A Parameterized Greedy Algorithm for Cluster Editing

Peter E. Shaw[1], Frans A. Henskens[1], Michael A. Langston[2], and Michael R. Hannaford[1]

[1] School of Electrical Engineering and Computer Science, The University of Newcastle, Australia pshaw@cs.newcastle.edu.au
[2] Department of Computer Science, Carleton University, Ottowa, Canada

Abstract. Computer technology has allowed the business and research communities to amass huge amounts of data in electronic form. The data per se is of little use. What owners of the data require is analysis of its content, resulting in information that can then be used, for example, to influence future decisions or solve problems. This paper presents a new algorithm for automated processing of data sets to produce information from the data, or to indicate that no such information exists. The techniques are proven to be mathematically correct, and examples are used to demonstrate application of the technique.

1 Introduction

Computer technology has allowed the business and research communities to amass huge amounts of data in electronic form. The data per se is of little use. What owners of the data require is analysis of its content, resulting in information that can then be used, for example, to influence future decisions or solve problems.

One approach to use of data for decision-making is to display the data in a form that assists observers to identify useful trends or features (e.g. visualisation [1]). It has been observed that display of large amounts of complex data actually causes further confusion on the part of the observer, leading to reduced ability to make appropriate decisions. What is preferable is that the computer performs initial analysis of the data, providing the decision-maker with post-analysis information of reasonable size. This approach supports, rather than hinders, decision making [2].

In practice, automated analysis of many data sets has proven to be non trivial. Indeed, work on such problems typically determines that they can be classified as intractable; meaning that use of algorithms to find exact solutions cannot be expected to yield the solutions in measurable time [3]. For example, in high school we learned an algorithm that always found the roots (or showed there were no Real roots) for quadratic polynomials, but there is no such generic algorithm to find the roots of polynomials of degree > 4. The fact of intractability led to a belief that it may be possible to obtain increasingly close approximations to solutions using Approximation Algorithms [4], but this was shown to

be infeasible for most of these cases [5]. This realisation created areas of pursuit such Neural Networks [6], Heuristics [7] and Genetic Algorithms [8] which use various motifs in an attempt to find solutions in reasonable time. The problem with the latter techniques is that it is not possible to define an upper bound on variance of the found solution from the optimum solution.

The data from which information is required is often very valuable, both inherently and as a result of the monetary and time costs of collection. It has been argued that it is inappropriate to be satisfied with imperfect information from data of such high value [9]. Fixed Parameter Tractability (FPT) [10] attempts to directly arrive at exact solution(s) by rephrasing the question in a form that can be solved in reasonable time, producing either an optimum solution or showing that no reasonably-sized solution (of size less than some fixed parameter) exists. For example, a generic problem may state "how can phone booths be most efficiently positioned in this map so that no resident has to walk more than one block to a phone?" According to FPT, this intractable problem would be reworded as "can at most 20 phones be positioned on this map so that no resident has to walk more than one block to a phone". The number 20 is the parameter, and changes the generic problem into one that is bound, and for which a solution can be found, either revealing the answer or informing that the problem is impossible to solve (i.e. because not enough phone booths have been provided - NOT because of algorithmic intractability).

It is convenient to express many problems by using graphs, which can be expressed for computer-based analysis in terms of entities (vertices) and the relationships between them (edges). These edges may be undirected or directed, and may have weights associated with them indicating the strength of the relationship between the connected entities. Problems expressed in this form may be simplified by considering groups of connected vertices (called clusters). Clusters may be classified in terms of the level of connectivity of the member vertices. A cluster for which every vertex is connected to every other vertex in the cluster is known as a clique [11]. If a cluster has a high degree of connectivity, then the entities in the cluster are clearly all inter-related. This may reflect, for example, collaboration in the case of authors, or insider trading in the case of investors. Identification of cliques, or of clusters with high level connectivity, can be extremely useful when attempting to derive information from graphs [12].

The following section introduces and explains Cluster Editing, the analysis of the graph to determine the effects of removal or insertion of edges on the existence of highly connected clusters or cliques.

2 Cluster Editing

The idea behind cluster editing comes from a realisation that complex structures (the graph) are built from components (clusters) comprising closely related subcomponents (dense sets of edges between vertices) and between which there may be looser relationships (few edges between vertices belonging to different clusters). For example, it is generally accepted that the human body is constructed from systems such as the respiratory, muscular, skeletal, nervous, circulatory, etc. Were the physical structure of the human body to be represented as a graph,

the components of each of these systems would be highly connected, but connections would also exist between components belonging to different systems. For example there are connections between components of the muscular and skeletal systems, and (perhaps arguably) connections between components of the circulatory system and all of the others.

Cluster editing is an automated process that attempts to identify:

- Non-existent edges whose creation would create cliques. The consequence of this analysis is that the pre-insertion cluster must have been connected at a significantly strong level.
- Edges that represent interactions between highly connected clusters, but each of whose individual existence does not indicate strong connection. The consequence of this analysis is that such edges represent interaction between subsystems rather than membership of the same subsystem.

A further issue is the inevitable occurrence of errors in the data collection itself, and in the transformation of the collected data into graphical form. Such errors can result from diverse sources such as limitations of collection equipment, selection of sampling thresholds, sampling of source data, and so on. These errors manifest themselves as existent edges that should not be present (noise) or missing edges whose non-existence is accidental rather than truly representative of the sample. It must be mentioned that a third class of edges is also identified by cluster editing. These edges neither form subsystems nor represent interaction between subsystems. Their existence is explained in the description of noise.

In practical use of cluster editing, initial analysis sometimes quickly identifies clusters for which internal connectivity is extremely high and external connectivity is almost non-existent. While not entirely discarding knowledge of such (existing external or non-existent internal) connections, it facilitates analysis of the remaining graph if such clusters are temporarily removed after identification and recognition of the relationships they represent.

FPT cluster editing limits the options available during simplification of this kind. Each change to the graph (e.g. insertion of an edge or removal of an edge) is termed an edit, and each set of edge manipulations is called a reduction [10]. The ultimate aim is to produce disjoint cliques. A growing set of reduction rules is being defined for use when decisions are made on possible edge edits. For example [9] describes a rule for reducing a graph when it is bi-connected or tri-connected. A fixed parameter (positive integer) is associated with the to-be-solved problem, and this parameter serves as a limit on the number of edits that are permitted in the quest for a solution. In effect, every reduction decrements the current parameter value by the number of edits involved in the reduction. If the current parameter value becomes zero (or negative) before a solution is found, the technique guarantees the problem cannot be solved in the parameter-defined number of edits. For an FPT problem the result of applying the reductions is that the graph is reduced to a form known as the kernel. The size of the kernel is such that it can be expressed as a function of the parameter. This means that the time required to exhaustively search the remaining graph for solutions has become a function of the parameter rather than a function of the initial size of the graph [13].

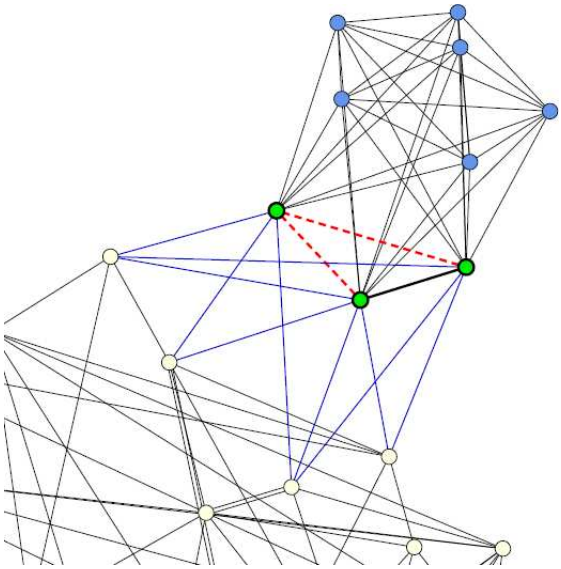

Fig. 1. Crown Decomposition

The Crown Rule, originally described in [14], takes a reduction rule that had been previously only applied to a fixed number of vertices, and generalises its application to an arbitrary number of vertices. The cluster editing form of the Crown Rule, described in [15] and depicted in Figure 1, is a reduction rule that partitions a graph into three disjoint sets of vertices. The first of these sets, known as the crown (depicted as black circles), comprises vertices that are fully connected with each other. The second set, known as the (depicted as grey circles), logically sits between the crown and the rest of the graph. Its vertices are fully connected to those in the crown, but not necessarily to the vertices in the head or the rest of the graph (depicted as open circles). Included in the Rule is a stipulation on the relative sizes of the crown and the head that prevents construction of an under-sized crown compared with the size of the head. Ultimately, application of the Crown Rule identifies a set of edits that produce a disconnected clique (the crown and the head) that can then be removed from further consideration in the quest for a solution.

Recent hybrid techniques use FPT cluster editing in combination with other techniques (e.g. improved approximation [16], heuristics [2], or use of improved approximation to influence choice of reduction rules [15, 17]).

Identifying locations for potential application of reduction rules, and then choosing the appropriate rule to apply, presents a challenge. This issue is further discussed in [18, 19]. A techniques known as Greedy Localization starts by iteratively processing and annotating the graph (to produce a maximal packing), after which it either identifies locations for application of reduction rules or determines that the question cannot be solved for the given parameter [20, 21]. The following section describes a new algorithm that uses a hybrid technique combining the greedy localization and the cluster editing Crown Rule to obtain an improved kernel for the cluster editing problem.

3 Improved Kernelization

In this section we consider the situation in which we are given an undirected graph G(V, E) where V is the set of vertices and E is the set of edges of the graph. We are also given a non-negative integer parameter k. The question to be answered, the k-Cluster Editing Problem, can be expressed as:

"Can we transform G by deleting and adding at most k edges into a graph that consists of a disjoint union of cliques?"

Solving this problem is achieved by initially reducing the graph by applying the Crown Reduction rule. This rule was first used in [15]. This will produce a reduced form of the graph, which will then be further processed. The Crown reduction used is unique in that it produces a smaller kernel than can be achieved by previous techniques, and it is fast enough that its use can be interleaved with the second phase, namely the search of the resultant reduced graph.

Definition 1. *A Crown Decomposition of a graph $G = (V, E)$ is a tripartition of the vertices of the graph into three sets H, C and X with the following properties:*

1. *C is a clique*
2. *Every vertex of C is adjacent to every vertex of H*
3. *H is a cutset, in the sense that there are no edges between C and X, the rest of the graph: the set of vertice in $V \setminus (C \cup H)$*
4. *$N = \{for\_all\_v \in V - (C - H) \exists u \in H$, with corresponding uv in $E\}$.*

Reduction Rule 1 The Cluster Crown Reduction Rule: *If (G, K) is an instance of the Cluster Editing problem, and G admits a cluster crown decomposition (C, H, N, X) where $|C| \geq |H| + |N| - 1$ then replace $G, k)$ with (G', k') where $G' = G - C - H$ and $k' = k - e - f$, where e is the number of edges that need to be added between vertices of H in order to make the union of C and H into a clique and f is the number of edges between H and N.*

We refer the reader to [15] for the proof of this rule.

4 The proof of kernel *k*-Cluster Editing

The following description uses the terminology:

- P_3 is a path with 3 vertices and 3 edges.
- C_i is the set of vertices in the i^{th} clique.
- Similarly H_i, N_i and X_i are the set contain it the i^{th} crown decomposition.
- $V(P)$ is the set oc vertices in the packing P.
- $< O >$ is the graph induced by the set of vertices of O.

4.1 Kernelization and Boundary Lemmas

Let $c_i = |C_i|$, $h_i = |H_i|$ $n_i = |N_i|$ and $x_i = |X_i|$

Before giving the preprocessing algorithm we make the following observations.

Observation 1 *If their exists a maximal packing P of vertex pair disjoint P_3's (paths of length 2) in G of size $|P| > k$ then (G, k) is a* No *-instance for* k-Cluster Editing.

In Table 1 we describe a polynomial $O(nE)$ time preprocessing algorithm which we apply to an instance (G, k) of k-Cluster Editing.

Step 1.	Generate a maximal packing of vertex pair disjoint P_3's. Let $O = V \setminus V(P)$ (We will argue in Lemma 2 that P implies that O is either the empty set of consists of disjoint cliques)
Step 2.	From observation 1 if $\|P\| > k$ answer No and halt.
Step 3.	For each of these disjoint cliques C_i in O determine the adjacent vertices in $V(P)$ and hence a cut-set H_i that separates it from the rest of the graph. To do this:
Step 3a.	Choose one vertex from each of the disjoint cliques and consider its neighbors. Those neighbor not in $V(P)$ are also in C_i and the others belong to H_i. Determine the size of an alternative packing P' (described in detail below).
Step 4.	If $\|P'\| > k$ then answer No and halt. If $\|G\| > 8k$ then by Lemma 1 then the instance (G, k) is a No instance Return No

Table 1. Kernel Algorithm

4.2 Kernelization and Boundary Lemmas

Lemma 1. *Algorithmic Boundary Lemma If $|V(G)| > 8k$ then the preprocessing algorithm with either decide the instance of (G, k) or it will reduce it.*

Proof by contradiction will follow.

Lemma 2. *Algorithmic Boundary Lemma If $G = (V, E)$ is a* Yes*-instance of the* k-Cluster Editing*problem for parameter k, a* No *-instance for parameter $k+1$, and G is reduced after applying the Crown Rule using the algorithm given in Table 1 , then $|V| \leq 8k$.*

Proof. 2 Assume in contradiction to Lemma 2 that $|V(G)| > 8k$, but that the algorithm has neither decided the instance (G, k) nor reduced it.

After the preprocessing algorithm has run, $V(G)$ is partitioned as follows:

- $V(P)$, the vertices in the maximal packing P of vertex pair disjoint $P_3's$ in G.
- $O = V \setminus V(P)$, the vertices not present in P.

Structural Claims:

Claim 1 *$< O >$ is a union of disjoint cliques.*

Claim 2 *Each maximal clique in C_i in $< O >$ induces a crown structure (C, H, X) in G. Furthermore $H \subseteq P$.*

Proof of Claim 2. Let C_i $0 \leq i \leq m$ be the independent cliques in $< O >$. Let H_i be those vertices in $V(P)$ adjacent to C_i. Suppose $\exists$ a pair of edges (u, v) such that $u \in C_i, v \in H_i$ *and* $(u, v) \notin E$ But as $u \in H, \exists w$ adjacent to u. Hence, *exist* a $P_3 = (u, w, v) \notin P$ which contracts the assumption that P is maximal.

Claim 3 *$|V(P)| \leq 3k$.*

Proof of Claim 3. For every $P_3 \in P$ one edge need or removed. However, as none of these P_3's share a pair of vertices at least one edit is needed for each. Thus $|P| \leq k$. But each P_3 can contribute at most 3 vertices.

Claim 4 *If u, v belong to different cliques in $< O >$, then u, v don't share a common neighbor in P*

Proof of Claim 4. Suppose, $u, v \notin P$ thus, u, x, v cannot share common vertex pairs with any P_3 in P. However, as u, v belong to different cliques there is no edge between them so $p_x = (u, x, v)$ is also a P_3. But, this contradicts the assumption that P is maximal we can increase P_3 by adding p_x.

Claim 5 *The heads of the crown structures are disjoint.*

Proof of Claim 5. Follows directly from claim 4

In the alternative packing one verticex in each P_3 are choosen from each of the sets C_i, H_i and N_i for each crown decomposition detected in the first packing.

Claim 6 *Observe that this implies that the edges in the P_3's contained in any Crown Decomposition are disjoint. Further, the number of P_3's contain in the crown decomposition (N_i, H_i, C_i) is at least $min[|N_i|, |C_i|]$.*

Proof. For each $n_i \in N_i$ choose an edge (n_j, h_j) (such an edge exist by def of N_i)

As $C_i \geq N_i$ and C_i and H_i are completely biconnected, there must also exist some edge $(h_j, c_j), c_j \in C_i$ such that (h_j, c_j) is not already in the packing. One P_3 is generated for each $n_j \in N_i$.

As C_i and H_i are completely biconnected, for $j = 1 \ldots |C_i|$, there must also exist some edge $(h_j, c_j), c_j \in C_i$ such that (h_j, c_j) is not already in the packing. Hence, at least $|C_i|$ P_3's are created.

Claim 7 *In the packing P^*, an edge is shared by at most two P_3*

> *Proof of Claim 8.* In the alternative packing, the P_3 are all edge disjoint for any individual crown decompositions used to generate it. Furthermore, in the alternative packing edges either lie between the head and the crown vertices of a crown decomposition or between two heads. But, as no two crowns share the same head, the edges connecting vertices in the heads are shared by at most two P_3.

A second lower bound, based on an alternative packing P' or P_3's, is then calculated.

Claim 8 $2|P'| \geq \sum_{C_i \in O} min[C_i, N_i]$. *Hence* $\sum_{i=1}^{r} C_i + \sum_{r+1}^{m} N_i \leq 2|P'| \leq 2k$

> *Proof of Claim 8.* This ensures that $\sum_{C_i \in <O>, st N_i > C_i} C_i + \sum_{C_i \in <O>, st C_i \geq N_i} N_i \leq 2|P| \leq 2k$

Although the minimum size of this packing is determined, it does not need to be constructed and is unlikely to reveal additional crowns $\sum C_i$ would be smaller.

Combined with the crown rule this then gives the new kernel size.

$$
\begin{aligned}
|V| &\leq |P| + |O| \\
&\leq |P| + \sum_{C_i \in O, stC_i \geq N_i} C_i + \sum_{C_i \in O, stC_i < N_i} Ci \\
&\leq |P| + \sum_{C_i \in O, stC_i > H_i + N_i - 1 \geq N_i} C_i + \sum_{C_i \in O, stH_i + N_i - 1 > C_i \geq N_i} C_i + \sum_{C_i \in O, stC_i < N_i} Ci \\
&\leq |P| + \sum_{C_i \in O, stC_i \geq N_i} (Hi + Ni - 2) + \sum_{C_i \in O, stC_i < N_i} C_i \; (By\ crown\ rule) \\
&\leq |P| + \sum_{C_i > N_i} (H_i - 2) + [\sum_{C_i > N_i} N_i + \sum_{C_i < N_i} C_i] \\
&\leq |P| + \sum_{C_i > N_i} (H_i - 2) + 2k \; (By\ claim\ 8) \\
&\leq 3k + 3k - 2 + 2k \; By\ claim\ 3) \\
&\leq 8k - 2
\end{aligned}
$$

Proof. 1 Assume in contradiction to the stated theorem that there exists a graph G of size $|V(G)| > 8k$ but has no k-Cluster Editing.

Let $k' < k$ be the largest k' for which G is a Yes-instance. By the Boundary Lemma 2 we know that $|V(G)| \leq 8k' - 2 < 8k - 2$. This contradicts the assumption.

Thus the total size $|V(G)| = |V(P)| + |O| \leq 8k - 2$.

5 Test Results

By way of preliminary discussion of the results, the number of vertices available in our search for cliques is at most $|V| - |V(P)|$. But as k can be as large as $n^2/4$ this may leave very little in the graph for cliques. As the compression form of the algorithm uses a 4-approximation to choose the crown heads, the amount of the graph remaining for crowns (cliques) is $|V| - |V(A)|$ [23]. Theoretical $V(A)$ (the number of vertices in the approximation) can be $2 \times 4k = 8k$ in size, which could be even worse than using the packing. But, if the approximation found lies on the boundary of two large cliques, this could be as small as the square root of k. i.e. some fraction of n. As a result we might obtain much better results.

This poses the question as to which algorithm gives the best result in practice. To help answer this question we have tested the performance of these algorithms on a variety of graphs. It was our initial intention to apply both algorithms to a range of real and random data. Current limitations on obtaining approximations below make using the 4–approximation was not possible for the microarray datasets being used. However, we also processed these datasets using a heuristic algorithm in order to measure what results could be obtained in this way.

6 Data

Two sets of data were processed. The first was pseud-random data previously evaluated in [9]. Real data representing micro-array (SH2-3) and protein-protein (globin) networks was also processed. This real data was converted from a weighted complete graph derived using a number of thresholds. The use of multiple thresholds resulted in graphs of a variety of densities.

7 Implementation decisions and justifications

The performance of these algorithms was evaluated using a 4 Xeon 3.4 GHz system with 12GB main memory. One processor was left idle to support other tasks un-associated with the experiments. All kernelization programs were written in C++ and compiled using GNU C++ compiler gcc 3.4.6 under a 64bit environment. The LEDA v 5.1 graph library was used to implement the graph structures. It is recognised that this uses adjacency lists, but due to the simple nature of the algorithms used, no extra $O(n)$ was incurred as a result.

When evaluating the performance of the compression kernelization algorithm [15] an approximation was needed. This was obtained using the lp-solve linear programming library. Slightly better performance may be observed if a different libray such as CPLEX was used. However, as the algorithm used equates to an $O(n^8)$ complexity [22] this speedup was not sufficient to allow it to be used on datasets large then a 100 nodes. Note that an $O(n^5)$ variation of this algorithm also exists, but no implementation of this was available to us. Tests were performed using a heuristic algorithm of complexity approx $O(n^5)$ so that larger datasets could be included.

8 Results

The Table 2 shows results obtained from the compression algorithm using a greedy approximation on protein-protein globin graph for various thresholds. This shows that greedy localization is effective when the number of edges is relatively small (low density graph), evidenced by the large reduction achieved. As the density of the graph increased, the technique became effective. Table 3 shows a comparison of processing the same data using greedy localization and compression algorithm [15] respectively. Two densities of original graph were used. An optimal solution is obtainable for both of these simple graphs. The best reduction in graph size is obtained on $50v_10c_18ed$, when a the compression algorithm was used with a greedy approximation in addition to packing based algorithm. However, for this problem using the 4-approximation in the compression algorithm also produced the same result.

No reduction is observed in the more dense graph ($50v_225edges_5c$). However, the improved lower bound obtained from the greedy localization indicates that the approximation required by the compression algorithm was, in fact, optimal. Thus neither the approximation nor greedy techniques reduced the graph.

Set Name	**Number Vertices**	**Number Edges**	**Kernel Size**	**Removed Edges**, δk	
Globin3	972	3898	770	647	92
Globin4	972	9160	916	64	20
Globin5	972	18986	965	7	4
Globin6	972	27671	967	4	3
Globin7	972	38557	969	4	3
Globin8	972	47797	969	4	3
Globin9	972	62525	969	4	3
Globin10	972	75386	971	1	1
Globin11	972	91317	972	0	0

Table 2. Effect of crown rule for varying graph density

In [9] unexpected results where obtained on the execution times for solving cluster editing. It was found that the processing cost of solving the FPT problem when k increased significantly as k increased above optimal. Thus, any refinement in the valid range of optimal solution time could significantly decrease the processing time in finding an optimal solution. While the optimal result obtained in the graph $50v_225edge_5c$ is the result of the approximation being optimum, without the lower bound obtained from the packing this would not be known. Table

Data name	**Vertices**	**Kernel size**	**Original Edges**	**Reduced Edges**	**Lower Bound** k	δk	**Approx Size**	**Method Used**
$50v_10c_18ed$	50	10	100	19	3	10	n/a	Greedy
$50v_10c_18ed$	50	0	100	0	1	18	34	Compression
$50v_225edges_5c$	50	50	225	225	44	0	139	Greedy
$50v_225edges_5c$	50	50	225	225	44	0	44	Compression

Table 3. Optimal solutions found by combining both algorithms

4 shows the processing of globin graphs using heuristic approximation combined with Crown rule reduction. The table shows calculation of an integrity factor, expressed as Integrity = Approximation Size / Lower Bound. It is not possible to calculate this integrity factor when using many heuristic techniques, and the data set is too large to be processed in reasonable time by the best-known approximation techniques [23]. However, the integrity figures obtained by the new greedy technique exceed those reported in [23].

Set Name Set Name	Num Vertices	Kernel Size	Num Edges	Reduced Edges	Lower Bound k	δk	Approx Size	**Integrity**
sh2-3	839	767	5860	5820	2894	40	5133	**1.77367**
sh2-3-2***	839	767	5860	5820	2894	0	5159	**1.77367**
sh2-4	839	827	13799	13797	6885	2	11321	**1.644299**
sh2-5	839	832	26612	26612	13294	0	20409	**1.535204**
globin10	972	971	75386	75385	37273	1	56821	**1.524455**

Table 4. Lower bound gives integrity for heuristic approximation

9 Conclusions and Further Research

This paper presented a new kernelization algorithm, combining greedy localization and the Crown rule, that produces an improved 8k kernel towards solution of the cluster editing problem. The techniques are proven to be mathematically correct, and examples show that application of the technique:

- produces results for dense graphs when the best approximation techniques cannot produce a result in reasonable time,
- produces results with better bounds than is otherwise achievable using heuristic or approximation algorithms, and
- Is $O(nE)$, (where E is the number of edges) meaning that its time-complexity is linear with respect to the density of the graph. This means that the new reduction technique is fast enough for interleaving with the search algorithm.

Work is in progress on production of an interleaved implementation of the technique. In previous work investigating the use of interleaving with linear kernelization (on easier problems), the manageable size for data sets was found to increase from about 200 vertices to about 50,000. It is expected that this increase will be exceeded through the application of greedy localization and the Crown reduction rule. The outcome of experiments using the new interleaved implementation will be reported in a future publication.

References

1. Eades, P., Feng, Q.W.: Multilevel visualization of clustered graphs. Lecture Notes in Computer Science (1996) 101–112
2. Michalewicz, A., e.a.: Case study: An intelligent decision-support system. IEEE Intelligent Systems **20** (2005) 43–49
3. Garey, M., Johnson, D.: Computers and Intractiblity. A Guide to the Theory of NP-Completeness. New Yourk: W. H. Freeman (1979)
4. Vazirani, V.: Approximation Algorithms. Springer-Verlag (2001)
5. Khanna, S.: On syntactic versus computational views of approximability. Sian Journal of Computing **28** (1999) 164–191
6. Anderson, J.: An introduction to neural networks. Cambridge, Mass.: MIT Press (1995)
7. Michalewicz: Heuristic methods for evolutionary computation techniques. Journal of Heuristics (1995)
8. Mitchell, M.: An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press (1996)
9. Dehne, F., Langston, M.A., Luo, X., Pitre, S., Shaw, P., Zhang, Y.: The cluster editing problem: Implementations and experiments. In: Proc. Int. Workshop on Parameterized and Exact Computation (IWPEC), Springer LNCS (2006)
10. Downey, R., Fellows, M.: Parameterized complexity. Springer Verlag. New York: Springer (1999)
11. Harary, F.: Graph Theory. Reading, MA: Addison-Wesley (1994)
12. Shamir, R., Sharan, R., D., T.: Cluster graph modification problems. Discrete and Applied Mathematics **144** (2004) 173–182 preliminary version in: 28th WG 2002, LNCS 2573 (2002), 379-390.
13. L. Cai, J. Chen, R.D., Fellows, M.R.: On the parameterized complexity of short computation and factorization. Arch. for Math. Logic **36** (1997)

14. Chor, B., Fellows, M., Juedes, D.: Linear kernels in linear tme or how to save k colors in $o(n^2)$ steps. In: Proceedings WG 2004-30th Workshop on Graph Theoretic Concepts in Computer Science,. Lecture Notes in Competer Science, Springer-Verlag (2004)
15. Fellows, M., Langston, M., Rosamond, F., Shaw, P.: Efficient preprocess algorithm for cluster editing. In: LNCS, Springer Virlag (2007) incomplete fix.
16. Prieto-Rodriguez, E.: Systematic Kernelization in FPT Algorithm Design. Ph. D. Thesis. PhD thesis, School of EE&CS, University of Newcastle, Australia (2005)
17. Guo J., e.a.: Improved fixed-parameter algorithms for two feedback set problems. In: WADS. (2005)
18. Estivill-Castro, V.e.a. In: FPT is P-Time extremal structure 1. King's College Publications (2005) 1–41
19. Fellows, M.: Blow-ups, win/wins and crown rules: Some new directions in fpt. In: Lecture Notes in Computer Science, Springer-Verlag (2003) 1–12
20. Dehne, F., Fellows, M., Rosamond, F., Shaw., P.: Greedy localization, iterative compression and modeled crown reductions: new fpt techniques, an improved algorithm for set splitting and a novel $2k$ kernelization for vertex cover. In: Proceedings of the First International Workshop on Parameterized and Exact Computation. Lecture Notes in Computer Science, Springer-Verlag (2004) 271–280
21. Fellows, M., E., P., Sloper, C.: Looking at the stars. In: IWPEC04, Springer Verlag (2004)
22. Wirth, A.: Approximation Algorithms for Clustering. PhD thesis, Princeton University (2005)
23. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. Journal of Computer and System Sciences **71** (2005) 360–383