Finding all maximal cliques in dynamic graphs

Volker Stix

Vienna University of Economics

Department of Information Business

Augasse 2–6

A-1090 Vienna / Austria

volker.stix@wu-wien.ac.at

April 25, 2001

Abstract

Clustering applications dealing with perception based or biased data lead to models with non-disjunct clusters. There, objects to be clustered are allowed to belong to several clusters at the same time which results in a fuzzy clustering. It can be shown that this is equivalent to searching all maximal cliques in dynamic graphs like $G_t = (V, E_t)$, where $E_{t-1} \subset E_t, \quad t = 1, \ldots, T; E_0 = \phi$. In this article algorithms are provided to track all maximal cliques in a fully dynamic graph. It is naturally to raise the question about the maximum clique, having all maximal cliques. Therefore this article discusses potentials and drawbacks for this problem as well.

Key words—maximal clique, dynamic graphs, fuzzy clustering

1 Introduction and motivation

Suppose a gray scale of 10 levels of gray and that the human eye can not distinguish between consecutive shades but between shades having a "distance" of at least 2. Shade number 4 can be identified to belong to the cluster of dark grays whereas shade number 6 belongs to the cluster of light grays. Shade number 5, however, belongs to both clusters at the same

time, because it can neither be distinguished from shade number 4 nor from shade number 6. The nature of perception under resolution constraints leads to *fuzzy clustering* and *maximal cliques* as we will see shortly.

Suppose n objects are given in a measure space with a metric $d(i,j)$, which measures the distance between objects i and j. As usual, $d(i,j) > 0, i \neq j$ and $d(i,j) = d(j,i)$. We define a parameterized relation $\sim_f$ among the n objects as:

$$i \sim_f j := d(i,j) \leq f.$$

It can be seen that this relation is reflexive, symmetric but not transitive. It can be illustrated by a undirected graph $G = (V, E)$, where V is the set of nodes (objects) and E is the set of edges. The number of nodes $|V|$ is equal to n and the number of edges $|E|$ is equal to m. There is an edge $(i,j) \in E$ connecting node i with node j, whenever $i \sim_f j$ with respect to a given f.

Example Suppose the following matrix reflecting the distances between 7 objects.

	1	2	3	4	5	6	7
1	0	7	6	8	2	1	4
2	7	0	3	7	9	8	6
3	6	3	0	6	9	6	7
4	8	7	6	0	1	9	2
5	2	9	9	1	0	2	1
6	1	8	6	9	2	0	2
7	4	6	7	2	1	2	0

Figure 1 illustrates the graphs for the relation of these 7 objects with three different threshold levels f. Note that the distances between nodes in a graphical illustration are usually not correlated to the distances measured by the metric, which is encoded in the matrix. This is reasonable because objects and their metrics often originate from a high dimensional space whereas their graphical representation can only offer a two dimensional projection. Only the relation coded through the edges is important here.

Suppose we do not or even can not distinguish between objects i and j having $d(i,j) \leq f$. This naturally forms sets S of objects of the following structure $\{i \in S \Leftrightarrow d(i,j) \leq f, \forall j \in S\}$. These can

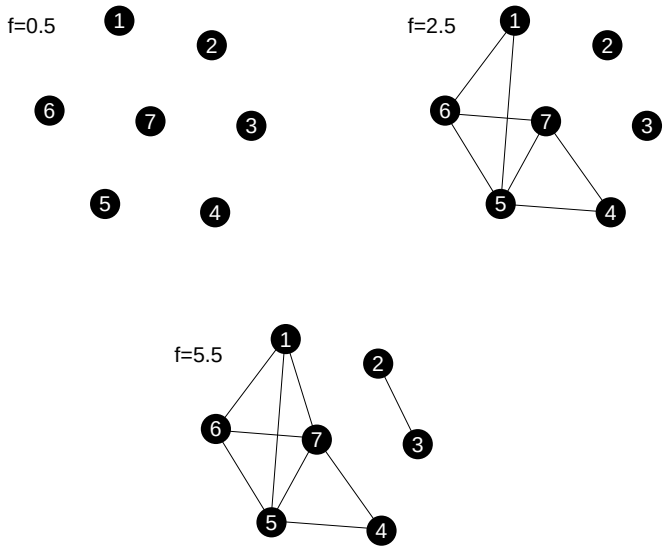

Figure 1: Graphical representations of three different f-levels.

be encapsulated in a sphere with a diameter not larger than f. Each of these sets can be seen to form a new single object (or cluster). The sets highly depend on the threshold level f. If $f = 0.5$ is small, as in the first picture of Figure 1, than every object can be identified. Setting $f = 2.5$ leaves 5 different clusters and with $f = 5.5$ three clusters remain as illustrated in Figure 2.

These sets reflect a set of nodes C in graph theory, where all nodes in C are pairwise connected, thus C is an induced complete subgraph of G. Such a set in graph theory is called a *clique* [1]. A *maximal clique* is a clique, not contained in any larger clique. We are interested in maximal cliques only, since we are interested in the clustering of objects as described before and illustrated in Figure 2.

The series of graphs parameterized by f can be seen as a dynamic graph, which has a constant set of nodes and a changing set of edges. The graphs are defined as follows:

$$G_f = (V, E_f) \quad f \in [0, \infty) \text{ where } E_{f'} \subseteq E_f, f' \leq f; E_0 = \phi. \tag{1}$$

Because the adding/removal process is monotone in f, this dynamic graph problem is an incremental/decremental one. Commonly investigated problems of these graphs focus mainly on connectivity problems as presented in [10], which are of interest in topics like VLSI design, communication networks and so on. The difference here is that clustering does not produce a partition of objects like various spanning or topology tree algorithms on dynamic graphs [11, 15, 9] but produces several sets of undistinguishable objects. This enables the algorithm to let objects belong to different clusters within the same clustering result. The usually forced condition that $\sim_f$ is transitive and thus it is an equality relation is violated here although the requirement of fuzzy clustering appears naturally

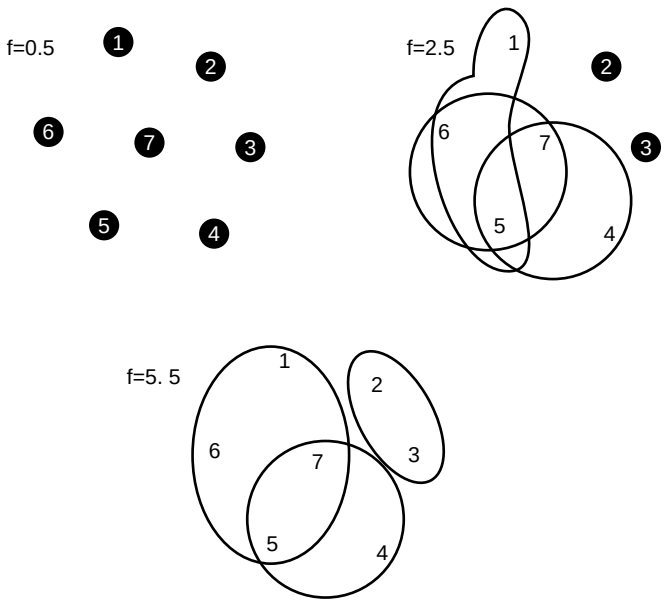

Figure 2: Formation of sets with different f-values.

as motivated above. It can be found especially in any kind of biased or subjective perception areas like e.g. music [7] as discussed later. The ability to create overlapping clusters distinguishes this approach from classical clustering techniques as e.g. described in Hartigan [14] as well.

Technically we are looking for all maximal cliques in a dynamic graph like (1). Bron and Kerbosch [4] provide an algorithm for finding all cliques in an undirected graph. In our paper, however, the information about maximal cliques gained for G_t is used to imply all maximal cliques whenever an edge is added or removed from G_t, avoiding the calculation of the whole problem from scratch. This algorithm will be presented in Section 2 and 4. Section 3 shows an example in music clustering analysis, where this algorithm was successfully applied. Knowing all maximal cliques implies that the largest clique of them, the so called *maximum clique*, is known as well. The maximum clique problem (MCP) is a well-known combinatorial optimization problem and is well discussed in [2]. We will address the MCP in Section 3 as well and discuss potentials and drawbacks of our approach.

In general it is known that the number of maximal cliques can grow exponentially with the order of a graph [17]. A reduction of the maximum clique problem to a standard quadratic optimization problem, first done by Motzkin and Straus [18], yields (in a later regularized formulation) to a one-to-one correspondence of maximal cliques and local optimizers [3] and thus delivers a lower bound for the maximal number of coexisting evolutionarily stable strategies (ESSs) in game theory [20, 8, 5]. Therefore our assigned goal of enumerating all maximal cliques might be for some applications of

theoretical nature only. This is because it is often impossible to enumerate all maximal cliques within reasonable memory and time requirements, simply because there are too many of them. This aspect is discussed in Section 5 and Section 6 illustrates some interesting experiments, which show the potential behavior of the number of maximal cliques in dynamic random graphs.

2 The basic iterative algorithm

The threshold f used in the motivation example above to parameterize the graphs is a real number and therefore not suitable to describe the discrete process of adding or removing of edges. Therefore we parameterize the dynamic graph G by an integer $t = 0, \ldots, T$ such that

$$G_t = (V, E_t) \quad t = 0, \ldots, T \text{ where } E_{t-1} \subset E_t; E_0 = \phi. \tag{2}$$

Thus t counts whenever there is a change in the edges set which is of course implied by f. It can be interpreted as (non-linear) time passing with the increase of f. Let C_t be the set of all maximal cliques of the graph G_t. For $t = 0$ clearly

$$C_0 = \{\{1\}, \ldots, \{n\}\}, \tag{3}$$

where again $n = |V|$ is the dimension of the graph G. C_0 represents the clustering where each object can still be identified. Let $[A]_t, A \subseteq V$ be the set of all cliques $C \in C_t$, such that $C \cap A \neq \phi$. $[A]_t$ represents all clusters where nodes out of A can be found at time t.

2.1 A single step decomposition

With the definitions and notations from the previous section we are able to develop a single step decomposition of the problem. It can be expected that at one single step in time more edges than only one are added (see e.g. Figure 1, where 7 edges are added from $t = 0$ to $t = 1$ and another 2 are added until step $t = 2$). Given a series of Graphs G_t as in (2), one can easily construct a new series G'_t which satisfies the more restrictive condition

$$|E'_t \setminus E'_{t-1}| = 1 \quad t = 1, \ldots, T\ , \tag{4}$$

i.e. one and only one edge is added at each step in time. This construction can be fulfilled by adding $k - 1$ intermediate time stamps whenever $|E_t \setminus E_{t-1}| = k > 1$. The order of these new time stamps can be arbitrarily chosen out of $k!$ possibilities, which can raise the amount of freedom enormously.

We will discuss this issue in Section 5. From now on all dynamic graphs are assumed to satisfy (4) as well.

Lemma 1 *Every node $v \in V$ is element in at least one set $C \in C_t$ for all $t = 0, \ldots, T$.*

The obvious proof is left to the reader.

Lemma 2 *Let $G_t = (V, E_t)$ be an arbitrary graph. Adding one edge (v, w) to G_t can only alter cliques in C_t which contain v or w, i.e. $[\{v, w\}]_t$.*

The proof is clear since cliques which are neither in $[v]_t$ nor in $[w]_t$ can not have an edge to v or w, thus they are not affected by the edge (v, w). Knowing this fact we will enumerate all possible scenarios and deduce the set C_{t+1} from these.

Lemma 3 *Let $G_t = (V, E_t)$ be an arbitrary graph and (v, w) the edge which is added from time t to $t + 1$. Then for all $A \in [v]_t$ and $B \in [w]_t$ the following statements are true.*

1. $[v]_t \neq \phi, [w]_t \neq \phi$
2. $A \setminus B \neq \phi$ *and* $B \setminus A \neq \phi$
3. $|A \Delta B| \geq 2$

Proof: $[v]_t$ can never be empty because of Lemma 1. The second statement is true because A can never be a subset of B (and vice versa) otherwise A would not satisfy the maximality criterion of a maximal clique. The third statement is directly implied by the second one.

Theorem 4 *Let $G_t = (V, E_t)$ be an arbitrary graph and (v, w) an edge which is added from time t to $t + 1$ then:*

1. *All cliques in C_t which are not in $[\{v, w\}]_t$ are in C_{t+1}.*
2. *For all $(A, B) \in [v]_t \times [w]_t$ the following statements are true:*
   (a) *$L = (A \cap B) \cup \{v, w\}$ is a clique and $L \in C_{t+1}$ if L is maximal.*
   (b) *$|A \setminus B| = 1 \quad \Rightarrow \quad A \notin C_{t+1}$; otherwise $A \in C_{t+1}$ if A is still maximal.*
   (c) *$|B \setminus A| = 1 \quad \Rightarrow \quad B \notin C_{t+1}$; otherwise $B \in C_{t+1}$ if B is still maximal.*
3. *The set C_{t+1} is fully determined by 1.) and 2.) statements.*

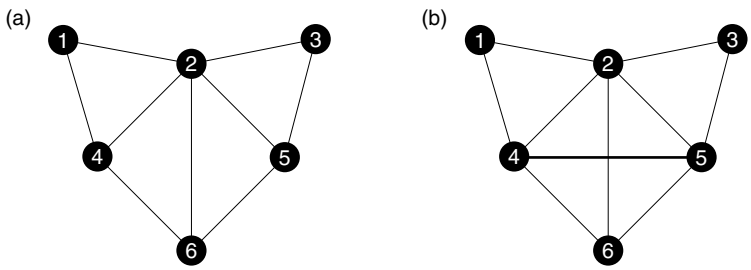

Figure 3: A check for the maximality criterion is necessary.

Proof: (1.) is clear by Lemma 2. Ad (2a.): Any intersection of two cliques is a clique*, so $(A \cap B)$ is a clique. All nodes of A and B are fully connected to v and w resp. thus by adding the last missing edge (v, w), L becomes a clique. It may happen, that this clique is not maximal with respect to the others newly generated in (2.) so only the maximal ones will be in C_{t+1} (see Figure 3). Ad (2b,c.) $|A \setminus B| = 1$ implies with Lemma 3 that $A \setminus B = \{v\}$. Therefore by (2a.) $A \subset L$ and $A \neq L$ so $A \notin C_{t+1}$. Ad (3.) let $M \in C_{t+1}$. If $\{v, w\} \cap M = \phi$ then $M \in C_t$ and by (1.) $M \in C_{t+1}$. If $\{v, w\} \subseteq M$ then remove the edge (v, w) to obtain two cliques $M_1 = M \setminus \{v\}, M_2 = M \setminus \{w\}$. Clearly M_1, M_2 are in $[v]_t, [w]_t$ resp. and therefore M was constructed through (2a.). If only $v \in M$ then $M \in C_t$ and so $M \in [v]_t$. Suppose there was a $B' \in C_t$ s.t. $|M \setminus B'| = 1$ then by (2a.) there was an $M' = (M \cap B') \cup \{v, w\} \in C_{t+1}$. Because $M \subset M'$ and $M \neq M'$, M could not have been in C_{t+1}. So there can not be such a B' and therefore $M \in C_t$ was not removed by (2b.). By symmetry this holds true if only $w \in M$.

Figure 3 shows a simple example, where the check for the maximality criterion is necessary. The edge $(4, 5)$ is added to the graph G_t. The set of all maximal cliques is

$$C_t = \{\{1, 2, 4\}, \{2, 3, 5\}, \{2, 4, 6\}, \{2, 5, 6\}\},$$

and all these cliques are in $[\{4, 5\}]_t$. If we take e.g. $\{1, 2, 4\} \in [\{4\}]_t$ and $\{2, 3, 5\} \in [\{5\}]_t$ then by (2.) of Theorem 4, $\{2, 4, 5\}$ is a clique in G_{t+1} but it is not maximal and therefore it is not in C_{t+1} whereas $\{2, 4, 5, 6\}$ is in C_{t+1} constructed by $\{2, 4, 6\} \in [\{4\}]_t$ and $\{2, 5, 6\} \in [\{5\}]_t$.

*Note that ϕ is a clique as well

2.2 The algorithm

In the following we present the algorithm which generates the series of maximal cliques C_t given the dynamic graph G_t, $t = 0, \ldots, T$.

1. Initialize the set C_0 as in (3); let $t = 0$.
2. Let $C_{t+1} = C_t$.
3. Let (v, w) be the edge to be added at time t.
4. For all $(A, B) \in [\{v\}]_t \times [\{w\}]_t$ do:
   (a) Insert $C = (A \cap B) \cup \{v, w\}$ into C_{t+1} if C is a maximal clique in G_{t+1}.
   (b) Remove A or B resp. from C_{t+1} if A or B resp. are no longer maximal cliques in G_{t+1}.
5. Let $t = t + 1$ and repeat with step 2 until $t = T$.

The maximality criterion for a set A as in step (4.) can be verified as follows:

1. For all $v \in V \setminus A$ do:
   (a) If v is connected to all nodes in A return *false*.
2. Return *true*.

If you prefer set-operations to verify the maximality criterion then for every new set A you are about to add to C_{t+1}, the following should be checked:

1. If A is a subset of any $B \in C_{t+1}$ then A is not added.
2. If A is a superset of any $B \in C_{t+1}$ then A is added and B is removed.
3. If nothing of the above applies then A is added.

For efficiency reasons it is sufficient if the set-manipulation operations from above are not applied to the whole set C_t but only to the newly added or altered sets during time t.

3 Applications

3.1 Fuzzy clustering analysis

As motivated in Section 1, the nature of perception leads to fuzzy clustering and maximal cliques. We can distinguish between two trivial clusterings which coincide with the start and end points of

our dynamic graph. On the one hand we have the clustering C_0 where every object forms a cluster of its own resulting in n clusters and on the other hand we have the clustering C_T where everything forms one single object resulting in one single cluster. The first one assumes the best resolution in perception and the last one the worst. Both are trivial clusters and are normally not desired. In-between, depending on f, the threshold of perception, there are many other different clustering possibilities $C_t, 0 < t < T$. The quality of these clusterings highly depend on the type of objects being clustered and additional requirements. On the one hand the clustering should not have too many clusters and on the other hand the overlapping of the cluster should not be too large, though some overlapping might be desired or even required. Quality will often be a more subjective judgement.

Suppose we have the objects and the distance measure, then we start with G_0 letting the graph grow while increasing the threshold of perception f. The proposed algorithm finds one clustering C_t for each G_t which is implied by the perception threshold f. For each C_t a quality measure (as shortly discussed later in this section) can be applied to decide whether the given threshold f implies a clustering as desired. There can exist more than one "good" clustering. Note that a maximal clique in G_s stays a clique (not necessarily a maximal one) in G_t, for $s \leq t \leq T$, because only edges are added. As a consequence, clusters at some time s are grouped together in larger clusters later. This implies a hierarchic structure of the clusters evolving in time.

We will show here an example taken from melodic clustering analysis, where our algorithm was successfully applied. Details of the work can be found in Cambouropoulos et al. [6]. They aim to cluster melodic segments into significant musical categories by assigning objects (musical pieces) a k-dimensional property vector like e.g. rhythm, tune, and so on. A distance measure based on Hamming distance is defined on this object space, consisting of 47 musical objects. Figure 4 shows the number of clusters found, while the threshold increases. For each clustering, a quality measure should be calculated. Figure 5 shows one simple possible measure, based on the degree of averaged overlapping of the clusters[†]. For each number of clusters (horizontally) the smallest averaged overlapping (vertically) is assigned. E.g. for a clustering of 5 clusters, the smallest overlapping is 159, averaged over 5 clusters results in a measure of 31.8, whereas for a clustering of 40 clusters the average overlapping is only 16.15 . Naturally a smaller number of clusters within one clustering is more desired which should also go into the quality measure and so on. Cambouropoulos et al. decided to use the so called *category utility* measure [12], a measure build on a-priori and a-posteriori

[†]This is measured for each clustering by the sum of all cluster-sizes minus 47 –the minimum required size– averaged by the number of clusters in the clustering

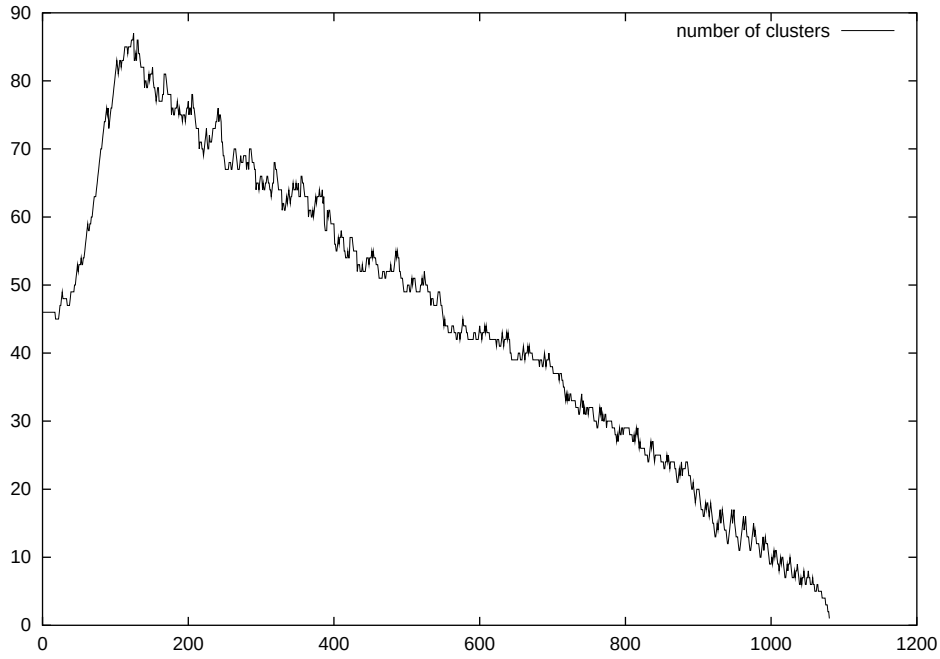

Figure 4: Number of clusters found during construction.

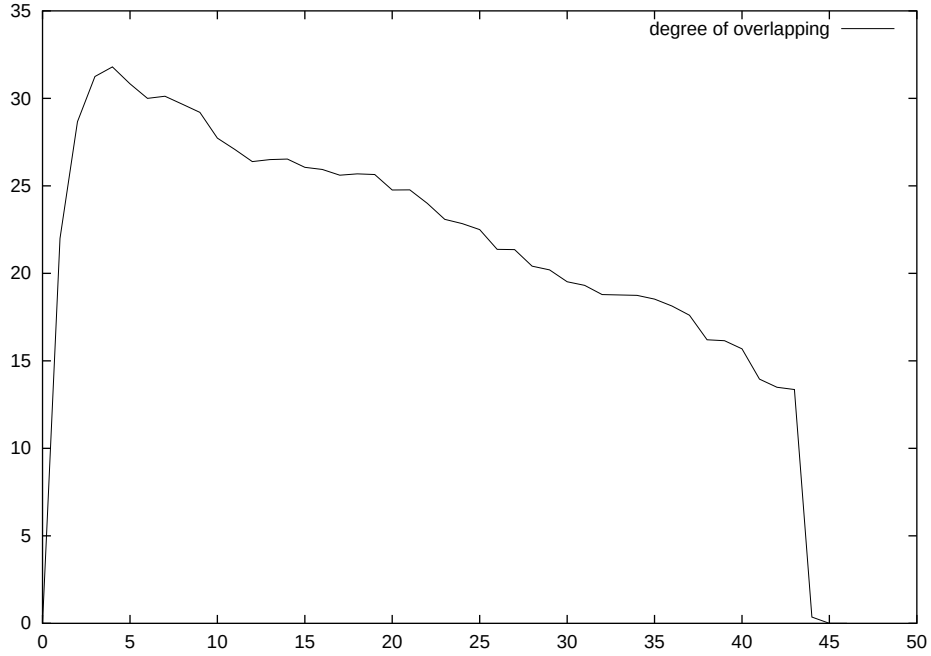

Figure 5: Average overlapping per clustering.

probabilities. It can be seen, that there is a wide range and possibility of measuring quality of a clustering and we will not discuss this issue further here.

3.2 Finding the maximum clique

Finding the maximum clique (i.e. the largest clique) if all maximal cliques of a graph are known is easy. The number of maximal cliques, however, exceed for hard graph instances as presented in [16] often time resources to find all of them and memory resources to store them. A known strategy to make graphs hard to compute with respect to the MCP, is to "hide" the maximum clique behind a huge number of inefficient local solutions, namely maximal cliques. A Moon/Moser graph [17] of dimension 60 has over three billion maximal cliques of size 20. Suppose this graph is embedded in a 100 dimensional graph trying to hide the maximum clique, clearly no computer can track easily this huge number of maximal cliques in random graphs. Section 6 will show some growth illustrations of the number of maximal cliques. Nevertheless this edge-oriented approach is different form the normally node-oriented approaches and can aid in finding the maximum clique as discussed below. A description of the MCP and its applications with over 300 references can be found in [2], as already mentioned in the introduction.

The algorithm in this paper can be used for finding the maximum clique in a graph $G(V, E)$. The only thing which has to be done is the generation of a series of edges to be added to the empty graph G_0 until the desired graph G is constructed. Even more, it is possible to render the maximum clique of a subgraph $\underline{G} = (V, \underline{E})$ of G, where $\underline{E} \subseteq E$. Thus $\underline{G}$ is constructed by removing some edges from G. If $\underline{G}$ is simple with respect to the MCP, e.g. a perfect graph [13], the maximum clique and all maximal cliques (or just large ones) can be extracted. After that the original graph G and all its cliques can be reconstructed (or at least better approximated) using our algorithm. The same idea can be followed with a supergraph $\overline{G}$ of G, where $\overline{E} \supseteq E$ constructed by adding edges to G. Again from this simpler graph $\overline{G}$ the maximum clique of G can be constructed (or approximated) backwards by removing edges, which is discussed in the following section. We are currently working on such sub- and super-graph approximations using transitive orientable (TRO) graphs [19].

In general, for the MCP, starting with G_0 we can add edges theoretically in $|E|!$ different sequences because we have no distance metrics which constrains us in adding edges. Experiments in Section 6 show, that there is a difference between edge adding strategies and this will be discussed in Section 5. After the algorithm finishes, of course, all strategies are leading to the same maximal clique set C_T but in-between there are differing sets.

4 A reverse approach

4.1 The single step decomposition

Dynamic graphs deal with the adding and removal of edges, therefore we offer here an algorithm for decremental graphs as well. It can be viewed as starting with a very large threshold level f, where no single object can be distinguished. Thus the reverse approach starts with a fully connected graph. For fuzzy clustering this approach can be desired when a maximum level of precision is defined and therefore rather coarse clusters are required. For the MCP it reduces the number of time steps for dense graphs and additionally enables the approximation from above through the discussed graph $\overline{G}$. There are e.g. 4455 edges in a graph with density 0.9 and dimension 100 out of 4950 possible edges inside graphs of dimension 100. Therefore it can be decided either to add 4455 edges from an empty graph or to remove 495 edges from a fully connected one. Additionally by removing edges the maximal cliques can only shrink. When aiming for the maximum clique, it is possible to reject inefficient maximal cliques in C_t if the sizes of these cliques are below a certain threshold. This threshold can be gained by known or pre-calculated lower bounds for the MC. If we know e.g. a maximal clique of order 15, we can reject all maximal cliques of order 14 and smaller. This was not possible in the edge adding approach. The algorithm also continually delivers a decreasing upper bound for the MCP. There are, however, drawbacks of this reverse algorithm. By looking more precisely at one removing step, you might expect an exponential growth of maximal cliques (especially at the beginning) with a basis close to 2. At $t = 1$ the number of cliques is two. It is then very likely that the next edge which is removed belongs to both maximal cliques which doubles the number of cliques again and so on. This drawback has the same source as the drawbacks mentioned before, namely the possible huge number of maximal cliques in a graph.

The basic idea of the algorithm is identical only a few adaptations must be considered. The same notation as before will be used with the small difference that G_0 now is a fully connected graph and therefore

$$C_0 = \{\{1, \ldots, n\}\} \tag{5}$$

is the one and only clique at time 0. With increasing time we subtract edges instead of adding them, again only one edge at one step in time, until we reach the graph G_T.

We have to adapt the theory from above to the removing process before presenting the algorithm for the reverse approach.

Lemma 5 *Let C be a maximal clique then only the removal of edges which are connecting two nodes in C can destroy the clique property of C.*

The proof is obvious. Further we define $[(v,w)]_t = \{C \in C_t : \{v,w\} \subseteq C\}$ as the subset of C_t where both nodes v and w are in one clique.

Theorem 6 *Let $G_t = (V, E_t)$ be an arbitrary graph and (v,w) an edge which is removed from time t to time $t+1$ then:*

1. *All cliques in C_t which do not contain both the nodes v and w are in C_{t+1}.*
2. *For all $A \in [(v,w)]_t$ the following statements are true:*
   (a) *$A \notin C_{t+1}$.*
   (b) *$L = A \setminus \{v\}$ is a clique and $L \in C_{t+1}$ if L is maximal.*
   (c) *$L = A \setminus \{w\}$ is a clique and $L \in C_{t+1}$ if L is maximal.*
3. *The set C_{t+1} is fully determined by those 4 statements.*

Proof: (1.) is clear by Lemma 5. (2a.) is clear because the edge (v,w) is missing in the clique A. Ad (2c+d.) A was a maximal clique and is therefore fully connected to all elements of A except for the node v and w resp. Ad (3.) Let $M \in C_{t+1}$. If neither v nor w is in M then $M \in C_t$ and by (1.) $M \in C_{t+1}$. If $v \in M$ then either $L = M \cup \{w\}$ is a clique then $L \in C_t$ and by (2a.) $M \in C_{t+1}$. If, however, L is not a clique then $M \in C_t$ and by (1.) $M \in C_{t+1}$. For symmetry reasons it holds also true if $w \in M$. $\{v,w\} \subset M$ is not possible because M is a clique.

As before it might happen, that a clique but not a maximal clique is constructed through step 3 or 4 due to the parallelism of the removal process (more than one clique is altered). Figure 6 illustrates this with a simple example. The edge $(4,5)$ is removed from the graph G_t. The set of all maximal cliques at t is

$$C_t = \{\{1,2,4\},\{2,3,5\},\{2,4,5\}\} ,$$

and $\{2,4,5\}$ is in $[(4,5)]_t$. By Theorem 6 (2b.) and (2c.) $\{2,4\}$ and $\{2,5\}$ are cliques in G_{t+1} but they are not maximal and therefore they are not in C_{t+1}.

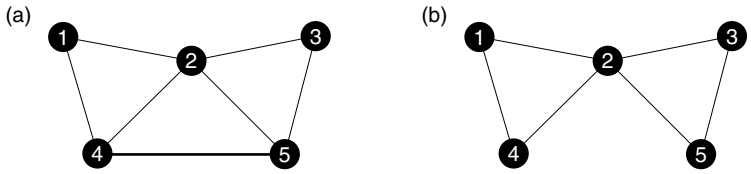

Figure 6: A check for the maximality criterion is necessary.

4.2 The algorithm

The following algorithm can be formulated using the findings from above.

1. Initialize the set C_0 as in (5); let t=0.
2. Let $C_{t+1} = C_t$.
3. Let (v, w) be the edge to be removed at time t.
4. For all $A \in [(v, w)]_t$ do:
   (a) Insert $C = A \setminus \{v\}$ into C_{t+1} if C is a maximal clique in G_{t+1}.
   (b) Insert $C = A \setminus \{w\}$ into C_{t+1} if C is a maximal clique in G_{t+1}.
   (c) Remove A from C_{t+1}.
5. Let $t = t + 1$ and repeat with step 2 until $t = T$.

5 Generating an optimal series of graphs

We have already mentioned in Section 2, that at some stages more than one edge is added a the time, which gives us freedom in choosing their order. For example in Figure 1, seven edges are added at the time which results in 5040 possible orderings. Even more degree of freedom arises when constructing all maximal cliques of the final graph G_T without constrains as proposed for the MCP, which leads to $|E_T|!$ possibilities. It can be expected that not all of the orderings have equal computational effort although the final result is the same.

As example we consider the simple graph given in Figure 7. This graph represents our final graph G_T. Figure 8 illustrates 3 different construction approaches. Each of them is in a separate column marked with a circle, a box and a cross. The rows represent these graphs at times $t = 0, \ldots, 6$. Which of these different edge orderings should be thought to be more efficient? Is there a difference? Figure 9 reveals the number of maximal cliques (vertical) for each of the three graphs in each step in

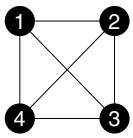

Figure 7: A simple Graph.

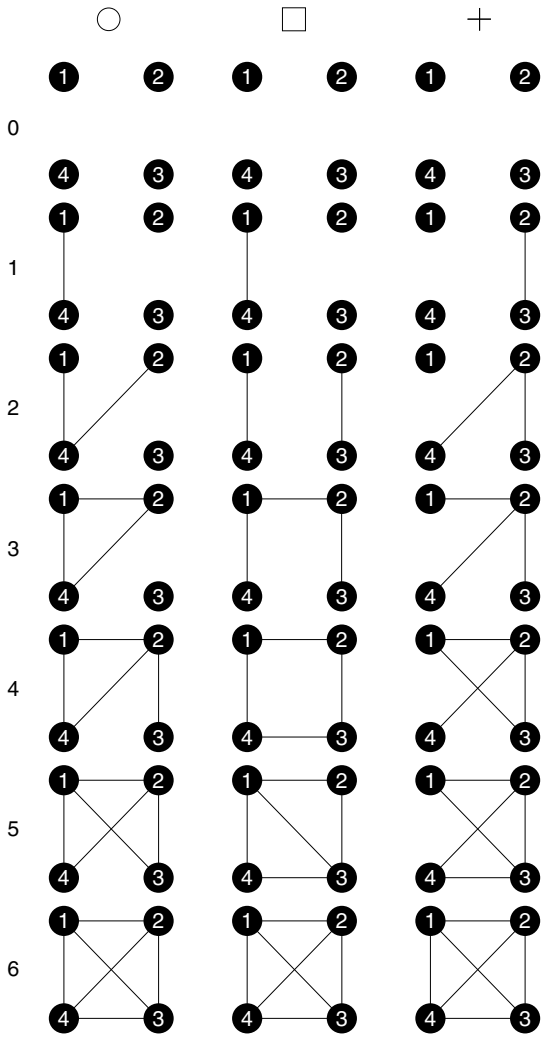

Figure 8: Different constructions of a simple Graph.

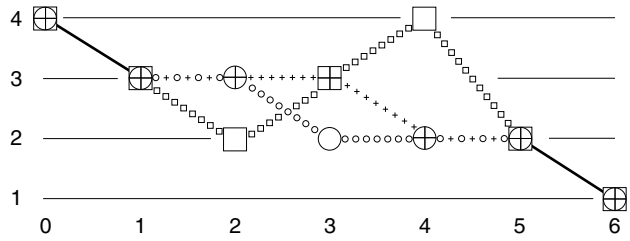

Figure 9: Different constructions of a simple Graph.

time (horizontal) during construction. The first and the last construction strategy (circle and cross) always reduce the number of maximal cliques which is favorable because of run-time and storage requirements. The second one (box) can be supposed as an inefficient construction although it has fewer cliques at $t = 2$ than the others but it increases at $t = 4$ to 4 maximal cliques. The first one (circle) should be considered to be the best one. It exceeds the third one (cross) because it reduces its numbers of cliques earlier and stays on this level. The first approach had 17 maximal cliques during its lifetime the second one 19 and the third one 18, respectively. This is, however, a very simple example and it can be imagined that in larger graphs a lot of construction methods like our second approach exist, i.e. that the number of cliques reduces earlier compared with other strategies but increases later again. In this example $6! = 720$ different construction possibilities can be found in principle. Because of isomorphism there exist, however, only one graph with one edge, two graphs with two, three graphs with three, two graphs with four and one graph with five edges. All of them are present in Figure 8. For every t, these G_t which have the minimum number of maximal cliques could be selected and then verified if this selected series of graphs is a valid construction method in accordance to the definition of the dynamic graphs considered in (2). This is not possible in our simple example. Problems become more complex in larger graphs.

6 Experimental results

This section illustrates the number of maximal cliques obtained via different adding and removal strategies. It is subdivided into two parts following the theory of the preceding sections.

For the experiments random graphs of different orders and densities were constructed. In the illustrations below random graphs of order 50 were used to outline the different adding and removing strategies. Graphs with equal densities are identical in both subsections. The figures plot the evolving time (horizontally) and the number of maximal cliques C_t (vertically).

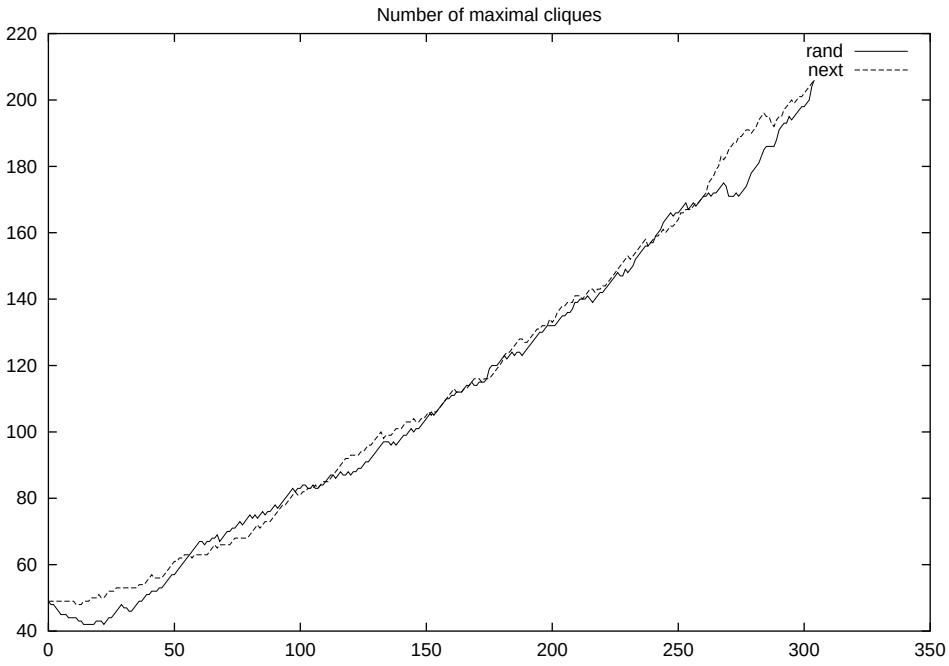

Figure 10: Adding strategies in a random graph of order 50 and density 25%.

6.1 Adding Edges

Figures 10–12 show two different simple adding methods. The first one, labelled with "next", iterates through the adjacency matrix and adds the edges as they appear. The second method, labelled with "rand", adds edges randomly. We compared these methods in random graphs with densities 25%, 50% and 75% respectively. It can be seen that these two strategies perform almost the same, even if it seems that the random adding strategy is a little bit superior. This advantage can also be observed for the MCP within at least one outcome, namely the 25% case. There, the random strategy found the maximum clique size after 58% of the iterations were completed, whereas the "next" strategy needed 79%. This result however holds only for the 25% case whereas in the other cases the maximum clique was found after approximately the same amount of time. It can be argued that it is not surprising, given random graphs, that the performance of the two strategies is almost identical. The next section, however, will contradict this intuition. In Figure 12, the fast growth of maximal cliques in dense graphs can nicely be observed (8823 cliques).

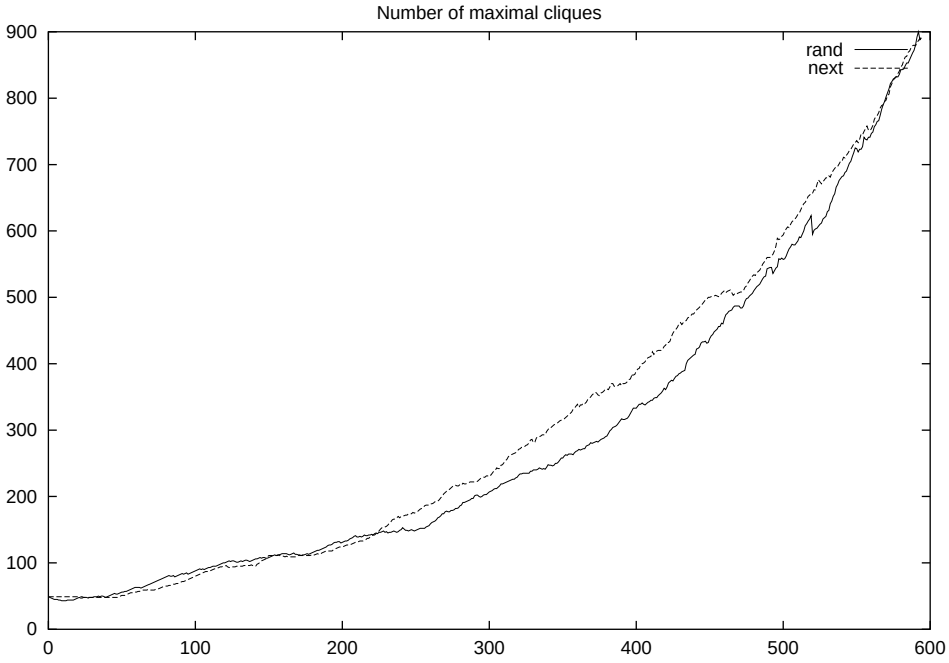

Figure 11: Adding strategies in a random graph of order 50 and density 50%.

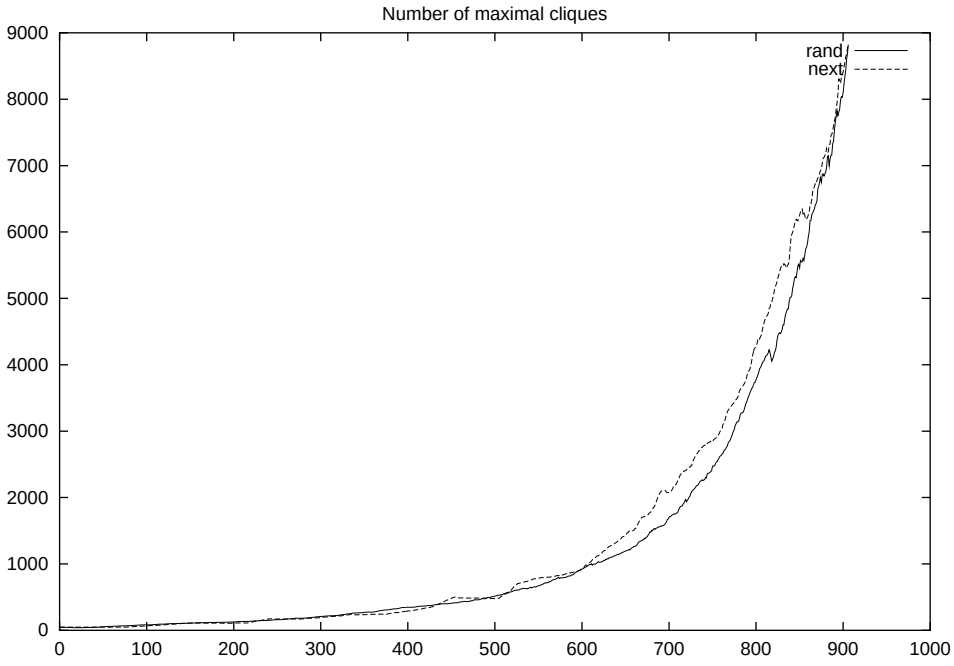

Figure 12: Adding strategies in a random graph of order 50 and density 75%.

6.2 Removing Edges

In these experiments three simple different removing methods were tested. The first one tries to eliminate large cliques inside G_t, in order to obtain upper bounds for the MCP faster. In order to achieve this, one edge (the first one found) inside one largest clique (the first one found) in C_t is removed in each iteration. This approach is labeled in the figures 13–16 with "down". The second method removes edges as they appear in the adjacency matrix and the third one removes edges randomly. They are labelled "next" and "rand" following the notation of the adding strategies before. Random graphs of densities 90%, 75% and 50% resp. were used. It can be observed that the paths of the different orderings are very different among the strategies. Here the next-strategy seems to be the best one. The rand-strategy seems to be unpracticable for the removing process. In Figure 14 the exponential growth for that strategy at the beginning of the removal can be seen as mentioned in the previous section. This strategy was removed in the Figures 15–16 to obtain a better scaling.

The next-strategy needed for these three graphs on average 95% of the iterations until the maximum clique size was found the, down-strategy needed only 84%. It should be additionally noticed that the down-strategy not only found the maximum clique size after 84% of the iterations, but also proved that the MCP is solved at that time. The next-strategy, however, must render all iterations and after that it can prove that the clique found after 95% of the iterations was the maximum clique. The down-strategy on the other hand needs more storage requirements because it produces more cliques during its lifetime.

6.3 Comparison

In our experiments, dense graphs (75% and 90%) were faster rendered with the removing algorithm, sparse graphs (25% and 50%) with the adding approach. It should be supposed that for the 50% graphs both algorithms work similar but comparing Figure 11 with Figure 16 it can be seen, that the number of cliques is almost monotonic increasing for the adding procedure, which is not the case for the removing strategy, especially for the down-strategy. The adding method seems to be more smooth in general. One reason could be that during the adding process the graph is getting denser and thus more complex with respect to the given problem. The removal process, however, starts with very dense and thus complex graphs and ends up with simpler ones. Therefore the number of maximal cliques is for the first one increasing and for the latter one decreasing, which can be observed

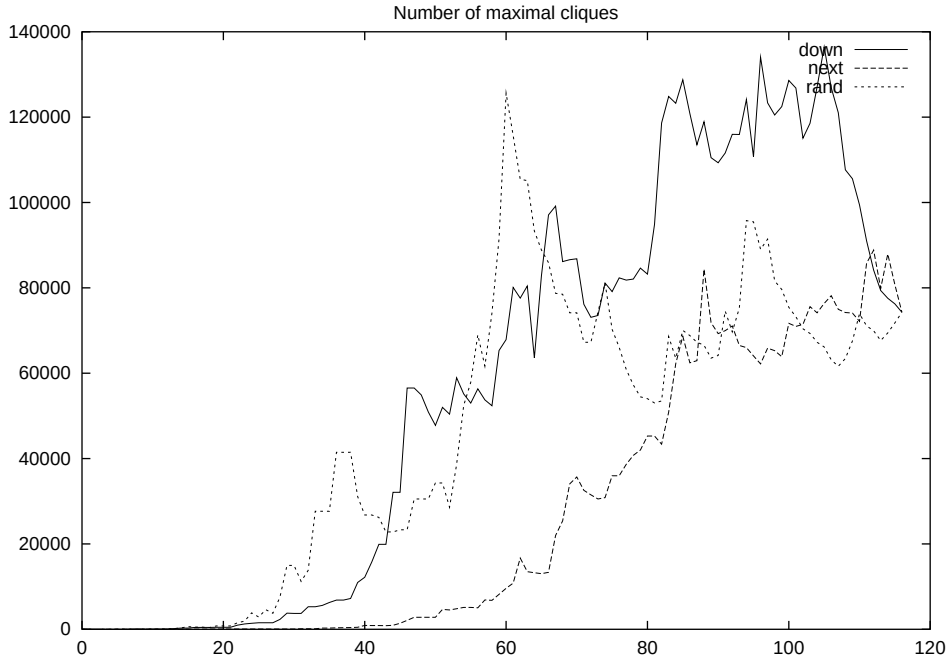

Figure 13: Removing strategies in a random graph of order 50 and density 90%.

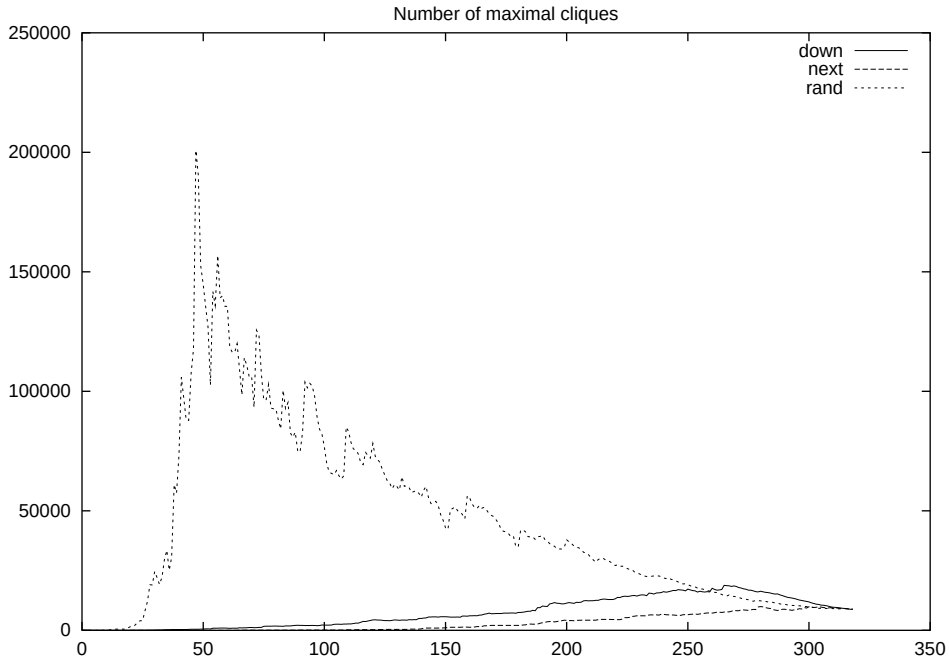

Figure 14: Removing strategies in a random graph of order 50 and density 75%.

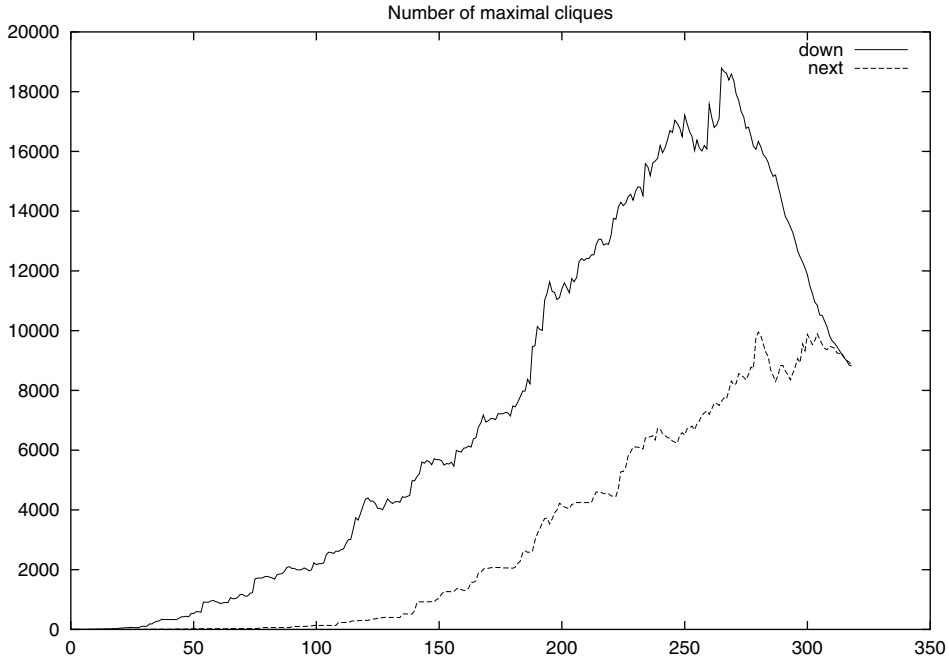

Figure 15: Removing strategies in a random graph of order 50 and density 75%.

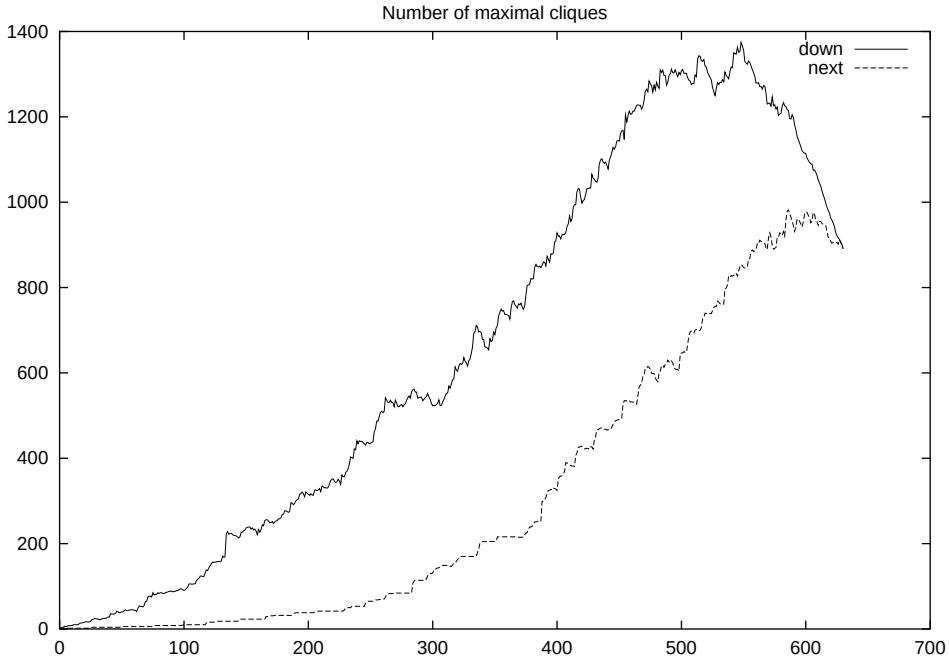

Figure 16: Removing strategies in a random graph of order 50 and density 50%.

at the end of all removing-experiments.

7 Conclusion

This paper provides two algorithms which keep track of all maximal cliques inside dynamic graphs. The purposes of these algorithms are on the one hand to obtain a fuzzy clustering of objects in a measure space with additional constraints on perception. There it additionally provides a hierarchical structuring of these objects as well. On the other hand the algorithms can aid in finding and approximating the maximum clique of a graph. The article empirically shows that there are differences in complexity concerning the ordering of incremental or decremental edges. In dynamic graphs this observation is hard to explain theoretically because of complex dependencies and isomorphic graphs. For clustering applications, where the number of simultaneous changing edges are not so big, it seems to us that these effects can be neglected.

References

[1] B. Bollobás. *Modern graph theory*. Springer, New York, 1998.

[2] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, volume Suppl. Vol. A:4, Boston, MA, 1999. Kluwer Academic Publishers.

[3] I. M. Bomze and V. Stix. Genetic engineering via negative fitness: Evolutionary dynamics for global optimization. *Annals of Oper. Res.*, 89, 1999.

[4] C. Bron and J. Kerbosch. Algorithm 457: Finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973.

[5] M. Broom, C. Cannings, and G. T. Vickers. On the number of local maxima of a constrained quadratic form. *Proc. R. Soc. Lond.*, 443:573–584, 1993.

[6] E. Cambouropoulos, A. Smaill, and G. Widmer. A clustering algorithm for melodic analysis. *Proceedings of the Diderot'99*, 1999.

[7] E. Cambouropoulos and G. Widmer. Melodic clustering: Motivic analysis of Schumann's Träumerei. In *Proceedings of JIM 2000*, Bordeaux, 2000.

[8] C. Cannings and G. T. Vickers. Patterns of ESSs II. *J. theor. Biol.*, 132:409–420, 1988.

[9] D. Eppstein. Clustering for faster network simplex pivots. *Proc. 5th ACM-SIAM Symp. Discrete Algorithms*, pages 160–166, 1994.

[10] D. Eppstein, Z. Galil, and G. F. Italiano. *Algorithms and theory of computation handbook*, chapter dynamic graph algorithms, pages 8–1—8–25. CRC press, New York, 1999.

[11] Z. Galil and G. F. Italiano. Fully dynamic algorithms for 2-edge-connectivity. *SIAM J. Comput.*, 21:1047–1069, 1992.

[12] M. A. Gluck and J. E. Corter. Information, uncertainty, and the utility of categories. *Proc. 7th Ann. Conf. of the Cognitive Science Society*, 1985.

[13] M. C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, New York, 1980.

[14] J. A. Hartigan. *Clustering Algorithms.* Wiley series in probability and mathematical statistics. Wiley, New York, 1975.

[15] M. R. Henzinger and V. King. Maintaining minimum spanning trees in dynamic graphs. *Proc. 24th Int. Coll. Automata, Languages and Programming*, pages 594–604, 1997.

[16] D. S. Johnson and M. A. Tricks (editors). Cliques, coloring and satisfiability: Second dimacs implementation challenge. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 26, American Mathematical Society, Procidence, 1996.

[17] J. W. Moon and L. Moser. On cliques in graphs. *Isr. J. Math.*, 3:23–28, 1965.

[18] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, 17(4):533–540, 1965.

[19] V. Stix. Approximations and solutions of the maximum clique problem using perfect sub- and supergraphs. Working paper, Department of Information Business, Vienna University of Economics, 2002.

[20] G. T. Vickers and C. Cannings. Patterns of ESSs I. *J. theor. Biol.*, 132:387–408, 1988.