KaMIS v1.0 – Karlsruhe Maximum Independent Sets
User Guide

Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash and Renato F. Werneck

Karlsruhe Institute of Technology (KIT), *Karlsruhe, Germany*

lamm@ira.uka.de, {sanders, christian.schulz, strash}@kit.edu

San Francisco, USA

rwerneck@acm.org

Abstract

This paper serves as a user guide to the framework KaMIS (Karlsruhe Maximum Independent Sets). The framwork computes high quality independent sets in huge sparse graphs. We give a rough overview of the techniques used within the framework and describe the user interface as well as the file formats used.

Contents

1 Introduction

The maximum independent set problem is an NP-hard problem that has attracted much attention in the combinatorial optimization community, due to its difficulty and its importance in many fields. Given a graph $G = (V, E)$, the goal of the maximum independent set problem is to compute a maximum cardinality set of vertices $\mathcal{I} \subseteq V$, such that no vertices in $\mathcal{I}$ are adjacent to one another. Such a set is called a *maximum independent set* (MIS). The maximum independent set problem has applications spanning many disciplines, including classification theory, information retrieval, and computer vision [4]. Independent sets are also used in efficient strategies for labeling maps [7], computing shortest paths on road networks [16], (via the complementary minimum vertex cover problem) computing mesh edge traversal ordering for rendering [23], and (via the complementary maximum clique problem) have applications in biology [5], sociology [11], and e-commerce [28].

It is easy to see that the complement of an independent set $\mathcal{I}$ is a vertex cover $V \backslash \mathcal{I}$ and an independent set in G is a clique in the complement graph $\overline{G}$. Since all of these problems are NP-hard [6], heuristic algorithms are used in practice to efficiently compute solutions of high quality on *large* graphs [2, 9]. However, small graphs with hundreds to thousands of vertices may often be solved in practice with traditional branch-and-bound methods [21, 22, 26], and medium-sized instances can be solved exactly in practice using reduction rules to kernelize the graph. Recently, Akiba and Iwata [1] used advanced reduction rules with a measure and conquer approach to solve the minimum vertex cover problem for medium-scale sparse graphs exactly in practice. Thus, their algorithm also finds the maximum independent set for these instances. However, none of these exact algorithms can handle huge sparse graphs. Furthermore, our experiments suggest that the quality of existing heuristic-based solutions tends to degrade at scale for inputs such as Web graphs and road networks. Therefore, we need new techniques to find high-quality independent sets in these graphs.

With KaMIS we provide a framework for computing high quality independent sets in huge sparse graphs. So far our framework contains the ReduMIS algorithm which is an advanced evolutionary algorithm based on graph partitioning and incorporates kernelization techniques to compute large independent sets. Our algorithm uses reduction techniques and recursively chooses vertices that are likely to be in a large independent set (using an evolutionary approach), to then further kernelize the graph. This opens up the reduction space – which not only speeds up the computation of large independent sets drastically, but also enables us to compute high-quality independent sets on massive instances.

2 Algorithmic components within KaMIS

We now give a rough overview over the algorithmic components implemented in our framework. For details on the individual components, we refer the interested reader to the corresponding papers.

ARW. There is a wide range of heuristics and local search algorithms for the maximum clique problem (see for example [3, 10, 8, 15, 20, 9]). They typically maintain a single solution and try to improve it by performing node deletions, insertions, and swaps, as well as *plateau* search. For the independent set problem, Andrade et al. [2] extended the notion of swaps to (j, k)-swaps, which remove j nodes from the current solution and insert k nodes. One *iteration* of the ARW algorithm consists of a perturbation and a local search step. The ARW *local search* algorithm uses simple 2-improvements or $(1, 2)$-swaps to gradually improve a single current solution. The simple version of the local search iterates over all nodes of the graph and looks for a $(1, 2)$-swap. By using a data structure that allows insertion and removal operations on nodes in time proportional to their degree, this procedure can find a valid $(1, 2)$-swap in $\mathcal{O}(m)$ time, if it exists. An even faster incremental version of the algorithm (which we use here) maintains a list of *candidates*, which are nodes that may be involved in $(1, 2)$-swaps. It ensures a node is not examined twice unless there is some change in its neighborhood.

EvoMIS. EvoMIS is a very natural evolutionary algorithm proposed by Lamm et al. [17]. It uses combination operations that are based on graph partitioning and ARW local search. They employ the partitioning framework KaHIP [25] to derive operations that make it possible to quickly exchange whole blocks of given independent sets.

The basic idea of the combine operations is to use a partition of the graph to exchange whole blocks of solution nodes and use local search afterwards to turn the solution into a maximal one. We explain one of the combine operations based on 2-way node separators in more detail. In its simplest form, the operator starts by computing a 2-way node separator $V = V_1 \cup V_2 \cup S$ of the input graph. The separator S is then used as a crossover point for the operation. The operator generates two children, $O_1 = (V_1 \cap \mathcal{I}_1) \cup (V_2 \cap \mathcal{I}_2)$ and $O_2 = (V_1 \cap \mathcal{I}_2) \cup (V_2 \cap \mathcal{I}_1)$. In other words, whole parts of independent sets are exchanged from the blocks V_1 and V_2 of the node separator. Note that the exchange can be implemented in time linear in the number of nodes. Recall that a node separator ensures that there are no edges between V_1 and V_2. Hence, the computed children are independent sets, but may not be maximal since separator nodes have been ignored and potentially some of them can be added to the solution. Therefore, the child is made maximal by using a greedy algorithm. The operator finishes with one iteration of the ARW algorithm to ensure that a local optimum is reached and to add diversification.

Reduction Techniques. Our algorithm uses exact and inexact reduction techniques to shrink the search space. Each reduction allows us to choose vertices that are in some MIS by following simple rules. If an MIS is found on the kernel graph $\mathcal{K}$, then each reduction may be undone, producing an MIS in the original graph. Refer to Akiba and Iwata [1] for a more thorough discussion, including implementation details. With these facts in mind, we apply the evolutionary algorithm on the kernelized graph instead of on the input graph, thus boosting its performance. We use intermediate solutions of the evoluationary algorithm to select good solution candidates (independent set vertices), remove them and their incident neighbors from the graph, and finally repeat the whole process. For more details on the algorithm, we refer the reader to [29].

3 Graph Format

3.1 Input File Format

The graph format used by our partitioning programs is the same as used by Metis [14], Chaco [12] and the graph format that has been used during the 10th DIMACS Implementation Challenge on Graph Clustering and Partitioning. The input graph has to be undirected, withoud self-loops and without parallel edges.

To give a description of the graph format, we follow the description of the Metis 4.0 user guide very closely. A graph $G = (V, E)$ with n vertices and m edges is stored in a plain text file that contains $n + 1$ lines (excluding comment lines). The first line contains information about the size and the type of the graph, while the remaining n lines contain information for each vertex of G. Any line that starts with % is a comment line and is skipped.

The first line in the file contains two integers n m. The two integers are the number of vertices n and the number of undirected edges of the graph, respectively. Note that in determining the number of edges m, an edge between any pair of vertices v and u is counted *only once* and not twice, i.e. we do not count the edge (v, u) from (u, v) separately. The remaining n lines of the file store information about the actual structure of the graph. In particular, the ith line (again excluding comment lines) contains information about the ith vertex. Each line has the following structure:

$$v_1 \, v_2 \, \ldots \, v_k$$

where $v_1, \ldots, v_k$ are the vertices adjacent to the vertex described by the line. The adjacent vertices must be *sorted* by increasing order. Note that the vertices are numbered starting from 1 (not from 0).

3.2 Output File Formats

The output format is basically a text file that represents the independent set. This file contains n lines. Each line indicates if the correspoding vertex is part of the independent set (1) or not (0), i.e. line i indicates if vertex i (here the vertices are numbered from 0 to $n - 1$) is part of the independent set.

3.3 Troubleshooting

KaMIS should not crash! If KaMIS crashes it is mostly due to the following reasons: the provided graph contains self-loops or parallel edges, the edges are not sorted, there exists a forward edge but the backward edge is missing, or the number of vertices or edges specified does not match the number of vertices or edges provided in the file. To sort the edges of an unsorted graph please use the `sort_metis.py` script in `misc/conversion`. Please use the *graphcheck* tool provided in our package to verify whether your graph has the right input format. If our graphcheck tool tells you that the graph that you provided has the correct format and KaMIS crashes anyway, please write us an email.
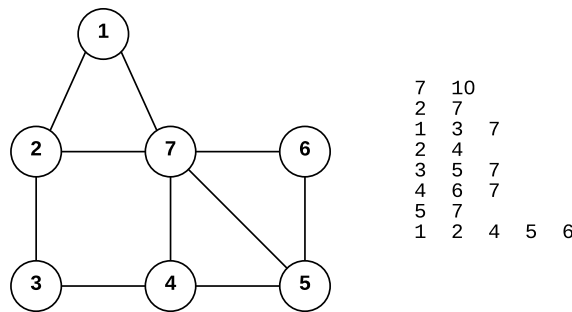

Figure 1: An example graph and its representation in the graph format.

4

4 User Interface

KaMIS contains the program `redumis` to compute an independent set. To compile this program you need to have Argtable, g++, OpenMP and scons installed (we use argtable-2.10, g++-4.8.0 and scons-1.2). Once you have that you can execute `compile.sh` in the main folder of the release. When the process is finished the binaries can be found in the folder `deploy`. We now explain the parameters of each of the programs briefly.

4.1 ReduMIS

Description: This is the evolutionary algorithm based on graph partitioning and reduction techniques.

Usage:
redumis   [--help] FILE [--seed=<int>] [--config=VARIANT] [--time_limit=<double>]
          [--output=<string>] [--console_log] [--all_reductions] [--red_thres=<int>]

Options:

FILE	Path to graph file.
–help	Print help.
–seed=<int>	Seed to use for the PRNG.
–config=VARIANT	Use a preconfiguration. (Default: standard) [standard|social|full_standard|full_social].
	Explanation: standard/social modify the graph partitioning mode.
	"full" configurations use more time consuming parameters.
–time_limit=<double>	Time limit. (Default: 1000s)
–output=<string>	Path to output file.
–console_log	Print verbose output to the console.
–disable_checks	Disable sortedness check during I/O.
–red_thres=<int>	Number of unsuccessful iterations of EA before reduction (Default: 350).

4.2 Graph Format Checker

Description: This program checks if the graph specified in a given file is valid.

Usage:
graphchecker   file

Options:

FILE	Path to the graph file.

References

[1] T. Akiba and Y. Iwata. Branch-and-reduce Exponential/FPT Algorithms in Practice: A Case Study of Vertex Cover. In *Proceedings of the Meeting on Algorithm Engineering & Experiments*, ALENEX '15, pages 70–81, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

[2] D. V. Andrade, M. G. C. Resende, and R. F. Werneck. Fast Local Search for the Maximum Independent Set Problem. *Journal of Heuristics*, 18(4):525–547, 2012.

[3] R. Battiti and M. Protasi. Reactive Local Search for the Maximum Clique Problem. *Algorithmica*, 29(4):610–637, 2001.

[4] T. A. Feo, M. G. C. Resende, and S. H. Smith. A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set. *Operations Research*, 42(5):860–878, 1994.

[5] Eleanor J. Gardiner, , Peter Willett, and Peter J. Artymiuk. Graph-theoretic techniques for macromolecular docking. *Journal of Chemical Information and Computer Science*, 40(2):273–279, 2000.

[6] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[7] A. Gemsa, M. Nöllenburg, and I. Rutter. Evaluation of labeling strategies for rotating maps. In *Experimental Algorithms*, volume 8504 of *LNCS*, pages 235–246. Springer, 2014.

[8] A. Grosso, M. Locatelli, and F. Della C. Combining Swaps and Node Weights in an Adaptive Greedy Approach for the Maximum Clique Problem. *Journal of Heuristics*, 10(2):135–152, 2004.

[9] A. Grosso, M. Locatelli, and W. Pullan. Simple Ingredients Leading to Very Efficient Heuristics for the Maximum Clique Problem. *Journal of Heuristics*, 14(6):587–612, 2008.

[10] P. Hansen, N. Mladenović, and D. Urošević. Variable Neighborhood Search for the Maximum Clique. *Discrete Applied Mathematics*, 145(1):117–125, 2004.

[11] F. Harary and I. C. Ross. A Procedure for Clique Detection Using the Group Matrix. *Sociometry*, 20(3):pp. 205–215, 1957.

[12] B. Hendrickson. Chaco: Software for Partitioning Graphs. `http://www.cs.sandia.gov/~bahendr/chaco.html`.

[13] Y. Iwata, K. Oka, and Y. Yoshida. Linear-time FPT Algorithms via Network Flow. In *Proceedings of 25th ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 1749–1761. SIAM, 2014.

[14] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[15] K. Katayama, A. Hamamoto, and H. Narihisa. An Effective Local Search for the Maximum Clique Problem. *Information Processing Letters*, 95(5):503–511, 2005.

[16] T. Kieritz, D. Luxen, P. Sanders, and C. Vetter. Distributed time-dependent contraction hierarchies. In Paola Festa, editor, *Experimental Algorithms*, volume 6049 of *LNCS*, pages 83–93. Springer Berlin Heidelberg, 2010.

[17] S. Lamm, P. Sanders, and C. Schulz. Graph Partitioning for Independent Sets. In *Proceedings of the 14th International Symposium on Experimental Algorithms (SEA'15)*, volume 8504, pages 68–81. Springer, 2015.

[18] H. Meyerhenke, P. Sanders, and C. Schulz. Partitioning Complex Networks via Size-constrained Clustering. In *13th Int. Symp. on Exp. Algorithms*, LNCS. Springer, 2014.

[19] G.L. Nemhauser and Jr. Trotter, L.E. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[20] W. J. Pullan and H. H. Hoos. Dynamic Local Search for the Maximum Clique Problem. *Journal of Artifical Intelligence Research(JAIR)*, 25:159–185, 2006.

[21] P. San Segundo, F. Matia, D. Rodriguez-Losada, and M. Hernando. An improved bit parallel exact maximum clique algorithm. *Optimization Letters*, 7(3):467–479, 2013.

[22] P. San Segundo, D. Rodríguez-Losada, and Agustín J. An exact bit-parallel algorithm for the maximum clique problem. *Computers & Operations Research*, 38(2):571–581, 2011.

[23] P. V. Sander, D. Nehab, E. Chlamtac, and H. Hoppe. Efficient traversal of mesh edges using adjacency primitives. *ACM Trans. Graph.*, 27(5):144:1–144:9, December 2008.

[24] P. Sanders and C. Schulz. KaHIP – Karlsruhe High Qualtity Partitioning Homepage. `http://algo2.iti.kit.edu/documents/kahip/index.html`.

[25] P. Sanders and C. Schulz. Think Locally, Act Globally: Highly Balanced Graph Partitioning. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, LNCS. Springer, 2013.

[26] E. Tomita, Y. Sutani, T. Higashi, S. Takahashi, and M. Wakatsuki. A simple and faster branch-and-bound algorithm for finding a maximum clique. In Md. Saidur Rahman and Satoshi Fujita, editors, *WALCOM: Algorithms and Computation*, volume 5942 of *LNCS*, pages 191–203. Springer Berlin Heidelberg, 2010.

[27] M. Xiao and H. Nagamochi. Confining sets and avoiding bottleneck cases: A simple maximum independent set algorithm in degree-3 graphs. *Theoretical Computer Science*, 469:92 – 104, 2013.

[28] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. In *3rd International Conference on Knowledge Discovery and Data Mining*, pages 283–286. AAAI Press, 1997.

[29] S. Lamm, P. Sanders, C. Schulz, D. Strash and R. F. Werneck Finding Near-Optimal Independent Sets at Scale. In *Proceedings of the 16th Workshop on Algorithm Engineering and Experimentation (ALENEX'16)*. 2016.