Advances in Cluster Editing: Linear FPT Kernels and Comparative Implementations

Peter E. Shaw
B. Comp. Sci. (Hons I)

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

School of Electrical Engineering
and Computer Science

The University of Newcastle
Callaghan N.S.W. 2308
Australia

February, 2010

STATEMENT OF ORIGINALITY

The thesis contains no material which has been accepted for the award of any other degree of diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying subject to the provisions of the Copyright Act 1968.

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

I hereby certify that parts of the work embodied in this thesis have been done in collaboration with other researchers. I have included as part of the thesis a statement clearly outlining the extent of collaboration, with whom and under what auspices.

I hereby certify that the work embodied in this thesis contains a number of published scholarly work (conference proceedings) of which I am a joint author. I have included a written statement, attesting to my contribution towards the joint publications/scholarly work.

Peter E. Shaw

Acknowledgements

Firstly, I would like to thank my darling wife and children for persevering with me in my pursuit for knowledge and understanding, for standing by me as I travelled round the world trying to do what was needed to complete this work.

I would also like to thank Professor Michael Fellows and Dr Fran Rosamond for introducing me to this interesting topic of parameterized complexity and for continuing to be available even after they left Newcastle. Next I would like to take time to thank Dr Mike Hannaford and Associate Professor Frans Henskens for taking on the role of supervisor at Newcastle University after Mike Fellows had to leave. Unfortunately, at the time the university had no available supervisors with intimate knowledge of my field of research. This left Frans Henskens and Mike Hannaford with the extra burden of understanding a field outside their core research areas and their efforts in this are appreciated. Frans Henskens, greatly assisted in facilitation my research including arranging for Professor Mike Langston to be appointed as a technical advisor. I would also like to thank Mike Hannaford and Frans Henskens for their perseverance in editing my work over the last three years. I have had a lot of challenges in my writing (as my sight is not good) and this has often been an arduous task. I also offer my thanks and appreciation to Mike Hannaford and Frans Henskens for their support and encouragement over this time. I would also like to give a special thank you to Michael Langston who, while he was under no obligation to do so, made himself available as a supervisor (technical advisor) after Mike Fellows left and provided invaluable assistance from that point onwards. I greatly appreciated his invaluable advice and his help, which, among other things, included: allowing opportunities for joint research, reviewing my dissertation, monitoring my progress, strict encouragement, and even traveling to Australia when special assistance was required. It was a real privilege to have his assistance and a demonstration of Mike Langston professionalism.

Finally, I would like to thank the many others who have been my friends during the trying time that I have had over the last five years, among them: God for his inspiration (Isaih28) and for giving me this chance in life; my wife and family; my mother – Pru; my father – Russell Shaw. My friends: Wagner Costa, Bayu Hendradjaya, Peter Harwood and Helen Lee for their many important discussions including how to give God glory for the work we do. The 'Uni Acts' Prayer Group for their friendship and prayers. My home group: Adrian and Paula Howson, Jim and Helen Walkom and all the children. The many people who encouraged me. Especially Helen Lee, Joel Kidd, and Helen Walkom, for their assistance in editing my thesis. Trevor Wilks and the others at the Adaptive Technology Centre for their assistance in enabling me do to my study. My older children's mother for assisting me with their many difficulties that they have faced during this time. My LORD and Saviour Jesus for helping me with both my inspiration and also my weaknesses that I have had to overcome. My colleagues at University.

Collaborators: Mike Langston; Henning Fernau; Elena Prieto; Matthew Sunderman; Frank Dehne; Yun Zhang, Xuemei Luo, Chris Symond, and Jon Scharff from the University at Tennessee; Kim Colyvas (Newcastle University statistics support group).

Editor:
I would also like to thank my editor, Dr Maria Freij for her proffessional editing of the final draft of my thesis.

To my darling wife Eunmee Chang-Shaw whom I love so much.

Rejoice in the Lord always. I will say it again: Rejoice! Let your gentleness be evident to all. The Lord is near. Do not be anxious about anything, but in everything, by prayer and petition, with thanksgiving, present your requests to God. And the peace of God, which transcends all understanding, will guard your hearts and your minds in Christ Jesus.

Philippians 4:4–7 (New International Version)

Contents

1 Introduction **33**

1.1 Overview 33

1.2 Cluster Analysis 34

1.2.1 Measures of Similarity 37

1.2.2 Clustering Algorithms 42

1.2.3 Measures of Clustering 44

1.3 The CLUSTER EDITING Problem 47

1.4 Biological Motivation 50

1.4.1 Demonstration of CLUSTER EDITING using Biological Data . . 54

1.5 Need for Exact Solutions in Biological Analysis 54

1.6 Intractability 55

1.7 Practical Approaches to NP-Hard problems 60

1.7.1 Approximation Algorithms 60

1.7.2 Average-Case Analysis 63

1.7.3 Heuristics 64

1.7.4 Parameterised Complexity 64

1.7.5 Stochastic Algorithms . . . 70

1.8 Summary . . . 71

1.8.1 Thesis Objective . . . 71

1.8.2 Thesis Overview . . . 73

2 Preliminaries 79

2.1 Terminology and Notations . . . 79

2.1.1 Approximations . . . 79

2.1.2 Combinatorics . . . 80

2.1.3 Graphs . . . 82

2.2 Favourite Problems . . . 84

2.3 The Parameterised Complexity Toolkit . . . 85

2.4 Reduction Rules and Kernelization . . . 91

2.4.1 Kernelization Categories . . . 93

2.4.2 Polynomial Kernels . . . 95

2.4.3 Proper Kernels . . . 95

2.4.4 Annotated Kernels . . . 97

2.4.5 Induced Kernels . . . 98

2.4.6 Interleaving Reduction Rules . . . 99

2.4.7 Parameter Independent Reduction Rules . . . 100

2.5 Global Reduction Rules . . . 101
2.5.1 Nemhauser Trotter . . . 102
2.5.2 Crown Rule . . . 103
2.5.3 Properties of Crowns . . . 106
2.5.4 Generalisations of the Crown Rule . . . 107
2.5.5 Modified Crown Rules . . . 107
2.5.6 Edge Disjoint Triangle Packing . . . 108
2.5.7 Other Global Reduction Rules . . . 109
2.5.8 Experimental Comparison of Reduction Rules . . . 109
2.5.9 Cutting Rules . . . 112
2.6 Kernelization Techniques . . . 112
2.7 Catalyzation . . . 113
2.8 Inductive Techniques . . . 114
2.8.1 Extremal Method . . . 117
2.8.2 Iterative Compression . . . 120
2.8.3 Reducing the Overhead of Iterative Compression . . . 125
2.8.4 Compression from a Constant Factor Solution . . . 129
2.9 Bounded Branching . . . 131
2.9.1 Bounded Search Trees . . . 131
2.9.2 Greedy Localization . . . 132

2.9.3 Nondeterministic Branching 133

2.10 Pseudo-Kernelization 135

2.11 Auxiliary Graph Structure 135

2.12 Win–Win 136

2.12.1 Imposing FPT Structure – Bounded Treewidth 136

2.12.2 Win–Win 136

2.12.3 Parametric Duals 138

2.12.4 Small kernels 139

3 Cluster Editing 141

3.1 Overview 141

3.2 The Cluster Editing Family of Problems 142

3.2.1 Complexity 146

3.3 Problem Characteristics 146

3.3.1 Non-monotonicity of the Cluster Edit Problem 146

3.3.2 Scarcity of Solutions 147

3.3.3 Previous Work 148

3.3.4 Our Results 149

3.3.5 A Crown Reduction Rule 150

3.3.6 Cluster Crown Rule 151

3.4 Polynomial-Time Linear Kernelization for Cluster Edit 153
3.5 A 24k Linear Kernel Using a Compression Routine 155
3.5.1 A 24*k* Linear Kernel Using a Compression Routine 156
3.5.2 Correctness of Cluster-Crown Rule 156
3.5.3 Optimality of Cluster-Crown Rule 160
3.6 A 6k many-to-one kernel 161
3.6.1 Efficient Pre-processing for Cluster Edit 162
3.6.2 6k Kernelization Algorithm 163
3.7 An 8k Many-to-One Kernel Using Greedy Localization 165
3.7.1 Description of the Algorithm and Running Time Analysis . 165
3.8 Remarks 176

4 Evaluation of the Kernelization Techniques 179
4.1 Outline 179
4.2 Overview 180
4.3 Analysis Approach 185
4.3.1 Statistical Methods 185
4.3.2 Graphics Used 194
4.3.3 Observations 198
4.3.4 Response Variables 199

4.3.5 Exploratory Variables 209

4.3.6 Experimental Validation 212

4.3.7 Summary 212

4.4 Choice of Test Data 213

4.4.1 Choice of Non-Synthetic Data and Random Data 213

4.4.2 Non-Synthetic Data 216

4.5 Synthetic Data 217

4.5.1 Random Data 218

4.5.2 Pseudo-Random Data 220

4.6 Implementation Decisions and Justifications 223

4.6.1 Session Info 228

4.7 Reduction Rules 229

4.7.1 Removing Outliers 229

4.7.2 Connected Components 230

4.7.3 Pendent Rule and Other Local Reduction Rules 231

4.7.4 2 and 3-Edge Connected Components 237

4.7.5 Greater than k-Common Neighbours 247

4.8 Modelled Crown Rule 265

4.8.1 Compression Routine Kernelization Algorithm 266

4.8.2 TWIN Auxiliary-Graph Search Algorithm 271

4.8.3 Greedy Kernelization Algorithm 293

4.9 Approximations: A Practical Solution 317

4.9.1 Linear Programming Approximations 318

4.9.2 Execution-Times of LP approximation 319

4.9.3 Heuristic Approximations 321

4.9.4 Analysis of the Execution-Time of the Approximation Algorithms 322

4.9.5 Heuristic Algorithm Performance 323

4.9.6 Performance-Ratio 325

4.9.7 Performance Issue with the LP-Approximation Algorithm . 327

4.9.8 Overcoming the Performance Issues 328

4.10 Summary 330

4.10.1 Kernel Size 332

4.10.2 Lower Bound 337

4.10.3 Execution Times 339

4.10.4 Approximations 340

4.10.5 Limitation of This Research 344

4.10.6 Recommendations 345

4.10.7 Acknowledgements 347

5 Conclusions **349**

5.1 Summary of Findings 349

5.2 Limitations 354

5.3 Open Problems and Future Work 356

5.3.1 Scale-up of Data Size Both for Optimisation and Approximation 356

5.3.2 Post-processing of data 356

6 Appendix A **359**

Abstract

Experience has shown that clustering objects into groups is a useful way to analyze and order information. It turns out that many clustering problems are intractable. Several heuristic and approximation algorithms exist, however in many applications what is desired is an optimum solution. Finding an optimum result for the CLUSTER EDITING problem has proven non-trivial, as CLUSTER EDITING is $\mathcal{NP}$-Hard [KM86], and $\mathcal{APX}$-Hard, and therefore cannot be approximated within a factor of $(1 + \epsilon)$ unless $\mathcal{P}oly = \mathcal{NP}$ [SST04].

The algorithmic technique of Parameterized Complexity has proven an effective tool to address hard problems. Recent publications have shown that the CLUSTER EDITING problem is Fixed Parameter Tractable ($\mathcal{FPT}$). That is, there is a fixed parameter algorithm that can be used to solve the CLUSTER EDITING problem.

Traditionally, algorithms, in computer science, are evaluated in terms of the time needed to determine the output as a function of input size only. However, typically in science most real datum contains inherent structure. For Fixed Parameter Tractable (FPT) algorithms, permitting one or more parameters to be given in the input, to further define the question, allows the algorithm to take advantage of any inherit structure in the data [ECFLR05].

A key concept of FPT is kernelization; that is, reducing a problem instance to a core hard sub-problem. The previous best kernelization technique for CLUSTER EDITING was able to reduce the input to within k^2 [GGHN05] vertices, when parameterized by k, the edit distance. The edit distance is the number of edit operations required to transform the input graph into a cluster graph (a disjoint union of cliques). Experimental comparisons in [DLL$^+$06] showed that significant improvements were obtained using this reduction rule for the CLUSTER EDITING problem. The study reported in this thesis presents three polynomial-time, many-to-one kernelization algorithms for the CLUSTER EDITING problem, the best of these algorithms produces a linear kernel of at most $6k$ vertices.

In this thesis, we discuss how using new $\mathcal{FPT}$ techniques including *extremal method compression routine* and *modelled crown* reductions [DFRS04] can be used to kernelize the input for the CLUSTER EDITING problem. Using these new kernelization techniques, it has been possible to improve the number of vertices in the data sets that can be solved optimally, from the previous maximum of around 150 vertices to over 900. More importantly, the edit distance of the graphs that could be solved as also increased from around $k = 40$ to more than $k = 400$.

This study also provides a comparison of three inductive algorithmic techniques:

- *compression routine using a constant factor approximation* – Compression Crown Rule Search Algorithm;
- *extremal method* (coordinatized kernel) [PR05], using a *constructive* form of the boundary lemma – Greedy Crown Rule Search Algorithm;
- *extremal method,* using an auxiliary (TWIN) graph structure – Crown Rule TWIN Search Algorithm.

Algorithms derived using each of the above techniques to obtain linear kernels for the CLUSTER EDITING problem have been evaluated using a variety of data with different exploratory properties. Comparisons have been made in terms of reduction in kernel size, lower bounds obtained and execution time.

Novel solutions have been required to obtain approximations within a reasonable time, for the CLUSTER EDITING problem that is within a factor of four of the edit distance (minimum solution size). Most approximation methods performed very badly for some graphs and well for others. Without any guide regarding the quality of the result, a very bad result can be assumed to be close to optimum.

Our study has found that just using the highest available lower bound for the approximation is insufficient to improve the result. However, by combining both the highest lower bound obtained and the reduction obtained using kernelization, a 30-fold improvement in the approximation performance ratio is achieved.

List of Figures

1.1 Classification based iris petal and sepal area [Fis36] [Hei08]. 36

1.2 A dendrogram of Edgar Anderson's Iris Data 43

1.3 A cluster graph. 49

1.4 Place in the pipeline 53

1.5 The optimum and optimal solutions differ considerably. 56

1.6 Cutting up intractability [DFS95] 67

1.7 Layout of the thesis. 74

2.1 A clique 83

2.2 Fixed parameter algorithmic techniques [ST08] 90

2.3 Pendent rule for vertex cover 92

2.4 Crown decomposition 104

2.5 The union of overlapping crowns is a crown 106

2.6 Kernelization techniques 113

2.7 Inductive techniques - witness structures 114

2.8 Inductive techniques [ST08] 115

2.9 Size $k + 1$ solution reveals crowns 123

3.1 The Cluster Edit Problem is non-monotonic [DLL$^+$06]. 148

3.2 A cluster graph - with modifications. 150

3.3 Quadratic sized solutions exist 153

3.4 Cluster Edit - Crown Rule. 154

3.5 Pendent rules. 154

3.6 The cost $\Gamma(\pi)$ of partitioning π. 157

3.7 *Cluster-crown* reduction rule is optimal. 160

3.8 Edge disjoint P_3s resolved by a single edge 166

3.9 Greedy Packing of P_3s 167

3.10 Alternative Packing of P_3s. 172

4.1 The effect of interleaving on the bounded-search 182

4.2 Scatter Plot Graph 190

4.3 Example Box Plot 194

4.4 Example rPart Diagram 195

4.5 Example Normal Q-Q and Scale-Location Graphs 197

4.6 Example VCD Graphic 198

4.7 Observed Kernel Size on Cluster-Graphs with Noise 200

4.8 Tree-based classification of the reduction rules 201

4.9 Cluster graphs with exist with quadratic edit distance. 204

4.10 Observed kernel size for sample graphs and rules 206

4.11 Average degree of sample of graphs 215

4.12 Micro-array data SH2 at Threshold 3 216

4.13 Scale-Free Graph Degree Distribution 221

4.14 Cluster Graph with Random Noise 223

4.15 Schema of data and experimental results used. 226

4.16 Document generation process 227

4.17 Three local reduction rules for Cluster Edit. 231

4.18 Summary of effect of pendent rules on kernel size 233

4.19 Effect of pendent rules on non-synthetic Data 234

4.20 Effect of pendent rules on kernel size 235

4.21 3-Edge-Connect Component Reduction 238

4.22 3-Edge-Connectivity counter-example 238

4.23 4-edge-connectivity counter example 239

4.24 Two 3-edge-connected components 240

4.25 Paths connect v_1 to u_1 and u_2. 241

4.26 Two 3-edge-connected components – an alternative partitioning exits242

4.27 2-connected Pseudo-Reduction Rule 243

4.28 Effectiveness of 2-vertex and 3-edge-connected reduction rules . . . 245

4.29 k–common neighbours reduction rules 248

4.30 Effect of parameter reduction on k 249

4.31 Effect of the number of clusters on kernel size 251

4.32 3D Scatter Plot edits, k and Applied Edits 253

4.33 k common neighbour kernel as a function of edit distance (lower bound) 254

4.34 k-Common-Neighbour Execution-Times 256

4.35 K (non)-common-neighbour execution linear model 257

4.36 k common neighbour execution times 259

4.37 Lattice plot of k-common-neighbour execution-times 260

4.38 Lattice plot of k-common-neighbour execution-times 261

4.39 k (non)-common-neighbour rule execution-time linear model . . . 262

4.40 Simple test graphs. 267

4.41 Execution time for Cluster Editing bounded-search 271

4.42 Size Reduction Using *TWIN auxiliary-graph search* algorithm 272

4.43 Edits Applied Using TWIN Search Algorithm 274

4.44 3D Scatter Plot Clusters versus Applied Edits 277

4.45 Size Reduction Using TWIN Search 279

4.46 Kernel size from *TWIN auxiliary-graph search* algorithm 280

4.47 Kernel size with respect to Lower Bound for TWIN Crown Rule . . 282

4.48 Kernel size as a ratio of approximation size for *TWIN auxiliary-graph search* 283

4.49 Lower Bound on cluster-graphs-with-noise 284

4.50 TWIN auxiliary-graph search execution time using random data with random density. 286

4.51 Execution-Time of *crown rule TWIN search*: Linear Model 287

4.52 *Crown rule TWIN search* execution times 289

4.53 *greedy crown rule search* Residuals vs. Fitted 289

4.54 Crown rule TWIN search execution linear model 291

4.55 Execution time for bounded-search for various size of (G, k) [DLL+06]. 295

4.56 Boxplots showing *greedy-crown search* reduction in k and V 296

4.57 Kernel size for *greedy crown rule search* algorithms 297

4.58 Kernel size by *greedy crown rule search* algorithm 298

4.59 Greedy-Crown Search reduction in V and k 299

4.60 Lower Bounds on Various Data Types 301

4.61 Lower Bounds on Random Data 302

4.62 3D Scatter Plot Clusters versus Applied Edits 304

4.63 Lower Bound on Pseudo-Random Data 305

4.64 Performance-ratio achieved by combining kernelization 306

4.65 Greedy Packing Search execution time 308

4.66 Execution-Time of *greedy crown rule search* : Linear Model 309

4.67 *greedy crown rule search* execution times 310

4.68 *greedy crown rule search* Residuals vs. Fitted 311

4.69 *greedy crown rule search* execution linear model 313

4.70 Crown Rule reduction in k 314

4.71 Comparison of the Crown Rule Algorithms on cluster-graphs-with-noise 315

4.72 Execution times of the *Greedy* and *TWIN auxiliary-graph search* . . . 318

4.73 Scatterplot matrix for the LP approximation execution times 320

4.74 Scatterplot matrix for the greedy approximation execution times . . 323

4.75 The approximations sizes using greedy and LP methods 325

4.76 The approximations' execution times using greedy and LP methods 326

4.77 Approximation quality 327

4.78 Interleaving 329

4.79 3D Scatter Plot Max k, noise and Clusters 335

4.80 Lower Bounds on Random Data 338

4.81 Identified Outliers in Approximations 341

4.82 Approximation quality achieved by combining kernelization 343

5.1 Addressing the problem of clustering 350

List of Tables

1.1 Correlation problems 62

1.2 Summary of contributions 76

2.1 The effects of parameter on performance [Fel03]. 89

2.2 Graph: SH2-3.dim, n = 839, k = 246. Times are given in seconds [AKLSS04, 9]. 111

2.3 Results of vertex cover on SH2-3 111

2.4 Results of Vertex Cover on SH2-10 112

3.1 Edit-to-clique problems. 143

3.2 TWIN graph algorithm for applying *cluster-crown rule*. 164

3.3 A greedy algorithm to apply the *cluster-crown rule*. 169

4.1 Cluster Edit Kernelization Algorithms Execution-Time 180

4.2 Key for rules used in Figure 4.8. 202

4.3 Synthetic data used. 218

4.4 Implementation Code Lengths. 224

4.5 The coefficients of the relationship between average vertex degree and edits applied. 235

4.6 ANOVA comparison of the linear models for the execution-times of the k-neighbour search models: $\log t = \log n$ and $\log t = \log; n + \log m$ 258

4.7 Summary of the coefficients for the execution-time for the k-(non)-common neighbour search model $logt = logn$ 258

4.8 ANOVA comparison of the linear models for the execution-times of the k-neighbour search models: $\log t = a \log n$ and $\log t = a \log n + b \log m$ 263

4.9 Summary of the coefficients for the execution-time for the k-neighbour search model $logt = logn + logm$ 263

4.10 Summary of the coefficients for the execution-time for the k-neighbour search model $logt = logn$ 263

4.11 Crown rule solutions found by combining kernelization and approximations 267

4.12 Lower bound gives quality for approximations 267

4.13 Effect of *crown rule* for varying graph density. 268

4.14 Results on non-synthetic graphs. 269

4.15 A summary of Linear model showed a significant relationship between the edit-distance and the amount of reduction. 276

4.16 Alternative lower bound form *TWIN auxiliary-graph search* 285

4.17 ANOVA comparison of the linear models for the execution-times of the *crown rule TWIN search*: $logt = logn$ and $logt = logn + logm$. 286

4.18 Summary of the coefficients for the execution-time for the TWIN search using the model $logt = logn + logm$. 288

4.19 ANOVA linear models for TWIN search – high clustering coeff data. 288

4.20 ANOVA $\log t = a \log n + b \log m$ mode for TWIN search – high clustering coeff data. 290

4.21 Summary of the coefficients for the execution-time for the crown rule TWIN search model $logt = logn$ 290

4.22 ANOVA comparison of the linear models for the execution-times of the *greedy crown rule search*: $logt = logn$ and $logt = logn + logm$. 308

4.23 Summary of the coefficients (model 2) for the execution-time for the *greedy crown rule search* model $logt = logn + logm$ 308

4.24 ANOVA comparison of the linear models for the execution-times of the *greedy crown rule search* models: $logt = logn$ and $logt = logn + logm$ 312

4.25 Summary of the coefficients for the execution-time for the greedy crown rule search model $logt = logn + logm$ on cluster-graphs-with-noise. 312

4.26 Summary of the coefficients for the execution-time for the greedy crown rule search model $logt = logn$ on cluster-graphs-with-noise. 312

4.27 ANOVA comparison of the linear models for the execution-times of the LP approximation model: $logt = logn$ and $logt = logn + logm$ 321

4.28 Summary of the coefficients for the execution-time for the LP approximation model $logt = logn$ 321

4.29 ANOVA comparison of the linear models for the execution-times of the heuristic approximation model: $logt = logn$ and $logt = logn + logm$ 324

4.30 Summary of the coefficients for the execution-time for the heuristic approximation model $logt = logn$ 324

4.31 Cluster Edit Kernelization Algorithms kernel sizes 333

4.32 Execution times functions for reduction search algorithms 340

4.33 Mean Performance-Ratios 340

6.1 Summary of Contributions 359