Karlsruhe Institute of Technology

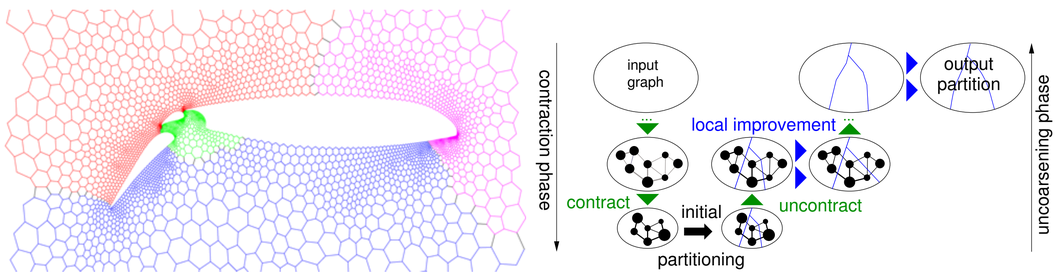
Institute of Theoretical Informatics
Algorithmics

KaHIP – Karlsruhe High Quality Partitioning

Peter Sanders and Christian Schulz

1. The Graph Partitioning Problem

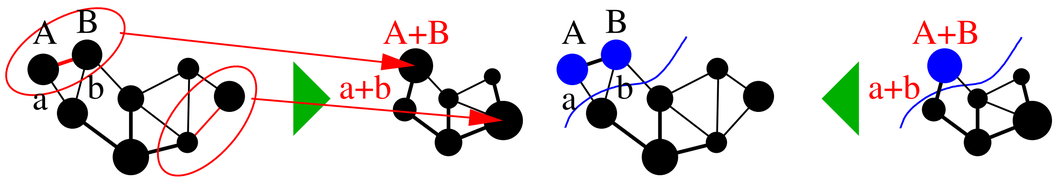
Due to many technical advances of the last decades, networks are used everywhere. Graphs can be used to model relationships in networks or other important data. Given an undirected graph $G = (V, E)$, the graph partitioning problem asks for a division of a graph's vertex set into k equally sized blocks $V_1, \ldots, V_k$ such that the number of edges that run between the blocks, i.e. $\{\{u, v\} \in E \mid u \in V_i, v \in V_j, i \neq j\}$, is minimized. Partitioning graphs has applications everywhere. For example, in parallel computing good partitionings of unstructured graphs are very valuable.



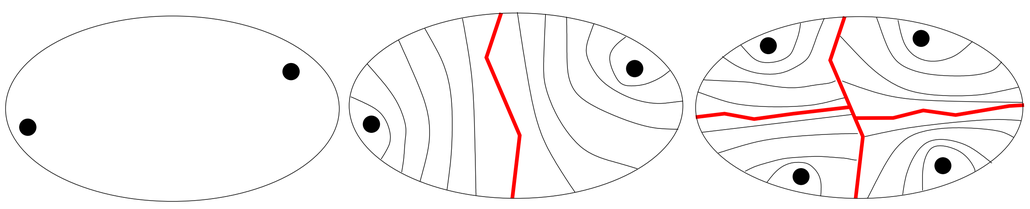
2. Multilevel Graph Partitioning

The most commonly used method in practice is the multilevel approach. During a coarsening phase, a multilevel algorithm reduces the graph size by contracting vertices and edges until the graph is small enough to be directly partitioned. A partition of the input graph is then constructed by successively transferring the solution to the next finer graph and applying a local search algorithm to improve the current solution.

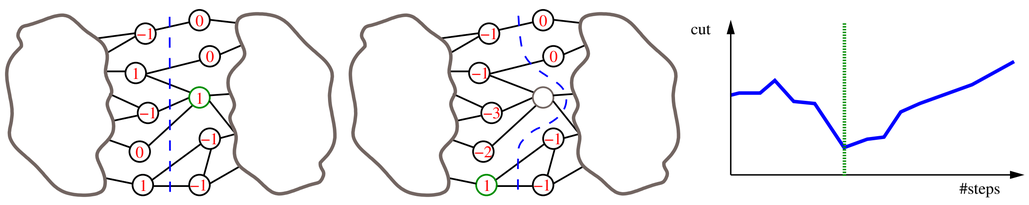
Coarsening Phase. There are multiple ways to define contraction schemes. One is by computing a matching in the graph. Each matched edge is contracted into a coarse vertex. It is ensured that a partition of a coarse graph can be easily transferred to a partition on the next finer level. Weights are defined such that objective and block balance of the transferred solution are identical to the partition on the coarse level. Contraction is done until the graph is small.



Initial Partitioning. Typically, when the graph is small enough a heuristic is applied to solve the problem. One such heuristics starts by finding to vertices that are far away from each other, e.g. by performing breadth first searches repeatedly. To a compute a bipartition of the graph, one performs an additional breadth first search that alternates between both sides and assigns the vertices accordingly. To compute a k-way partition one can recurse on each of the blocks.



Local Search. Local search moves vertices between the blocks to improve an objective function. One possibility is the algorithm by Fiduccia and Mattheyses. The algorithm assigns each vertex a gain that equals the amount that the objective will be reduced when the vertex is moved to the opposite block. It proceeds by alternating between the blocks – always moving the vertex with the largest gain. One a vertex is moved, it cannot be moved back and gains of adjacent vertices are updated. In the end the best partition that has been found is reconstructed.



3. Open Source Graph Partitioning Software

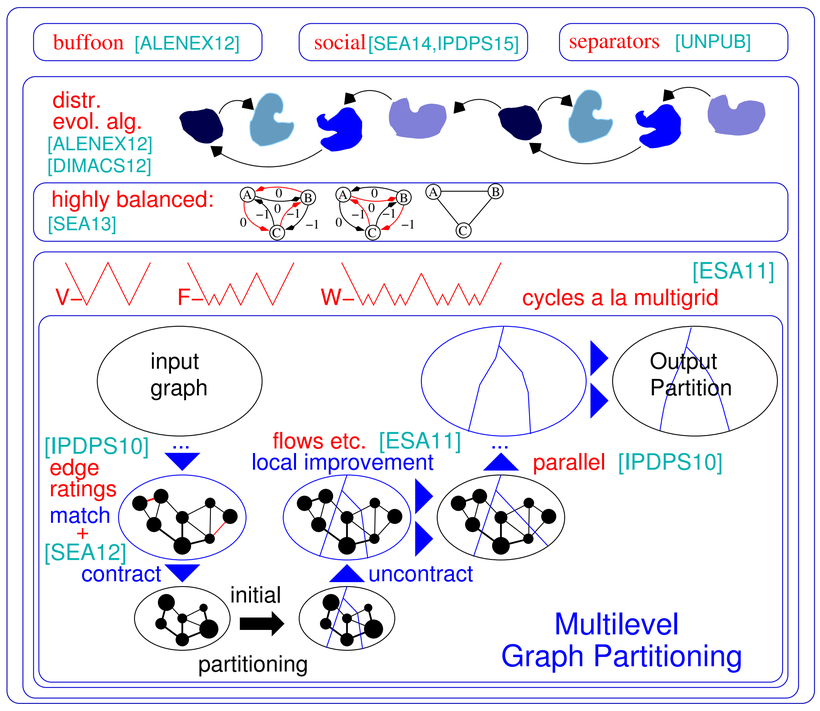
Although several successful multilevel partitioners have been developed in the last decades, we had the impression that certain aspects of the multilevel method are not well understood. This motivated us to make a fresh start, putting all aspects of the scheme on trial.

3.1 High Solution Quality

The perspective taken is that we developed our graph partitioners in a benchmark driven way achieving almost all optimal entries in the Walshaw benchmark as well as scoring most of the points in the graph partitioning subchallenge of the 10th DIMACS Implementation Challenge on Graph Partitioning and Graph Clustering. Another equally valid perspective is that we apply the methodology of algorithm engineering to all aspects of the multilevel graph partitioning approach, achieving improvements in coarsening, refinement, parallelization, ...

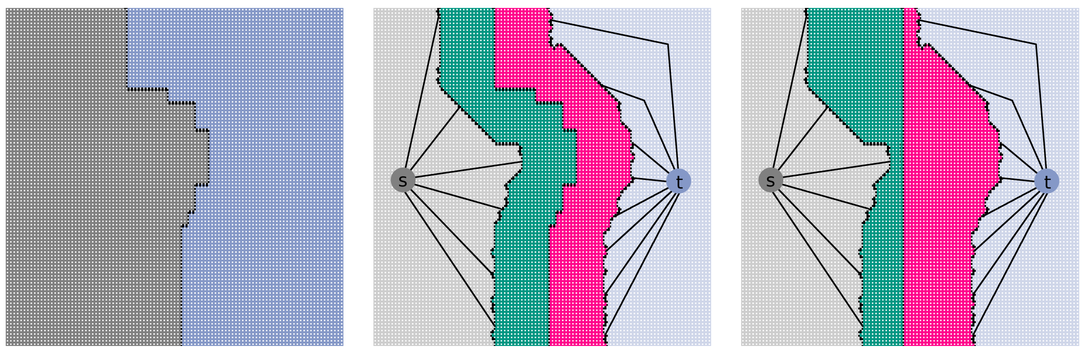
3.2 KaHIP – Karlsruhe High Quality Partitioning

KaHIP - **Ka**rlsruhe **Hi**gh Quality **P**artitioning - is our family of graph partitioning programs. It contains multilevel graph partitioning algorithms, a parallel evolutionary algorithm that provide effective combine and mutation operations, as well as algorithms that handle the case where blocks have to have almost equal size.



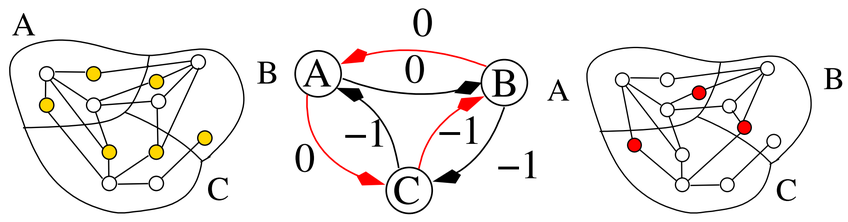
3.3 Example: Max-Flow Min-Cut based Local Search

The main idea of the algorithm is to define an a max-flow problem around the cut of a given partition. The area of the flow problem is defined in such a way that each cut in this area directly yields a feasible partition of the original problem. A standard algorithm is then used to find a max-flow for the network which yields in turn a minimum cut in the defined area.



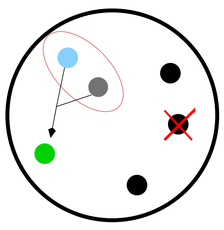
3.4 Example: Local Search for the Highly Balanced Case

In this case, state-of-the-art local search algorithms are restricted to pair-wise node exchanges. We introduced new techniques that relax the balance constraint for node movements but globally maintain balance by combining multiple local searches. The combination problem is reduced to finding negative cycles in a directed graph.



3.5 Example: Parallel Evolutionary Algorithm

Our framework modifies the multilevel scheme to provide new effective combine and mutation operations. This is combined with a scalable communication protocol to provide a coarse-grained parallelization. We are able to compute record setting solution quality in Walshaw's benchmark archive *within a few minutes* for graphs of moderate size. Previous methods required up to one week for graphs of that size.



Selected References

[1] Peter Sanders and Christian Schulz. Engineering Multilevel Graph Partitioning Algorithms. In *Proceedings of the 19th European Symposium on Algorithms (ESA'11)*, vol. 6942 of LNCS, pages 469–480. Springer, 2011.

[2] Peter Sanders and Christian Schulz. Distributed Evolutionary Graph Partitioning. In *Proceedings of the 12th Workshop on Algorithm Engineering and Experimentation (ALENEX'12)*, pages 16-19, 2012.

[3] Peter Sanders and Christian Schulz. Think Locally, Act Globally: Highly Balanced Graph Partitioning. In *Proceedings of the 12th Symposium on Experimental Algorithms (SEA'13)*, vol. 7933 of Lecture Notes in Computer Science, pages 164–175. Springer, 2013.

[4] KaHIP Homepage – `http://algo2.iti.kit.edu/documents/kahip/index.html` .