Theory and Practice on Sparse Graphs

Marcin Pilipczuk, Michał Pilipczuk and Sebastian Siebertz

Countless real life problems can be formulated as abstract combinatorial optimization problems, and many of them can be formulated naturally in terms of graphs. Graphs are fundamental mathematical structures which can be used to represent any kind of objects which are pairwise related. Formally, a graph consists of a set of vertices and a set of edges connecting the vertices. For instance, a telecommunication network may be modelled by a graph in which terminals are represented by vertices and transmission links between the terminals by edges of the graph. Graphs find important applications in many different areas. In computer science they are used to represent networks of communication or transportation, computational devices, flows of computation and many more. Graph theory is also applied to study molecules in chemistry and physics or in DNA representations in biology.

Once a graph theoretical formulation of a real world situation has been established, it is a challenging task to efficiently solve various optimization problems or to extract information from ever growing amounts of structured data. An example of such an optimization problem in a communication network is to place a minimum number of communication antennas to cover all important locations with wireless network. In its graph theoretic formulation the problem corresponds to the problem of finding a small set of vertices in the network graph which dominate all other vertices, that is, which are connected to all other vertices by an edge.

It has long been realized that a large class of important algorithmic graph problems seems to evade all attempts of solving them efficiently in general. However, on restricted graph classes, the complexity of a problem may be quite different from the general worst-case complexity. In particular, instances of graphs arising in applications often have more structure than general graphs. For instance, road or railway maps correspond to nearly planar graphs and telecommunication networks are modelled by sparse graphs, that is, by graphs with a moderate number of edges.

Researchers have studied many structural properties of graphs which can be used to design efficient algorithms for otherwise hard problems. Among the most important ones are properties of planar graphs or, much more generally, properties of graph classes that exclude a fixed minor. Very recent results indicate that a natural and very *general model of uniform sparseness* in graphs, called *bounded expansion*, leads to strong tractability results.

The goals of our proposal are to obtain new theoretical results in the rapidly growing field of algorithmic graph structure theory on graphs of bounded expansion as well as to provide empirical evaluations of theoretical algorithmic results on real world instances.

An example of a theoretical question we may consider is to *find a constant factor approximation algorithm for tree-depth*. The above described theory of sparse graphs uses tree like structures as building blocks. In this context, similarity to a tree is defined in terms of a measure called tree-depth. While every depth-first search of a graph of tree-depth k produces a tree-depth decomposition of depth at most 2^k, it is open whether one can compute better approximations in polynomial time. We conjecture that there exist even a constant factor approximation algorithm running in polynomial time and we propose to work on this subject together with a student. Some background in graph theory and graph algorithms may be of advantage.

An example of a practical project is the *implementation of an algorithm to solve the dominating set problem (and similar problems) on sparse graphs*. We have recently presented a polynomial time preprocessing procedure which, on an input graph G of bounded expansion which has a dominating set of size k, computes a subgraph G' of size at most $c \cdot k$ (where c is a constant depending only on the expansion properties and not on the size of G) such that every dominating set of G' is also a dominating set of G. Hence, after running the preprocessing procedure, a brute force algorithm may be able to solve the resulting dominating set instance for small values of k exactly. We want to implement and evaluate our algorithm on practical instances. Other algorithms for similar graph problems may be developed and implemented. Some deeper background in programming or algorithmic competitions (such as Topcoder, ICPC, Codeforces) may be of advantage.