Massive Quasi-Clique Detection

James Abello[1], Mauricio G.C. Resende[1], and Sandra Sudarsky[2⋆]

[1] AT&T Labs Research, Florham Park, NJ 07032 USA
{abello,mgcr}@research.att.com
[2] Siemens Corporate Research, Inc, Princeton, NJ 08540 USA
sudarsky@scr.siemens.com

Abstract. We describe techniques that are useful for the detection of dense subgraphs (quasi-cliques) in massive sparse graphs whose vertex set, but not the edge set, fits in RAM. The algorithms rely on efficient semi-external memory algorithms used to preprocess the input and on greedy randomized adaptive search procedures (GRASP) to extract the dense subgraphs. A software platform was put together allowing graphs with hundreds of millions of nodes to be processed. Computational results illustrate the effectiveness of the proposed methods.

1 Introduction

A variety of massive 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 useful for storage organization and information retrieval. The availability of computers with gigabytes of RAM allows us to make the realistic assumption that the vertex set (but not the edge set) of $D(M)$ fits in main memory. Moreover, it has been observed experimentally, in data gathered in the telecommunications industry, the internet, and geographically based information systems, that $D(M)$ is a sparse graph with very skewed distribution and low undirected diameter [1]. These observations have made the processing of $D(M)$ feasible. We present here an approach for discovering large dense subgraphs (quasi-cliques) in such large sparse multi-digraphs with millions of vertices and edges. We report a sample of our current experimental results.

Before proceeding any further let us agree on the following notational conventions.

Let $G = (V, E)$ be a graph where V is the set of vertices and E is the set of edges in G. A multi-graph M is just a graph with an integer multiplicity associated with every edge. Whenever it becomes necessary to emphasize that the underlying graph is directed we use the term multi-digraph.

For a subset $S \subseteq V$, we let G_S denote the subgraph induced by S.

A graph $G = (V, E)$ is γ-dense if $|E(G)| \geq \gamma \binom{|V(G)|}{2}$. A γ-clique S, also called a quasi-clique, is a subset of V such that the induced graph G_S is connected

⋆ Work completed as an AT&T consultant and DIMACS visitor.

S. Rajsbaum (Ed.): LATIN 2002, LNCS 2286, pp. 598–612, 2002.

and γ-dense. The maximum γ-clique problem is to find a γ-clique of maximum cardinality in a graph G.

For a graph $G = (V, E)$, $S \subset V$, $E' \subset E$, and $x \in V$, let

- $\deg(x) = |\mathcal{N}(x)|$ where $\mathcal{N}(x) = \{y \in V(G) \big| (x, y) \in E(G)\}$;
- $\deg(x)\big|_S = |\mathcal{N}(x)\big|_S|$ where $\mathcal{N}(x)\big|_S = \mathcal{N}(x) \cap S$;
- $\mathcal{N}(S) = \cup_{x \in S} \mathcal{N}(x)$;
- $E(S) = \{(x, y) \in E(G) : x \in S, y \in S\}$;
- $G_S = (S, E(S))$;
- For $S, R \subset V(G)$, $E(S, R) = \{(x, y) \in E(G) : x \in S, y \in R\}$;

Finding a maximum 1-clique is a classical *NP*-hard problem and therefore one can expect exact solution methods to have limited performance on large instances. In terms of approximation, a negative breakthrough result by Arora et al. [4,5] together with results of Feige et al. [6], and more recently Håstad [9], prove that no polynomial time algorithm can approximate the maximum clique size within a factor of n^{ϵ} ($\epsilon > 0$), unless $P = NP$. Given this current state of knowledge, it is very unlikely (with our current models of computation) that general heuristics exist which can provide answers with certifiable measures of optimality. A suitable approach is then to devise heuristics equipped with mechanisms that allow them to escape from poor local optimal solutions. These include heuristics such as simulated annealing [11], tabu search [8], and genetic algorithms [10], that move from one solution to another, as well as the multistart heuristic GRASP [7,12], which samples different regions of the solution space, finding a local minimum each time.

In this paper, we use the concept of quasi-cliques as a unifying notion that drives heuristics towards the detection of dense subgraphs in very large but sparse multi-digraphs. (It is assumed here that the vertex set of the graph, but not the edge set, fits in RAM. These graphs are termed semi-external in [2]).

Our main contributions include the introduction of a very intuitive notion of potential of a vertex set (edge set) with respect to a given quasi-clique (bi-clique), the use of edge pruning and an external memory breadth first search (BFS) traversal to decompose a disk resident input digraph into smaller subgraphs to make the search feasible. We remark that the techniques described here are being applied to very large daily telecommunications traffic graphs containing hundreds of millions of vertices.

The paper is organized as follows. In Section 2, we consider several graph decomposition schemes and pruning approaches used in our computations to reduce the search space. Section 3 and Section 4 present basic quasi-clique notions and define the potential function that is central to our approach for discovering maximal quasi-cliques. Section 5 contains a brief discussion of the main ingredients necessary to design greedy randomized adaptive search procedures (GRASP). Section 6 uses the machinery presented in the preceeding sections to describe a GRASP tailored for finding maximal quasi-cliques in both bipartite and non-bipartite graphs. Section 7 contains the description of our semi-external approach to handle very large sparse multi-digraphs. Sample computational results and concluding remarks appear in Section 8 and Section 9.

2 Graph Decomposition and Pruning

We introduce in this section two decomposition schemes that make the processing of very large graphs feasible.

First, we identify sources, sinks and transmitter vertices in the input graph. Namely, consider the underlying directed graph

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

and the corresponding underlying undirected graph

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

It is worthwhile recalling here that for data gathered in certain telecommunications applications (such as phone calls), internet data (such as URL links), and geographical information systems, $U(M)$ is a sparse graph with very low diameter [1].

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, let $\text{outdeg}(u) = |\text{out}(u)|$, and $\text{indeg}(u) = |\text{in}(u)|$.

In a preprocessing phase, we use efficient external memory algorithms for computing the undirected 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 edge set of each connected component by traversing the corresponding subgraph in a breadth first search manner. The assumption that the vertex set fits in main memory together with the fact that $U(M)$ has low diameter allow us to perform the Breadth First Search in few passes over the data. We store each connected component as a collection of subgraphs according to the BFS order. Namely, the subgraphs induced by vertices at the same undirected distance, from the root of the corresponding BFS tree, are stored contiguously. The edges between vertices at adjacent levels are also stored contigously. Clearly, 1-cliques can appear only in these subgraphs. We explot this fact to localize (and parallelize) the quasi-clique search by using maximal 1-cliques as seeds.

A complementary but very effective processing scheme that also helps to localize quasi-clique detection, consists in pruning those edges that do not contribute to a better solution. These edges are called *peelable*. In the case of 1-cliques, if a lower bound k on the cardinality of maximum clique is known, we can delete all those edges incident with a vertex of degree less than k. This process affects the degrees of other vertices, hence, further simplification is possible by reapplying the same reduction scheme. To control the pruning we recursively delete edges incident to vertices of degree i, from $i = 1$ to $k - 1$, in that order, updating the degree of both endpoints.

When γ is less than one, we use the notion of γk-*peelable* vertices. Namely, a vertex v is γk-*peelable* if v and all its neighbors have degree smaller than γk. γk-*peelable* vertices cannot be added to a quasi-clique of density γ and cardinality at least k to obtain a larger quasiclique with density greater than or equal to γ. Because of this, if we know the existence of a quasi-clique of density γ and of cardinality at least k, then we can prune all those edges incident to γk *peelable* vertices, in increasing degree order, updating every time the degrees of both endpoints.

We want to remark here that designing a good and efficient peeling schedule is by itself a problem that could be useful in the exploration of massive data sets. Our experimental results indicate that the proposed approach works well for sparse and low diameter graphs that are reducible to trees by a sequence of dense subgraph contractions. In Section 8 we report results obtained by a combination of these techniques when applied to multi-graphs extracted from telecommunications data sets. From now on, we assume that we have an index structure to the subgraphs induced by vertices on the same level of the BFS and to the subgraphs induced by the union of the vertices in two consecutive BFS levels.

3 Quasicliques

Quasicliques are subgraphs with specified edge density. Two optimization problems related to quasicliques arise naturally. In the first, we fix an edge density and seek a quasiclique of maximum cardinality with at least the specified edge density. In the other, we specify a fixed cardinality and seek a quasiclique of maximum edge density. In this section, we describe general properties about quasicliques in an undirected graph $G = (V, E)$. We denote by S the set of vertices of the subgraph G_S we wish to find, i.e. the subgraph induced by S on G. Let γ be a real parameter such that $0 < \gamma \leq 1$. Recall that a set of vertices $S \subseteq V(G)$ is a γ-clique if G_S is connected and $|E(G_S)| \geq \gamma\binom{|S|}{2}$. We also refer to γ-cliques as quasicliques. A vertex $x \in \bar{S}$ is called a γ-vertex, with respect to a γ-clique S, if $G_{S\cup\{x\}}$ is a γ-clique. Similarly, a set of vertices $R \subseteq \bar{S}$ is called a γ-set with respect to S if $S \cup R$ is a γ-clique. The set of γ-vertices with respect to S is denoted by $\mathcal{N}_\gamma(S)$. Notice that $\mathcal{N}_\gamma(S)$ is not necessarily a γ-clique.

One basic property of γ-cliques is the following.

Lemma 1. *Let S and R be disjoint γ-cliques with*

$$\frac{|E(G_S)|}{\binom{|S|}{2}} = \gamma_S \text{ and } \frac{|E(G_R)|}{\binom{|R|}{2}} = \gamma_R.$$

$S \cup R$ is a γ-clique, if and only if

$$|E(S,R)| \geq \gamma(|R||S|) - (\gamma_R - \gamma)\binom{|R|}{2} - (\gamma_S - \gamma)\binom{|S|}{2}. \quad (1)$$

The proof of this simple but fundamental lemma follows easily from the definitions. It provides a general framework to find quasicliques. Namely, given a γ-clique S, find another γ-clique R with $S \cap R = \emptyset$, such that (1) holds. In order to guarantee that the joint condition (1) is satisfied one can restrict R to be a γ-clique in $\mathcal{N}_\gamma(S)$. More generally, the objective is to find large γ-sets with respect to S. One approach to achieve this is to use the notion of set potential. Define the potential of a set R to be

$$\phi(R) = |E(R)| - \gamma \binom{|R|}{2}$$

and the potential of a set R with respect to a disjoint set S to be

$$\phi_S(R) = \phi(S \cup R).$$

Sets with nonnegative potential are precisely γ-sets. We seek γ-sets R with large potential $\phi_S(R)$. Ideally, in a construction algorithm, one would prefer sets R of maximum cardinality. Finding such sets is computationally intractable. Our approach is to build a maximal γ-clique incrementally. In the next section, we describe such an algorithm.

4 Finding Maximal Quasicliques

Assume S is a γ-clique. We seek a vertex $x \in \mathcal{N}_\gamma(S)$ to be added to S. One strategy for selecting x is to measure the effect of its selection on the potential of the other vertices in $\mathcal{N}_\gamma(S)$. To accomplish this, define the potential difference of a vertex $y \in \mathcal{N}_\gamma(S) \setminus \{x\}$ to be

$$\delta_{S,x}(y) = \phi_{S\cup\{x\}}(\{y\}) - \phi_S(\{y\}).$$

It follows from the definitions that

$$\delta_{S,x}(y) = \deg(x)\big|_S + \deg(y)\big|_{\{x\}} - \gamma(|S|+1).$$

The above equation shows that the potential of γ-neighbors of x does not decrease with the inclusion of x. On the other hand, the potential of the non-γ-neighbors of x may decrease when the potential of the γ-neighbors of x increases by less than a unit. If the increase in the potential of a γ-neighbor is exactly one unit, there is no change in the potential of the non-γ-neighbors. It also follows that if x and y are adjacent γ-vertices and x is added to S, then y remains a γ-vertex with respect to $S \cup \{x\}$.

The total effect on the potentials, caused by the selection of x, on the remaining vertices of $\mathcal{N}_\gamma(S)$ is

$$\Delta_{S,x} = \sum_{y\in\mathcal{N}_\gamma(S)\setminus\{x\}} \delta_{S,x}(y) = \left|\mathcal{N}_\gamma(\{x\})\right| + \left|\mathcal{N}_\gamma(S)\right| \left(\deg(x)\big|_S - \gamma(|S|+1)\right). \tag{2}$$

Note that $\left|\mathcal{N}_\gamma(\{x\})\right| = \deg(x)\big|_{\mathcal{N}_\gamma(S)}$. The vertex x that maximizes $\Delta_{S,x}$ is one with a high number of γ neighbors and with high degree with respect to S. A greedy algorithm that recursively selects such a vertex will eventually terminate with a maximal γ-set.

A construction procedure builds a quasiclique, one vertex at a time. Ideally, the cardinality of the set of γ-neighbors of a given partial solution S should not increase. This could occur in the case that a vertex with nonpositive potential gains enough potential with the inclusion of a vertex to make it a γ-vertex. To insure that the set of γ-neighbors does not increase, we use a control variable $\gamma^* \geq \gamma$ that corresponds to the density of the current partial solution. The construction procedure, whose pseudo-code is shown in Figure 1, incorporates this idea. Its time complexity is $O(|S|\ |V|^2)$, where S is the set of vertices of the constructed maximal quasiclique.

```
procedure construct_dsubg(I:γ,V,E,O:S)
1   γ* = 1;
2   Select x ∈ V;
3   S* = {x};
4   while γ* ≥ γ do
5       S = S*;
6       if (N_γ*(S) ≠ ∅) then
7           Select x ∈ N_γ*(S);
8       else
9           if (N(S) \ S = ∅) return(S);
10          Select x ∈ N(S) \ S;
11      end;
12      S* = S ∪ {x};
13      γ* = |E(S*)|/(|S*| choose 2);
14  end while;
15  return(S);
end construct_dsubg;
```

Fig. 1. Pseudo-code of construction procedure for maximal dense subgraphs

Notice that by modifying the stopping criterion of the while loop (to $|S| < K$), the procedure can be used to find a K cardinality quasiclique of large density.

Given a maximal γ-clique S and an objective function $f(S)$, define

$$H(S; f) = \left\{X \subseteq S;\ Y \subseteq \bar{S} \;\middle|\; f((S \setminus X) \cup Y) > f(S)\right\}$$

to be the set of subset exchanges that improve the objective function. A local improvement procedure that makes use of this neighborhood structure can further improve the quasiclique. A restricted version of this local search with $|X| = 1$ and $|Y| = 2$ called a $(1, 2)$ local exchange procedure, is used to potentially im-

prove the γ-clique in the case that $f(S) = |S|$. When $f(S) = |E(S)|/\binom{|S|}{2}$, we use local search with $|X| = |Y| = 1$ to try to find a denser K cardinality quasiclique.

This local search, coupled with a greedy randomized version of the construction procedure `construct_dsubg` is described in Section 6. That type of algorithm is called a greedy randomized adaptive search procedure (GRASP), and we review the overall idea in the next section.

5 GRASP

A GRASP, or greedy randomized adaptive search procedure [7], is a multi-start or iterative process, in which each GRASP iteration consists of two phases, a construction phase, in which a feasible solution is produced, and a local search phase, in which a local optimum in the neighborhood of the constructed solution is sought. The best overall solution is kept as the result. The pseudo-code in Figure 2 illustrates a GRASP procedure for maximization in which `maxitr` GRASP iterations are done. For a recent survey of GRASP, see [12].

```
procedure grasp(f(·),g(·),maxitr,x*)
1   f* = −∞;
2   for k = 1, 2, ... , maxitr do
3       grasp_construct(g(·),α,x);
4       local(f(·),x);
5       if f(x) > f* do
6           x* = x;
7           f* = f(x*);
9       end if;
10  end for;
end grasp;
```

Fig. 2. GRASP pseudo-code

In the construction phase, a feasible solution is iteratively constructed, one element at a time. At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a restricted candidate list (RCL) C with respect to a greedy function $g : C \to \mathbb{R}$, and randomly choosing one of the candidates in the list. Let $\alpha \in [0, 1]$ be a given real parameter. The pseudo code in Figure 3 describes a basic GRASP construction phase. The pseudo-code shows that the parameter α controls the amounts of greediness and randomness in the algorithm. A value $\alpha = 1$ corresponds to a purely greedy construction procedure, while $\alpha = 0$ produces a purely random construction.

A solution generated by a GRASP construction is not guaranteed to be locally optimal with respect to simple neighborhood definitions. It is almost always

```
procedure grasp_construct(g(·),α,x)
1   x = ∅;
2   Initialize candidate set C;
3   while C ≠ ∅ do
4       s̲ = min{g(t) | t ∈ C};
5       s̄ = max{g(t) | t ∈ C};
6       RCL = {s ∈ C | g(s) ≥ s̲ + α(s̄ − s̲)};
7       Select s, at random, from the RCL;
8       x = x ∪ {s};
9       Update candidate set C;
10  end while;
end grasp_construct;
```

Fig. 3. GRASP construction pseudo-code

beneficial to apply a local improvement procedure to each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The *neighborhood structure* N for a problem P relates a solution s of the problem to a subset of solutions $N(s)$. A solution s is said to be *locally optimal* if there is no better solution in $N(s)$. The pseudo-code in Figure 4 describes a basic local search procedure.

```
procedure local(f(·),N(·),x)
1   H = {y ∈ N(x) | f(y) > f(x)};
2   while |H| > 0 do
3       Select x ∈ H;
4       H = {y ∈ N(x) | f(y) > f(x)};
5   end while;
end local;
```

Fig. 4. GRASP local search pseudo-code

6 GRASP for Finding Quasicliques

In this section, we describe a GRASP for finding γ-cliques. First, we describe a procedure for the nonbipartite case. Then, we describe a procedure for the bipartite case.

6.1 Nonbipartite Case

This GRASP uses a greedy randomized version of the construction procedure `construct_dsubg` described in Section 4 and a $(2,1)$-exchange local search.

First, we consider the problem of finding a high cardinality quasiclique of specified density γ. We need to specify how the selections are made in lines 2, 7, and 10 of the procedure `construct_dsubg`. In line 2, the greedy function used is the vertex degree $g(x) = \deg(x)\big|_V$. In line 7, we use the total potential difference $g(x) = \Delta_{S,x}$, as given in equation (2). In line 10, the greedy function is the vertex degree with respect to the current solution S, i.e. $g(x) = \deg(x)\big|_S$.

For the problem of finding a high density quasiclique of specified cardinality K, the only difference is that in line 7 the greedy function becomes the vertex degree with respect to the current solution S, i.e. $g(x) = \deg(x)\big|_S$.

6.2 Bipartite Case

For a bipartite graph $G = (V, E)$, where $V(G) = L \cup R$ and $L \cap R = \emptyset$, the procedure builds a quasiclique one edge at a time. A balanced bipartite quasiclique (γ-biclique) is a subgraph G_S such that $V(G_S) = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$ such that $|S_1| = |S_2|$ and $|E(S_1, S_2)| \geq \gamma |L||R|$. We seek a balanced bipartite quasiclique with large cardinality.

In this case, we consider the following potential function that takes as arguments the pair of disjoint sets of vertices L and R. The potential $\phi((L, R)) = |E(L, R)| - \gamma |L||R|$. The potential of a γ-biclique is nonnegative. For a γ-biclique S with $V(S) = L \cup R$ and $L \cap R = \emptyset$, an edge $(x, y) \in E(\mathcal{N}_\gamma(S) \setminus S$ is a γ-neighbor of S if $|E(S \cup \{x, y\})| \geq \gamma((|L| + 1)(|R| + 1))$.

The notions of potential and potential difference of an edge, and the total effect on the potentials of the remaining edges by the inclusion of an edge can be extended to this case using the modified potential given above. Likewise, the construction procedure `construct_dsubg` can be adapted to find large balanced bipartite quasicliques. The edge selection greedy function is now

$$g(x, y) = \deg(x)\big|_{\mathcal{N}_\gamma(S)} \deg(y)\big|_{\mathcal{N}_\gamma(S)} / (\deg(x)\big|_{\mathcal{N}_\gamma(S)} + \deg(y)\big|_{\mathcal{N}_\gamma(S)}).$$

The complexity in this case becomes $O(|Ma||V|^2)$ where $|Ma|$ is the size of a maximum matching in G.

7 A Semi External Memory Approach for Finding Quasicliques

The procedure described in the previous sections requires access to the edges and vertices of the input graph. This limits its use to graphs small enough to fit in memory. We now propose a semi-external procedure [2,3] that works only with vertex degrees and a subset of the edges in main 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. We assume that very large graphs appearing in massive datasets are sparse. The approach described next takes advantage of this situation in two respects. First, the largest cardinality clique is bounded from above by approximately the square root of the number of edges. Second, exploring the vertices of the graph in a breadth first search manner will eventually get to a maximal γ-clique.

In order to proceed, we first define a `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. This peeling operation is used in the case of cliques, i.e. quasicliques with $\gamma = 1$.

In the case of γ cliques when γ is strictly less than 1, we use a more constrained peeling operation `localpeel`.

`localpeel`(G, q) recursively deletes from G only those vertices with degree less than q, all of whose neighbors have also degree less than q, along with their incident edges.

We proceed now with the details for cliques and later on we discuss how to extend them to general quasicliques.

First, we sample 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 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 the 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, the 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 partition the edges that remain into sets that are smaller than the threshhold and apply `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 5 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 perform a semi-external breadth first search on the subgraph in the component and store the edges for sampling in the order determined by the search.

```
procedure clique(V,E,maxitr_s, maxitr_l,T_G,Q)
1     Let G = (V, E) be a subgraph of G = (V, E) such that | E | ≤ T_G;
2     while G ≠ G do
3           G^+ = G;
4           grasp(V, E, maxitr_s, Q);
5           q = |Q|;
6           peel(V, E, q);
7           Let G = (V, E) be a subgraph of G = (V, E) such that | E | ≤ T_G;
8           if G^+ == G break;
9     end while;
10    Partition E into E_1, ..., E_k such that | E_j | ≤ T_G, for j = 1, ..., k;
11    for j = 1, ..., k do
12          Let V_j be the set of vertices in E_j;
13          grasp(V_j, E_j, maxitr_l, Q_j);
14    end for;
15    G^+ = G;
16    q = max {|Q_1|, |Q_2|, ..., |Q_k|};
17    peel(V, E, q);
18    if G^+ ≠ G then
19          clique(V,E,maxitr_s,maxitr_l, T_G,Q);
20    end if;
end clique;
```

Fig. 5. Semi-external approach for maximum clique

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 k and density γ. In the original subgraph, edges adjacent to at least one γk-peelable vertex are removed in increasing degree order as justified in Section 2. To prevent the peeling process from bypassing a large portion of the search space, we generate disjoint maximal γ-cliques, using the `construct-dsubg` procedure of Figure 1. In this case, we set the peeling parameter

$$q = \min\{q_1\gamma_1, q_2\gamma_2, \ldots, q_i\gamma_i, \ldots, q^*\gamma\},$$

where q_i and γ_i denote the cardinalities of the obtained maximal quasi-cliques and their densities, respectively, and q^* is the minimum cardinality of the maximal quasi-cliques under consideration. Therefore, lines 16 and 17 of Figure 5, are modified to adapt the clique semi-external approach to general quasi-cliques. Namely, the peeling parameter q is chosen more conservatively (as indicated above) and the procedure `peel` is substituted by the procedure `localpeel`. Recall that in this case a vertex is peeled only if its degree and that of all its neighbors is smaller than q. It is important to point out that we use different

threads on the different subgraphs obtained by the external memory BFS. In general, different quasi-cliques are obtained by starting the construction from different vertices in an independent manner. It is with regard to maximality within the local vicinity of the current solution that peeling becomes very effective. Global peeling as proposed above, i.e. with respect to the current set of obtained quasi-cliques, becomes a necessity, specially when we are dealing with graphs with hundreds of millions of vertices. The main idea is to use the obtained quasicliques as a guide to the exploration of the remaining portions of the graph. In the case of 1-cliques, information among the threads is used to peel off vertices that do not satisfy the lower bound degree. In this case, the peeling can be done in a more aggressive manner specially if the aim is to maximize cardinality.

8 Experiments with a Large Graph

In this section, we report typical results obtained with telecommunications data sets. 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 undirected connected components out of which only 302,468 were 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. The giant component has 13,799,430 Directed 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. By counting the edges in the trees, one observes that there are very few edges that go between trees and consequently it is more likely that cliques are within the subgraphs induced by the nodes of a tree. 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 searched for large cliques. The GRASP was repeated 1000 times, with each iteration producing a locally optimal clique. 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. It is interesting to observe that the cliques found, even though distinct, share a large number of vertices. 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.

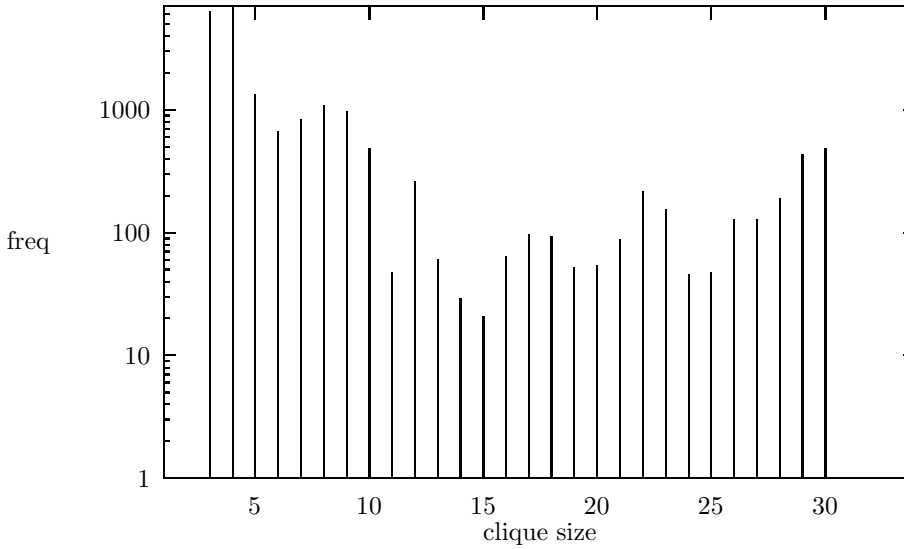

Fig. 6. Frequency of clique sizes found on entire dataset

Fig. 7. Local search improvement

Having found cliques of size 26 in a quarter of the graph, we next intensified our search on the entire giant 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 close to optimal. 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. To increase our confidence that the cliques of size 30 were maximum, we applied $\texttt{peel}(V, E, 30)$, 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. 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.

9 Concluding Remarks

We introduced a very intuitive notion of potential of a vertex set with respect to a given quasi-clique. This potential is used to devise a local search procedure that finds either a larger quasi-clique with the same density or a denser quasi-clique with the same cardinality. Iterating this procedure eventually produces maximal quasi-cliques. In a similar vein, a potential function of a set of edges with respect to the set of edges in a given bi-clique was used to find balanced and maximally dense subgraphs of a given bipartite graph.

In order to make these procedures applicable to very large but sparse multi-digraphs, we used a specialized graph edge pruning and an external memory breadth first search traversal to decompose the input graph into a smaller collection of subgraphs on which the detection of quasi-cliques becomes possible.

We presented a sample of our experimental results when the algorithms were applied to very large graphs collected in the telecommunications industry. Our main intention was to show the feasibility of massive multi-digraph processing. In fact, our platform is currently being used to experimentally process phone data. We expect to be able to report, in the full journal version, our quasi-clique analysis of the largest AT&T telephone traffic day in history which occurred on September 11, 2001. Given the similarities exhibited by telephone traffic and internet data (i.e. skew distribution, sparsity and low diameter) we are currently applying the techniques described here to internet routing data.

There are many natural questions being raised by the processing of massive multi-digraphs. One is how to devise a greedy randomized adaptive search procedure when the input is a semi-external but weighted multi-digraph. Similarly,

it is tantalizing to study the case when the input graph is fully external, i.e. not even the vertex set fits in RAM.

10 Acknowledgements

We thank the members of the Network Services Research Center at AT&T Labs Research for maintaining a reservoir of challenging research problems. We also acknowledge the suggestions made by three anonymous referees that helped to improve the paper presentation.

References

1. J. Abello, P. Pardalos and M. Resende, editors. *Handbook of Massive Data Sets*, Kluwer Academic Publishers, 2002.
2. J. Abello, A. Bushbaum, and J. Westbrook. A functional approach to external memory algorithms. In *European Symposium on Algorithms*, volume 1461 of *Lecture Notes in Computer Science*, pages 332–343. Springer-Verlag, 1998.
3. J. Abello and J. Vitter, editors. *External Memory Algorithms*, volume 50 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1999.
4. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Proc. 33rd IEEE Symp. on Foundations of Computer Science*, pages 14–23, 1992.
5. S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *J. of the ACM*, volume 45, pages 70–122, 1998.
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*, volume 8, pages 67–71, 1989.
8. F. Glover. Tabu search. Part I. *ORSA J. Comput.*, volume 1, pages 190–206, 1989.
9. J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, volume 182, pages 105–142, 1999.
10. J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.
11. S. Kirkpatrick, C. D. Gellat Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, volume 220, 671–680, 1983.
12. L.S. Pitsoulis and M.G.C. Resende. Greedy randomized adaptive search procedures. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*. Oxford University Press, pages 168–182, 2002.