My Publications until 2020

Frank Kammer

This is a survey of my publications arranged by topics.

Space-efficient Graph Algorithms

Sorting and Ranking of Self-Delimiting Numbers with Applications to Tree Isomorphism [31]

Assume that an N-bit sequence S of k self-delimiting numbers is given as input. We present space-efficient algorithms for sorting, dense ranking and (competitive) ranking S on the word RAM model with word size $\Omega(\log N)$ bits. Our algorithms run in $O(k+\frac{N}{\log N})$ time and use $O(N)$ bits. The sorting algorithm returns the given numbers in sorted order, stored within a bit-vector of N bits, whereas our ranking algorithms construct data structures that allow us subsequently to return the (dense) rank of each number x in S in constant time if the position of x in S is given together with x.

As an application of our algorithms above we give an algorithm for tree isomorphism, which runs in $O(n)$ time and uses $O(n)$ bits on n-node trees. The previous best linear-time algorithm for tree isomorphism uses $\Theta(n \log n)$ bits.

FPT-space Graph Kernelizations [30]

Let n be the size of a parametrized problem and k the parameter. We present polynomial-time kernelizations for Cluster Editing/Deletion, Path Contractions and Feedback Vertex Set that run with $O(\texttt{poly}(k) \log n)$ bits and compute a kernel of size polynomial in k. By first executing the new kernelizations and subsequently the best known polynomial-time kernelizations for the problem under consideration, we obtain the best known kernels in polynomial time with $O(\texttt{poly}(k) \log n)$ bits.

Our kernelization for Feedback Vertex Set computes in a first step an approximated solution, which can be used to build a simple algorithm for undirected s-t-connectivity (USTCON) that runs in polynomial time and with $O(\texttt{poly}(k) \log n)$ bits.

Space-Efficient Euler Partition and Bipartite Edge Coloring [14, 15]

We describe space-efficient algorithms for two problems on undirected multigraphs: Euler partition (partitioning the edges into a minimum number of trails); and bipartite edge coloring (coloring the edges of a bipartite multigraph with the minimum number of colors). Let n, m and $\Delta \geq 1$ be the numbers of vertices and of edges and the maximum degree, respectively, of the input multigraph. For Euler partition we reduce the amount of working memory needed by a logarithmic factor, from $\Theta((n+m)\log(n+m))$ bits to $O(n+m)$ bits, while preserving a running time of $O(n+m)$. For bipartite edge coloring, still using $O(n+m)$ bits of working memory, we achieve a running time of $O(n+m\min\{\Delta, \log\Delta \log^*\Delta(1+(\log m \log\Delta)/\Delta)\})$. This is $O(m \log\Delta \log^*\Delta)$ if $m =$

$\Omega(n \log n \log\log n)$, to be compared with $O(m \log \Delta)$ for the fastest known algorithm. Instrumental in obtaining the latter result is a new data structure presented here, the *subgraph stack*, that allows graph algorithms centered around recursive calls on substantially smaller subgraphs to be implemented in a space-efficient manner.

Space-Efficient Biconnected Components and Recognition of Outerplanar Graphs [23, 24]

We present space-efficient algorithms for computing cut vertices in a given graph with n vertices and m edges in linear time using $O(n + \min\{m, n \log\log n\})$ bits. With the same time and using $O(n + m)$ bits, we can compute the biconnected components of a graph. We use this result to show an algorithm for the recognition of (maximal) outerplanar graphs in $O(n \log\log n)$ time using $O(n)$ bits.

Linear-Time In-Place DFS and BFS in the Restore Model [29]

We present an in-place depth first search (DFS) and an in-place breadth first search (BFS) that runs on a word RAM in linear time such that, if the adjacency arrays of the input graph are given in a sorted order, the input is restored after running the algorithm. To obtain our results we use properties of the representation used to store the given graph and show several linear-time in-place graph transformations from one representation into another.

Simple 2^f-Color Choice Dictionaries [28]

A c-color choice dictionary of size $n \in \mathbb{N}$ is a fundamental data structure in the development of space-efficient algorithms that stores the colors of n elements and that supports operations to get and change the color of an element as well as an operation choice that returns an arbitrary element of that color. For an integer $f > 0$ and a constant $c = 2^f$, we present a word-RAM algorithm for a c-color choice dictionary of size n that supports all operations above in constant time and uses only $nf + 1$ bits, which is optimal if all operations have to run in $o(n/w)$ time where w is the word size.

In addition, we extend our choice dictionary by an operation union without using more space.

Extra Space during Initialization of Succinct Data Structures and Dynamical Initializable Arrays [27]

Many succinct data structures on the word RAM require precomputed tables to start operating. Usually, the tables can be constructed in sublinear time. In this time, most of a data structure is not initialized, i.e., there is plenty of unused space allocated for the data structure. We present a general framework to store temporarily extra buffers between the real data so that the data can be processed immediately, stored first in the buffers, and then moved into the real data structure after finishing the tables. As an application, we apply our framework to Dodis, Pătraşcu, and Thorup's data structure (STOC 2010) that emulates c-ary memory and to Farzan and Munro's succinct encoding of arbitrary graphs (TCS 2013). We also use our framework to present an in-place dynamical initializable array.

On-the-Fly Array Initialization in Less Space [13]

We show that for all given $n, t, w \in \{1, 2, \ldots\}$ with $n < 2^w$, an array of n entries of w bits each can be represented on a word RAM with a word length of w bits in at most $nw + \lceil n(t/(2w))^t \rceil$ bits of uninitialized memory to support constant-time initialization of the whole array and $O(t)$-time reading and writing of individual array entries. At one end of this tradeoff, we achieve initialization and access (i.e., reading and writing) in constant time with $nw + \lceil n/w^t \rceil$ bits for arbitrary fixed t, to be compared with $nw + \Theta(n)$ bits for the best previous solution, and at the opposite end, still with constant-time initialization, we support $O(\log n)$-time access with just $nw + 1$ bits, which is optimal for arbitrary access times if the initialization executes fewer than n steps.

Succinct Choice Dictionaries [12]

The *choice dictionary* is introduced as a data structure that can be initialized with a parameter $n \in \mathbb{N} = \{1, 2, \ldots\}$ and subsequently maintains an initially empty subset S of $\{1, \ldots, n\}$ under insertion, deletion, membership queries and an operation *choice* that returns an arbitrary element of S. The choice dictionary appears to be fundamental in space-efficient computing. We show that there is a choice dictionary that can be initialized with n and an additional parameter $t \in \mathbb{N}$ and subsequently occupies $n + O(n(t/w)^t + \log n)$ bits of memory and executes each of the four operations *insert*, *delete*, *contains* (i.e., a membership query) and *choice* in $O(t)$ time on a word RAM with a word length of $w = \Omega(\log n)$ bits. In particular, with $w = \Theta(\log n)$, we can support *insert*, *delete*, *contains* and *choice* in constant time using $n + O(n/(\log n)^t)$ bits for arbitrary fixed t. We extend our results to maintaining several pairwise disjoint subsets of $\{1, \ldots, n\}$.

A static representation of a subset S of $\{1, \ldots, n\}$ that consists of $n + s$ bits $b_1, \ldots, b_{n+s}$ is called *systematic* if $b_\ell = 1 \Leftrightarrow \ell \in S$ for $\ell = 1, \ldots, n$ and is said to have *redundancy* s. We extend the former definition to dynamic data structures and prove that the minimum redundancy of a systematic choice dictionary with parameter n that executes every operation in $O(t)$ time on a w-bit word RAM is $\Theta(n/(tw))$, provided that $tw = O(n/\log n)$. Allowing a redundancy of $\Theta(n \log(t \log n)/(t \log n) + n^\epsilon)$ for arbitrary fixed $\epsilon > 0$, we can support additional $O(t)$-time operations *p-rank* and *p-select* that realize a bijection from S to $\{1, \ldots, |S|\}$ and its inverse. The bijection may be chosen arbitrarily by the data structure, but must remain fixed as long as S is not changed. In particular, an element of S can be drawn uniformly at random in constant time with a redundancy of $O(n \log\log n/\log n)$.

We study additional space-efficient data structures for subsets S of $\{1, \ldots, n\}$, including one that supports only insertion and an operation *extract-choice* that returns and deletes an arbitrary element of S. All our main data structures can be initialized in constant time and support efficient iteration over the set S, and we can allow changes to S while an iteration over S is in progress. We use these abilities crucially in designing the most space-efficient algorithms known for solving a number of graph and other combinatorial problems in linear time. In particular, given an undirected graph G with n vertices and m edges, we can output a spanning forest of G in $O(n+m)$ time with at most $(1+\epsilon)n$ bits of working memory for arbitrary fixed $\epsilon > 0$, and if G is connected, we can output a shortest-path spanning tree of G rooted at a designated vertex in $O(n+m)$ time with $n \log_2 3 + O(n/(\log n)^t)$ bits of working memory for arbitrary fixed $t \in \mathbb{N}$.

Space-Efficient Plane-Sweep Algorithms [3]

We introduce space-efficient plane-sweep algorithms for basic planar geometric problems. It is assumed that the input is in a read-only array of n items and that the available workspace is $\Theta(s)$ bits, where $\lg n \leq s \leq n \cdot \lg n$. Three techniques that can be used as general tools in different space-efficient algorithms are introduced and employed within our algorithms. In particular, we give an almost-optimal algorithm for finding the closest pair among a set of n points that runs in $O(n^2/s + n \cdot \lg s)$ time. We also give a simple algorithm to enumerate the intersections of n line segments that runs in $O((n^2/s^{2/3}) \cdot \lg s + k)$ time, where k is the number of intersections. The counting version can be solved in $O((n^2/s^{2/3}) \cdot \lg s)$ time. When the segments are axis-parallel, we give an $O((n^2/s) \cdot \lg^{4/3} s + n^{4/3} \cdot \lg^{1/3} n)$-time algorithm for counting the intersections, and an algorithm for enumerating the intersections that runs in $O((n^2/s) \cdot \lg s \cdot \lg \lg s + n \cdot \lg s + k)$ time, where k is the number of intersections. We finally present an algorithm that runs in $O((n^2/s + n \cdot \lg s) \cdot \sqrt{(n/s) \cdot \lg n})$ time to calculate Klee's measure of axis-parallel rectangles.

Space-Efficient Basic Graph Algorithms [2]

We reconsider basic algorithmic graph problems in a setting where an n-vertex input graph is read-only and the computation must take place in a working memory of $O(n)$ bits or little more than that. For computing connected components and performing breadth-first search, we match the running times of standard algorithms that have no memory restrictions, for depth-first search and related problems we come within a factor of $\Theta(\log \log n)$, and for computing minimum spanning forests and single-source shortest-paths trees we come close for sparse input graphs.

Tree Decompositions, FPT and Kernelizations

FPT-space Graph Kernelizations [30]

Let n be the size of a parametrized problem and k the parameter. We present polynomial-time kernelizations for Cluster Editing/Deletion, Path Contractions and Feedback Vertex Set that run with $O(\texttt{poly}(k) \log n)$ bits and compute a kernel of size polynomial in k. By first executing the new kernelizations and subsequently the best known polynomial-time kernelizations for the problem under consideration, we obtain the best known kernels in polynomial time with $O(\texttt{poly}(k) \log n)$ bits.

Our kernelization for Feedback Vertex Set computes in a first step an approximated solution, which can be used to build a simple algorithm for undirected s-t-connectivity (USTCON) that runs in polynomial time and with $O(\texttt{poly}(k) \log n)$ bits.

Space-Efficient Vertex Separators for Treewidth [26]

For n-vertex graphs with treewidth $k = O(n^{1/2-\epsilon})$ and an arbitrary $\epsilon > 0$, we present a word-RAM algorithm to compute vertex separators using only $O(n)$ bits of working memory. As an application of our algorithm, we give an $O(1)$-approximation algorithm for tree decomposition. Our algorithm computes a tree decomposition in $c^k n(\log\log n)\log^* n$ time using $O(n)$ bits for some constant $c > 0$.

We finally use the tree decomposition obtained by our algorithm to solve Vertex Cover, Independent Set, Dominating Set, MaxCut and q-Coloring by using $O(n)$ bits as long as the treewidth of the graph is smaller than $c' \log n$ for some problem dependent constant $0 < c' < 1$.

Multistage Problems on a Global Budget [16]

Time-evolving or temporal graphs gain more and more popularity when studying the behavior of complex networks. In this context, the multistage view on computational problems is among the most natural frameworks. Roughly speaking, herein one studies the different (time) layers of a temporal graph (effectively meaning that the edge set may change over time, but the vertex set remains unchanged), and one searches for a solution of a given graph problem for each layer. The twist in the multistage setting is that the solutions found must not differ too much between subsequent layers. We relax on this already established notion by introducing a global instead of the local budget view studied so far. More specifically, we allow for few disruptive changes between subsequent layers but request that overall, that is, summing over all layers, the degree of change is moderate. Studying several classical graph problems (both NP-hard and polynomial-time solvable ones) from a parameterized complexity angle, we encounter both fixed-parameter tractability and parameterized hardness results. Somewhat surprisingly, we find that sometimes the global multistage versions of NP-hard problems such as Vertex Cover turn out to be computationally more tractable than the ones of polynomial-time solvable problems such as Matching.

Approximate Tree Decompositions of Planar Graphs in Linear Time [35, 38]

Many algorithms have been developed for NP-hard problems on graphs with small treewidth k. For example, all problems that are expressible in linear extended monadic second order can be solved in linear time on graphs of bounded treewidth. It turns out that the bottleneck of many algorithms for NP-hard problems is the computation of a tree decomposition of width $O(k)$. In particular, by the bidimensional theory, there are many linear extended monadic second order problems that can be solved on n-vertex planar graphs with treewidth k in a time linear in n and subexponential in k if a tree decomposition of width $O(k)$ can be found in such a time.

We present the first algorithm that, on n-vertex planar graphs with treewidth k, finds a tree decomposition of width $O(k)$ in such a time. In more detail, our algorithm has a running time of $O(nk^2 \log k)$. We show the result as a special case of a result concerning so-called weighted treewidth of weighted graphs.

Linear-Time Kernelization for the Rooted k-Leaf Outbranching Problem [21, 22]

In the Rooted k-Leaf Outbranching Problem, a digraph $G = (V, E)$, a vertex r of G, and an integer k are given, and the goal is to find an r-rooted spanning outtree of G with $\geq k$ leaves (a subtree of G with vertex set V, all edges directed away from r, and $\geq k$ leaves). We present a linear-time algorithm that computes a problem kernel with $O(k^6)$ vertices and $O(k^7)$ edges for the Rooted k-Leaf Outbranching Problem. By combining the new result with a result of Daligault and Thomassé [IWPEC 2009], a kernel with a quadratic number of vertices and edges can be found on n-vertex m-edge digraphs in time $O(n + m + k^{14})$.

The Complexity of Minimum Convex Coloring [17, 36]

A coloring of the vertices of a graph is called convex if each subgraph induced by all vertices of the same color is connected. We consider three variants of recoloring a colored graph with minimal cost such that the resulting coloring is convex. Two variants of the problem are shown to be $\mathcal{NP}$-hard on trees even if in the initial coloring each color is used to color only a bounded number of vertices. For graphs of bounded treewidth, we present a polynomial-time $(2+\epsilon)$-approximation algorithm for these two variants and a polynomial-time algorithm for the third variant. Our results also show that, unless $\mathcal{NP} \subseteq DTIME(n^{O(\log \log n)})$, there is no polynomial-time approximation algorithm with a ratio of size $(1-o(1)) \ln \ln N$ for the following problem: Given pairs of vertices in an undirected N-vertex graph of bounded treewidth, determine the minimal possible number l for which all except l pairs can be connected by disjoint paths.

Treelike and Chordal Graphs: Algorithms and Generalizations [20]

We present algorithms to compute tree decompositions on planar and chordal graphs. Using these algorithms and other algorithms to find a tree decomposition we solve the k-disjoint paths problem on chordal graphs and minimum convex coloring in graphs of small treewitdh. Finally, we show several algorithms on intersection graphs.

Linear-Time Computation of a Linear Problem Kernel for Dominating Set on Planar Graphs [1]

We present a linear-time data reduction algorithm that transforms a given planar graph G with domination number $\gamma(G)$ into a planar graph G' of size $O(\gamma(G))$ with $\gamma(G) = \gamma(G')$. In addition, a minimum dominating set for G can be inferred from a minimum dominating set for G'. In terms of parameterized algorithmics, this implies a linear-size problem kernel for the NP-hard Dominating Set problem on planar graphs, where the kernelization takes linear time. This improves on previous kernelization algorithms that provide linear-size kernels in cubic time.

The k-Disjoint Paths Problem on Chordal Graphs [33]

Algorithms based on a bottom-up traversal of a tree decomposition are used in literature to develop very efficient algorithms for graphs of bounded treewidth. However, such algorithms can also be used to efficiently solve problems on chordal graphs, which

in general do not have a bounded treewidth. By combining this approach with a sparsification technique we obtain the first linear-time algorithm for chordal graphs that solves the k-disjoint paths problem. In this problem k pairs of vertices are to be connected by pairwise vertex-disjoint paths. We also present the first polynomial-time algorithm for chordal graphs capable of finding disjoint paths solving the k-disjoint paths problem with minimal total length. Finally we prove that the version of the disjoint paths problem, where k is part of the input, is NP-hard on chordal graphs.

Determining the smallest k such that G is k-outerplanar [19]

The outerplanarity index of a planar graph G is the smallest k such that G has a k-outerplanar embedding. We show how to compute the outerplanarity index of an n-vertex planar graph in $O(n^2)$ time, improving the previous best bound of $O(k^3n^2)$. Using simple variations of the computation we can determine the radius of a planar graph in $O(n^2)$ time and its width in $O(n^3)$ time.

We also give a linear-time 4-approximation algorithm for the outerplanarity index and show how it can be used to solve maximum independent set and several other NP-hard problems faster on planar graphs with outerplanarity index within a constant factor of their treewidth.

Connectivity [32]

We are mainly concerned with the strength of connections between vertices with respect to the number of vertex- or edge-disjoint paths. As we shall see, this is equivalent to the question of how many nodes or edges must be removed from a graph to destroy all paths between two (arbitrary or specified) vertices.

We present algorithms which check k-vertex (k-edge) connectivity, compute the vertex (edge) connectivity, and compute the maximal k-connected components of a given graph.

A Lower Bound for the Treewidth of k-Outerplanar Graphs [34]

Many optimization problems can be solved efficiently if a tree-decomposition of small width is given. Unfortunately, all known algorithms computing, for general graphs, a tree decomposition of width k, if one exists, have a running time exponential in k. However, Bodlaender observed that each k-outerplanar graph has a tree decomposition of width at most $3k-1$ and his analysis implicitly leads to an $O(kn)$-time algorithm for computing such a tree-decomposition. In this paper we show that the bound $3k-1$ is tight, i.e., for every $k \in I\!N$, there are k-outerplanar graphs having treewidth $3k-1$.

Temporal Graphs

Two Moves per Time Step Make a Difference [9]

A temporal graph is a graph whose edge set can change over time. We only require that the edge set in each time step forms a connected graph. The temporal exploration problem asks for a temporal walk that starts at a given vertex, moves over at most one

edge in each time step, visits all vertices, and reaches the last unvisited vertex as early as possible. We show in this paper that every temporal graph with n vertices can be explored in $O(n^{1.75+\varepsilon})$ time steps for arbitrary $\varepsilon > 0$ provided that either the degree of the graph is bounded in each step or the temporal walk is allowed to make two moves per step. This result is interesting because it breaks the lower bound of $\Omega(n^2)$ steps that holds for the worst-case exploration time if only one move per time step is allowed and the graph in each step can have arbitrary degree. We complement this main result by a logarithmic inapproximability result and a proof that for sparse temporal graphs (i.e., temporal graphs with $O(n)$ edges in the underlying graph) making $O(1)$ moves per time step can improve the worst-case exploration time at most by a constant factor.

Multistage Problems on a Global Budget [16]

Time-evolving or temporal graphs gain more and more popularity when studying the behavior of complex networks. In this context, the multistage view on computational problems is among the most natural frameworks. Roughly speaking, herein one studies the different (time) layers of a temporal graph (effectively meaning that the edge set may change over time, but the vertex set remains unchanged), and one searches for a solution of a given graph problem for each layer. The twist in the multistage setting is that the solutions found must not differ too much between subsequent layers. We relax on this already established notion by introducing a global instead of the local budget view studied so far. More specifically, we allow for few disruptive changes between subsequent layers but request that overall, that is, summing over all layers, the degree of change is moderate. Studying several classical graph problems (both NP-hard and polynomial-time solvable ones) from a parameterized complexity angle, we encounter both fixed-parameter tractability and parameterized hardness results. Somewhat surprisingly, we find that sometimes the global multistage versions of NP-hard problems such as Vertex Cover turn out to be computationally more tractable than the ones of polynomial-time solvable problems such as Matching.

On Temporal Graph Exploration [7]

A temporal graph is a graph in which the edge set can change from step to step. The temporal graph exploration problem TEXP is the problem of computing a foremost exploration schedule for a temporal graph, i.e., a temporal walk that starts at a given start node, visits all nodes of the graph, and has the smallest arrival time. We consider only temporal graphs that are connected at each step. For such temporal graphs with n nodes, we show that it is **NP**-hard to approximate TEXP with ratio $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$. We also provide an explicit construction of temporal graphs that require $\Theta(n^2)$ steps to be explored. We then consider TEXP under the assumption that the underlying graph (i.e. the graph that contains all edges that are present in the temporal graph in at least one step) belongs to a specific class of graphs. Among other results, we show that temporal graphs can be explored in $O(n^{1.5}k^2 \log n)$ steps if the underlying graph has treewidth k and in $O(n \log^3 n)$ steps if the underlying graph is a $2 \times n$ grid. We also show that sparse temporal graphs with regularly present edges can always be explored in $O(n)$ steps.

Online- and Approximationalgorithms

Query-Competitive Algorithms for Cheapest Set Problems under Uncertainty [6, 8]

Considering the model of computing under uncertainty where element weights are uncertain but can be obtained at a cost by query operations, we study the problem of identifying a cheapest (minimum-weight) set among a given collection of feasible sets using a minimum number of queries of element weights. For the general case we present an algorithm that makes at most $d \cdot OPT + d$ queries, where d is the maximum cardinality of any given set and OPT is the optimal number of queries needed to identify a cheapest set. For the minimum multi-cut problem in trees with d terminal pairs, we give an algorithm that makes at most $d \cdot OPT + 1$ queries. For the problem of computing a minimum-weight base of a given matroid, we give an algorithm that makes at most $2 \cdot OPT$ queries, generalizing a known result for the minimum spanning tree problem. For each of the above algorithms we give matching lower bounds. We also settle the complexity of the verification version of the general cheapest set problem and the minimum multi-cut problem in trees under uncertainty.

Approximation Algorithms for Intersection Graphs [37, 39]

We study three complexity parameters called *k-perfectly groupable*, *k-simplicial*, and *k-perfectly orientable*. These parameters measure in some sense how chordal-like a graph is. The similarity to chordal graphs is used to construct simple polynomial-time approximation algorithms with constant approximation ratio for many NP-hard problems, when restricted to graphs for which at least one of the three complexity parameters is bounded by a constant. As applications we present approximation algorithms with constant approximation ratio for maximum weighted independent set, minimum (independent) dominating set, minimum vertex coloring, maximum weighted clique, and minimum clique partition for large classes of intersection graphs.

Practical Approaches to Partially Guarding a Polyhedral Terrain [25]

We study the problem of placing guard towers on a terrain such that the terrain can be seen from at least one tower. This problem is important in many applications, and has an extensive history in the literature (known as, e.g. multiple observer siting). In this paper, we consider the problem on polyhedral terrains, and we allow the guards to see only a fixed fraction of the terrain, rather than everything. We experimentally evaluate how the number of required guards relates to the fraction of the terrain that can be covered. In addition, we introduce the concept of dominated guards, which can be used to preprocess the potential guard locations and speed up the subsequent computations.

Removing local extrema from imprecise terrains [10, 11]

In this paper we consider imprecise terrains, that is, triangulated terrains with a vertical error interval in the vertices. In particular, we study the problem of *removing* as many local extrema (minima and maxima) as possible from the terrain; that is, finding an assignment of one height to each vertex, within its error interval, so that the resulting

terrain has minimum number of local extrema. We show that removing only minima or only maxima can be done optimally in $O(n \log n)$ time, for a terrain with n vertices. Interestingly, however, the problem of finding a height assignment that minimizes the total number of local extrema (minima as well as maxima) is NP-hard, and is even hard to approximate within a factor of $O(\log \log n)$ unless P = NP. Moreover, we show that even a simplified version of the problem where we can have only three different types of intervals for the vertices is already NP-hard, a result we obtain by proving hardness of a special case of 2-Disjoint Connected Subgraphs, a problem that has lately received considerable attention from the graph-algorithms community.

Maximising Lifetime for Fault-Tolerant Target Coverage in Sensor Networks [4, 5]

We study the problem of maximising the lifetime of a sensor network for fault-tolerant target coverage in a setting with composite events. Here, a composite event is the simultaneous occurrence of a combination of atomic events, such as the detection of smoke and high temperature. We are given sensor nodes that have an initial battery level and can monitor certain event types, and a set of points at which composite events need to be detected. The points and sensor nodes are located in the Euclidean plane, and all nodes have the same sensing radius. The goal is to compute a longest activity schedule with the property that at any point in time, each event point is monitored by at least two active sensor nodes. We present a $(6 + \varepsilon)$-approximation algorithm for this problem by devising an approximation algorithm with the same ratio for the dual problem of minimising the weight of a fault-tolerant sensor cover. The algorithm generalises previous approximation algorithms for geometric set cover with weighted unit disks and is obtained by enumerating properties of the optimal solution that guide a dynamic programming approach.

Graph Drawing

Simultaneous Embedding of Two Planar Graphs with Two Bends per Edge [18]

The simultaneous embedding problem is, given two planar graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$, to find planar embeddings $\varphi(G_1)$ and $\varphi(G_2)$ such that each vertex $v \in V$ is mapped to the same point in $\varphi(G_1)$ and in $\varphi(G_2)$. This article presents a linear-time algorithm for the simultaneous embedding problem such that edges are drawn as polygonal chains with at most two bends and all vertices and all bends of the edges are placed on a grid of polynomial size. An extension of this problem with so-called fixed edges is also considered.

A further linear-time algorithm of this article solves the point-set-embedding problem: Given a planar graph G and a set of distinct points, find a planar embedding for G that maps each vertex to one of the given points. The solution presented also uses at most two bends per edge and a grid whose size is polynomial in the size of the grid that includes all given points.

An example shows two bends per edge to be optimal for both problems.

References

[1] René van Bevern, Sepp Hartung, Frank Kammer, Rolf Niedermeier, and Mathias Weller. Linear-time computation of a linear problem kernel for dominating set on planar graphs. In *Proc 6th International Symposium on Parameterized and Exact Computation (IPEC 2011)*, volume 7112 of *LMCS*, pages 194–206. Springer, 2011. doi:10.1007/978-3-642-28050-4_16.

[2] Amr Elmasry, Torben Hagerup, and Frank Kammer. Space-efficient basic graph algorithms. In *Proc. 32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30 of *LIPIcs*, pages 288–301. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.288.

[3] Amr Elmasry and Frank Kammer. Space-efficient plane-sweep algorithms. In *Proc. 27th International Symposium on Algorithms and Computation (ISAAC 2016)*, volume 64 of *LIPIcs*, pages 30:1–30:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ISAAC.2016.30.

[4] Thomas Erlebach, Tom Grant, and Frank Kammer. Maximising lifetime for fault-tolerant target coverage in sensor networks. *Sustain. Computing: Inform. Systems*, 1(3):213–225, 2011. doi:10.1016/j.suscom.2011.05.005.

[5] Thomas Erlebach, Tom Grant, and Frank Kammer. Maximising lifetime for fault-tolerant target coverage in sensor networks. In *Proc. 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2011)*, pages 187–196. ACM, 2011. doi:10.1145/1989493.1989521.

[6] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. In *Proc. 39th International Symposium on Mathematical Foundations of Computer Science 2014 (MFCS 2014), Part II*, volume 8635 of *LNCS*, pages 263–274. Springer, 2014. doi:10.1007/978-3-662-44465-8_23.

[7] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. In *Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP 2015), Part I*, volume 9134 of *LNCS*, pages 444–455. Springer, 2015. doi:10.1007/978-3-662-47672-7_36.

[8] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. *Theor. Comput. Sci.*, 613:51–64, 2016. doi:10.1016/j.tcs.2015.11.025.

[9] Thomas Erlebach, Frank Kammer, Kelin Luo, Andrej Sajenko, and Jakob T. Spooner. Two moves per time step make a difference. In *Proc. 46th International Colloquium on Automata, Languages, and Programming, (ICALP 2019)*, volume 132 of *LIPIcs*, pages 141:1–141:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.141.

[10] Chris Gray, Frank Kammer, Maarten Löffler, and Rodrigo I. Silveira. Removing local extrema from imprecise terrains. *Comput. Geom.*, 45(7):334–349, 2012. doi:10.1016/j.comgeo.2012.02.002.

[11] Chris Gray, Frank Kammer, Maarten Löffler, and Rodrigo I. Silveira. Removing local extrema from imprecise terrains, Proc. 26th European Workshop on Computational Geometry (EuroCG 2010), 2010. arXiv:1002.2580.

[12] Torben Hagerup and Frank Kammer. Succinct choice dictionaries, 2016. arXiv:1604.06058.

[13] Torben Hagerup and Frank Kammer. On-the-fly array initialization in less space. In *Proc. 28th International Symposium on Algorithms and Computation, ISAAC 2017*, volume 92 of *LIPIcs*, pages 44:1–44:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ISAAC.2017.44.

[14] Torben Hagerup, Frank Kammer, and Moritz Laudahn. Space-efficient Euler partition and bipartite edge coloring. In *Proc. 10th International Conference on Algorithms and Complexity (CIAC 2017)*, LNCS. Springer, 2017. Accepted for publication.

[15] Torben Hagerup, Frank Kammer, and Moritz Laudahn. Space-efficient Euler partition and bipartite edge coloring. *Theor. Comput. Sci.*, 754:16–34, 2019. doi:10.1016/j.tcs.2018.01.008.

[16] Klaus Heeger, Anne-Sophie Himmel, Frank Kammer, Rolf Niedermeier, Malte Renken, and Andrej Sajenko. Multistage problems on a global budget. *CoRR*, abs/1912.04392, 2019. arXiv:1912.04392.

[17] F. Kammer and T. Tholey. The complexity of minimum convex coloring. In *Proc. 19th Annual International Symposium on Algorithms and Computation (ISAAC 2008)*, volume 5369 of *LNCS*, pages 16–27. Springer, 2008. doi:10.1007/978-3-540-92182-0_5.

[18] Frank Kammer. Simultaneous embedding with two bends per edge in polynomial area. In *Proc. 10th Scandinavian Workshop on Algorithm Theory (SWAT 2006)*, volume 4059 of *LNCS*, pages 255–267. Springer, 2006. doi:10.1007/11785293_25.

[19] Frank Kammer. Determining the smallest k such that G is k-outerplanar. In *Proc. 15th Annual European Symposium on Algorithms (ESA 2007)*, volume 4698 of *LNCS*, pages 359–370. Springer, 2007. doi:10.1007/978-3-540-75520-3_33.

[20] Frank Kammer. *Treelike and Chordal Graphs: Algorithms and Generalizations*. PhD thesis, University of Augsburg, 2012.

[21] Frank Kammer. A linear-time kernelization for the rooted k-leaf outbranching problem. In *Proc. 39th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2013)*, volume 8165 of *LNCS*, pages 310–320. Springer, 2013. doi:10.1007/978-3-642-45043-3_27.

[22] Frank Kammer. A linear-time kernelization for the rooted k-leaf outbranching problem. *Discret. Appl. Math.*, 193:126–138, 2015. doi:10.1016/j.dam.2015.04.028.

[23] Frank Kammer, Dieter Kratsch, and Moritz Laudahn. Space-efficient biconnected components and recognition of outerplanar graphs. In *Proc. 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*,

volume 58 of *LIPIcs*, pages 56:1–56:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. Full version:. arXiv:1606.04679.

[24] Frank Kammer, Dieter Kratsch, and Moritz Laudahn. Space-efficient biconnected components and recognition of outerplanar graphs. *Algorithmica*, 81(3):1180–1204, 2019. doi:10.1007/s00453-018-0464-z.

[25] Frank Kammer, Maarten Löffler, Paul Mutser, and Frank Staals. Practical approaches to partially guarding a polyhedral terrain. In *Proc. 8th International Conference on Geographic Information Science (GIScience 2014)*, volume 8728 of *LNCS*, pages 318–332. Springer, 2014. doi:10.1007/978-3-319-11593-1_21.

[26] Frank Kammer, Johannes Meintrup, and Andrej Sajenko. Space-efficient vertex separators for treewidth. *CoRR*, abs/1907.00676, 2019. arXiv:1907.00676.

[27] Frank Kammer and Andrej Sajenko. Extra space during initialization of succinct data structures and dynamical initializable arrays. In *Proc. 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018)*, volume 117 of *LIPIcs*, pages 65:1–65:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.MFCS.2018.65.

[28] Frank Kammer and Andrej Sajenko. Simple 2^f-color choice dictionaries. In *Proc. 29th International Symposium on Algorithms and Computation (ISAAC 2018)*, volume 123 of *LIPIcs*, pages 66:1–66:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ISAAC.2018.66.

[29] Frank Kammer and Andrej Sajenko. Linear-time in-place DFS and BFS on the word RAM. In *Proc. 11th International Conference on Algorithms and Complexity (CIAC 2019)*, volume 11485 of *LNCS*, pages 286–298. Springer, 2019. doi:10.1007/978-3-030-17402-6_24.

[30] Frank Kammer and Andrej Sajenko. FPT-space graph kernalizations. *CoRR*, 2020. arXiv:2007.11643.

[31] Frank Kammer and Andrej Sajenko. Sorting and ranking of self-delimiting numbers with applications to tree isomorphism. *CoRR*, 2020. arXiv:2002.07287.

[32] Frank Kammer and Hanjo Täubig. Connectivity. In *Network Analysis: Methodological Foundations*, volume 3418 of *LNCS*, pages 143–177. Springer, 2004. doi:10.1007/b106453.

[33] Frank Kammer and Torsten Tholey. The k-disjoint paths problem on chordal graphs. In *Proc. 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2009)*, volume 5911 of *LNCS*, pages 190–201. Springer, 2009. doi:10.1007/978-3-642-11409-0_17.

[34] Frank Kammer and Torsten Tholey. A lower bound for the treewidth of k-outerplanar graphs. Technical report, Universität Augsburg, 2009.

[35] Frank Kammer and Torsten Tholey. Approximate tree decompositions of planar graphs in linear time. In *Proc. 23th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pages 683–698. SIAM, 2012.

[36] Frank Kammer and Torsten Tholey. The complexity of minimum convex coloring. *Discrete Applied Mathematics*, 160(6):810–833, 2012. doi:10.1016/j.dam.2011.09.022.

[37] Frank Kammer and Torsten Tholey. Approximation algorithms for intersection graphs. *Algorithmica*, 68(2):312–336, 2014. doi:10.1007/s00453-012-9671-1.

[38] Frank Kammer and Torsten Tholey. Approximate tree decompositions of planar graphs in linear time. *Theor. Comput. Sci.*, 645:60–90, 2016. doi:10.1016/j.tcs.2016.06.040.

[39] Frank Kammer, Torsten Tholey, and Heiko Voepel. Approximation algorithms for intersection graphs. In *Proc. 13th International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX 2010)*, volume 6302 of *LNCS*, pages 260–273. Springer, 2010. doi:10.1007/978-3-642-15369-3_20.