ISSN 2186-7437

NII Shonan Meeting Report

No.144

Parameterized Graph Algorithms & Data Reduction

Bart M.P. Jansen
Christian Schulz
Hisao Tamaki

March 04–08, 2019

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-Ku, Tokyo, Japan

Parameterized Graph Algorithms & Data Reduction

Organizers:
Bart M.P. Jansen (Eindhoven University of Technology, The Netherlands)
Christian Schulz (University of Vienna, Austria)
Hisao Tamaki (Meiji University, Japan)

March 04–08, 2019

Many real-world optimization problems concerning route planning (the Traveling Salesman Problem), scheduling (Graph Coloring), and facility location (Dominating Set), can be formulated as solving an optimization problem on a graph. These three problems, and many others, are NP-hard: it is expected that no efficient (polynomial-time) algorithm exists that always finds an optimal solution. However, many NP-hard graph problems have been shown to be fixed-parameter tractable (FPT): large inputs can be solved efficiently and provably optimally, as long as some problem parameter is small. Here the parameter measures the 'difficulty' of the input in some mathematically well-defined way, for example using the treewidth of the underlying graph. Over the last two decades, significant advances have been made in the design and analysis of FPT algorithms for a wide variety of graph problems. This has resulted in a rich algorithmic toolbox spanning techniques such as bounded-depth search trees, color coding, iterative compression, cut&count, important separators, kernelization, algebraic methods, and dynamic programming on tree decompositions. By now, these techniques are well-established and are described in several textbooks and surveys. They lead to algorithms that are theoretically efficient: they allow problems of size n with a parameter value of k to be solved in time $f(k)n^c$ for some (exponential) function f and constant c, thereby restricting the exponential dependence of the running time to the parameter k only.

These theoretical algorithmic ideas have received very little attention from the practical perspective. Few FPT algorithms are implemented and tested on real datasets, and their practical potential is far from understood. The goal of this workshop has been to bring algorithmic researchers from the FPT community together with algorithm engineers who implement, optimize, and experiment with algorithms running on large data sets, thereby stimulating an exchange of ideas and techniques. On the one hand, experimental results can trigger new theoretical questions and suggest new properties of inputs that are relevant parameters to use in theoretical analysis. In the other direction, the rich toolbox of parameterized algorithm theory offers a rich set of algorithmic ideas that are challenging to implement and engineer in practical settings. By applying techniques from FPT algorithms in nontrivial ways, algorithms can be obtained that perform surprisingly well on real-world instances for NP-hard problems. The viability of this approach has been demonstrated in recent years through the Parameterized Algorithms and Computational Experiments challenge, in which teams compete to solve real-world inputs using ideas from parameterized algorithm design.

The workshop brought together algorithm designers and algorithm engineers to solve algorithmic graph problems. This led to a very fruitful exchange of ideas. There were 30 participants, of which 26 were international. The open problem session, closing session, and talks, led to lively discussions. Topics that were discussed include the importance of parallelism in preprocessing and problem solving, finding problem parameters to explain the effectiveness of preprocessing, how to efficiently implement algorithms that work on tree decomposition, how to develop new reduction rules, and how to prove that heuristic reductions can lead to optimal solutions. We now list some concrete areas in which significant exchange of ideas has been achieved through invited presentations and discussions.

1. **Algorithms exploiting graph decompositions.** Sometimes an input graph can be decomposed into pieces connecting to the remaining graph is limited to a small number of vertices, then this decomposition facilitates a divide-and-conquer approach to solve the problem. One way to formalize the existence and quality of such decompositions theoretically is using notions such as treewidth and pathwidth. From a theoretical perspective there has been a lot of research on computing good decompositions, and exploiting them to solve other graph problems. From the practical side, graph algorithms that exploit a hierarchical partitioning algorithm have been shown to be very successful for computing maximum independent sets, and various types of longest/shortest path problems. The workshop brought researches from both sides together to discuss on how to use partitioning algorithms for other problems that can exploit good decompositions.

2. **Preprocessing algorithms and reduction rules.** A well-known technique for developing FPT algorithms is a reduction to a problem kernel, where one uses reduction rules that shrink the input to size bounded by a function in the parameter, without changing the answer. A variety of interesting mathematical techniques is available for obtaining such reduction rules, using local replacements, matroids, and linear-programming based techniques. Many of the talks that took place during the workshop that presented ideas from the area. For example, the talk by Strash showed how such rules can be used to solve huge independent set instances in practice, and the open problem by Nichterlein shows interesting ideas in that direction for the matching problem.

3. **Lossy kernelization and preprocessing.** It may be that at some point one gets stuck in a situation where no further preprocessing rule can shrink the input under the guarantee that the optimal solution value does not change and the reduced problem is still too large to be solved by an exact solver. However, sometimes it is possible to shrink the input while guaranteeing that the optimal solution value changes only slightly, so that a good approximate solution of the reduced input can be lifted to a good approximate solution of the original. Hence, when computing optimal solutions is computationally infeasible, such lossy preprocessing techniques

may be instrumental in efficiently computing good approximations. A theory of lossy kernelization has recently been proposed, but its effectiveness in practice has not yet been analyzed. During the workshop, we looked at various lossy kernelization techniques, that come without a guarantee but have shown to work well on real-world instances – the practical importance of such techniques, with or without guarantees, has been emphasized.

For most of the meeting talks, the slides can be obtained from the meeting website (`https://www.win.tue.nl/~bjansen/shonan-144/`).

Workshop Outcomes

The long-term impact of the discussions, collaborations, and work on open problems during the workshop is hard to ascertain at the time of writing. But in this section we list several concrete outcomes of the workshop that show its benefits to science.

Finding improving k-OPT moves in bounded-degree graphs

One of the workshop participants, Prof. Yoichi Iwata, posed a theoretical open problem that is motivated by practical considerations. It concerns the classic Traveling Salesman Problem (TSP), in which the goal is to find a tour (a cycle) among n cities that visits each city exactly one. For each pair of cities (i, j) there is a cost $c(i, j)$ for moving from city i to city j, and the goal is to find a tour that minimizes the total cost. This problem has numerous applications, for example in logistics, but also in optimizing the movement of robot arms on an assembly line. While finding optimal tours is often computationally infeasible, a popular strategy for finding good solutions is local search: start from a greedily or randomly constructed tour, and then repeatedly improve the tour by replacing k edges, for some small constant k. Such a replacement step is called a k-OPT move. When no further improvement can be obtained by swapping k edges, the resulting tour is used as the solution.

Existing solvers for TSP are very good at identifying, for each city, the *most promising* edges incident to that city for use on good tours. Since finding k-OPT improvements in general is quite a computationally costly step, one can search only for improvement steps involving these most promising edges, in the hope of finding good solutions faster. This is effective in practice; one of the workshop participants asked whether it is also possible to give theoretical guarantees that improving k-OPT steps can be found quickly in the constant-degree graphs consisting of the most promising edges. During the workshop, this question was resolved negatively by a $W[1]$-hardness proof. This result triggered a team of four workshop participants (Édouard Bonnet, Yoichi Iwata, Bart M. P. Jansen, and Lukasz Kowalik) which had never cooperated before, to work on various theoretical aspects of this practically motivated problem. It led to the submission of a joint paper, which is under review at the time of writing.

Relevance of algorithms exploiting tree decompositions

The Thursday of the workshop was devoted to the discussion of algorithms that compute tree decompositions of graphs, or use such tree decompositions to solve optimization problems efficiently. There was a rich interaction between researchers that work on computing such decompositions, and researchers working on utilizing such decompositions to solve optimization problems of interest. This led to various important realizations that impact the work in these areas.

Concerning the development of algorithms that compute graph decompositions, a widely-held belief was that such tree decompositions can only be practically exploited when their width is quite small, up to roughly 30. This is due to the fact that typical algorithms that solve an optimization problem using such a graph decomposition, perform a dynamic-programming approach whose number of states is exponential in the width of the decomposition. For width values significantly beyond 30, this causes such algorithms to quickly run out of memory. This would suggest that tree decompositions of width beyond 30 are not useful in practice, so that for graphs whose treewidth exceeds 30, there is no practical interest in computing optimal tree decompositions. But during one of the discussions, a custom *hybrid* approach was presented which exploits a tree decomposition not by a direct dynamic-programming algorithm, but by using other solvers such as Answer-Set programming to solve the subproblems. Experiments show that such solvers can even benefit from tree decompositions of width values beyond 100, thereby giving new motivations for computing optimal tree decompositions of graphs of treewidth around 100.

When it comes to algorithms that exploit tree decompositions to solve problems such as HAMILTONIAN CYCLE and STEINER TREE, several presentations highlighted the importance of having a subproblem formulation in the dynamic programming step that allows for early *pruning* of subproblems that provably will not lead to the optimal solution. The pruning can, for example, be done using the rank-based approach (as discussed by Prof. Marcin Pilipczuk and Prof. Hans L. Bodlaender), or by turning the techniques that allow the problem to be solved efficiently on restricted types of inputs, into pruning steps that speed up the solution of general instances (as discussed by Prof. Yoichi Iwata). Such pruning steps of the dynamic program mean that in practice, it can outperform different formulations for which better worst-case guarantees exist. The combination of theoretical and applied talks on this topic identified several interesting follow-up experiments to be done, as well as theoretical questions concerning the approximation of a row basis of a matrix.

Verifying heuristic reductions

The effectiveness of applying reduction rules when solving a large-scale graph problem is well known. For problems such as INDEPENDENT SET, there are powerful reduction rules which can shrink the size of the input in such a way that an optimal solution for the resulting smaller problem, can quickly be translated back into an optimal solution for the original problem. Such reduction rules were discussed by Dr. Darren Strash on Tuesday. Reduction rules can also cascade: applying a rule can trigger new situations in which reduction rules can be applied. Some inputs on thousands of vertices can be solved completely by applying preprocessing rules!

When preprocessing gets stuck, and no reduction rule is applicable for which it can be guaranteed to always result in an optimal solution, the standard approach is to do a branching step: pick a vertex, and branch on either including this vertex in the solution, or not including it. The two generated subproblems are then solved recursively. The simplification done by the branching step may cause preprocessing rules to become applicable, so the two techniques are interleaved. But when the initial preprocessing phase gets stuck on a graph that is very large, the branching process will lead to a search tree of infeasible large size. One approach that is used in practice, is to apply *heuristic* reduction steps: steps for which it cannot be guaranteed that they always lead to an optimal solution, but which reduce the graph and trigger other (safe) reduction rules for further reductions. Situations in which to apply such heuristic reductions can be found using local search and graph partitioning, and have been shown to lead to high-quality solutions on large inputs in practice. Experimental results presented during the workshop revealed that in many situations, different runs of the algorithm (using different heuristic reductions) actually lead to exactly the same final solution, thereby suggesting that this solution may be the optimal one. This led to the following challenge: can it be *proven* based on the local structure of the graph, that a reduction step that is found using local search actually leads to an optimal solution on the given instance? Some preliminary experiments were performed during the workshop, and a deeper exploration has been identified as an interesting challenge for future work.

Overview of Talks

Parameterization for Current and Future Hardware

Petteri Kaski, Aalto University

Computer hardware is becoming increasingly parallel, hierarchical, and constrained by fundamental physical parameters such as the speed of light and atomic scale. For example, a single high-end compute node today can easily possess more than 10,000 cores, and a leading-edge supercomputer consists of more than 100,000,000 cores with a clock-cycle-length of less than a nanosecond. Algorithm design for such systems must take into account the characteristics of the system and its subsystems via parameters such as capacity, bandwidth, latency, and energy consumption of the different subsystems and their communication-interconnects. This talk looks at the parameters and challenges of current hardware illustrated with examples, and takes a speculative outlook towards the future. For example, we will look at our engineering effort (ALENEX'18) to bring a family of delegatable and error-tolerant "Camelot" algorithms (Björklund & K, PODC'16) arising from Williams' (CCC'16) noninteractive Merlin-Arthur proof systems for batch evaluation from theory to the computing practice.

Computing sparsity stuff in real world graphs

Marcin Pilipczuk, University of Warsaw

Graphs of bounded expansion and nowhere dense graphs are robust notions capturing many sparse graph classes. Rich algorithmic theory has been developed in the last decade and recently we have seen a number of experimental works in the area. After a brief survey, I will mostly focus on two primitives in the theory: generalized coloring numbers and the uniform quasi-wideness property from the point of view of algorithmic engineering. Furthermore, we will discuss performance of the algorithms in various real-world graphs (joint work with W. Nadara, R. Rabinovich, F. Reidl, and S. Siebertz, SEA 2018) as well as application of them to kernelization of the dominating set problem (experiments by W. Nadara).

Recent advances in kernelization for the maximum independent set problem

Darren Strash, Hamilton College

In kernelization, a collection of simple rules (called reduction rules) are repeatedly applied to an input instance until the input is reduced to its smallest hardest-to-solve part. Kernelization has long been a powerful technique for developing fixed-parameter tractable algorithms for NP-hard combinatorial problems. Research in the area has exploded in the last 3 years, with a significant number of these results concentrating on the maximum independent set problem. In this talk, we give a thorough overview of the recent theoretical and practical results for this problem, and introduce recent kernelization results for its weighted variant.

A Practical Analysis of Kernelization Techniques for the Maximum Cut Problem

Sebastian Lamm, Karlsruhe Institute of Technology

We discuss existing and new kernelization techniques for a well-known NP-Complete problem: Max-Cut. Given an undirected graph, the task here is to find a bipartition of the vertex set that maximizes the total weight of edges that have their endpoints in different partitions. Practical fields for Max-Cut include modeling of social networks, portfolio risk analysis, network design, and statistical physics. We primarily focus on the unweighted case, but we also consider the signed and weighted versions to some degree. Our work heavily focuses on the practical utility of kernelization techniques for Max-Cut. Kernelization techniques compute a smaller problem instance called a kernel that if solved allows one to compute a solution for the initially given instance. Our work includes both a theoretical analysis of existing reduction rules as well as the development of brand new ones. Using our set of reduction rules we are able to show that kernelization has a significant impact on a multitude of cases: First, we are able to significantly improve the performance of current state of the art Max-Cut solvers. Second, we are able to solve instances that previously took over 6 hours in just a few seconds. Finally, we are able to compute a maximum cut for instances that were previously infeasible.

The Parameterized Complexity of Matrix Completion

Robert Ganian, Technical University Vienna

We consider the parameterized complexity of fundamental matrix completion problems, in which we are given a matrix with missing entries and the task is to complete the matrix in a way that optimizes a desired structural measure. In the first part of the talk, we will be focusing on completing the matrix in a way which either minimizes its rank or minimizes the number of distinct rows that appear in the matrix. Afterwards, we will turn our attention to the problem of completing the matrix in order to obtain a clustering of the rows into a small number of clusters, each with small diameter. The presentation will mainly target fixed-parameter algorithms for these problems, showcasing how kernelization and graph-theoretic tools can be used to solve problems which lie outside of the usual frontiers of parameterized complexity.

Shared-Memory Exact Minimum Cuts

Christian Schulz, University of Vienna

The minimum cut problem for an undirected edge-weighted graph asks us to divide its set of nodes into two blocks while minimizing the weight sum of the cut edges. We engineer the fastest known exact algorithm for the problem. State-of-the-art algorithms like the algorithm of Padberg and Rinaldi or the algorithm of Nagamochi, Ono and Ibaraki identify edges that can be contracted to reduce the graph size such that at least one minimum cut is maintained in the contracted graph. Our algorithm achieves improvements in running time over these algorithms by a multitude of techniques. First, we use a recently developed fast and parallel inexact minimum cut algorithm to obtain a better bound for the problem. Afterwards, we use reductions that depend on this bound to reduce the size of the graph much faster than previously possible. We use improved data structures to further lower the running time of our algorithm. Additionally, we parallelize the contraction routines of Nagamochi et al. . Overall, we arrive at a system that outperforms the fastest state-of-the-art solvers for the exact minimum cut problem significantly.

Diameter: A Case Study for FPT in P and the Graph Parameter Hierarchy

Matthias Bentert, TU Berlin

Parameterized Algorithmics was introduced to gain practical algorithms for relevant special cases of NP-hard problems; however, the approach is not limited to NP-hard problems. In this talk I will give a brief introduction to the recently advocated research direction of “FPT in P”. Afterwards I will present two tools (namely “Graphana” and the “parameter hierarchy”) that we are working on and that we believe to be valuable when working with parameterized algorithms (both for NP-hard and polynomial-time solvable problems and for both theoretical and practical work). Finally, I will demonstrate the usefulness of the parameter hierarchy using the polynomial-time solvable problem of computing the diameter of an undirected graph as an example. The overview is based on our paper (`https://arxiv.org/abs/1802.10048`) and the references therein.

Exact Algorithms for Finding Well-Connected 2-Clubs in Sparse Real-World Graphs: Theory and Experiments

Andre Nichterlein, TU Berlin

Finding (maximum-cardinality) “cliquish” subgraphs is a central topic in graph mining and community detection. A popular clique relaxation are 2-clubs: instead of asking for subgraphs of diameter one (these are cliques), one asks for subgraphs of diameter at most two (these are 2-clubs). A drawback of the 2-club model is that it produces hub-and-spoke structures (typically star-like) as maximum-cardinality solutions. Hence, we study 2-clubs with the additional constraint to be well-connected. More specifically, we investigate the algorithmic complexity for three variants of well-connected 2-clubs, all established in the literature: robust, hereditary, and “connected” 2-clubs. Finding these more dense 2-clubs is NP-hard; nevertheless, we develop an exact combinatorial algorithm, extensively using efficient data reduction rules. Besides several theoretical insights we provide a number of empirical results based on an engineered implementation of our exact algorithm. In particular, the algorithm significantly outperforms existing algorithms on almost all (sparse) real-world graphs we considered.

Algorithms and Software for Large and Complex Networked Systems

Henning Meyerhenke, Humboldt University Berlin

Motivated by big graph problems, I present a brief overview of the algorithmic research in my group in three application fields: network analysis, combinatorial scientific computing and applied combinatorial optimization for the (life) sciences.

Recent Advances on the Parameterized Complexity of Integer Linear Programming

Sebastian Ordyniak, Technical University Vienna

Integer Linear Programming (ILP) is probably the archetypical NP-complete optimisation problem, allowing for the efficient solution of many otherwise intractable optimisation problems in computer science by a translation to ILP. Surprisingly, until very recently, only few tractable classes of ILP had been identified, i.e., ILP is known to be solvable in polynomial-time if the constraint matrix is totally uni-modular (Papadimitriou, Steiglitz 1982) and if the number of variables (or constraints) is constant (Lenstra, 1983). In particular, in contrast to SAT and CSP, ILP had not been studied w.r.t. structural restrictions on the constraint matrix.

The aim of this talk is to survey recent results on the (parameterized) complexity of ILP under structural restrictions on the constraint matrix. Namely, we consider structural restrictions on two graphical models of the constraint matrix (that have had a huge impact for our understanding of SAT and CSP): (1) the primal graph having a vertex for every variable and an edge between every two variables occurring together in a constraint and (2) the incidence graph having a vertex for every variable and every constraint and an edge between a variable and a constraint if the variable occurs (with a non-zero coefficient) in the constraint. After providing an overview about known tractability (and intractability) results w.r.t. well-known decompositional parameters such as treedepth, treewidth, and clique-width, we will show that the well known block-structured ILPs (e.g., n-fold, two-stage stochastic, 4-block N-fold, and tree-fold ILPs) can be modelled (and generalised) in terms of certain structural parameters on the primal and incidence graph.

Theoretical and practical algorithms for CSP parameterized by the number of variables with value-independent constraints

Gregory Gutin, Royal Holloway University of London

CSP parameterized by the number of variables is W[1]-hard and in W[2]. Fortunately, CSP parameterized by the number of variables with value-independent constraints is FPT and of interest in (security) access control. We'll discuss theoretical results for such a CSP as well as results of computational experiments with FPT algorithms and SAT and CSP solvers.

Towards an efficient implementation of a generic solver for fixed-cardinality optimization in graphs

Christian Komusiewicz, University of Marburg

In fixed-cardinality optimization problems in graphs, we are given a graph $G = (V, E)$, an objective function f , and an integer k and search for a set $S \subseteq V$ of k vertices that maximizes $f(G[S])$. We implement an enumeration-based algorithm for solving fixed-cardinality optimization problems when $G[S]$ needs to be connected. To avoid enumerating all connected subgraphs of order k, we present several generic pruning rules. Finally, we perform an experimental analysis of the performance of the algorithm and the usefulness of the pruning rules for eight example problems.

Computing treewidth via exact and heuristic lists of minimal separators

Hisao Tamaki, Meiji University

We develop practically efficient algorithms for computing the treewidth $\mathrm{tw}(G)$ of a graph G. The core of our approach is a new dynamic programming algorithm which, given a graph G, a positive integer k, and a set Delta of minimal separators of G, decides if G has a tree-decomposition of width at most k of a certain canonical form that uses minimal separators only from Delta, in the sense that the intersection of every pair of adjacent bags belongs to Delta. This algorithm is used to show a lower bound of $k + 1$ on $\mathrm{tw}(G)$, setting Delta to be the set of all minimal separators of cardinality at most k and to show an upper bound of k on $\mathrm{tw}(G)$, setting Delta to be some, hopefully rich, set of such minimal separators. Combining this algorithm with new algorithms for exact and heuristic listing of minimal separators, we obtain exact algorithms for treewidth which overwhelmingly outperform previously implemented treewidth algorithms.

Separator-based Pruned Dynamic Programming for Steiner Tree

Yoichi Iwata, National Institute of Informatics

Steiner tree is a classical NP-hard problem that has been extensively studied both theoretically and empirically. When parameterized by the number of terminals, a dynamic programming algorithm can solve the problem in FPT time. In this talk, I present a novel separator-based pruning technique for speeding up the DP algorithm, which is the key technique used in our winning solver for

the PACE challenge 2018. The idea behind our pruning came from a theoretical algorithm for a special case where the graph is planar and all the terminals on a single face. Although this special case is too special to apply in practice, we found that the idea behind this improvement can be used for pruning. Our empirical evaluation shows that our pruned DP algorithm is quite effective against real-world instances admitting small separators, scales to more than a hundred terminals, and is competitive with a branch-and-cut solver.

Finding Hamiltonian Cycle in Graphs of Bounded Treewidth: Experimental Evaluation

Marcin Pilipczuk, University of Warsaw

The notion of treewidth, introduced by Robertson and Seymour in their seminal Graph Minors series, turned out to have tremendous impact on graph algorithmics. Many hard computational problems on graphs turn out to be efficiently solvable in graphs of bounded treewidth: graphs that can be sweeped with separators of bounded size. These efficient algorithms usually follow the dynamic programming paradigm. In the recent years, we have seen a rapid and quite unexpected development of involved techniques for solving various computational problems in graphs of bounded treewidth. One of the most surprising directions is the development of algorithms for connectivity problems that have only single-exponential dependency (i.e., $2^{O(t)}$) on the treewidth in the running time bound, as opposed to slightly superexponential (i.e., $2^{O(t \log t)}$) stemming from more naive approaches. In this work, we perform a thorough experimental evaluation of these approaches in the context of one of the most classic connectivity problem, namely Hamiltonian Cycle.

Projected model counting using small treewidth

Markus Hecher, Technical University Vienna

We discuss algorithms to solve projected model counting (PMC). PMC asks to count solutions of a Boolean formula with respect to a given set of projected variables, where multiple solutions that are identical when restricted to the projected variables count as only one solution. Our algorithm exploits small treewidth of the primal graph of the input instance. It runs in double exponential time in the treewidth and is polynomial in the size of the instance. In other words, we obtain that the problem PMC is fixed-parameter tractable when parameterized by treewidth. Further, we take the exponential time hypothesis (ETH) into consideration and establish lower bounds of bounded treewidth algorithms for PMC, yielding that the runtime bounds of our algorithm cannot be significantly improved. Finally, we generalize our result to PMC for quantified boolean formulas (QBFs). For solving the decision problem QSAT for QBFs there are already competitive solvers based on treewidth as, e.g., dynqbf. In order to successfully extend dynqbf for counting and solving PMC, we present our idea of practical hybrid parameterized solving. In particular, hybrid solvers aim at leveraging from the interplay between parameterized dynamic programming algorithms and established monolithic solvers.

Weighted Model Counting on the GPU by Exploiting Small Treewidth

Johannes Fichte, TU Dresden

We propose a novel solver that efficiently finds almost the exact number of solutions of a Boolean formula (#SAT) and the weighted model count of a weighted Boolean formula (WMC) if the treewidth of the given formula is sufficiently small. The basis of our approach are dynamic programming algorithms on tree decompositions, which we engineered towards efficient parallel execution on the GPU. We provide thorough experiments and compare the runtime of our system with state-of-the-art #SAT and WMC solvers. Our results are encouraging in the sense that also complex reasoning problems can be tackled by parameterized algorithms executed on the GPU if instances have treewidth at most 30, which is the case for more than half of counting and weighted counting benchmark instances. In addition, we will present the latest version gpusat2, which uses improved data structures and preprocessing and allows for solving instances up to large treewidth (before reduction).

From theory to practice in k-OPT heuristic for Travelling Salesman Problem

Lukasz Kowalik, University of Warsaw

Given a traveling salesman problem (TSP) tour H in graph G, a k-move is an operation which removes k edges from H, and adds k edges of G so that a new tour H' is formed. The popular k-opt heuristic for TSP finds a local optimum by starting from an arbitrary tour H and then improving it by a sequence of k-moves. In 2016 and 2017 two papers improved the $O(n^k)$ running time of the naive algorithm for finding an improving k-move. I will present the second algorithm, which runs in time $O(n^{c_k \cdot k})$, where $\lim c_k = 1/4$. While this algorithm, implemented "as is" is unlikely to be useful in practice, I will present a new heuristic based on it, and some preliminary experimental results.

An ETH-Tight Exact Algorithm for Euclidean TSP

Hans Bodlaender, Utrecht University

In this talk, we look at the exact complexity of the Euclidean Travelling Salesman Problem; the input for TSP is a set of points in d-dimensional space with Euclidean distances. Assuming the Exponential Time Hypothesis, we settle the asymptotic complexity of this problem, and give an algorithm that runs in $2^{O(n^{1-1/d})}$ time and show a lowerbound conditional to the ETH of $2^{\Omega(n^{1-1/d})}$ time, for $d \geq 2$.

Edge disjoint path packing

Peter Rossmanith, RWTH Aachen

Packing edge disjoint paths into a graph is a notoriously complicated problem from a complexity theoretic view. To pack even a single path into a general graph is NP-hard. For k paths the problem is W[1]-hard even on graphs of tree-width two. It becomes fixed parameter tractable on forests and an implementation of this algorithms seems to work well in practice. There is also an algorithm running in time $n^{O(k^2)}$ for an arbitrary number of paths if the graph has maximal degree bounded by k, at most k components, and at most k vertices with degree higher than two. This running time is almost optimal and the algorithm is practical on small instances.

Overview of Open Problems

k-exchange TSP for bounded-degree graphs

Prof. Yoichi Iwata, National Institute of Informatics

For general graphs, it is known that this problem is W[1]-hard and there is an XP algorithm running in $n^{(1/4+\epsilon_k)k}$ time. How about when the graph is sparse?

I'm interested in this bounded-degree case because, before running the local search, state-of-the-art TSP solvers sparsify the input graph. At the Shonan meeting, we obtained proof of W[1]-hardness and a faster XP algorithm running in $n^{(0.174+\epsilon_k)k}$ time.

Simple Kernel for Maximum Cardinality Matching

Dr. Andre Nichterlein, TU Berlin

Data Reduction Rule 1:
Let v be a vertex of degree one. Delete v and its neighbor.

Data Reduction Rule 2:
Let v be a degree-two vertex. Then delete v and merge its two neighbors.

For both rules it holds that the starting instance has a matching of size k if and only if the resulting instance has a matching of size $k-1$. Essentially the same rules work also for Vertex Cover and Independent Set. My question is: What parameter describes graphs where the above rules work well? Experiments show that the kernel with respect to the parameter feedback edge set is insufficient to explain the success of the rules [1].

[1] V. Korenwein, A. Nichterlein, R. Niedermeier, P. Zschoche. Data Reduction for Maximum Matching on Real-World Graphs: Theory and Experiments. In Proceedings of the 26th Annual European Symposium on Algorithms (ESA '18), pages 53:153:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

Finding Balanced Separators

Dr. Sebastian Ordyniak, Technical University Vienna

Finding a balanced separators in a graph is one of the fundamental problems in graph theory with numerous applications for the design of efficient algorithms on graphs. Given an undirected graph G and a number k, the problem asks whether there is set S of at most k vertices such that removing S from G results in “small” components (usually components of size at most αn, for some $1/2 \leq \alpha < 1$). Finding balanced separators allows one to recursively decompose the graph into small pieces, which can be processed independently. While the theoretical implications of balanced separators for the design of efficient algorithms for many problems on graphs are well understood, there seems to be a distinct lack of research into how balanced separators can best be exploited for the design of practically efficient algorithms (exact, heuristic, or approximation). Given that balanced separators are applicable for the majority of problems on graphs (such as all problems that benefit from bounded treewidth), I think it would therefore be interesting and worthwhile to explore the potential of balanced separators for the efficient solution of real-world instances.

Extended Travelling Salesman Problem

Dr. Sergey Pupyrev, Facebook

Let $G = (V, E, w)$ be a directed weighted graph where an edge $(s, t) \in E$ has a weight, $w(s, t) > 0$. The goal is to find an ordering of V, $\Pi : 1, \ldots, |V| \to V$, so as to maximize the following expression, called ExtTSP:

$$\text{ExtTSP} = \sum_{(s,t)\in E} w(s,t) \times \begin{cases} k_f(\Pi(t)\Pi(s)) & \text{if } \Pi(s) \leq \Pi(t) \\ k_b(\Pi(s)\Pi(t)) & \text{if } \Pi(s) > \Pi(t) \end{cases},$$

where the summation is taken over all edges of G and $k_f, k_b : [0, |V|) \to R$ are distance coefficients for forward and backward edges, respectively. Notice that the classical (maximum) Travelling Salesman Problem is an instance of ExtTSP when $k_f(1) = 1$ and $k_b(i) = 0$ for all i. We assume that both k_f and k_b are decreasing functions, that is, short edges are more important for the objective than the long ones. We are particularly interested in the case when $k_f(i) = k_b(i) = 0$ for all $i \geq C$ for some constant $C > 0$. ExtTSP has practical practical instances in the context of compiler optimization.

- Is there a (constant-factor) approximation algorithm or PTAS for the problem?
- Can the above problem be solved optimally on control flow graphs (e.g., having small treewidth)?

Call for Reductions

Dr. Christian Schulz, University of Vienna

I am interested in reductions with provable guarantees for balanced graph partitioning, modularity and conductance clustering as well as the longest path problem. For each of those problems, we have good solvers in practice, but currently reductions are missing to compute problem kernels and potentially make the solvers more efficient.

Exploiting Reductions for Graph Coloring

Dr. Darren Strash, Hamilton College

While several simple reductions exist for graph coloring, little has been done with them in the literature (both in theory and practice). Kernelization with these reductions may be a dead end, but what about exact exponential algorithms? The best branching algorithms in theory include the algorithm of Beigel and Eppstein [Journal of Algorithms, 2004] for 3-coloring in $O^*(1.3289^n)$ time, using data reductions for CSP; an algorithm for finding a minimum-color proper coloring in exponential space in $O^*(2^n)$ time using inclusion-exclusion [Koivisto, FOCS 2006]; and a polynomial space algorithm [Gaspers and Lee, COCOON 2017] that uses maximal independent set counting as a subroutine, with running time $O^*(2.2358^n)$. In contrast, the fastest algorithms for independent set utilize problem-specific data reductions. What is possible if we utilize simple reductions in a branch-and-reduce algorithm? Is this worth doing in practice? Do reductions give way to a better analysis with measure and conquer?

Cluster Editing

Prof. Christian Komusiewicz, University of Marburg

For the NP-hard CLUSTER EDITING problem, a practical branch and bound algorithm can be obtained by computing a lower bound based on the size of a maximal set of vertex-pair disjoint induced P_3s [1]. Herein, two P_3s are vertex-pair disjoint if their vertex sets have at most one common element. We would like to explain the fact that these branch and bound algorithms perform well in practice by answering whether the following problem fixed-parameter tractable.

> ABOVE-GUARANTEE CLUSTER EDITING
> **Input:** A graph $G = (V, E)$, a set $\mathcal{P}$ of vertex-pair disjoint induced P_3s in G, and an integer $k \in \mathbb{N}$.
> **Question:** Can G be transformed into a cluster graph by at most k edge modifications?
> **Parameter:** $k - |\mathcal{P}|$.

A graph is a cluster graph if and only if it does not contain a P_3 as an induced subgraph. Since the graphs in $\mathcal{P}$ are vertex-pair disjoint, every edge modification may destroy at most one of the forbidden subgraphs in P, hence we have $k \geq |\mathcal{P}|$. If the forbidden induced subgraphs in $\mathcal{P}$ are even vertex-disjoint, then above-guarantee parameterization is fixed-parameter tractable [2].

[1] Sepp Hartung, Holger H. Hoos. Programming by Optimisation Meets Parameterised Algorithmics: A Case Study for Cluster Editing. In Learning and Intelligent Optimization - 9th International Conference, LION 9. Lecture Notes in Computer Science, volume 8994, pages 43–58, Springer, 2015.

[2] René van Bevern, Vincent Froese and Christian Komusiewicz. Parameterizing Edge Modification Problems Above Lower Bounds. In Theory Computing Systems, volume 62, no 3, pages 739–770, 201.

Data reduction for Graph Coloring

Prof. Bart M. P. Jansen, Eindhoven University of Technology

The q-Coloring problem asks whether the vertices of a given undirected graph can be colored with q colors, so that the endpoints of each edge receive different colors. Since it is already famously NP-complete for $q = 3$, its parameterized complexity has been investigated using parameters that express the structural complexity of the input graph. This led to a preprocessing algorithm by Jansen and Pieterse in IPEC 2017 (`http://dx.doi.org/10.4230/LIPIcs.IPEC.2017.22`), which is guaranteed to reduce the number of vertices in an input instance to $O(k^{q-1})$, where k is the size of a minimum vertex cover of the input graph. A novel aspect of this preprocessing algorithm is that it uses linear-algebraic dependence testing to find irrelevant vertices, which means the algorithm does not require knowledge of (an approximation of) a vertex cover in the graph to be able to work. It would be interesting to see whether this preprocessing algorithm leads to a large amount of data reduction in practice. To implement the preprocessing algorithm efficiently for large graphs, there are several engineering challenges in order to deal with the linear-algebraic dependency testing.

List of Participants

- Max Bannach, M. Sc., University of Luebeck
- Mr. Matthias Bentert, TU Berlin
- Prof. Hans Bodlaender, Utrecht University
- Dr. Edouard Bonnet, Middlesex University
- Prof. Benjamin Burton, University of Queensland
- Dr. Johannes Fichte, TU Dresden
- Dr. Robert Ganian, Technical University Vienna
- Prof. Gregory Gutin, Royal Holloway University of London
- Mr. Markus Hecher, Technical University Vienna
- Prof. Yoichi Iwata, National Institute of Informatics
- Prof. Bart M. P. Jansen, Eindhoven University of Technology
- Prof. Petteri Kaski, Aalto University
- Prof. Christian Komusiewicz, University of Marburg
- Prof. Lukasz Kowalik, University of Warsaw
- Prof. Stefan Kratsch, Humboldt University Berlin
- Mr. Sebastian Lamm, Karlsruhe Institute of Technology
- Prof. Henning Meyerhenke, Humboldt University Berlin
- Dr. Andre Nichterlein, TU Berlin
- Prof. Yoshio Okamoto, University of Electro-Communications
- Dr. Sebastian Ordyniak, Technical University Vienna
- Prof. Yota Otachi, Kumamoto University
- Dr. Marcin Pilipczuk, University of Warsaw
- Dr. Sergey Pupyrev, Facebook
- Prof. Peter Rossmanith, RWTH Aachen
- Dr. Christian Schulz, University of Vienna
- Dr. Florian Sikora, University Paris Dauphine LAMSADE
- Dr. Darren Strash, Hamilton College
- Prof. Hisao Tamaki, Meiji University
- Prof. Takeaki Uno, National Institute of Informatics
- Prof. Yushi Uno, Osaka Prefecture University

Meeting Schedule

Sunday March 3rd, 2019

- 15:00 Check-in
- 19:00 - 21:00 Welcome banquet

Monday March 4th, 2019: Welcome & open problems

- 09:00 - 10:00 Shonan introduction & participant introductions
- 10:00 - 10:30 Coffee break
- 10:30 - 11:00 Continuation of participant introductions
- 11:00 - 12:00 Petteri Kaski: Parameterization for current and future hardware
- 12:00 - 13:30 Lunch
- 14:00 - 15:00 Open problem session
- 15:00 - 15:30 Coffee break
- 15:30 - 16:30 Marcin Pilipczuk: Computing sparsity stuff in real world graphs
- 16:30 - 18:00 Free discussion time
- 18:00 - 19:00 Dinner

Tuesday March 5th, 2019: Kernelization & preprocessing

- 09:00 - 10:00 Darren Strash: Recent advances in kernelization for the maximum independent set problem
- 10:00 - 10:30 Coffee break
- 10:30 - 11:00 Sebastian Lamm: A Practical Analysis of Kernelization Techniques for the Maximum Cut Problem
- 11:00 - 11:30 Robert Ganian: The Parameterized Complexity of Matrix Completion
- 11:30 - 11:40 Group photo
- 12:00 - 13:30 Lunch
- 14:00 - 14:30 Christian Schulz: Shared-Memory Exact Minimum Cuts
- 14:30 - 15:00 Matthias Bentert: Diameter: A Case Study for FPT in P and the Graph Parameter Hierarchy
- 15:00 - 15:30 Coffee break
- 15:30 - 16:00 Andre Nichterlein: Exact Algorithms for Finding Well-Connected 2-Clubs in Sparse Real-World Graphs: Theory and Experiments

- 16:00 - 16:30 Henning Meyerhenke: Algorithms and Software for Large and Complex Networked Systems
- 16:30 - 18:00 Free discussion time
- 18:00 - 19:00 Dinner

Wednesday March 6th, 2019: Integer linear programming & Constraint Satisfaction

- 09:00 - 10:00 Sebastian Ordyniak: Recent Advances on the Parameterized Complexity of Integer Linear Programming
- 10:00 - 10:30 Coffee break
- 10:30 - 11:30 Gregory Gutin: Theoretical and practical algorithms for CSP parameterized by the number of variables with value-independent constraints
- 11:30 - 12:00 Christian Komusiewicz: Towards an efficient implementation of a generic solver for fixed-cardinality optimization in graphs
- 12:00 - 13:30 Lunch
- 13:30 - 18:15 Excursion
- 18:15 - 20:15 Main Banquet

Thursday March 7th, 2019: Approaches based on treewidth

- 09:00 - 10:00 Hisao Tamaki: Computing treewidth via exact and heuristic lists of minimal separators
- 10:00 - 10:30 Coffee break
- 10:30 - 11:00 Yoichi Iwata: Separator-based Pruned Dynamic Programming for Steiner Tree
- 11:00 - 11:30 Marcin Pilipczuk: Finding Hamiltonian Cycle in Graphs of Bounded Treewidth: Experimental Evaluation
- 12:00 - 13:30 Lunch
- 14:00 - 14:30 Markus Hecher: Projected model counting using small treewidth
- 14:30 - 15:00 Johannes Fichte: Weighted Model Counting on the GPU by Exploiting Small Treewidth
- 15:00 - 15:30 Coffee break
- 15:30 - 16:00 Lukasz Kowalik: From theory to practice in k-OPT heuristic for Travelling Salesman Problem
- 16:00 - 18:00 Free discussion time
- 18:00 - 19:00 Dinner

Friday March 8th, 2019: Parameterized algorithms

- 09:00 - 10:00 Hans Bodlaender: An ETH-Tight Exact Algorithm for Euclidean TSP
- 10:00 - 10:30 Coffee break
- 10:30 - 11:00 Peter Rossmanith: Edge disjoint path packing
- 11:00 - 11:30 Closing session
- 11:30 - 13:00 Lunch
- 14:00 - 18:00 Free discussion and departure