Publications of Christopher Weyand

This document lists all peer-reviewed publications of Christopher Weyand, Chair for Algorithm Engineering, Hasso Plattner Institute, Potsdam, Germany. This listing was automatically generated on April 23, 2024. An up-to-date version is available online at hpi.de/friedrich/docs/publist/weyand.pdf.

Conference papers

[1] Böther, M., Kißig, O., Weyand, C., Efficiently Computing Directed Minimum Spanning Trees. In: *SIAM Symposium on Algorithm Engineering and Experiments (ALENEX)*, pp. 86–95, 2023.

Computing a directed minimum spanning tree, called arborescence, is a fundamental algorithmic problem, although not as common as its undirected counterpart. In 1967, Edmonds discussed an elegant solution. It was refined to run in O(min(n2, m log n)) by Tarjan which is optimal for very dense and very sparse graphs. Gabow et al. gave a version of Edmonds' algorithm that runs in O(n log n +m), thus asymptotically beating the Tarjan variant in the regime between sparse and dense. Despite the attention the problem received theoretically, there exists, to the best of our knowledge, no empirical evaluation of either of these algorithms. In fact, the version by Gabow et al. has never been implemented and, aside from coding competitions, all readily available Tarjan implementations run in O(n2). In this paper, we provide the first implementation of the version by Gabow et al. as well as five variants of Tarjan's version with different underlying data structures. We evaluate these algorithms and existing solvers on a large set of real-world and random graphs.

[2] Bläsius, T., Friedrich, T., Stangl, D., Weyand, C., An Efficient Branch-and-Bound Solver for Hitting Set. In: *Algorithm Engineering and Experiments (ALENEX)*, 2022.

The hitting set problem asks for a collection of sets over a universe U to find a minimum subset of U that intersects each of the given sets. It is NP-hard and equivalent to the problem set cover. We give a branch-and-bound algorithm to solve hitting set. Though it requires exponential time in the worst case, it can solve many practical instance from different domains in reasonable time. Our algorithm outperforms a modern ILP solver, the state-of-the-art for hitting set, by at least an order of magnitude on most instances.

[3] Bläsius, T., Fischbeck, P., Gottesbüren, L., Hamann, M., Heuer, T., Spinner, J., Weyand, C., Wilhelm, M., A Branch-and-Bound Algorithm for Cluster Editing. In: *Symposium on Experimental Algorithms (SEA)*, pp. 13:1–13:19, 2022.

The editing problem asks to transform a given graph into a disjoint union of cliques by inserting and deleting as few edges as possible. We describe and evaluate an exact branch-and-bound algorithm for cluster editing. For this, we introduce new reduction rules and adapt existing ones. Moreover, we generalize a known packing technique to obtain lower bounds and experimentally show that it contributes significantly to the performance of the solver. Our experiments further evaluate the effectiveness of the different reduction rules and examine the effects of structural properties of the input graph on solver performance. Our solver won the exact track of the 2021 PACE challenge.

[4] Bläsius, T., Friedrich, T., Weyand, C., Efficiently Computing Maximum Flows in Scale-Free Networks. In: *European Symposium on Algorithms (ESA)*, pp. 21:1–21:14, 2021.

We study the maximum-flow/minimum-cut problem on scale-free networks, i.e., graphs whose degree distribution follows a power-law. We propose a simple algorithm that capitalizes on the fact that often only a small fraction of such a network is relevant for the flow. At its core, our algorithm augments Dinitz's algorithm with a balanced bidirectional search. Our experiments on a scale-free random network model indicate sublinear run time. On scale-free real-world networks, we outperform the commonly used highest-label Push-Relabel implementation by up to two orders of magnitude. Compared to Dinitz's original algorithm, our modifications reduce the search space, e.g., by a factor of 275 on an autonomous systems graph. Beyond these good run times, our algorithm has an additional advantage compared to Push-Relabel. The latter computes a preflow, which makes the extraction of a minimum cut potentially more difficult. This is relevant, for example, for the computation of Gomory-Hu trees. On a social network with 70000 nodes, our algorithm computes the Gomory-Hu tree in 3 seconds compared to 12 minutes when using Push-Relabel.

[5] Scheibel, W., Weyand, C., Bethge, J., Döllner, J., Algorithmic Improvements on Hilbert and Moore Treemaps for Visualization of Large Tree-structured Datasets. In: *EUROVIS 2021 - Shorts*, 2021.

[6] Bläsius, T., Fischbeck, P., Gottesbüren, L., Hamann, M., Heuer, T., Spinner, J., Weyand, C., Wilhelm, M., PACE Solver Description: The KaPoCE Exact Cluster Editing Algorithm. In: *International Symposium on Parameterized and Exact Computation (IPEC)*, pp. 27:1–27:3, 2021.

The cluster editing problem is to transform an input graph into a cluster graph by performing a minimum number of edge editing operations. A cluster graph is a graph where each connected component is a clique. An edit operation can be either adding a new edge or removing an existing edge. In this write-up we outline the core techniques used in the exact cluster editing algorithm of the KaPoCE framework (contains also a heuristic solver), submitted to the exact track of the 2021 PACE challenge.

[7] Bläsius, T., Fischbeck, P., Gottesbüren, L., Hamann, M., Heuer, T., Spinner, J., Weyand, C., Wilhelm, M., PACE Solver Description: KaPoCE: A Heuristic Cluster Editing Algorithm. In: *International Symposium on Parameterized and Exact Computation (IPEC)*, pp. 31:1–31:4, 2021.

The cluster editing problem is to transform an input graph into a cluster graph by performing a minimum number of edge editing operations. A cluster graph is a graph where each connected component is a clique. An edit operation can be either adding a new edge or removing an existing edge. In this write-up we outline the core techniques used in the heuristic cluster editing algorithm of the Karlsruhe and Potsdam Cluster Editing (KaPoCE) framework, submitted to the heuristic track of the 2021 PACE challenge.

[8] Bläsius, T., Friedrich, T., Katzmann, M., Meyer, U., Penschuck, M., Weyand, C., Efficiently Generating Geometric Inhomogeneous and Hyperbolic Random Graphs. In: *European Symposium on Algorithms (ESA)*, pp. 21:2–21:14, 2019. EATCS Best Paper Award.

Hyperbolic random graphs (HRG) and geometric inhomogeneous random graphs (GIRG) are two similar generative network models that were designed to resemble complex real world networks. In particular, they have a power-law degree distribution with controllable exponent β, and high clustering that can be controlled via the temperature T. We present the first implementation of an efficient GIRG generator running in expected linear time. Besides varying temperatures, it also supports underlying geometries of higher dimensions. It is capable of generating graphs with ten million edges in under a second on commodity hardware. The algorithm can be adapted to HRGs. Our resulting implementation is the fastest sequential HRG generator, despite the fact that we support non-zero temperatures. Though non-zero temperatures are crucial for many applications, most existing generators are restricted to $T = 0$. We also support parallelization, although this is not the focus of this paper. Moreover, we note that our generators draw from the correct probability distribution, i.e., they involve no approximation. Besides the generators themselves, we also provide an efficient algorithm to determine the non-trivial dependency between the average degree of the resulting graph and the input parameters of the GIRG model. This makes it possible to specify $\bar{d}$ as input and obtain a graph with expected average degree $\bar{d}$. Moreover, we investigate the differences between HRGs and GIRGs, shedding new light on the nature of the relation between the two models. Although HRGs represent, in a certain sense, a special case of the GIRG model, we find that a straight-forward inclusion does not hold in practice. However, the difference is negligible for most use cases.

[9] Scheibel, W., Weyand, C., Döllner, J., EvoCells – A Treemap Layout Algorithm for Evolving Tree Data. In: *9th International Conference on Information Visualization Theory and Applications (IVAPP)*. INSTICC, pp. 273–280, 2018.

We propose the rectangular treemap layout algorithm EvoCells that maps changes in tree-structured data onto an initial treemap layout. Changes in topology and node weights are mapped to insertion, removal, growth, and shrinkage of the layout rectangles. Thereby, rectangles displace their neighbors and stretch their enclosing rectangles with a run-time complexity of O(nlogn). An evaluation using layout stability metrics on the opensource ElasticSearch software system suggests EvoCells as a valid alternative for stable treemap layouting.