_i such that $C_i \cup \{v\}$ is not a

Table 8: k-**plex2** Results

G	$\omega_2(G)$	seconds	BBN	$\omega_3(G)$	seconds	BBN	$\omega_4(G)$	seconds	BBN
brock200-1	23	≥3600	983266826	24	≥3600	1061715139	27	≥3600	1090413176
brock200-2	13*	74	19636408	15	≥3600	973470713	17	≥3600	1118582507
brock200-4	19	≥3600	917681365	20	≥3600	1078546865	21	≥3600	1158926695
brock400-2	23	≥3600	996101972	27	≥3600	1112675498	29	≥3600	1152442843
brock400-4	23	≥3600	989222371	22	≥3600	1132833870	30	≥3600	1159027166
brock800-2	21	≥3600	926436143	20	≥3600	1077097050	23	≥3600	1142792112
brock800-4	20	≥3600	919220732	22	≥3600	1044131631	24	≥3600	1109841414
c-fat200-1	12*	0	3677	12*	0	123687	12*	19	4452622
c-fat200-2	24*	0	1895	24*	0	30227	24*	2	759423
c-fat200-5	58*	0	760	58*	0	6566	58*	0	88301
c-fat500-1	14*	0	19733	14*	8	1183127	14*	1416	134845416
c-fat500-2	26*	0	10081	26*	2	266957	26*	91	12812559
c-fat500-5	64*	0	4195	64*	0	52676	64*	6	956345
c-fat500-10	126*	0	2141	126*	0	18391	126*	2	221427
hamming6-2	32*	0	396	32*	1	195054	40*	872	226208834
hamming6-4	6*	0	4526	8*	0	37113	10*	1	395729
hamming8-2	128*	0	7448	128*	2744	182864232	112	≥3600	808066329
hamming8-4	16*	47	7982728	17	≥3600	1053193592	19	≥3600	1045849871
hamming10-2	512*	33	147919	448	≥3600	208033053	430	≥3600	799045131
hamming10-4	41	≥3600	595181620	46	≥3600	1058346589	51	≥3600	1042679457
johnson8-2-4	5*	0	2621	8*	0	10472	9*	0	151028
johnson8-4-4	14*	0	40896	18*	39	14014988	21	≥3600	1160638581
keller4	15*	1000	284120617	21	≥3600	909167108	16	≥3600	1183954679
MANN-a9	26*	0	53102	36*	1	375502	36*	129	36511315
MANN-a27	235	≥3600	33275514	351	≥3600	23112644	351	≥3600	93317993
MANN-a45	661	≥3600	3326395	990	≥3600	2840051	990	≥3600	16526869
p-hat300-1	10*	6	1561119	12*	552	128854637	13	≥3600	1001501766
p-hat300-2	29	≥3600	704387512	29	≥3600	909116307	33	≥3600	987023800
p-hat300-3	34	≥3600	850772303	37	≥3600	956121775	36	≥3600	1050444550
p-hat700-1	13*	667	90760266	13	≥3600	687249461	13	≥3600	964257683
p-hat700-2	37	≥3600	655931022	39	≥3600	804657901	42	≥3600	937295855
p-hat700-3	51	≥3600	820473217	43	≥3600	991432587	44	≥3600	1063865458
san200-0.7-2	24	≥3600	1079696544	36	≥3600	940350668	48	≥3600	862621323
san200-0.9-1	90	≥3600	485362621	125*	253	64534321	50	≥3600	1006342152
COMP-GEOM-0	22*	397	2150878	5	≥3600	156104820	4	≥3600	265372393
COMP-GEOM-1	10*	1118	4875730	4	≥3600	172334737	4	≥3600	267076112
COMP-GEOM-2	8*	1145	5034335	4	≥3600	162641029	4	≥3600	266562165
ERDOS-97-1	8*	0	25770	9*	19	2391684	11*	1897	256331821
ERDOS-97-2	8*	1253	12602187	3	≥3600	216016439	4	≥3600	331758265
ERDOS-98-1	8*	0	27246	9*	20	2552046	11*	1675	212106816
ERDOS-98-2	8*	1514	14265207	3	≥3600	209965048	4	≥3600	318616472
ERDOS-99-1	8*	0	28035	9*	21	2632545	11*	1783	223030325
ERDOS-99-2	8*	1757	15748963	3	≥3600	205078469	4	≥3600	308498817

* optimal

co-k-plex. This general version of DSATUR was implemented and tested (McClosky 2008). However, the results did not improve upon ICCH and FCCH.

The final exact algorithm, k-**plex2**, dominates the Type 1 algorithms with respect to number of instances solved to optimality. Moreover, k-**plex2** converges quickly, when it converges at all. However, the final solution can be far from optimal when k-**plex2** fails to converge. The algorithm demonstrates this phenomenon on the collaboration networks for $k = 3, 4$. The reason for this behavior is that the algorithm can spend much of its time optimizing over a subset of V. Consequently, if vertices at the end of the vertex ordering are needed to make large k-plexes, good solutions cannot be found until the entire vertex set is

Table 9: Results Summary

Algorithm	$k = 2$	$k = 3$	$k = 4$	Total
basicPlex	16	11	5	32
k**-plex1a**	20	8	5	33
k**-plex1b**	21	7	4	32
k**-plex2**	28	18	14	60

processed. This illustrates the importance of vertex ordering for this type of algorithm.

Type 1 algorithms spend time at each branch and bound node to approximate $\chi_k(G[U])$ for the candidate set U. Unfortunately, $\chi_k(G)$ could be an inaccurate bound on $\omega_k(G)$ in general. The k**-plex2** algorithm spends no time estimating $\chi_k(G)$ but benefits from the bound obtained using the c_k array. In any case, the purpose of these bounds is to prune the candidate set. The requirement for membership in the candidate set becomes less stringent as k increases, so the set becomes harder to prune. This contributes to the increase in runtime as k grows.

When comparing these results to those found in Balasundaram et al. (2006), the combinatorial algorithms outperform branch-and-cut on the DIMACS graphs. On the other hand, branch-and-cut works better on the larger social network graphs. This suggests that combinatorial methods are superior for graphs on a few hundred vertices, but branch-and-cut becomes the preferred method as the size of the graphs grow.

Another area for future research is an exact co-k-plex coloring algorithm. The co-k-plex chromatic number is unknown for most of the benchmark graphs. Therefore, it is difficult to evaluate the performance of the co-k-plex coloring heuristics in Section 2. It would also be beneficial to study additional heuristics for both the upper and lower bounds on $\omega_k(G)$.

Acknowledgments

This research was partially supported by NSF grants DMI-0521209 and DMS-0729251. Svyatoslav Trukhanov has independently generalized Östergård's clique algorithm to find maximum k-plexes (Trukhanov 2008).

References

Applegate, D. and Johnson, D. 1993. dfmax.c. ftp://dimacs.rutgers.edu/pub/challange/graph/solvers/.

Atamtürk, A., Nemhauser, G.L., and Savelsbergh, M.W.P. 2000. Conflict graphs in solving integer programming problems. *European Journal of Operational Research* **121** 40–55.

Balas, E. and Xue, J. 1996. Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica* **15** 397–412.

Balasundaram, B., Butenko, S., Hicks, I.V. and Sachdeva, S. 2006. Clique relaxations in social network analysis: The maximum k-plex problem. *Submitted*.

V. Batagelj and A. Mrvar, 2006. Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/, 2006.

Beebe, N.H.F. (2002): Nelson H.F. Beebe's Bibliographies Page.

Butenko, S. and Wilhelm, W. 2006. Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research*, **173** 1–17.

Carraghan, R. and Pardalos, P.M. 1990. An exact algorithm for the maximum clique problem. *Oper Res. Lett.* **9** 375–382.

Chen, Y. P., Liestman, A. L., and Liu, J. 2004. Clustering algorithms for ad hoc wireless networks. *Ad Hoc and Sensor Networks* (Y. Pan and Y. Xiao eds.), Nova Science Publishers.

DIMACS. Cliques, Coloring, and Satisability: Second DIMACS Implementation Challenge. http://dimacs.rutgers.edu/Challenges/, 1995. Accessed 2006.

Festinger, L. 1949. *The analysis of sociograms using matrix algebra.* Human Relations 10:153-58.

Friedkin, N.E.1984. *Structural cohesion and equivalence explanations of social homogeneity.* Sociological Methods & Research 12: 235-261.

Jones, B., Computational Geometry Database, February 2002; FTP / HTTP

Luce, R. and Perry, A. 1949. *A method of matrix analysis of group structure.* Psychometrika 14, 95-116.

McClosky, B. and Hicks, I. V. 2007. The co-2-plex polytope and integral systems, *SIAM Journal of Discrete Mathematics,* to appear.

McClosky, B. 2008. *Independence Systems and Stable Set Relaxations.* Ph.D. thesis, Computational and Applied Mathematics Department, Rice University, Houston, TX.

Moon, J.W. and Moser, L. 1965. On cliques in graphs. *Israel J. Math.* **3** 23–28.

Östergård, P. R. J. 2002. A fast algorithm for the maximum clique problem. *Discrete Appl. Math.* **120** 197–207.

V. Batagelj and A. Mrvar, 2006. Pajek datasets. http://vlado.fmf.uni-lj.si/pub/networks/data/. Accessed 2006.

Régin, J.C. 2003. Solving the maximum clique problem with constraint programming. *Fifth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. 166–179.

Seidman, S. B. and Foster, B. L. 1978. A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology* **6** 139–154.

Sewell, E.C. 1998. A branch and bound algorithm for the stability number of a sparse graph. *INFORMS J. Comput.* **10** 438–447.

Tomita, E. and Seki, T. 2003. An efficient branch-and-bound algorithm for finding a maximum clique. *Lecture Notes in Computer Science Series* **2731** 278–289.

Trukhanov, S. 2008, personal communication.

Washio, T. and Motoda, H. 2003. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*. **5(1)** 59–68.

Wasserman, S. and Faust, K. 1994. *Social Network Analysis*, Cambridge University Press.

Wood, D.R. 1997. An algorithm for finding a maximum clique in a graph. *Oper. Res. Lett.* **21** 211-217.