ON MAXIMUM CLIQUE PROBLEMS IN VERY LARGE GRAPHS

J. ABELLO, P. M. PARDALOS, AND M. G.C. RESENDE

Abstract. We present an approach for clique and quasi-clique computations in very large multi-digraphs. We discuss graph decomposition schemes used to break up the problem into several pieces of manageable dimensions. A semi-external greedy randomized adaptive search procedure (GRASP) for finding approximate solutions to the maximum clique problem and maximum quasi-clique problem in very large sparse graphs is presented. We experiment with this heuristic on real data sets collected in the telecommunications industry. These graphs contain on the order of millions of vertices and edges.

1. Introduction

The proliferation of massive data sets brings with it a series of special computational challenges. Many of these data sets can be modeled as very large multi-digraphs M with a special set of edge attributes that represent special characteristics of the application at hand [1]. Understanding the structure of the underlying digraph $D(M)$ is essential for storage organization and information retrieval. In this paper we present a new approach for finding large cliques in large sparse multi-digraphs with millions of vertices and edges.

Let $G = (V, E)$ be an undirected graph where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and E is the set of edges in G. For a subset $S \subseteq V$, we let $G(S)$ denote the subgraph induced by S.

A graph $G = (V, E)$ is *complete* if its vertices are pairwise adjacent, i.e. $\forall\ i, j \in V, \{i, j\} \in E$. A *clique* C is a subset of V such that the induced graph $G(C)$ is complete. The maximum clique problem is to find a clique of maximum cardinality in a graph G. A multigraph M is just an undirected graph with an integer multiplicity associated with every edge. A multi-digraph is defined similarly.

The maximum clique problem is known to be *NP*-complete. Furthermore, the complexity of its approximation remained an open question until recently. In [16], Papadimitriou and Yannakakis introduced the complexity class *MAX SNP* and showed that many natural problems are complete in this class, relative to a reducibility that preserves the quality of approximation. For example, the vertex cover problem (for constant degree graphs), minimum cut problem, dominating set problem, and the MAX 3-SAT problem are such complete problems [19].

If the solution to any of these complete problems can be approximated to arbitrary small constant factors, then the optimal solution to any problem in the class can be approximated to arbitrarily small constant factors. The question of whether

Date: October 13, 1998.
Key words and phrases. maximum clique problem, GRASP, heuristics, experimentation, large data sets, very large sparse graphs, applications.
AT&T Labs Research Technical Report: TR 98.32.1.

such approximation schemes can be found for the complete problems in this class was left unresolved. In [5], Berman and Schnitger show that if one of the *MAX SNP* problems does not have polynomial time approximation schemes, then there is an $\epsilon > 0$ such that the maximum clique problem cannot be approximated in polynomial time with performance ratio (size of maximum clique to size of approximate clique) of $O(n^\epsilon)$, where n is the number of vertices in the graph (see also Feige et al. [6], where a connection between approximation complexity and interactive proof systems is discussed).

A breakthrough in approximation complexity is the recent result by Arora et al. [3, 4]. It is shown that the maximum number of satisfiable clauses in a 3-SAT formula (MAX 3-SAT) cannot be approximated to arbitrary small constants (unless $P = NP$), thus resolving the open question in [16]. This immediately shows the difficulty of finding good approximate solutions to all the above listed problems. In particular, it is shown that no polynomial time algorithm can approximate the maximum clique size within a factor of n^ϵ ($\epsilon > 0$), unless $P = NP$ (by using the results of Feige et al. [6]).

Although these complexity results characterize worst case instances, they nevertheless indicate that the maximum clique problem is indeed a very difficult problem to solve. The maximum clique problem can be approached in two different ways: by exact solutions or heuristic approximation. Since the problem is *NP*-complete, one can expect exact solution methods to have limited performance on large dense problems. On the other hand, with a heuristic one can never know how close to the actual maximum clique the solution is.

The main difficulty with heuristic approximation of the maximum clique problem is that a local optimum can be far from a global optimum. This is handled in many heuristics by devices that allow escape from poor local optimal solutions. Such devices are present in heuristics, such as simulated annealing [15], tabu search [10, 11], and genetic algorithms [13], that move from one solution to another, as well as in the multistart heuristic GRASP [7, 8], which samples different regions of the solution space, finding a local minimum each time. For further information on various algorithms and heuristics see [14, 17].

The maximum clique problem has many practical applications in science and engineering. Some of them are project selection, classification theory, fault tolerance, coding theory, computer vision, economics, information retrieval, signal transmission theory and aligning DNA with protein sequences [17, 12].

The paper is organized as follows. In Section 2 we consider several graph decomposition schemes used in the experiment. Section 3 proposes a GRASP for finding large cliques in very large graphs. Preliminary computational results are described in Section 4 and concluding remarks are made in Section 5.

2. Graph decomposition

In this section, we discuss some decomposition schemes that make very large sparse graphs suitable for processing by graph optimization algorithms. We discuss two schemes.

First, we take the approach proposed by Abello [1] to partition the edge set of a multi-digraph M. Consider the underlying directed graph

$$D(M) = \{(x, y) \mid (x, y) \text{ is an edge in } M\}$$

and the corresponding underlying undirected graph

$$U(M) = \{\{x, y\} \mid (x, y) \text{ is an edge in } D(M)\ \}.$$

For a vertex $u \in M$, let

$$\text{out}(u) = \{x \mid (u, x) \in D(M)\} \text{ and } \text{in}(u) = \{y \mid (y, u) \in D(M)\}.$$

Furthermore, define the out-degree $\text{outdeg}(u) = |\text{out}(u)|$, the in-degree $\text{indeg}(u) = |\text{in}(u)|$, and for C, a subset of vertices in M, let the neighborhood of u with respect to C be $N_C(u) = \text{out}(u) \cup \text{in}(u) \cap C$. The degree of u with respect to C is $\deg_C(u) = |\ N_C(u)\ |$.

In a preprocessing phase, we use efficient external memory algorithms for computing the connected components of $U(M)$ [2]. For each connected component, consider the sub-digraph of $D(M)$ induced by its vertex set and classify its vertices as sources (indeg $= 0$), sinks (outdeg $= 0$) and transmitters (indeg and outdeg > 0). We then partition the directed edge set of each connected component by computing directed Depth First Search Trees (DFST) from each source vertex (or from high degree transmitters). These DFSTs impose an ordering of the edges which is used in a greedy randomized adaptive search procedure (GRASP) tailored for the local maximum clique problem that is described in the next section. The collection of DFSTs defines a neighborhood structure which is used to search for an improvement to a local clique or it can be taken itself as a directed weighted digraph where each DFST now becomes a vertex and the edges running between two DFSTs are collected into a weighted directed edge. It is worth to notice that this digraph $\text{DAG}(M)$ is acyclic, suggesting a topological order of processing. Undirected cliques on $\text{DAG}(M)$ give global structural information about M. The product of the maximum clique size in $\text{DAG}(M)$ times the maximum local clique size in the DFSTs gives an upper bound on the maximum clique size of M. We are currently experimenting with heuristics that use $\text{DAG}(M)$ as the neighborhood structure to move from one DFST to another comparing their local max cliques and gluing them together to search for an improvement.

Another simple decomposition scheme works on large connected components of the graph $U(M)$. The approach is repeatedly applied until no further reduction in the size of the graph is possible. In this reduction scheme, we assume that cliques of size k or smaller are of no interest. All edges having at least one vertex of degree less than k are deleted. This, of course, affects the degrees of other vertices, hence, further simplification is possible by reapplying the reduction scheme. In Section 4 we illustrate these reduction schemes on a graph that arises from real telecommunications data.

3. GRASP for maximum clique and maximum quasi-clique

Feo, Resende, and Smith [9, 18] proposed a greedy randomized adaptive search procedure (GRASP) for the maximum independent set problem. In this paper, we propose a GRASP, tailored for the maximum clique problem and the maximum quasi-clique problem on very large sparse graphs, based on the procedure described in [9].

GRASP [7, 8] is an iterative method that, at each iteration, constructs a randomized, greedily biased, solution and then finds a locally optimal solution in the neighborhood of the constructed solution. The role of randomization is to generate different solutions from which to initiate local search. Greediness leads to

```
procedure construct($V$,$E$,$\alpha$,$Q$)
1    Set initial clique $Q = \emptyset$;
2    Set $C = V$;
3    while $|C| > 0$ do
4        Let $G(C)$ be the subgraph induced by the vertices in $C$;
5        Let $\deg_{G(C)}(u)$ be the degree of $u \in C$ with respect to $G(C)$;
6        $\underline{d} = \min\{\deg_{G(C)}(u) \mid u \in C\}$;
7        $\bar{d} = \max\{\deg_{G(C)}(u) \mid u \in C\}$;
8        RCL $= \{u \in C \mid \deg_{G(C)}(u) \geq \underline{d} + \alpha(\bar{d} - \underline{d})\}$;
9        Select $u$ at random from the RCL;
10       $Q = Q \cup \{u\}$;
11       $C = N_C(u)$;
12   end while;
end construct;
```

FIGURE 1. GRASP construction procedure

constructed solutions of good quality. Local search applied from these greedily biased solutions usually converges quickly to a local minimum and consequently many probes of different, potentially good, neighborhoods can be carried out in a limited amount of time. We initially describe the GRASP for maximum clique. To describe a GRASP, one needs to specify a construction mechanism and a local search procedure.

The construction phase of the GRASP for maximum clique proposed here builds a clique, one vertex at a time. It uses vertex degrees as a guide for construction. Assume that a clique is being constructed. At each step of the construction phase we have a clique on hand. We call the vertices of this clique the *selected* vertices. A vertex is a *candidate* to be included in the clique if it is adjacent to all previously selected clique vertices. Let C denote the set of candidate vertices. Initially, all vertices are candidates, i.e. $C = V$. A *restricted candidate list* (RCL) contains all candidate vertices of high degree in the subgraph $G(C)$ induced by the candidate vertices. A vertex $u \in C$ is said to have high degree in the graph induced by the candidate vertices if its degree $\deg_{G(C)}(u)$ with respect to the subgraph induced by the candidate nodes is at least $\underline{d} + \alpha(\bar{d} - \underline{d})\}$, where $\underline{d} = \min\{\deg_{G(C)}(u) \mid u \in C\}$ and $\bar{d} = \max\{\deg_{G(C)}(u) \mid u \in C\}$ and α is a real parameter in the interval [0,1]. One vertex, among those in the RCL, is selected at random and is added to the clique under construction. To take this selection into consideration, the newly chosen vertex and all other candidate vertices not adjacent to it are eliminated from the set of candidate vertices, i.e. the new candidate set is simply the neighborhood of the newly selected vertex with respect to the old candidate set. This process is repeated until the set of candidate vertices is empty. Figure 1 shows pseudo-code for the construction phase of GRASP.

Local search can be implemented in many ways. A simple $(2, 1)$-exchange approach seeks a vertex in the clique whose removal allows two adjacent vertices not in the clique to be included in the clique, thus increasing the clique size by one. Likewise, one could look for pairs of vertices in the clique whose removal would

```
procedure local($V$,$E$,$Q$)
1    $H = \{(v,u,w) \mid v,u,w \in V, (v,u) \in E, w \in Q,$
         and $v$ and $u$ are adjacent to all vertices of $Q$ except $w\}$;
2    while $|H| > 0$ do
3        Select $(u,v,w) \in H$;
4        $Q = Q \cup \{u,v\} \setminus \{w\}$;
5        $H = \{(v,u,w) \mid v,u,w \in V, (v,u) \in E, w \in Q,$
             and $v$ and $u$ are adjacent to all vertices of $Q$ except $w\}$;
6    end while;
end local;
```

FIGURE 2. GRASP $(2,1)$-exchange local search procedure

```
procedure grasp($V$,$E$,maxitr,$Q^*$)
1    CliqueSize $= -\infty$;
2    for $k = 1, 2, \ldots,$ maxitr do
3        Select $\alpha$, at random, from interval [0,1];
4        construct($V$,$E$,$\alpha$,$Q$);
5        local($V$,$E$,$Q$);
6        if $|Q| >$ CliqueSize do
7            CliqueSize $= |Q|$;
8            $Q^* = Q$;
9        end do;
10   end for;
end grasp;
```

FIGURE 3. GRASP for maximum clique problem

permit three vertices not in the clique to be placed in the clique. Figure 2 shows pseudo-code for the $(2,1)$-exchange local search used in our implementation.

Figure 3 shows pseudo-code indicating how the two procedures described above make up the GRASP for maximum clique.

The GRASP described in this section requires access to the edges and vertices of the graph. This limits its use to graphs small enough to fit in memory. We next propose a semi-external procedure that works only with vertex degrees and a subset of the edges in-memory, while most of the edges can be kept in secondary disk storage. Besides enabling its use on smaller memory machines, the procedure we describe below also speeds up the computation of the GRASP.

To describe this procedure, we first define the `peel` operation on a graph. Given a parameter q, `peel`(G, q) recursively deletes from G all vertices having degree less than q along with their incident edges.

The semi-external procedure `clique` first samples a subset $\mathcal{E}$ of edges of E such that $\mid \mathcal{E} \mid < \mathcal{T}_G$, where $\mathcal{T}_G$ is a threshhold function of the graph G. The subgraph corresponding to $\mathcal{E}$ is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set of $\mathcal{G}$. The procedure `grasp` is applied to $\mathcal{G}$ to produce a clique Q. Let the size of the clique found be $q = |Q|$. Since q is a lower bound on the largest clique in G, any vertex

```
procedure clique($V$,$E$,$\texttt{maxitr}_s$, $\texttt{maxitr}_l$,$\mathcal{T}_G$,$Q$)
1     Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a subgraph of $G = (V, E)$ such that $| \mathcal{E} | \leq \mathcal{T}_G$;
2     while $\mathcal{G} \neq G$ do
3         $\mathcal{G}^+ = \mathcal{G}$;
4         grasp($\mathcal{V}, \mathcal{E}, \texttt{maxitr}_s, Q$);
5         $q = |Q|$;
6         peel($V, E, q$);
7         Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a subgraph of $G = (V, E)$ such that $| \mathcal{E} | \leq \mathcal{T}_G$;
8         if $\mathcal{G}^+ == \mathcal{G}$ break;
9     end while;
10    Partition $E$ into $E_1, \ldots, E_k$ such that $| E_j | \leq \mathcal{T}_G$, for $j = 1, \ldots, k$;
11    for $j = 1, \ldots, k$ do
12        Let $V_j$ be the set of vertices in $E_j$;
13        grasp($V_j, E_j, \texttt{maxitr}_l, Q_j$);
14    end for;
15    $\mathcal{G}^+ = \mathcal{G}$;
16    $q = \max \{|Q_1|, |Q_2|, \ldots, |Q_k|\}$;
17    peel($V, E, q$);
18    if $\mathcal{G}^+ \neq G$ then
19        clique($V$,$E$,$\texttt{maxitr}_s$,$\texttt{maxitr}_l$, $\mathcal{T}_G$,$Q$);
20    end if;
end clique;
```

FIGURE 4. Semi-external approach for maximum clique

in G with degree less than q cannot possibly be in a maximum clique and can be therefore discarded from further consideration. This is done by applying `peel` to G with parameter q. Procedures `grasp` and `peel` are reapplied until no further reduction is possible. The aim is to delete irrelevant vertices and edges, allowing `grasp` to focus on the subgraph of interest. Reducing the size of the graph allows GRASP to explore portions of the solution space at greater depth, since GRASP iterations are faster on smaller graphs. If the reduction results in a subgraph smaller than the specified threshhold, GRASP can be made to explore the solution space in more detail by increasing the number of iterations to $\texttt{maxitr}_l$. This is what usually occurs, in practice, when the graph is very sparse. However, it may be possible that the repeated applications of `grasp` and `peel` do not reduce the graph to the desired size. In this case, we propose partitioning the edges that remain into sets that are smaller than the threshhold and applying `grasp` to each resulting subgraph. The size of the largest clique found is used as parameter q in a `peel` operation and if a reduction is achieved the procedure `clique` is recursively called. Figure 4 shows pseudo-code for this semi-external approach.

In procedure `clique`, edges of the graph are sampled. As discussed earlier, we seek to find a clique in a connected component, examining one component at a time. Within each component, we wish to maintain edges that share vertices close together so that when they are sampled in `clique` those edges are likely to be selected together. To do this, we compute Depth First Search Trees on the directed subgraph in the component and store the edges for sampling in the order determined by the trees.

A related problem of interest is to find quasi-cliques in very large sparse graphs. A *quasi-clique* is defined to be a dense subgraph. A clique is a completely dense quasi-clique. A quasi-clique $\mathcal{Q}(\gamma, q)$ is defined to be a subgraph having q vertices and edge density γ, i.e. the number of edges in $\mathcal{Q}(\gamma, q)$ is $\lfloor \gamma q(q-1)/2 \rfloor$. One can define several optimization problems for quasi-cliques. Three examples are:

- max γq,
- fix q and max γ,
- fix γ and max q.

The approach described above for finding cliques can be extended to address the third problem as follows.

The GRASP for quasi-clique constructs a clique to serve as a seed to grow the quasi-clique from, using a modified local search procedure. The construction procedure is identical to the one used in the case of cliques. The local search procedure looks for vertices (v, u, w) such that $(v, u) \in E$, and $w \in \mathcal{Q}$ and v and u are adjacent to at least $\gamma(q-1)$ vertices of $\mathcal{Q} \setminus \{w\}$. If such vertices exist, then w is removed from the quasi-clique and v and u are added to the quasi-clique, increasing its size by one. Note that if $\gamma = 1$, then this local search procedure is identical to the one used for cliques.

A semi-external procedure similar to the one proposed for finding large cliques can be derived for quasi-cliques. As before, the procedure samples edges from the original graph such that the subgraph induced by the vertices of those edges is of a specified size. The GRASP for quasi-cliques is applied to the subgraph producing a quasi-clique $\mathcal{Q}$ of size q. In the original graph, any vertex having degree not greater than γq can be discarded from further consideration, since it cannot possibly be in a quasi-clique of size at least q.

4. Experiments with a very large graph

In this section, we outline the preliminary exploratory experiments done with a sample dataset. The experiments were done on a Silicon Graphics Challenge computer (20 MIPS 196MHz R10000 processors with 6.144 Gbytes of main memory). A substantial amount of disk space was also used.

Our current data comes from telecommunications traffic. The corresponding multi-graph has 53,767,087 vertices and over 170 million edges. We found 3,667,448 connected components out of which only 302,468 were components of size greater than 3 (there were 255 self-loops, 2,766,206 pairs and 598,519 triplets).

A giant component with 44,989,297 vertices was detected. It is tantalizing to suggest then that we may be witnessing here a behavior similar to the one predicted by random graph theory even though our graphs are certainly not random. The giant component has 13,799,430 Depth First Search Trees (DFSTs) and one of them is a giant DFST (it has 10,355,749 vertices and 19,072,448 edges). Most of the DFSTs have no more than 5 vertices. The interesting trees have sizes between 5 and 100. Their corresponding induced subgraphs are most of the time very sparse ($|E| < |V| \ \log |V|$), except for some occasional dense subgraphs ($|E| > |V| \ \sqrt{|V|}$) with 11 to 32 vertices.

We argue that the largest clique in this component has size not greater than 32. Cliques are either within a subgraph induced by the vertices of a DFST, or

TABLE 1. Cliques found by `construct` and `local`

	cliques found by		distinct
size	`construct`	`local`	cliques
2	63	62	
3	473	320	
4	95	176	
5	73	103	14
6	116	95	11
7	59	38	25
8	54	63	28
9	22	33	14
10	17	10	9
11	15	38	35
12	10	32	22
13	1	26	18
14	0	3	3
15	0	1	1

distributed among the different DFSTs. We expect the former to occur. There are several large DFSTs, the largest having about 19 million edges. By counting the edges in the trees, one observes that there remain very few edges to go between trees and consequently it is more likely that cliques are within the graphs induced by the nodes of a tree. Since the largest dense subgraph induced by the vertices of a tree has 32 vertices, we should not expect many cliques larger that 32 to be found.

In this exploratory phase of our work, we did not integrate the `grasp` and `peel` procedures into procedure `clique`, but rather applied them manually. To begin our experimentation, we considered 10% of the edges in the large component from which we recursively removed all vertices of degree one by applying `peel`$(V, E, 1)$. This resulted in a graph with 2,438,911 vertices and 5,856,224 edges, which fits in memory. In this graph we search for large cliques. Our first motivation is to identify a lower bound on the size of the maximum clique so that we can delete higher-degree vertices on larger portions of the graph to possibly identify larger cliques. The GRASP was repeated 1000 times, with each iteration producing a locally optimal clique. Though applying local search on every constructed solution may not be efficient from a running time point of view, we applied local search to all constructed solutions to explore its effect in improving clique sizes. Because of the independent nature of the GRASP iterations and since our computer is configured with 20 processors, we created 10 threads, each independently running GRASP starting from a different random number generator seed.

Table 1 summarizes the first part of the experimental results. It shows, for each clique size found, the number of GRASP iterations that constructed or improved such solution, and from sizes 5 to 15, the number of distinct cliques that were found by the GRASP iterations. It is interesting to observe that these cliques, even though distinct, share a large number of vertices. Applying a greedy procedure to these cliques to identify a disjoint set of cliques produced one clique of size 15, 12, 9, and 7, and 4 cliques of size 6, and 5 of size 5.

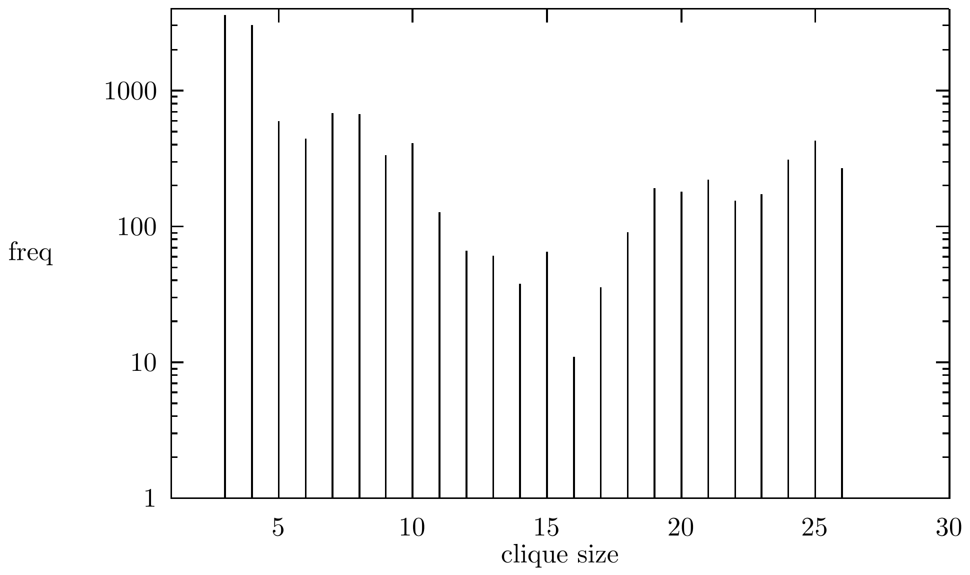

FIGURE 5. Frequency of clique sizes found on 25% of data

Next, we considered 25% of the edges in the large component from which we recursively removed all vertices of degree 10 or less. The resulting graph had 291,944 vertices and 2,184,751 edges. 12,188 iterations of GRASP produced cliques of size 26. In 271 iterations, a clique of size 26 was produced. In 431 iterations, a clique of size 25 was produced. In 313 iterations, a clique of size 24 was produced. Figure 5 shows the frequencies of cliques of different sizes found by the algorithm.

Having found cliques of size 26 in a quarter of the graph, we next intensified our search on the entire huge connected component. In this component, we recursively removed all vertices of degree 20 or less. The resulting graph has 27,019 vertices and 757,876 edges. Figure 6 shows the frequencies of cliques of different sizes found by the algorithm. Figure 7 shows the statistics of the improvement attained by local search.

Over 20,000 GRASP iterations were carried out on the 27,019 vertex – 757,876 edge graph. Cliques of 30 vertices were found. These cliques are very likely to be optimal because we do not expect cliques larger than 32 vertices to be found. The local search can be seen to improve the constructed solution not only for large constructed cliques, but also for small cliques. In fact, in 26 iterations, constructed cliques of size 3 were improved by the local search to size 30.

Finally to increase our confidence that the cliques of size 30 found are maximum, we applied `peel`$(V, E, 30)$, resulting in a graph with 8724 vertices and about 320 thousand edges. We ran 100,000 GRASP iterations on the graph taking 10 parallel processors about one and a half days to finish. The largest clique found had 30 vertices. Of the 100,000 cliques generated, 14,141 were distinct, although many of them share one or more vertices.

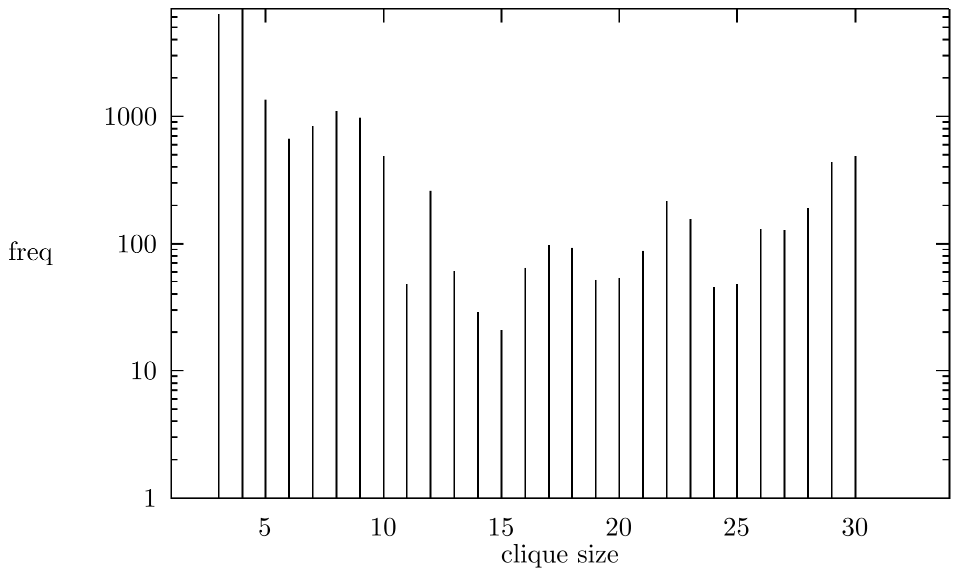

FIGURE 6. Frequency of clique sizes found on entire dataset

Finally, to compute quasi-cliques on this test data, we looked for large quasi-cliques with density parameters $\gamma = .9, .8, .7$, and $.5$. Quasi-cliques of sizes 44, 57, 65, and 98, respectively, were found.

5. Concluding remarks

We presented an approach for clique and quasi-clique computations in very large multi-digraphs. We discussed graph decomposition schemes used to break up the problem into several pieces of manageable dimensions. A greedy randomized adaptive search procedure (GRASP) for finding approximate solutions to the maximum clique (quasi-clique) problem in very large sparse graphs was presented. We experimented with this heuristic on real data sets collected in the telecommunications industry. These graphs contain on the order of millions of vertices and edges.

In a variety of industrial applications it is necessary to deal explicitly with weights, edge directions, and multiplicities. In this regard, we are currently extending the methods presented here to find directed weighted quasi-cliques. These are subgraphs that satisfy special local density conditions which make them amenable to local search based techniques. Our findings in this direction will be reported in a forthcoming paper.

References

[1] J. Abello. A dynamic multigraph model for massive data sets. Technical report, AT&T Labs Research, Florham Park, NJ, December 1997.

[2] J. Abello, A. Bushbaum, and J. Westbrook. A functional aproach to external memory algorithms. In *European Symposium on Algorithms*, volume 1461 of *Lecture Notes in Computer Science*, pages 332–343. Springer-Verlag, 1998.

[3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. On the intractability of approximation problems. Technical report, University of Rochester, 1992.

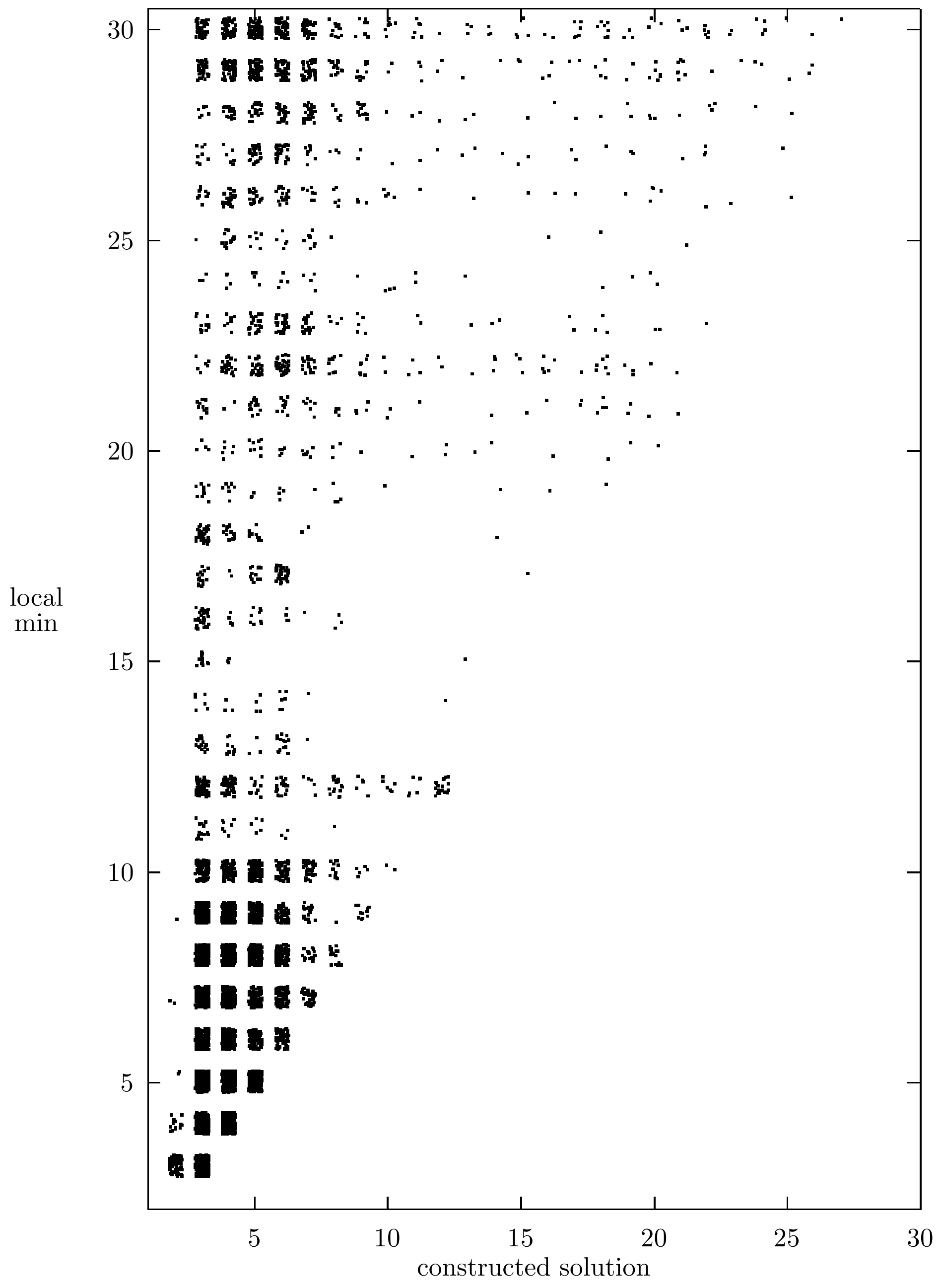

FIGURE 7. Local search improvement

[4] S. Arora and S. Safra. Approximating the maximum clique is *NP*-complete. Technical report, University of Rochester, 1992.

[5] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Lecture Notes in Computer Science*, 349:256–267, 1989.

[6] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating the maximum clique is almost *NP*-complete. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 2–12, 1991.
[7] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.
[8] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.
[9] T.A. Feo, M.G.C. Resende, and S.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878, 1994.
[10] F. Glover. Tabu search. Part I. *ORSA J. Comput.*, 1:190–206, 1989.
[11] F. Glover. Tabu search. Part II. *ORSA J. Comput.*, 2:4–32, 1990.
[12] J. Hasselberg, P. M. Pardalos, and G. Vairaktarakis. Test case generators and computational results for the maximum clique problem. *Journal of Global Optimization*, 3:463–482, 1993.
[13] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.
[14] D. S. Johnson and M. A. Trick, editors. *Cliques, Graph Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.
[15] S. Kirkpatrick, C. D. Gellat Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
[16] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. In *Proc. of the Twentieth Annual ACM STOC*, pages 229–234, 1988.
[17] P. M. Pardalos and J. Xue. The maximum clique problem. *Journal of Global Optimization*, pages 301–328, 1994.
[18] M.G.C. Resende, T.A. Feo, and S. H. Smith. FORTRAN subroutines for approximate solution of maximum independent set problems using grasp. Technical report, AT&T Labs Research, Florham Park, NJ, 1997. To appear in *ACM Trans. Math. Software*.
[19] M. Yannakakis. On the approximation of maximum satisfiability. In *Proc. 3rd Annual ACM-SIAM Symp. on Discrete Algorithms*, 1992.

(J. Abello) Network Services Research Center, AT&T Labs – Research, Shannon Laboratory, Florham Park, NJ 07932 USA
E-mail address, J. Abello: `abello@research.att.com`

Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611 USA.
E-mail address: `pardalos@ufl.edu`

Information Sciences Research, AT&T Labs Research, Florham Park, NJ 07932 USA.
E-mail address: `mgcr@research.att.com`