K

k-Best Enumeration

David Eppstein
Donald Bren School of Information and Computer Sciences, Computer Science Department, University of California, Irvine, CA, USA

Keywords

k minimum-weight matchings; k shortest paths; k shortest simple paths; k smallest spanning trees

Years and Authors of Summarized Original Work

1959; Hoffman, Pavley
1963; Clarke, Krikorian, Rausen
1968; Murty
1971; Yen
1972; Lawler
1977; Gabow
1982; Katoh, Ibaraki, Mine
1985; Hamacher, Queyranne
1987; Chegireddy, Hamacher
1993; Frederickson
1997; Eppstein, Galil, Italiano, Nissenzweig
1998; Eppstein
2007; Hershberger, Maxel, Suri
2013; Chen, Kanj, Meng, Xia, Zhang

Problem Definition

K-best enumeration problems are a type of combinatorial enumeration in which, rather than seeking a single best solution, the goal is to find a set of k solutions (for a given parameter value k) that are better than all other possible solutions. Many of these problems involve finding structures in a graph that can be represented by subsets of the graph's edges. In particular, the k shortest paths between two vertices s and t in a weighted network are a set of k distinct paths that are shorter than all other paths, and other problems such as the k smallest spanning trees of a graph or the k minimum weight matchings in a graph are defined in the same way.

Key Results

One of the earliest works in the area of k-best optimization was by Hoffman and Pavley [10] formulating the k-shortest path problem; their paper cites unpublished work by Bock, Kantner, and Hayes on the same problem. Later research by Lawler [12], Gabow [7], and Hamacher and Queyranne [8] described a general approach to k-best optimization, suitable for many of these problems, involving the hierarchical partitioning of the solution space into subproblems. One way

This material is based upon work supported by the National Science Foundation under Grant CCF-1228639 and by the Office of Naval Research under Grant No. N00014-08-1-1015.

M.-Y. Kao (ed.), *Encyclopedia of Algorithms*,
DOI 10.1007/978-1-4939-2864-4

of doing this is to view the optimal solution to a problem as a sequence of edges, and define one subproblem for each edge, consisting of the solutions that first deviate from the optimal solution at that edge. Continuing this subdivision recursively leads to a tree of subproblems, each having a worse solution value than its parent, such that each possible solution is the best solution for exactly one subproblem in the hierarchy. A best-first search of this tree allows the k-best solutions to be found. Alternatively, if both the first and second best solutions can be found, and differ from each other at an edge e, then one can form only two subproblems, one consisting of the solutions that include e and one consisting of the solutions that exclude e. Again, the subdivision continues recursively; each solution (except the global optimum) is the second-best solution for exactly one subproblem, allowing a best-first tree search to find the k-best solutions. An algorithm of Frederickson [6] solves this tree search problem in a number of steps proportional to k times the degree of the tree; each step involves finding the solution (or second-best solution) to a single subproblem.

Probably the most important and heavily studied of the k-best optimization problems is the problem of finding k shortest paths, first formulated by Hoffman and Pavley [10]. In the most basic version of this problem, the paths are allowed to have repeated vertices or edges (unless the input is acyclic, in which case repetitions are impossible). An algorithm of Eppstein [4] solves this version of the problem in the optimal time bound $O(m + n \log n + k)$, where m and n are the numbers of edges and vertices in the given graph; that is, after a preprocessing stage that is dominated by the time to use Dijkstra's algorithm to find a single shortest-path tree, the algorithm takes constant time per path. Eppstein's algorithm follows Hoffman and Pavley in representing a path by its sequence of *deviations*, the edges that do not belong to a tree T of shortest paths to the destination node. The deviation edges that can be reached by a path in T from a given node v are represented as a binary heap (ordered by how much additional length the deviation would cause) and these heaps are used to define a partition of the solution space into subproblems, consisting of the paths that follow a certain sequence of deviations followed by one more deviation from a specified heap. The best path in a subproblem is the one that chooses the deviation at the root of its heap, and the remaining paths can be partitioned into three sub-subproblems, two for the children of the root and one for the paths that use the root deviation but then continue with additional deviations. In this way, Eppstein constructs a tree of subproblems to which Frederickson's tree-searching method can be applied.

In a graph with cycles (or in an undirected graph which, when its edges are converted to directed edges, has many cycles), it is generally preferable to list only the k shortest simple (or loopless) paths, not allowing repetitions within a path. This variation of the k shortest paths problem was formulated by Clarke et al. [3]. Yen's algorithm [14] still remains the one with the best asymptotic time performance, $O(kn(m + n \log n))$; it is based on best-solution partitioning using Dijkstra's algorithm to find the best solution in each subproblem. A more recent algorithm of Hershberger et al. [9] is often faster, but is based on a heuristic that can sometimes fail, causing it to become no faster than Yen's algorithm. In the undirected case, it is possible to find the k shortest simple paths in time $O(k(m + n \log n))$ [11].

Gabow [7] introduced both the problem of finding the k minimum-weight spanning trees of an edge-weighted graph, and the technique of finding a binary hierarchical subdivision of the space of solutions, which he used to solve the problem. In any graph, the best and second-best spanning trees differ only by one edge swap (the removal of one edge from a tree and its replacement by a different edge that reconnects the two subtrees formed by the removal), a property that simplifies the search for a second-best tree as needed for Gabow's partitioning technique. The fastest known algorithms for the k-best spanning trees problem are based on Gabow's partitioning technique, together with dynamic graph data structures that keep track of the best swap in a network as that network undergoes a sequence of edge insertion and deletion operations. To use this technique, one initializes a fully-persistent best-swap data structure (one in

which each update creates a new version of the structure without modifying the existing versions, and in which updates may be applied to any version) and associates its initial version with the root of the subproblem tree. Then, whenever an algorithm for selecting the k best nodes of the subproblem tree generates a new node (a subproblem formed by including or excluding an edge from the allowed solutions) the parent node's version of the data structure is updated (by either increasing or decreasing the weight of the edge to force it to be included or excluded in all solutions) and the updated version of the data structure is associated with the child node. In this way, the data structure can be used to quickly find the second-best solution for each of the subproblems explored by the algorithm. Based on this method, the k-best spanning trees of a graph with n vertices and m edges can be found (in an implicit representation based on sequences of swaps rather than explicitly listing all edges in each tree) in time $O(\mathrm{MST}(m,n) + k \min(n,k)^{1/2})$ where $\mathrm{MST}(m,n)$ denotes the time for finding a single minimum spanning tree (linear time, if randomized algorithms are considered) [5].

After paths and spanning trees, probably the next most commonly studied k-best enumeration problem concerns matchings. The problem of finding the k minimum-weight perfect matchings in an edge-weighted graph was introduced by Murty [13]. A later algorithm by Chegireddy and Hamacher [1] solves the problem in time $O(kn^3)$ (where n is the number of vertices in the graph) using the technique of building a binary partition of the solution space. Other problems whose k-best solutions have been studied include the Chinese postman problem, the traveling salesman problem, spanning arborescences in a directed network, the matroid intersection problem, binary search trees and Huffman coding, chess strategies, integer flows, and network cuts.

For many NP-hard optimization problems, where even finding a single best solution is difficult, an approach that has proven very successful is *parameterized complexity*, in which one finds an integer parameter describing the input instance or its solution that is often much smaller than the input size, and designs algorithms whose running time is a fixed polynomial of the input size multiplied by a non-polynomial function of the parameter value. Chen et al. [2] extend this paradigm to k-best problems, showing that, for instance, many NP-hard k-best problems can be solved in polynomial time per solution for graphs of bounded treewidth.

Applications

The k shortest path problem has many applications. The most obvious of these are in the generation of alternative routes, in problems involving communication networks, transportation networks, or building evacuation planning. In bioinformatics, it has been applied to shortest-path formulations of dynamic programming algorithms for biological sequence alignment and also applied in the reconstruction of metabolic pathways, and reconstruction of gene regulation networks. The problem has been used frequently in natural language and speech processing, where a path in a network may represent a hypothesis for the correct decoding of an utterance or piece of writing. Other applications include motion tracking, genealogy, the design of power, communications, and transportation networks, timing analysis of circuits, and task scheduling.

The problem of finding the k-best spanning trees has been applied to point process intensity estimation, the analysis of metabolic pathways, image segmentation and classification, the reconstruction of pedigrees from genetic data, the parsing of natural-language text, and the analysis of electronic circuits.

Cross-References

▸ Minimum Spanning Trees
▸ Single-Source Shortest Paths

Recommended Reading

1. Chegireddy CR, Hamacher HW (1987) Algorithms for finding K-best perfect matchings. Discret Appl Math 18(2):155–165. doi:10.1016/0166-218X(87)90017-5

2. Chen J, Kanj IA, Meng J, Xia G, Zhang F (2013) Parameterized top-K algorithms. Theor Comput Sci 470:105–119. doi:10.1016/j.tcs.2012.10.052
3. Clarke S, Krikorian A, Rausen J (1963) Computing the N best loopless paths in a network. J SIAM 11:1096–1102
4. Eppstein D (1998) Finding the k shortest paths. SIAM J Comput 28(2):652–673. doi:10.1137/S0097539795290477
5. Eppstein D, Galil Z, Italiano GF, Nissenzweig A (1997) Sparsification–a technique for speeding up dynamic graph algorithms. J ACM 44(5):669–696. doi:10.1145/265910.265914
6. Frederickson GN (1993) An optimal algorithm for selection in a min-heap. Inf Comput 104(2):197–214. doi:10.1006/inco.1993.1030
7. Gabow HN (1977) Two algorithms for generating weighted spanning trees in order. SIAM J Comput 6(1):139–150. doi:10.1137/0206011
8. Hamacher HW, Queyranne M (1985) K best solutions to combinatorial optimization problems. Ann Oper Res 4(1–4):123–143. doi:10.1007/BF02022039
9. Hershberger J, Maxel M, Suri S (2007) Finding the k shortest simple paths: a new algorithm and its implementation. ACM Trans Algorithms 3(4):A45. doi:10.1145/1290672.1290682
10. Hoffman W, Pavley R (1959) A method for the solution of the Nth best path problem. J ACM 6(4):506–514. doi:10.1145/320998.321004
11. Katoh N, Ibaraki T, Mine H (1982) An efficient algorithm for K shortest simple paths. Networks 12(4):411–427. doi:10.1002/net.3230120406
12. Lawler EL (1972) A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem. Manag Sci 18:401–405. doi:10.1287/mnsc.18.7.401
13. Murty KG (1968) Letter to the editor–an algorithm for ranking all the assignments in order of increasing cost. Oper Res 16(3):682–687. doi:10.1287/opre.16.3.682
14. Yen JY (1971) Finding the K shortest loopless paths in a network. Manag Sci 17:712–716. http://www.jstor.org/stable/2629312

Kernelization, Bidimensionality and Kernels

Daniel Lokshtanov
Department of Informatics, University of Bergen, Bergen, Norway

Keywords

Bidimensionality; Graph algorithms; Kernelization; Parameterized complexity; Polynomial time pre-processing

Years and Authors of Summarized Original Work

2010; Fomin, Lokshtanov, Saurabh, Thilikos

Problem Definition

The theory of bidimensionality simultaneously provides subexponential time parameterized algorithms and efficient approximation schemes for a wide range of optimization problems on planar graphs and, more generally, on classes of graphs excluding a fixed graph H as a minor. It turns out that bidimensionality also provides *linear kernels* for a multitude of problems on these classes of graphs. The results stated here unify and generalize a number of kernelization results for problems on planar graphs and graphs of bounded genus; see [2] for a more thorough discussion.

Kernelization

Kernelization is a mathematical framework for the study of polynomial time preprocessing of instances of computationally hard problems. Let $\mathcal{G}$ be the set of all graphs. A *parameterized graph problem* is a subset Π of $\mathcal{G} \times \mathbb{N}$. An *instance* is a pair $(G, k) \in \mathcal{G} \times \mathbb{N}$. The instance (G, k) is a "yes"-instance of Π if $(G, k) \in \Pi$ and a "no"-instance otherwise. A *strict kernel with ck vertices* for a parameterized graph problem Π and constant $c > 0$ is an algorithm $\mathcal{A}$ with the following properties:

- $\mathcal{A}$ takes as input an instance (G, k), runs in polynomial time, and outputs another instance (G', k').
- (G', k') is a "yes"-instance of Π if and only if (G, k) is.
- $|V(G')| \leq c \cdot k$ and $k' \leq k$.

A *linear kernel* for a parameterized graph problem is a strict kernel with ck vertices for some constant c. We remark that our definition of a linear kernel is somewhat simplified compared to the classic definition [8], but that it is essentially equivalent. For a discussion of the definition of a kernel, we refer to the textbook of Cygan et al. [4].

Graph Classes

Bidimensionality theory primarily concerns itself with graph problems where the input graph is restricted to be in a specific *graph class*. A *graph class* $\mathcal{C}$ is simply a subset of the set $\mathcal{G}$ of all graphs. As an example, the set of all planar graphs is a graph class. Another example of a graph class is the set of all *apex* graphs. Here a graph H is *apex* if H contains a vertex v such that deleting v from H leaves a planar graph. Notice that every planar graph is apex.

A graph H is a *minor* of a graph G if H can be obtained from G by deleting vertices, deleting edges, or contracting edges. Here *contracting* the edge $\{u, v\}$ in G means identifying the vertices u and v and removing all self-loops and double edges. If H can be obtained from G just by contracting edges, then H is a *contraction* of G.

A graph class $\mathcal{C}$ is *minor closed* if every minor of a graph in $\mathcal{C}$ is also in $\mathcal{C}$. A graph class $\mathcal{C}$ is *minor-free* if $\mathcal{C}$ is minor closed and there exists a graph $H \notin \mathcal{C}$. A graph class $\mathcal{C}$ is *apex-minor-free* if $\mathcal{C}$ is minor closed and there exists an apex graph $H \notin \mathcal{C}$. Notice that $H \notin \mathcal{C}$ for a minor closed class $\mathcal{C}$ implies that H cannot be a minor of any graph $G \in \mathcal{C}$.

CMSO Logic

CMSO logic stands for *Counting Monadic Second Order* logic, a formal language to describe properties of graphs. A *CMSO-sentence* is a formula ψ with variables for single vertices, vertex sets, single edges and edge sets, existential and universal quantifiers ($\exists$ and $\forall$), logical connectives $\vee$, $\wedge$ and $\neg$, as well as the following operators:

- $v \in S$, where v is a vertex variable and S is a vertex set variable. The operator returns true if the vertex v is in the vertex set S. Similarly, CMSO has an operator $e \in X$ where e is an edge variable and X is an edge set variable.
- $v_1 = v_2$, where v_1 and v_2 are vertex variables. The operator returns true if v_1 and v_2 are the same vertex of G. There is also an operator $e_1 = e_2$ to check equality of two edge variables e_1 and e_2.
- $\mathbf{adj}(v_1, v_2)$ is defined for vertex variables v_1 and v_2 and returns true if v_1 and v_2 are adjacent in G.
- $\mathbf{inc}(v, e)$ is defined for a vertex variable v and edge variable e. $\mathbf{inc}(v, e)$ returns true if the edge e is incident to the vertex v in G, in other words, if v is one of the two endpoints of e.
- $\mathbf{card}_{p,q}(S)$ is defined for every pair of integers p, q, and vertex or edge set variable S. $\mathbf{card}_{p,q}(S)$ returns true if $|S| \equiv q \mod p$. For an example, $\mathbf{card}_{2,1}(S)$ returns true if $|S|$ is odd.

When we quantify a variable, we need to specify whether it is a vertex variable, edge variable, vertex set variable, or edge set variable. To specify that an existentially quantified variable x is a vertex variable we will write $\exists x \in V(G)$. We will use $\forall e \in E(G)$ to universally quantify edge variables and $\exists X \subseteq V(G)$ to existentially quantify vertex set variables. We will always use lower case letters for vertex and edge variables and upper case letters for vertex set and edge set variables.

A graph G on which the formula ψ is true is said to *model* ψ. The notation $G \models \psi$ means that G models ψ. As an example, consider the formula

$$\psi_1 = \forall v \in V(G)\ \forall x \in V(G)\ \forall y \in V(G)\ \forall z \in V(G) : \\ (x = y) \vee (x = z) \vee (y = z) \vee \neg\mathbf{adj}(v, x) \vee \neg\mathbf{adj}(v, y) \vee \neg\mathbf{adj}(v, z)$$

The formula ψ_1 states that for every four (not necessarily distinct) vertices v, x, y, and z, if x, y, and z are distinct, then v is not adjacent to all of $\{x, y, z\}$. In other words, a graph G models ϕ_1 if and only if the degree of every vertex G is at most 2. CMSO can be used to express many

graph properties, such as G having a Hamiltonian cycle, G being 3-colorable, or G being planar.

In CMSO, one can also write formulas where one uses *free variables*. These are variables that are used in the formula but never quantified with an $\exists$ or $\forall$ quantifier. As an example, consider the formula

$$\psi_{DS} = \forall u \in V(G)\, \exists v \in V(G) : (v \in S) \wedge (u = v \vee \mathbf{adj}(u, v))$$

The variable S is a free variable in ψ_{DS} because it is used in the formula, but is never quantified. It does not make sense to ask whether a graph G models ψ_{DS} because when we ask whether the vertex v is in S, the set S is not well defined. However, if the set $S \subseteq V(G)$ is provided together with the graph G, we can evaluate the formula ψ_{DS}. ψ_{DS} will be true for a graph G and set $S \subseteq V(G)$ if, for every vertex $u \in V(G)$, there exists a vertex $v \in V(G)$ such that v is in S and either $u = v$ or u and v are neighbors in G. In other words, the pair (G, S) models ψ_{DS} (written $(G, S) \models \psi_{DS}$) if and only if S is a dominating set in G (i.e., every vertex not in S has a neighbor in S).

CMSO-Optimization Problems

We are now in position to define the parameterized problems for which we will obtain kernelization results. For every CMSO formula ψ with a single free vertex set variable S, we define the following two problems:

ψ-CMSO-Min (Max):

INPUT: *Graph G and integer k.*
QUESTION: *Does there exist a vertex set $S \subseteq V(G)$ such that $(G, S) \models \psi$ and $|S| \leq k$ ($|S| \geq k$ for Max).*

Formally, ψ-CMSO-MIN (MAX) is a parameterized graph problem where the "yes" instances are exactly the pairs (G, k) such that there exists a vertex set S of size at most k (at least k) and $(G, S) \models \psi$. We will use the term *CMSO-optimization problems* to refer to ψ-CMSO-MIN (MAX) for some CMSO formula ψ.

Many well-studied and not so well-studied graph problems are CMSO-optimization problems. Examples include VERTEX COVER, DOMINATING SET, CYCLE PACKING, and the list goes on and on (see [2]). We encourage the interested reader to attempt to formulate the problems mentioned above as CMSO-optimization problems. We will be discussing CMSO-optimization problems *on planar graphs* and on minor-free classes of graphs.

Our results are for problems where the input graph is promised to belong to a certain graph class $\mathcal{C}$. We formalize this by encoding membership in $\mathcal{C}$ in the formula ψ. For an example, ψ_{DS}-CMSO-MIN is the well-studied DOMINATING SET problem. If we want to restrict the problem to planar graphs, we can make a new CMSO logic formula ψ_{planar} such that $G \models \psi_{\text{planar}}$ if and only if G is planar. We can now make a new formula

$$\psi'_{DS} = \psi_{DS} \wedge \psi_{\text{planar}}$$

and consider the problem ψ'_{DS}-CMSO-MIN. Here (G, k) is a "yes" instance if G has a dominating set S of size at most k *and* G is planar. Thus, this problem also forces us to check planarity of G, but this is polynomial time solvable and therefore not an issue with respect to kernelization. In a similar manner, one can restrict any CMSO-optimization problem to a graph class $\mathcal{C}$, as long as there exists a CMSO formula $\psi_{\mathcal{C}}$ such that $G \models \psi_{\mathcal{C}}$ if and only if $G \in \mathcal{C}$. Luckily, such a formula is known to exist for every minor-free class $\mathcal{C}$. We will say that a parameterized problem Π is a problem *on the graph class $\mathcal{C}$* if, for every "yes" instance (G, k) of Π, the graph G is in $\mathcal{C}$.

For any CMSO-MIN problem Π, we have that $(G, k) \in \Pi$ implies that $(G, k') \in \Pi$ for all $k' \geq k$. Similarly, for a CMSO-MAX problem Π, we have that $(G, k) \in \Pi$ implies that $(G, k') \in \Pi$ for all $k' \leq k$. Thus, the notion of "optimality" is well defined for CMSO-optimization problems. For the problem $\Pi = \psi$-CMSO-MIN, we define

$$OPT_{\Pi}(G) = \min\{k : (G, k) \in \Pi\}.$$

If no k such that $(G,k) \in \Pi$ exists, $OPT_\Pi(G)$ returns $+\infty$. Similarly, for the problem $\Pi = \psi$-CMSO-MAX,

$$OPT_\Pi(G) = \max\{k \ : \ (G,k) \in \Pi\}.$$

If no k such that $(G,k) \in \Pi$ exists, $OPT_\Pi(G)$ returns $-\infty$. We define $SOL_\Pi(G)$ to be a function that given as input a graph G returns a set S of size $OPT_\Pi(G)$ such that $(G,S) \models \psi$ and returns **null** if no such set S exists.

Bidimensionality

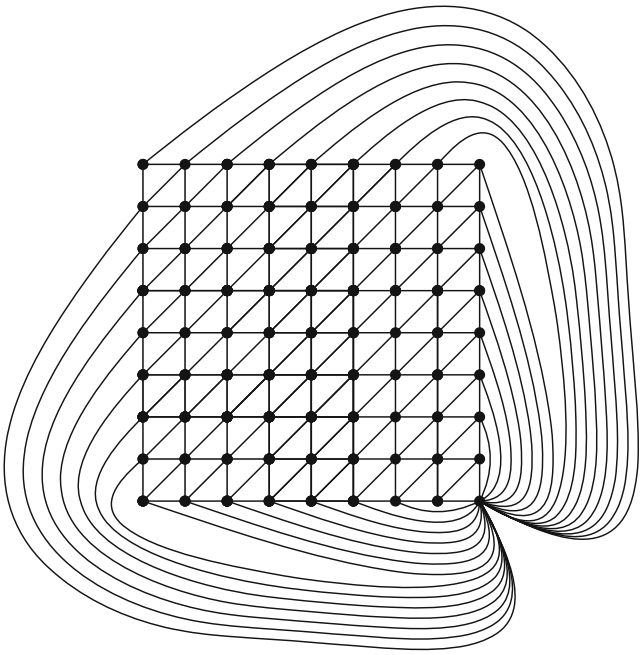
For many problems, it holds that contracting an edge cannot increase the size of the optimal solution. We will say that such problems are contraction closed. Formally, a CMSO-optimization problem Π is *contraction closed* if for any G and $uv \in E(G)$, $OPT_\Pi(G/uv) \leq OPT_\Pi(G)$. If contracting edges, deleting edges, and deleting vertices cannot increase the size of the optimal solution, we say that the problem is *minor closed*.

Informally, a problem is *bidimensional* if it is minor closed and the value of the optimum grows with both dimensions of a grid. In other words, on a $(k \times k)$-grid, the optimum should be approximately quadratic in k. To formally define bidimensional problems, we first need to define the $(k \times k)$-grid $\boxplus_k$, as well as the related graph Γ_k.

For a positive integer k, a $k \times k$ *grid*, denoted by $\boxplus_k$, is a graph with vertex set $\{(x,y) \ : \ x,y \in \{1,\ldots,k\}\}$. Thus, $\boxplus_k$ has exactly k^2 vertices. Two different vertices (x,y) and (x',y') are adjacent if and only if $|x-x'|+|y-y'| = 1$. For an integer $k > 0$, the graph Γ_k is obtained from the grid $\boxplus_k$ by adding in every grid cell the diagonal edge going up and to the right and making the bottom right vertex of the grid adjacent to all border vertices. The graph Γ_9 is shown in Fig. 1.

We are now ready to give the definition of bidimensional problems. A CMSO-optimization problem Π is *contraction-bidimensional* if it is contraction closed, and there exists a constant $c > 0$ such that $OPT_\Pi(\Gamma_k) \geq ck^2$. Similarly, Π is *minor-bidimensional* if it is minor closed, and there exists a constant $c > 0$ such that $OPT_\Pi(\boxplus_k) \geq ck^2$.

As an example, the DOMINATING SET problem is contraction-bidimensional. It is easy to verify that contracting an edge may not increase the size of the smallest dominating set of a graph G and that Γ_k does not have a dominating set of size smaller than $\frac{(k-2)^2}{7}$.



Kernelization, Bidimensionality and Kernels, Fig. 1 The graph Γ_9

Separability

Our kernelization algorithms work by recursively splitting the input instance by small separators. For this to work, the problem has to be somewhat well behaved in the following sense. Whenever a graph is split along a small separator into two independent sub-instances L and R, the size of the optimum solution for the graph $G[L]$ is relatively close to the size of the intersection between L and the optimum solution to the original graph G. We now proceed with a formal definition of what it means for a problem to be well behaved.

For a set $L \subseteq V(G)$, we define $\partial(L)$ to be the set of vertices in L with at least one neighbor outside L. A CMSO-optimization problem Π is *linear separable* if there exists a constant $c \geq 0$ such that for every set $L \subseteq V(G)$, we have

$$|SOL_\Pi(G) \cap L| - c \cdot |\partial(L)| \leq OPT_\Pi(G[L]) \leq |SOL_\Pi(G) \cap L| + c \cdot |\partial(L)|.$$

For a concrete example, we encourage the reader to consider the DOMINATING SET problem and to prove that for DOMINATING SET the inequalities above hold. The crux of the argument is to augment optimal solutions of G and $G[L]$ by adding all vertices in $\partial(L)$ to them.

Key Results

We can now state our main theorem.

Theorem 1 *Let Π be a separable CMSO-optimization problem on the graph class $\mathcal{C}$. Then, if Π is minor-bidimensional and $\mathcal{C}$ is minor-free, or if Π is contraction-bidimensional and $\mathcal{C}$ is apex-minor-free, Π admits a linear kernel.*

The significance of Theorem 1 is that it is, in general, quite easy to formulate graph problems as CMSO-optimization problems and prove that the considered problem is bidimensional and separable. If we are able to do this, Theorem 1 immediately implies that the problem admits a linear kernel on all minor-free graph classes, or on all apex-minor-free graph classes. As an example, the DOMINATING SET problem has been shown to have a linear kernel on planar graphs [1], and the proof of this fact is quite tricky. However, in our examples, we have shown that DOMINATING SET is a CMSO-MIN problem, that it is contraction-bidimensional, and that it is separable. Theorem 1 now implies that DOMINATING SET has a linear kernel not only on planar graphs but on all apex-minor-free classes of graphs! One can go through the motions and use Theorem 1 to give linear kernels for quite a few problems. We refer the reader to [9] for a non-exhaustive list.

We remark that the results stated here are generalizations of results obtained by Bodlaender et al. [2]. Theorem 1 is proved by combining "algebraic reduction rules" (fully developed by Bodlaender et al. [2]) with new graph decomposition theorems (proved in [9]). The definitions here differ slightly from the definitions in the original work [9] and appear here in the way they will appear in the journal version of [9].

Cross-References

- Bidimensionality
- Data Reduction for Domination in Graphs

Recommended Reading

1. Alber J, Fellows MR, Niedermeier R (2004) Polynomial-time data reduction for dominating set. J ACM 51(3):363–384
2. Bodlaender HL, Fomin FV, Lokshtanov D, Penninkx E, Saurabh S, Thilikos DM (2013) (Meta) Kernelization. CoRR abs/0904.0727. http://arxiv.org/abs/0904.0727
3. Borie RB, Parker RG, Tovey CA (1992) Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families. Algorithmica 7(5&6):555–581
4. Cygan M, Fomin FV, Kowalik Ł, Lokshtanov D, Marx D, Pilipczuk M, Pilipczuk M, Saurabh S (2015, to appear) Parameterized algorithms. Springer, Heidelberg
5. Demaine ED, Hajiaghayi M (2005) Bidimensionality: new connections between FPT algorithms and PTASs. In: Proceedings of the 16th annual ACM-SIAM symposium on discrete algorithms (SODA), Vancouver. SIAM, pp 590–601

6. Demaine ED, Hajiaghayi M (2008) The bidimensionality theory and its algorithmic applications. Comput J 51(3):292–302
7. Demaine ED, Fomin FV, Hajiaghayi M, Thilikos DM (2005) Subexponential parameterized algorithms on graphs of bounded genus and H-minor-free graphs. J ACM 52(6):866–893
8. Downey RG, Fellows MR (2013) Fundamentals of parameterized complexity. Texts in computer science. Springer, London
9. Fomin FV, Lokshtanov D, Saurabh S, Thilikos DM (2010) Bidimensionality and kernels. In: Proceedings of the 20th annual ACM-SIAM symposium on discrete algorithms (SODA), Austin. SIAM, pp 503–510
10. Fomin FV, Lokshtanov D, Raman V, Saurabh S (2011) Bidimensionality and EPTAS. In: Proceedings of the 21st annual ACM-SIAM symposium on discrete algorithms (SODA), San Francisco. SIAM, pp 748–759

Kernelization, Constraint Satisfaction Problems Parameterized above Average

Gregory Gutin
Department of Computer Science, Royal Holloway, University of London, Egham, UK

Keywords

Bikernel; Kernel; MaxCSP; MaxLin; MaxSat

Years and Authors of Summarized Original Work

2011; Alon, Gutin, Kim, Szeider, Yeo

Problem Definition

Let r be an integer, let $V = \{v_1, \ldots, v_n\}$ be a set of variables, each taking values -1 (TRUE) and 1 (FALSE), and let Φ be a set of Boolean functions, each involving at most r variables from V. In the problem MAX-r-CSP, we are given a collection $\mathcal{F}$ of m Boolean functions, each $f \in \mathcal{F}$ being a member of Φ and each with a positive integral weight. Our aim is to find a truth assignment that maximizes the total weight of satisfied functions from $\mathcal{F}$. We will denote the maximum by $\mathrm{sat}(\mathcal{F})$.

Let A be the average weight (over all truth assignments) of satisfied functions. Observe that A is a lower bound for $\mathrm{sat}(\mathcal{F})$. In fact, A is a tight lower bound, whenever the family Φ is closed under replacing each variable by its complement [1]. Thus, it is natural to parameterize MAX-r-CSP as follows (AA stands for *Above Average*).

MAX-r-CSP-AA

Instance: A collection $\mathcal{F}$ of m Boolean functions, each $f \in \mathcal{F}$ being a member of Φ, each with a positive integral weight, and a nonnegative integer k.
Parameter: k.
Question: $\mathrm{sat}(\mathcal{F}) \geq A + k$?

If Φ is the set of clauses with at most r literals, then we get a subproblem of MAX-r-CSP-AA, abbreviated MAX-r-SAT-AA, whose unparameterized version is simply MAX-r-SAT. Assign -1 or 1 to each variable in V randomly and uniformly. Since a clause c of an MAX-r-SAT-AA instance can be satisfied with probability $1 - 2^{r_c}$, where r_c is the number of literals in c, we have $A = \sum_{c \in \mathcal{F}} (1 - 2^{r_c})$. Clearly, A is a tight lower bound.

If Φ is the set S of equations $\prod_{i \in I_j} v_i = b_j$, $j = 1, \ldots, m$, where $v_i, b_j \in \{-1, 1\}$, b_js are constants, $|I_j| \leq r$, then we get a subproblem of MAX-r-CSP-AA, abbreviated MAX-r-LIN2-AA, whose unparameterized version is simply MAX-r-LIN2. Assign -1 or 1 to each variable in V randomly and uniformly. Since each equation of $\mathcal{F}$ can be satisfied with probability $1/2$, we have $A = W/2$, where W is the sum of the weights of equations in $\mathcal{F}$. For an assignment $v = v^0$ of values to the variables, let $\mathrm{sat}(S, v^0)$ denote the total weight of equations of S satisfied by the assignment. The difference $\mathrm{sat}(S, v^0) - W/2$ is called the *excess* of x^0. Let $\mathrm{sat}(S)$ be the maximum of $\mathrm{sat}(S, v^0)$ over all possible assignments v^0.

The following notion was introduced in [1]. Let Π and Π' be parameterized problems. A *bikernel* for Π is a polynomial-time algorithm that maps an instance (I,k) of Π to an instance (I',k') of Π' such that (i) $(I,k) \in \Pi$ if and only if $(I',k') \in \Pi'$ and (ii) $k' \leq g(k)$ and $|I'| \leq g(k)$ for some function g. The function $g(k)$ is called the *size* of the bikernel. It is known that a decidable problem is fixed-parameter tractable if and only if it admits a bikernel [1]. However, in general a bikernel can have an exponential size, in which case the bikernel may not be useful as a data reduction. A bikernel is called a *polynomial* bikernel if both $f(k)$ and $g(k)$ are polynomials in k.

When $\Pi = \Pi'$ we say that a bikernel for Π is simply a *kernel* of Π. A great deal of research has been devoted to decide whether a problem admits a polynomial kernel.

The following lemma of Alon et al. [1] shows that polynomial bikernels imply polynomial kernels.

Lemma 1 *Let Π, Π' be a pair of decidable parameterized problems such that the nonparameterized version of Π' is in NP and the nonparameterized version of Π is NP-complete. If there is a bikernelization from Π to Π' producing a bikernel of polynomial size, then Π has a polynomial-size kernel.*

Key Results

Following [2], for a Boolean function f of weight $w(f)$ and on $r(f) \leq r$ Boolean variables $v_{i_1}, \ldots, v_{i_{r(f)}}$, we introduce a polynomial $h_f(v)$, $v = (v_1, \ldots, v_n)$ as follows. Let $S_f \subseteq \{-1,1\}^{r(f)}$ denote the set of all satisfying assignments of f. Then

$$h_f(v) = w(f)2^{r-r(f)} \sum_{(a_1,\ldots,a_{r(f)}) \in S_f} \left[\prod_{j=1}^{r(f)} (1 + v_{i_j} a_j) - 1 \right].$$

Let $h(v) = \sum_{f \in \mathcal{F}} h_f(v)$. It is easy to see (cf. [1]) that the value of $h(v)$ at some v^0 is precisely $2^r(U - A)$, where U is the total weight of the functions satisfied by the truth assignment v^0. Thus, the answer to MAX-r-CSP-AA is YES if and only if there is a truth assignment v^0 such that $h(v^0) \geq k2^r$.

Algebraic simplification of $h(v)$ will lead us the following (Fourier expansion of $h(v)$, cf. [7]):

$$h(v) = \sum_{S \in \mathcal{F}} c_S \prod_{i \in S} v_i, \tag{1}$$

where $\mathcal{F} = \{\emptyset \neq S \subseteq \{1,2,\ldots,n\} : c_S \neq 0, |S| \leq r\}$. Thus, $|\mathcal{F}| \leq n^r$. The sum $\sum_{S \in \mathcal{F}} c_S \prod_{i \in S} v_i$ can be viewed as the excess of an instance of MAX-r-LIN2-AA, and, thus, we can reduce MAX-r-CSP-AA into MAX-r-LIN2-AA in polynomial time (since r is fixed, the algebraic simplification can be done in polynomial time and it does not matter whether the parameter of MAX-r-LIN2-AA is k or $k' = k2^r$). It is proved in [5] that MAX-r-LIN2-AA has a kernel with $O(k^2)$ variables and equations. This kernel is a bikernel from MAX-r-CSP-AA to MAX-r-LIN2-AA. Thus, by Lemma 1, we obtain the following theorem of Alon et al. [1].

Theorem 1 MAX-r-CSP-AA *admits a polynomial-size kernel.*

Applying a reduction from MAX-r-LIN2-AA to MAX-r-SAT-AA in which each monomial in (1) is replaced by 2^{r-1} clauses, Alon et al. [1] obtained the following:

Theorem 2 MAX-r-SAT-AA *admits a kernel with $O(k^2)$ clauses and variables.*

It is possible to improve this theorem with respect to the number of variables in the kernel. The following result was first obtained by Kim and Williams [6] (see also [3]).

Theorem 3 MAX-r-SAT-AA *admits a kernel with $O(k)$ variables.*

Crowston et al. [4] studied the following natural question: How parameterized complexity of MAX-r-SAT-AA changes when r is no longer

a constant, but a function $r(n)$ of n. They proved that MAX-$r(n)$-SAT-AA is para-NP-complete for any $r(n) \geq \lceil \log n \rceil$. They also proved that assuming the exponential time hypothesis, MAX-$r(n)$-SAT-AA is not even in XP for any integral $r(n) \geq \log\log n + \phi(n)$, where $\phi(n)$ is any real-valued unbounded strictly increasing computable function. This lower bound on $r(n)$ cannot be decreased much further as they proved that MAX-$r(n)$-SAT-AA is (i) in XP for any $r(n) \leq \log\log n - \log\log\log n$ and (ii) fixed-parameter tractable for any $r(n) \leq \log\log n - \log\log\log n - \phi(n)$, where $\phi(n)$ is any real-valued unbounded strictly increasing computable function. The proofs use some results on MAXLIN2-AA.

Cross-References

- Kernelization, MaxLin Above Average
- Kernelization, Permutation CSPs Parameterized above Average

Recommended Reading

1. Alon N, Gutin G, Kim EJ, Szeider S, Yeo A (2011) Solving MAX-k-SAT above a tight lower bound. Algorithmica 61:638–655
2. Alon N, Gutin G, Krivelevich M (2004) Algorithms with large domination ratio. J Algorithms 50: 118–131
3. Crowston R, Fellows M, Gutin G, Jones M, Kim EJ, Rosamond F, Ruzsa IZ, Thomassé S, Yeo A (2014) Satisfying more than half of a system of linear equations over GF(2): a multivariate approach. J Comput Syst Sci 80:687–696
4. Crowston R, Gutin G, Jones M, Raman V, Saurabh S (2013) Parameterized complexity of MaxSat above average. Theor Comput Sci 511:77–84
5. Gutin G, Kim EJ, Szeider S, Yeo A (2011) A probabilistic approach to problems parameterized above tight lower bound. J Comput Syst Sci 77: 422–429
6. Kim EJ, Williams R (2012) Improved parameterized algorithms for above average constraint satisfaction. In: IPEC 2011, Saarbrücken. Lecture notes in computer science, vol 7112, pp 118–131
7. O'Donnell R (2008) Some topics in analysis of Boolean functions. Technical report, ECCC report TR08-055. Paper for an invited talk at STOC'08. www.eccc.uni-trier.de/eccc-reports/2008/TR08-055/

Kernelization, Exponential Lower Bounds

Hans L. Bodlaender
Department of Computer Science, Utrecht University, Utrecht, The Netherlands

Keywords

Composition; Compression; Fixed parameter tractability; Kernelization; Lower bounds

Years and Authors of Summarized Original Work

2009; Bodlaender, Downey, Fellows, Hermelin

Problem Definition

Research on kernelization is motivated in two ways. First, when solving a hard (e.g., NP-hard) problem in practice, a common approach is to first *preprocess* the instance at hand before running more time-consuming methods (like integer linear programming, branch and bound, etc.). The following is a natural question. Suppose we use polynomial time for this preprocessing phase: what can be predicted of the size of the instance resulting from preprocessing? The theory of kernelization gives us such predictions. A second motivation comes from the fact that a decidable parameterized problem belongs to the class FPT (i.e., is *fixed parameter tractable*,) if and only if the problem has kernelization algorithm.

A parameterized problem is a subset of $\Sigma^* \times \mathbf{N}$, for some finite set Σ. A *kernelization algorithm* (or, in short *kernel*) for a parameterized problem $Q \subseteq \Sigma^* \times \mathbf{N}$ is an algorithm A that receives as input a pair $(x,k) \in \Sigma^* \times \mathbf{N}$ and outputs a pair $(x',k') = A(x,k)$, such that:

- A uses time, polynomial in $|x| + k$.
- $(x,k) \in Q$, if and only if $(x',k') \in Q$.
- There are functions f, g, such that $|x'| \leq f(k)$ and $k' \leq g(k)$.

In the definition above, f and g give an upper bound on the size, respectively the parameter of the reduced instance. Many well-studied problems have kernels with $k' \leq k$. The running time of an exact algorithm that starts with a kernelization step usually is exponential in the size of the kernel (i.e., $f(k)$), and thus small kernels are desirable. A kernel is said to be *polynomial*, if f and g are bounded by a polynomial. Many well-known parameterized problems have a polynomial kernel, but there are also many for which such a polynomial kernel is not known.

Recent techniques allow us to show, under a complexity theoretic assumption, for some parameterized problems that they do not have a polynomial kernel. The central notion is that of *compositionality*; with the help of transformations and cross compositions, a larger set of problems can be handled.

Key Results

Compositionality

The basic building block of showing that problems do not have a polynomial kernel (assuming $NP \not\subseteq coNP/poly$) is the notion of compositionality. It comes in two types: or-composition and and-composition.

Definition 1 An *or-composition* for a parameterized problem $Q \subseteq \Sigma^* \times \mathbf{N}$ is an algorithm that:

- Receives as input a sequence of instances for Q with the same parameter $(s_1, k), (s_2, k), \ldots, (s_r, k)$
- Uses time, polynomial in $k + \sum_{i=1}^{r} |s_i|$
- Outputs one instance for Q, $(s', k') \in \Sigma^* \times \mathbf{N}$, such that:

  1. $(s', k') \in Q$, if and only if there is an i, $1 \leq i \leq r$, with $(s_i, k) \in Q$.
  2. k' is bounded by a polynomial in k.

The notion of *and-composition* is defined similarly, with the only difference that condition (1) above is replaced by

$(s', k') \in Q$, if and only if for all i, $1 \leq i \leq r$: $(s_i, k) \in Q$.

We define the *classic variant* of a parameterized problem $Q \subseteq \Sigma^* \times \mathbf{N}$ as the decision problem, denoted Q^c where we assume that the parameter is encoded in unary, or, equivalently, an instance (s, k) is assumed to have size $|s| + k$.

Combining results of three papers gives the following results.

Theorem 1 *Let $Q \subseteq \Sigma^* \times \mathbf{N}$ be a parameterized problem. Suppose that the classic variant of Q, Q^c is NP-hard. Assume that $NP \not\subseteq coNP/poly$.*

1. *(Bodlaender et al. [3], Fortnow and Santhanam [12]) If Q has an or-composition, then Q has no polynomial kernel.*
2. *(Bodlaender et al. [3], Drucker [11]) If Q has an and-composition, then Q has no polynomial kernel.*

The condition that $NP \not\subseteq coNP/poly$ is equivalent to $coNP \not\subseteq NP/poly$; if it does not hold, the polynomial time hierarchy collapses to the third level [19].

For many parameterized problems, one can establish (sometimes trivially, and sometimes with quite involved proofs) that they are or-compositional or and-compositional. Taking the disjoint union of instances often gives a trivial composition. A simple example is the LONG PATH problem; it gets as input a pair (G, k) with G an undirected graph and asks whether G has a simple path of length at least k.

Lemma 1 *If $NP \not\subseteq coNP/poly$, then* LONG PATH *has no kernel polynomial in k.*

Proof LONG PATH is well known to be NP-complete. Mapping $(G_1, k), \ldots, (G_r, k)$ to the pair (H, k) with H the disjoint union of $G_1, \ldots, G_r$ is an or-composition. So, the result follows directly as a corollary of Theorem 1. □

The TREEWIDTH problem gets as input a pair (G, k) and asks whether the treewidth of G is most k. As it is NP-hard to decide if the treewidth of a given graph G is at most a given

number k [1] and the treewidth of a graph is the maximum of its connected components, taking the disjoint union gives an and-composition for the TREEWIDTH problem and shows that TREEWIDTH has no polynomial kernel unless $NP \subseteq coNP/poly$. Similar proofs work for many more problems. Many problems can be seen to be and- or or-compositional and thus have no polynomial kernels under the assumption that $NP \not\subseteq coNP/poly$. See, e.g., [3,5,9,17].

Transformations

Several researchers observed independently (see [2,5,9]) that transformations can be used to show results for additional problems. The formalization is due to Bodlaender et al. [5].

Definition 2 A *polynomial parameter transformation* (*ppt*) from parameterized problem $Q \subseteq \Sigma^* \times \mathbf{N}$ to parameterized problem $R \subseteq \Sigma^* \times \mathbf{N}$ is an algorithm A that:

- Has as input an instance of Q, $(s,k) \in \Sigma^* \times \mathbf{N}$.
- Outputs an instance of R, $(s',k') \in \Sigma^* \times \mathbf{N}$.
- $(s,k) \in Q$ if and only if $(s',k') \in R$.
- A uses time polynomial in $|s| + k$.
- k' is bounded by a polynomial in k.

The differences with the well-known polynomial time or Karp reductions from NP-completeness theory are small: note in particular that it is required that the new value of the parameter is polynomially bounded in the old value of the parameter. The following theorem follows quite easily.

Theorem 2 (See [5,6]) *Let R have a polynomial kernel. If there is a ppt from Q to R, and a polynomial time reduction from R to the classic variant of Q, then Q has a polynomial kernel.*

This implies that if we have a ppt from Q to R, Q^c is NP-hard, $R^c \in NP$, then when Q has no polynomial kernel, R has no polynomial kernel.

Cross Composition

Bodlaender et al. [4] introduced the concept of *cross composition*. It gives a more powerful mechanism to show that some problems have no polynomial kernel, assuming $NP \not\subseteq coNP/poly$. We need first the definition of a polynomial equivalence relation.

Definition 3 A *polynomial equivalence relation* is an equivalence relation on Σ^* that can be decided in polynomial time and has for each n, a polynomial number of equivalence classes that contain strings of length at most n.

A typical example may be that strings represent graphs and two graphs are equivalent if and only if they have the same number of vertices and edges.

Definition 4 Let L be a language, R a polynomial equivalence relation, and Q a parameterized problem. An *OR cross composition* of L to Q (w.r.t. R) is an algorithm that:

- Gets as input a sequence of instances $s_1, \ldots, s_r$ of L that belong to the same equivalence class of R.
- Uses time, polynomial in $\sum_{i=1}^{r} |s_i|$.
- Outputs an instance (s',k) of Q.
- k is polynomial in $\max |s_i| + \log k$.
- $(s',k) \in Q$ if and only if there is an i with $s_i \in L$.

The definition for an AND cross composition is similar; the last condition is replaced by

$(s',k) \in Q$ if and only if for all i with $s_i \in L$.

Theorem 3 (Bodlaender et al. [4]) *If we have an OR cross composition, or an AND cross composition from an NP-hard language L into a parameterized problem Q, then Q does not have a polynomial kernel, unless $NP \subseteq coNP/poly$.*

The main differences with or-composition and and-composition are we do not need to start with a collection of instances from Q, but can use a collection of instances of any NP-hard language; the bound on the new value of k usually allows us to restrict to collections of at most 2^k instances, and with the polynomial equivalence relation, we can make assumptions on "similarities" between these instances.

For examples of OR cross compositions, and of AND cross compositions, see, e.g., [4, 8, 13, 17].

Other Models and Improvements

Different models of compressibility and stronger versions of the lower bound techniques have been studied, including more general models of compressibility (see [11] and [7]), the use of co-nondeterministic composition [18], weak composition [15], Turing kernelization [16], and a different measure for compressibility based on witness size of problems in NP [14].

Problems Without Kernels

Many parameterized problems are known to be hard for the complexity class $W[1]$. As decidable problems are known to have a kernel, if and only if they are fixed parameter tractable, it follows that $W[1]$-hard problems do not have a kernel, unless $W[1] = FPT$ (which would imply that the exponential time hypothesis does not hold). See, e.g., [10].

Cross-References

- ▸ Kernelization, Polynomial Lower Bounds
- ▸ Kernelization, Preprocessing for Treewidth
- ▸ Kernelization, Turing Kernels

Recommended Reading

1. Arnborg S, Corneil DG, Proskurowski A (1987) Complexity of finding embeddings in a k-tree. SIAM J Algebr Discret Methods 8:277–284
2. Binkele-Raible D, Fernau H, Fomin FV, Lokshtanov D, Saurabh S, Villanger Y (2012) Kernel(s) for problems with no kernel: on out-trees with many leaves. ACM Trans Algorithms 8(5):38
3. Bodlaender HL, Downey RG, Fellows MR, Hermelin D (2009) On problems without polynomial kernels. J Comput Syst Sci 75:423–434
4. Bodlaender HL, Jansen BMP, Kratsch S (2011) Cross-composition: a new technique for kernelization lower bounds. In: Schwentick T, Dürr C (eds) Proceedings 28th international symposium on theoretical aspects of computer science, STACS 2011, Dortmund. Schloss Dagstuhl – Leibnitz-Zentrum fuer Informatik, Leibniz International Proceedings in Informatics (LIPIcs), vol 9, pp 165–176
5. Bodlaender HL, Thomassé S, Yeo A (2011) Kernel bounds for disjoint cycles and disjoint paths. Theor Comput Sci 412:4570–4578
6. Bodlaender HL, Jansen BMP, Kratsch S (2012) Kernelization lower bounds by cross-composition. CoRR abs/1206.5941
7. Chen Y, Flum J, Müller M (2011) Lower bounds for kernelizations and other preprocessing procedures. Theory Comput Syst 48(4):803–839
8. Cygan M, Kratsch S, Pilipczuk M, Pilipczuk M, Wahlström M (2012) Clique cover and graph separation: new incompressibility results. In: Czumaj A, Mehlhorn K, Pitts AM, Wattenhofer R (eds) Proceedings of the 39th international colloquium on automata, languages and programming, ICALP 2012, Part I, Warwick. Lecture notes in computer science, vol 7391. Springer, pp 254–265
9. Dom M, Lokshtanov D, Saurabh S (2009) Incompressibility through colors and IDs. In: Albers S, Marchetti-Spaccamela A, Matias Y, Nikoletseas SE, Thomas W (eds) Proceedings of the 36th international colloquium on automata, languages and programming, ICALP 2009, Part I, Rhodes. Lecture notes in computer science, vol 5555. Springer, pp 378–389
10. Downey RG, Fellows MR (2013) Fundamentals of parameterized complexity. Texts in computer science. Springer, London
11. Drucker A (2012) New limits to classical and quantum instance compression. In: Proceedings of the 53rd annual symposium on foundations of computer science, FOCS 2012, New Brunswick, pp 609–618
12. Fortnow L, Santhanam R (2011) Infeasibility of instance compression and succinct PCPs for NP. J Comput Syst Sci 77:91–106
13. Gutin G, Muciaccia G, Yeo: A (2013) (Non-)existence of polynomial kernels for the test cover problem. Inf Process Lett 113:123–126
14. Harnik D, Naor M (2010) On the compressibility of $\mathcal{NP}$ instances and cryptographic applications. SIAM J Comput 39:1667–1713
15. Hermelin D, Wu X (2012) Weak compositions and their applications to polynomial lower bounds for kernelization. In: Rabani Y (ed) Proceedings of the 22nd annual ACM-SIAM symposium on discrete algorithms, SODA 2012, Kyoto. SIAM, pp 104–113
16. Hermelin D, Kratsch S, Soltys K, Wahlström M, Wu X (2013) A completeness theory for polynomial (turing) kernelization. In: Gutin G, Szeider S (eds) Proceedings of the 8th international symposium on parameterized and exact computation, IPEC 2013, Sophia Antipolis. Lecture notes in computer science, vol 8246. Springer, pp 202–215
17. Jansen BMP, Bodlaender HL (2013) Vertex cover kernelization revisited – upper and lower bounds for a refined parameter. Theory Comput Syst 53:263–299
18. Kratsch S (2012) Co-nondeterminism in compositions: a kernelization lower bound for a Ramsey-type

problem. In: Proceedings of the 22nd annual ACM-SIAM symposium on discrete algorithms, SODA 2012, Kyoto, pp 114–122
19. Yap HP (1986) Some topics in graph theory. London mathematical society lecture note series, vol 108. Cambridge University Press, Cambridge

Kernelization, Matroid Methods

Magnus Wahlström
Department of Computer Science, Royal Holloway, University of London, Egham, UK

Keywords

Kernelization; Matroids; Parameterized complexity

Years and Authors of Summarized Original Work

2012; Kratsch, Wahlström

Problem Definition

Kernelization is the study of the power of polynomial-time instance simplification and preprocessing and relates more generally to questions of compact information representation. Given an instance x of a decision problem $\mathcal{P}$, with an associated *parameter* k (e.g., a bound on the solution size in x), a *polynomial kernelization* is an algorithm which in polynomial time produces an instance x' of $\mathcal{P}$, with parameter k', such that $x \in \mathcal{P}$ if and only if $x' \in \mathcal{P}$ and such that both $|x'|$ and k' are bounded by $p(k)$ for some $p(k) = \text{poly}(k)$. A *polynomial compression* is the variant where the output x' is an instance of a new problem $\mathcal{P}'$ (and may not have any associated parameter).

Matroid theory provides the tools for a very powerful framework for kernelization and more general information-preserving sparsification. As an example application, consider the following question. You are given a graph $G = (V, E)$ and two sets $S, T \subseteq V$ of terminal vertices, where potentially $|V| \gg |S|, |T|$. The task is to reduce G to a smaller graph $G' = (V', E')$, with $S, T \subseteq V'$ and $|V'|$ bounded by a function of $|S| + |T|$, such that for any sets $A \subseteq S$, $B \subseteq T$, the minimum (A, B)-cut in G' equals that in G. Here, all cuts are vertex cuts and may overlap A and B (i.e., the terminal vertices are also deletable). It is difficult to see how to do this without using both exponential time in $|S| + |T|$ (due to the large number of choices of A and B) and an exponential dependency of $|V'|$ on $|S|$ and $|T|$ (due to potentially having to include one min cut for every choice of A and B), yet using the appropriate tools from matroid theory, we can in polynomial time produce such a graph G' with $|V'| = O(|S| \cdot |T| \cdot \min(|S|, |T|))$. Call (G, S, T) a *terminal cut system*; we will revisit this example later.

The main power of the framework comes from two sources. The first is a class of matroids known as *gammoids*, which enable the representation of graph-cut properties as linear independence of vectors; the second is a tool known as the *representative sets lemma* (due to Lovász [4] via Marx [5]) applied to such a representation. To describe these closer, we need to review several definitions.

Background on Matroids

We provide only the bare essential definitions; for more, see Oxley [6]. Also see the relevant chapters of Schrijver [8] for a more computational perspective and Marx [5] for a concise, streamlined, and self-contained presentation of the issues most relevant to our concerns. For $s \in \mathbb{N}$, we let $[s]$ denote the set $\{1, \ldots, s\}$.

A *matroid* is a pair $M = (V, \mathcal{I})$ where V is a *ground set* and $\mathcal{I} \subseteq 2^V$ a collection of *independent sets*, subject to three axioms:

1. $\emptyset \in \mathcal{I}$.
2. If $B \in \mathcal{I}$ and $A \subseteq B$, then $A \in \mathcal{I}$.
3. If $A, B \in \mathcal{I}$ and $|A| < |B|$, then there exists some $b \in (B \setminus A)$ such that $A \cup \{b\} \in \mathcal{I}$.

All matroids we deal with will be finite (i.e., have finite ground sets). A set $S \subseteq V$ is *independent* in M if and only if $S \in \mathcal{I}$. A *basis* is a maximal independent set in M; observe that all bases of a matroid have the same cardinality. The *rank* of a set $X \subseteq V$ is the maximum cardinality of an independent set $S \subseteq X$; again, observe that this is well defined.

Linearly Represented Matroids

A prime example of a matroid is a *linear matroid*. Let A be a matrix over some field $\mathbb{F}$, and let V index the column set of A. Let $\mathcal{I}$ contain exactly those sets of columns of A that are linearly independent. Then $M = (V, \mathcal{I})$ defines a matroid, denoted $M(A)$, known as a linear matroid. For an arbitrary matroid M, if M is isomorphic to a linear matroid $M(A)$ (over a field $\mathbb{F}$), then M is *representable* (*over* $\mathbb{F}$), and the matrix A *represents* M. Observe that this is a compact representation, as $|\mathcal{I}|$ would in the general case be exponentially large, while the matrix A would normally have a coding size polynomial in $|V|$. In general, more powerful tools are available for linearly represented matroids than for arbitrary matroids (see, e.g., the MATROID MATCHING problem [8]). In particular, this holds for the representative sets lemma (see below).

Gammoids

The class of matroids central to our concern is the class of *gammoids*, first defined by Perfect [7]. Let $G = (V, E)$ be a (possibly directed) graph, $S \subseteq V$ a set of *source vertices*, and $T \subseteq V$ a set of *sink vertices* (where S and T may overlap). Let $X \subseteq T$ be independent if and only if there exists a collection of $|X|$ pairwise vertex-disjoint directed paths in G, each path starting in S and ending in X; we allow paths to have length zero (e.g., we allow a path from a vertex $x \in S \cap X$ to itself). This notion of independence defines a matroid on the ground set T, referred to as the *gammoid* defined by G, S, and T. By Menger's theorem, the rank of a set $X \subseteq T$ equals the cardinality of an (S, X)-min cut in G.

Gammoids are representable over any sufficiently large field [6], although only randomized procedures for computing a representation are known. An explicit randomized procedure was given in [2], computing a representation of the gammoid (G, S, T) in space (essentially) cubic in $|S| + |T|$. Hence, gammoids imply a polynomial-sized representation of terminal cut systems, as defined in the introduction. This has implications in kernelization [2], though it is not on its own the most useful form, since it is not a representation in terms of graphs.

Representative Sets

Let $M = (V, \mathcal{I})$ be a matroid, and X and Y independent sets in M. We say that Y *extends* X if $X \cup Y$ is independent and $X \cap Y = \emptyset$. The *representative sets lemma* states the following.

Lemma 1 ([4, 5]) *Let $M = (V, \mathcal{I})$ be a linearly represented matroid of rank $r + s$, and let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a collection of independent sets, each of size s. In polynomial time, we can compute a set $\mathcal{S}^* \subseteq \mathcal{S}$ such that $|\mathcal{S}^*| \leq \binom{r+s}{s}$, and for any independent set X, there is a set $S \in \mathcal{S}$ that extends X if and only if there is a set $S' \in \mathcal{S}^*$ that extends X.*

We refer to $\mathcal{S}^*$ as a *representative set* for $\mathcal{S}$ in M. This result is due to Lovász [4], made algorithmic by Marx [5]; recently, Fomin et al. [1] improved the running time and gave algorithmic applications of the result. The power of the lemma is extended by several tools which construct new linearly represented matroids from existing ones; see Marx [5]. For a particularly useful case, for each $i \in [s]$ let $M_i = (V_i, \mathcal{I}_i)$ be a matroid, where each set V_i is a new copy of an original ground set V. Given a representation of these matroids over the same field $\mathbb{F}$, we can form a represented matroid $M = (V_1 \cup \ldots \cup V_s, \mathcal{I}_1 \times \ldots \times \mathcal{I}_s)$ as a *direct sum* of these matroids, where an independent set X in M is the union of an independent set X_i in M_i for each i. For an element $v \in V$, let $v(i)$ denote the copy of v in V_i. Then the set $\{v(1), \ldots, v(s)\}$ extends $X = X_1 \cup \ldots \cup X_s$ if and only if $\{v(i)\}$ extends X_i for each $i \in [s]$. In other words, we have constructed an *AND operation* for the notion of an extending set.

Closest Sets and Gammoids

We need one last piece of terminology. For a (possibly directed) graph $G = (V, E)$ and sets $A, X \subseteq V$, let $R_G(A, X)$ be the set of vertices reachable from A in $G \setminus X$. The set X is *closest to* A if there is no set X' such that $|X'| \leq |X|$ and X' separates X from A, i.e., $X \cap R_G(A, X') = \emptyset$. This is equivalent to X being the unique minimum (A, X)-vertex cut. For every pair of sets $A, B \subseteq V$, there is a unique minimum (A, B)-vertex cut closest to A, which can be computed in polynomial time. Finally, for sets S and X, the set X *pushed towards* S is the unique minimum (S, X)-vertex cut closest to S; this operation is well defined and has no effect if X is already closest to S. The following is central to our applications.

Lemma 2 *Let M be a gammoid defined from a graph $G = (V, E)$ and source set S. Let X be independent in M, and let X' be X pushed towards S. For any $v \in V$, the set $\{v\}$ extends X if and only if $v \in R_G(S, X')$.*

Key Results

The most powerful version of the terminal cut system result is the following.

Theorem 1 *Let $G = (V, E)$ be a (possibly directed) graph, and $X \subseteq V$ a set of vertices. In randomized polynomial time, we can find a set $Z \subseteq V$ of $|Z| = O(|X|^3)$ vertices such that for every partition $X = A \cup B \cup C \cup D$, the set Z contains a minimum (A, B)-vertex cut in the graph $G \setminus D$.*

There is also a variant for cutting into more than two parts, as follows.

Theorem 2 *Let $G = (V, E)$ be an undirected graph, and $X \subseteq V$ a set of vertices. In randomized polynomial time, we can find a set $Z \subseteq V$ of $|Z| = O(|X|^{s+1})$ vertices such that for every partition of X into at most s parts, the set Z contains a minimum solution to the corresponding multiway cut problem.*

We also have the following further kernelization results; see [3] for problem statements.

Theorem 3 *The following problems admit randomized polynomial kernels parameterized by the solution size*: ALMOST 2-SAT, VERTEX MULTICUT *with a constant number of cut requests, and* GROUP FEEDBACK VERTEX SET *with a constant-sized group.*

Applications

We now review the strategy behind kernelization usage of the representative sets lemma.

Representative Sets: Direct Usage

There have been various types of applications of the representative sets lemma in kernelization, from the more direct to the more subtle. We briefly review one more direct and one indirect. The most direct one is for reducing *constraint systems*. We illustrate with the DIGRAPH PAIR CUT problem (which is closely related to a central problem in kernelization [3]). Let $G = (V, E)$ be a digraph, with a source vertex $s \in V$, and let $\mathcal{P} \subseteq V^2$ be a set of pairs. The task is to find a set X of at most k vertices (with $s \notin X$) such that $R_G(s, X)$ does not contain any complete pair from $\mathcal{P}$. We show that it suffices to keep $O(k^2)$ of the pairs $\mathcal{P}$. For this, replace s by a set S of $k + 1$ copies of s, and let M be the gammoid of (G, S, V). By Lemma 2, if X is closest to S and $|X| \leq k$, then $\{u, v\} \subseteq R_G(s, X)$ if and only if both $\{u\}$ and $\{v\}$ extend X in M. Hence, using the direct sum construction, we can construct a representative set $\mathcal{P}^* \subseteq \mathcal{P}$ with $|\mathcal{P}^*| = O(k^2)$ such that for any set X closest to S, the set $R_G(s, X)$ contains a pair $\{u, v\} \in \mathcal{P}$ if and only if it contains a pair $\{u', v'\} \in \mathcal{P}^*$. Furthermore, for an arbitrary set X, pushing X towards S yields a set X' that can only be an improvement on X (i.e., the set of pairs in $R_G(s, X)$ shrinks); hence for any set X with $|X| \leq k$, either pushing X towards S yields a solution to the problem, or there is a pair in $\mathcal{P}^*$ witnessing that X is not a solution. Thus, the set $\mathcal{P}^*$ may be used to replace $\mathcal{P}$, taking the first step towards a kernel for the problem.

Indirect Usage

For more advanced applications, we "force" the lemma to reveal some set Z of special vertices in G, as follows. Let M be a linearly represented matroid, and let $\mathcal{S} = \{S(v) : v \in V\}$ be a collection of subsets of M of bounded size. Assume that we have shown that for every $z \in Z$, there is a carefully chosen set $X(z)$, such that $S(v)$ extends $X(z)$ if and only if $v = z$. Then, necessarily, the representative set $\mathcal{S}^*$ for $\mathcal{S}$ must contain $S(z)$ for every $z \in Z$, by letting $X = X(z)$ in the statement of the lemma. Furthermore, we do not need to provide the set $X(z)$ ahead of time, since the (possibly non-constructive) *existence* of such a set $X(z)$ is sufficient to force $S(z) \in \mathcal{S}^*$. Hence, the set $V^* = \{v \in V : S(v) \in \mathcal{S}^*\}$ must contain Z, among a polynomially bounded number of other vertices. The critical challenge, of course, is to construct the matroid M and sets $S(v)$ and $X(z)$ such that $S(z)$ indeed extends $X(z)$, while $S(v)$ fails to extend $X(z)$ for every $v \neq z$.

We illustrate the application to reducing terminal cut systems. Let $G = (V, E)$ be an undirected graph (the directed construction is similar), with $S, T \subseteq V$, and define a set of vertices Z where $z \in Z$ if and only if there are sets $A \subseteq S, B \subseteq T$ such that every minimum (A, B)-vertex cut contains z. We wish to learn Z. Let a *sink-only copy* of a vertex $v \in V$ be a copy v' of v with all edges oriented towards v'. Then the following follows from Lemma 2 and the definition of closest sets.

Lemma 3 *Let $A, B \subseteq V$, and let X be a minimum (A, B)-vertex cut. Then a vertex $v \in V$ is a member of every minimum (A, B)-vertex cut if and only if $\{v'\}$ extends X in both the gammoid (G, A, V) and the gammoid (G, B, V).*

Via a minor modification, we can replace the former gammoid by the gammoid (G, S, V) and the latter by (G, T, V) (for appropriate adjustments to the set X); we can then compute a set V^* of $O(|S| \cdot |T| \cdot k)$ vertices (where k is the size of an (S, T)-min cut) which contains Z. From this, we may compute the sought-after smaller graph G', by iteratively bypassing a single vertex $v \in V \setminus (S \cup T \cup V^*)$ and recomputing V^*, until $V^* \cup S \cup T = V$; observe that bypassing v does not change the size of any (A, B)-min cut. Theorem 1 follows by considering a modification of the graph G, and Theorem 2 follows by a generalization of the above, pushing into s different directions.

Further Applications

A polynomial kernel for MULTIWAY CUT (in the variants with only s terminals or with deletable terminals) essentially follows from the above, but the further kernelization applications in Theorem 3 require a few more steps. However, they follow a common pattern: First, we find an approximate solution X of size poly(k) to "bootstrap" the process; second, we use X to transform the problem into a more manageable form (e.g., for ALMOST 2-SAT, this manageable form is DIGRAPH PAIR CUT); and lastly, we use the above methods to kernelize the resulting problem. This pattern covers the problems listed in Theorem 3.

Finally, the above results have some implications beyond kernelization. In particular, the existence of the smaller graph G' computed for terminal cut systems, and correspondingly an implementation of a gammoid as a graph with poly$(|S| + |T|)$ vertices, was an open problem, solved in [3].

Cross-References

- Kernelization, Exponential Lower Bounds
- Kernelization, Polynomial Lower Bounds
- Matroids in Parameterized Complexity and Exact Algorithms

Recommended Reading

1. Fomin FV, Lokshtanov D, Saurabh S (2014) Efficient computation of representative sets with applications in parameterized and exact algorithms. In: SODA, Portland, pp 142–151
2. Kratsch S, Wahlström M (2012) Compression via matroids: a randomized polynomial kernel for odd cycle transversal. In: SODA, Kyoto, pp 94–103
3. Kratsch S, Wahlström M (2012) Representative sets and irrelevant vertices: new tools for kernelization. In: FOCS, New Brunswick, pp 450–459

4. Lovász L (1977) Flats in matroids and geometric graphs. In: Proceedings of the sixth British combinatorial conference, combinatorial surveys, Egham, pp 45–86
5. Marx D (2009) A parameterized view on matroid optimization problems. Theor Comput Sci 410(44):4471–4479
6. Oxley J (2006) Matroid theory. Oxford graduate texts in mathematics. Oxford University Press, Oxford
7. Perfect H (1968) Applications of Menger's graph theorem. J Math Anal Appl 22:96–111
8. Schrijver A (2003) Combinatorial optimization: polyhedra and efficiency. Algorithms and combinatorics. Springer, Berlin/New York

Kernelization, Max-Cut Above Tight Bounds

Mark Jones
Department of Computer Science, Royal Holloway, University of London, Egham, UK

Keywords

Kernel; Lambda extendible; Max Cut; Parameterization above tight bound

Years and Authors of Summarized Original Work

2012; Crowston, Jones, Mnich

Problem Definition

In the problem MAX CUT, we are given a graph G with n vertices and m edges, and asked to find a bipartite subgraph of G with the maximum number of edges.

In 1973, Edwards [5] proved that if G is connected, then G contains a bipartite subgraph with at least $\frac{m}{2} + \frac{n-1}{4}$ edges, proving a conjecture of Erdős. This lower bound on the size of a bipartite subgraph is known as the *Edwards-Erdős bound*. The bound is tight – for example, it is an upper bound when G is a clique with odd number of vertices. Thus, it is natural to consider parameterized MAX CUT above this bound, as follows (AEE stands for *Above Edwards-Erdős*).

MAX CUT AEE

Instance: A connected graph G with n vertices and m edges, and a nonnegative integer k.
Parameter: k.
Question: Does G have a bipartite subgraph with at least $\frac{m}{2} + \frac{n-1}{4} + k$ edges?

Mahajan and Raman [6], in their first paper on above-guarantee parameterizations, asked whether this problem is fixed-parameter tractable. As such, the problem was one of the first open problems in above-guarantee parameterizations.

λ-Extendibility and the Poljak-Turzík Bound

In 1982, Poljak and Turzík [8] investigated extending the Edwards-Erdős bound to cases when the desired subgraph is something other than bipartite. To this end, they introduced the notion of λ-extendibility, which generalizes the notion of "bipartiteness." We will define the slightly stronger notion of *strong* λ-extendibility, introduced in [7], as later results use this stronger notion.

Recall that a *block* of a graph G is a maximal 2-connected subgraph of G. The blocks of a graph form a partition of its edges, and a vertex that appears in two or more blocks is a cut-vertex of the graph.

Definition 1 For a family of graphs Π and $0 \leq \lambda \leq 1$, we say Π is *s*trongly λ-extendible if the following conditions are satisfied:

1. If G is connected and $|G| = 1$ or 2, then $G \in \Pi$.
2. G is in Π if and only if each of its blocks is in Π.
3. For any real-valued positive weight function w on the edges of G, if $X \subseteq V(G)$ is such that $G[X]$ is connected and $G[X], G - X \in \Pi$, then G has a subgraph $H \in \Pi$ that uses all the edges of $G[X]$, all the edges of $G - X$, and at least a fraction λ (by weight) of the edges between X and $V(G) \setminus X$.

The definition of λ-extendibility given in [8] is the same as the above, except that the third condition is only required when $|X| = 2$. Clearly strong λ-extendibility implies λ-extendibility; it is an open question whether the converse holds.

The property of being bipartite is strongly λ-extendible for $\lambda = 1/2$. Other strongly λ-extendible properties include being acyclic for directed graphs ($\lambda = 1/2$) and being r-colorable ($\lambda = 1/r$).

Poljak and Turzík [8] extended Edwards' result by showing that for any connected graph G with n vertices and m edges, and any λ-extendible property Π, G contains a subgraph in Π with at least $\lambda m + \frac{1-\lambda}{2}(n-1)$ edges.

Thus, for any λ-extendible property Π, we can consider the following variation of MAX CUT AEE, for any λ-extendible Π (APT stands for *Above Poljak-Turzík*).

Π-SUBGRAPH APT

Instance: A connected graph G with n vertices and m edges, and a nonnegative integer k.
Parameter: k.
Question: Does G have a subgraph in Π with at least $\lambda m + \frac{1-\lambda}{2}(n-1) + k$ edges?

Key Results

We sketch a proof of the polynomial kernel result for MAX CUT AEE, first shown in [2] (although the method described here is slightly different to that in [2]).

For a connected graph G with n vertices and m edges, let $\beta(G)$ denote the maximum number of edges of a bipartite subgraph of G, let $\gamma(G) = \frac{m}{2} + \frac{n-1}{4}$, and let $\varepsilon(G) = \beta(G) - \gamma(G)$. Thus, for an instance (G, k), our aim is to determine whether $\varepsilon(G) \geq k$.

Now consider a connected graph G with a set X of three vertices such that $G' = G - X$ is connected and $G[X] = P_3$, the path with two edges. Note that $G[X]$ is bipartite. Let H' be a subgraph of G' with $\beta(G')$ edges. As bipartiteness is a $1/2$-extendible property, we can create a bipartite subgraph H of G using the edges of H', the edges of $G[X]$, and at least half of the edges between X and $G - X$. It follows that $\beta(G) \geq \beta(G') + \frac{|E(X,V(G)\setminus X)|}{2} + 2$. As $\gamma(G) = \gamma(G') + \frac{|E(X,V(G)\setminus X)|+2}{2} + \frac{3}{4}$, we have that $\varepsilon(G) = \beta(G) - \gamma(G) \geq \beta(G') - \gamma(G') + 2 - \frac{2}{2} - \frac{3}{4} = \varepsilon(G') + \frac{1}{4}$.

Consider a reduction rule in which, if there exists a set X as described above, we delete X from the graph. If we were able to apply such a reduction rule $4k$ times on a graph G, we would end up with a reduced graph G' such that G' is connected and $\varepsilon(G) \geq \varepsilon(G') + \frac{4k}{4} \geq 0 + k$, and therefore we would know that (G, k) is a YES-instance. Of course there may be many graphs for which such a set X cannot be found. However, we can adapt this idea as follows. Given a connected graph G, we recursively calculate a set of vertices $S(G)$ and a rational number $t(G)$ as follows:

- If G is a clique or G is empty, then set $S(G) = \emptyset$ and $t(G) = 0$.
- If G contains a set X such that $|X| = 3$, $G' = G - X$ is connected and $G[X] = P_3$, then set $S(G) = S(G') \cup X$ and set $t(G) = t(G') + \frac{1}{4}$.
- If G contains a cut-vertex v, then there exist non-empty sets of vertices X, Y such that $X \cap Y = \{v\}$, $G[X]$ and $G[Y]$ are connected, and all edges of G are in $G[X]$ or $G[Y]$. Then set $S(G) = S(G[X]) \cup S(G[Y])$ and set $t(G) = t(G[X]) + t(G[Y])$.

It can be shown that for a connected graph G, one of these cases will always hold, and so $S(G)$ and $t(G)$ are well defined. In the first case, we have that $\varepsilon(G) \geq 0$ by the Edwards-Erdős bound. In the second case, we have already shown that $\varepsilon(G) \geq \varepsilon(G') + \frac{1}{4}$. In the third case, we have that $\varepsilon(G) = \varepsilon(G[X]) + \varepsilon(G[Y])$ (note that the union of a bipartite subgraph of $G[X]$ and a bipartite subgraph of $G[Y]$ is a bipartite subgraph of G). It follows that $\varepsilon(G) \geq t(G)$. Note also that $|S(G)| \leq 12t(G)$. If we remove $S(G)$ from G, the resulting graph can be built by joining disjoint graphs at a single vertex, using only cliques as the initial graphs. Thus, $G - S(G)$ has the property

that each of its blocks is a clique. We call such a graph a *forest of cliques*.

We therefore get the following lemma.

Lemma 1 ([2]) *Given a connected graph G with n vertices and m edges, and an integer k, we can in polynomial time either decide that (G, k) is a* YES*-instance of* MAX CUT AEE*, or find a set S of at most $12k$ vertices such that $G - S$ is a forest of cliques.*

By guessing a partition of S and then using a dynamic programming algorithm based on the structure of $G - S$, we get a fixed-parameter algorithm.

Theorem 1 ([2]) MAX CUT AEE *can be solved in time $2^{O(k)} \cdot n^4$.*

Using the structure of $G - S$ and the fact that $|S| \leq 12k$, it is possible (using reduction rules) to show first that the number of blocks in $G - S$ must be bounded for any NO-instance, and then that the size of each block must be bounded (see [2]).

Thus, we get a polynomial kernel for MAX CUT AEE.

Theorem 2 ([2]) MAX CUT AEE *admits a kernel with $O(k^5)$ vertices.*

Crowston et al. [3] were later able to improve this to a kernel with $O(k^3)$ vertices.

Extensions to Π-SUBGRAPH APT

A similar approach can be used to show polynomial kernels for Π-SUBGRAPH APT, for other $1/2$-extendible properties. In particular, the property of being an acyclic directed graph is $1/2$-extendible, and therefore every directed graph with n vertices and m arcs has an acyclic subgraph with at least $\frac{m}{2} + \frac{n-1}{4}$ arcs. The problem of deciding whether there exists an acyclic subgraph with at least $\frac{m}{2} + \frac{n-1}{4} + k$ arcs is fixed-parameter tractable, and has a $O(k^2)$-vertex kernel [1].

The notion of a bipartite graph can be generalized in the following way. Consider a graph G with edges labeled either $+$ or $-$. Then we say G is *balanced* if there exists a partition V_1, V_2 of the vertices of G, such that all edges between V_1 and V_2 are labeled $-$ and all other edges are labeled $+$. (Note that if all edges of a graph are labeled $-$, then it is balanced if and only if it is bipartite.) The property of being a balanced graph is $1/2$-extendible, just as the property of being bipartite is. Therefore a graph with n vertices and m vertices, and all edges labeled $+$ or $-$, will have a balanced subgraph with at least $\frac{m}{2} + \frac{n-1}{4}$ edges. The problem of deciding whether there exists a balanced subgraph with at least $\frac{m}{2} + \frac{n-1}{4} + k$ edges is fixed-parameter tractable and has a $O(k^3)$-vertex kernel [3].

Mnich et al. [7] showed that Lemma 1 applies not just for MAX CUT AEE, but for Π-SUBGRAPH APT for any Π which is strongly λ-extendible for some λ (with the bound $12k$ replaced with $\frac{6k}{1-\lambda}$). Thus, Π-SUBGRAPH APT is fixed-parameter tractable as long as it is fixed-parameter tractable on graphs which are close to being a forest of cliques. Using this observation, Mnich et al. showed fixed-parameter tractability for a number of versions of Π-SUBGRAPH APT, including when Π is the family of acyclic directed graphs and when Π is the set of r-colorable graphs.

Crowston et al. [4] proved the existence of polynomial kernels for a wide range of strongly λ-extendible properties:

Theorem 3 ([4]) *Let $0 < \lambda < 1$, and let Π be a strongly λ-extendible property of (possibly oriented and/or labeled) graphs. Then Π-*SUBGRAPH APT *has a kernel on $O(k^2)$ vertices if Condition 1 or 2 holds, and a kernel on $O(k^3)$ vertices if only Condition 3 holds:*

1. *$\lambda \neq \frac{1}{2}$.*
2. *All orientations and labels (if applicable) of the graph K_3 belong to Π.*
3. *Π is a hereditary property of simple or oriented graphs.*

Open Problems

The Poljak-Turzík's bound extends to edge-weighted graphs. The weighted version is as follows: for a graph G with nonnegative real weights on the edges, and a λ-extendible family

of graphs Π, there exists a subgraph H of G such that $H \in \Pi$ and H has total weight $\lambda \cdot w(G) + \frac{1-\lambda}{2} \cdot MST(G)$, where $w(G)$ is the total weight of G and $MST(G)$ is the minimum weight of a spanning tree in G.

Thus, we can consider the weighted versions of MAX CUT AEE and Π-SUBGRAPH APT. It is known that a weighted equivalent of Lemma 1 holds (in which all edges in a block of $G-S$ have the same weight), and as a result, the integer-weighted version of MAX CUT AEE can be shown to be fixed-parameter tractable. However, nothing is known about kernelization results for these problems. In particular, it remains an open question whether the integer-weighted version of MAX CUT AEE has a polynomial kernel.

Cross-References

- Kernelization, Constraint Satisfaction Problems Parameterized above Average
- Kernelization, MaxLin Above Average

Recommended Reading

1. Crowston R, Gutin G, Jones M (2012) Directed acyclic subgraph problem parameterized above the Poljak-Turzík bound. In: FSTTCS 2012, Hyderabad. LIPICS, vol 18, pp 400–411
2. Crowston R, Jones M, Mnich M (2012) Max-Cut parameterized above the Edwards-Erdős bound. In: ICALP 2012, Warwick. Lecture notes in computer science, vol 7391, pp 242–253
3. Crowston R, Gutin G, Jones M, Muciaccia G (2013) Maximum balanced subgraph problem parameterized above lower bounds. Theor Comput Sci 513:53–64
4. Crowston R, Jones M, Muciaccia G, Philip G, Rai A, Saurabh S (2013) Polynomial kernels for λ-extendible properties parameterized above the Poljak-Turzík bound. In: FSTTCS 2013, Guwahati. LIPICS, vol 24, pp 43–54
5. Edwards CS (1973) Some extremal properties of bipartite subgraphs. Can J Math 25:475–485
6. Mahajan M, Raman V (1999) Parameterizing above guaranteed values: MaxSat and MaxCut. J Algorithms 31(2):335–354
7. Mnich M, Philip G, Saurabh S, Suchý O (2014) Beyond Max-Cut: λ-extendible properties parameterized above the Poljak-Turzík bound. J Comput Syst Sci 80(7):1384–1403
8. Poljak S, Turzík D (1982) A polynomial algorithm for constructing a large bipartite subgraph, with an application to a satisfiability problem. Can J Math 34(4):519–524

Kernelization, MaxLin Above Average

Anders Yeo
Engineering Systems and Design, Singapore University of Technology and Design, Singapore, Singapore
Department of Mathematics, University of Johannesburg, Auckland Park, South Africa

Keywords

Fixed-parameter tractability above lower bounds; Kernelization; Linear equations; M-sum-free sets

Years and Authors of Summarized Original Work

2010; Crowston, Gutin, Jones, Kim, Ruzsa
2011; Gutin, Kim, Szeider, Yeo
2014; Crowston, Fellows, Gutin, Jones, Kim, Rosamond, Ruzsa, Thomassé, Yeo

Problem Definition

The problem MAXLIN2 can be stated as follows. We are given a system of m equations in variables $x_1, \ldots, x_n$ where each equation is $\prod_{i \in I_j} x_i = b_j$, for some $I_j \subseteq \{1, 2, \ldots, n\}$ and $x_i, b_j \in \{-1, 1\}$ and $j = 1, \ldots, m$. Each equation is assigned a positive integral weight w_j. We are required to find an assignment of values to the variables in order to maximize the total weight of the satisfied equations. MAXLIN2 is a well-studied problem, which according to Håstad [8] "is as basic as satisfiability."

Note that one can think of MAXLIN2 as containing equations, $\sum_{i \in I_j} y_i = a_j$ over $\mathbb{F}_2$. This is equivalent to the previous definition by letting $y_i = 0$ if and only if $x_i = 1$ and letting $y_i = 1$ if and only if $x_i = -1$ (and $a_j = 1$ if and only if $b_j = -1$ and $a_j = 0$ if and only if $b_j = 1$). We will however use the original definition as this was the formulation used in [1].

Let W be the sum of the weights of all equations in an instance, S, of MAXLIN2 and let $\text{sat}(S)$ be the maximum total weight of equations that can be satisfied simultaneously. To see that $W/2$ is a tight lower bound on $\text{sat}(S)$, choose assignments to the variables independently and uniformly at random. Then $W/2$ is the expected weight of satisfied equations (as the probability of each equation being satisfied is $1/2$) and thus $W/2$ is a lower bound. It is not difficult to see that this bound is tight. For example, consider a system consisting of pairs of equations of the form $\prod_{i \in I} x_i = -1$, $\prod_{i \in I} x_i = 1$ of the same weight, for some nonempty sets $I \subseteq \{1, 2, \ldots, n\}$.

As MAXLIN2 is an NP-hard problem, we look for parameterized algorithms. We will give the basic definitions of fixed-parameter tractability (FPT) here and refer the reader to [4, 5] for more information. A *parameterized problem* is a subset $L \subseteq \Sigma^* \times \mathbb{N}$ over a finite alphabet Σ. L is *fixed-parameter tractable* (FPT, for short) if membership of an instance (x, k) in $\Sigma^* \times \mathbb{N}$ can be decided in time $f(k)|x|^{O(1)}$, where f is a function of the parameter k only.

If we set the parameter, k, of an instance, S, of MAXLIN2 to $\text{sat}(S)$, then it is easy to see that there exists an $O(f(k)|S|^c)$ algorithm, due to the fact that $k = \text{sat}(S) \geq W/2 \geq |S|/2$. Therefore, this parameter is not of interest (it is never small in practice), and a better parameter would be k, where we want to decide if $\text{sat}(S) \geq W/2 + k$. Parameterizing above tight lower bounds in this way was first introduced in 1997 in [11]. This leads us to define the following problem, where AA stands for *Above Average*.

MAXLIN2-AA

Instance: A system S of equations $\prod_{i \in I_j} x_i = b_j$, where $x_i, b_j \in \{-1, 1\}$, $j = 1, \ldots, m$ and where each equation is assigned a positive integral weight w_j and a nonnegative integer k.

Question: $\text{sat}(S) \geq W/2 + k$?

The above problem has also been widely studied when the number of variables in each equation is bounded by some constant, say r, which leads to the following problem.

MAX-r-LIN2-AA

Instance: A system S of equations $\prod_{i \in I_j} x_i = b_j$, where $x_i, b_j \in \{-1, 1\}$, $|I_j| \leq r$, $j = 1, \ldots, m$; equation j is assigned a positive integral weight w_j and a nonnegative integer k.

Question: $\text{sat}(S) \geq W/2 + k$?

Given a parameterized problem, Π, a *kernel* of Π is a polynomial-time algorithm that maps an instance (I, k) of Π to another instance, (I', k'), of Π such that (i) $(I, k) \in \Pi$ if and only if $(I', k') \in \Pi'$, (ii) $k' \leq f(k)$, and (iii) $|I'| \leq g(k)$ for some functions f and g. The function $g(k)$ is called the *size* of the kernel. It is well known that a problem is FPT if and only if it has a kernel.

A kernel is called a polynomial kernel if both $f(k)$ and $g(k)$ are polynomials in k. A great deal of research has been devoted to finding small-sized kernels and in particular to decide if a problem has a polynomial kernel.

We will show that both the problems stated above are FPT and in fact contain kernels with a polynomial number of variables. The number of equations may be non-polynomial, so these kernels are not real polynomial kernels. The above problems were investigated in a number of papers; see [1–3, 6].

Key Results

We will below outline the key results for both MaxLin2-AA and Max-r-Lin2-AA. See [1] for all the details not given here.

MaxLin2-AA

Recall that MaxLin2-AA considers a system S of equations $\prod_{i\in I_j} x_i = b_j$, where $x_i, b_j \in \{-1, 1\}$, $j = 1, \ldots, m$ and where each equation is assigned a positive integral weight w_j. Let $\mathcal{F}$ denote the m different sets I_j in the equations of S and let $b_{I_j} = b_j$ and $w_{I_j} = w_j$ for each $j = 1, 2, \ldots, m$.

Let $\varepsilon(x) = \sum_{I\in\mathcal{F}} w_I b_I \prod_{i\in I} x_i$ and note that $\varepsilon(x)$ is the difference between the total weight of satisfied and falsified equations. Crowston et al. [3] call $\varepsilon(x)$ the *excess* and the maximum possible value of $\varepsilon(x)$ the *maximum excess*.

Remark 1 Observe that the answer to MaxLin2-AA and Max-r-Lin2-AA is YES if and only if the maximum excess is at least $2k$.

Let A be the matrix over $\mathbb{F}_2$ corresponding to the set of equations in S, such that $a_{ji} = 1$ if $i \in I_j$ and 0, otherwise. Consider the following two reduction rules, where Rule 1 was introduced in [9] and Rule 2 in [6].

Reduction Rule 1 ([9]) *If we have, for a subset I of $\{1, 2, \ldots, n\}$, an equation $\prod_{i\in I} x_i = b'_I$ with weight w'_I, and an equation $\prod_{i\in I} x_i = b''_I$ with weight w''_I, then we replace this pair by one of these equations with weight $w'_I + w''_I$ if $b'_I = b''_I$ and, otherwise, by the equation whose weight is bigger, modifying its new weight to be the difference of the two old ones. If the resulting weight is 0, we delete the equation from the system.*

Reduction Rule 2 ([6]) *Let $t = \mathrm{rank}A$ and suppose columns $a^{i_1}, \ldots, a^{i_t}$ of A are linearly independent. Then delete all variables not in $\{x_{i_1}, \ldots, x_{i_t}\}$ from the equations of S.*

Lemma 1 ([6]) *Let S' be obtained from S by Rule 1 or 2. Then the maximum excess of S' is equal to the maximum excess of S. Moreover, S' can be obtained from S in time polynomial in n and m.*

If we cannot change a weighted system S using Rules 1 and 2, we call it *irreducible*. Let S be an irreducible system of MaxLin2-AA. Consider the following algorithm introduced in [3]. We assume that, in the beginning, no equation or variable in S is marked.

ALGORITHM $\mathcal{H}$
While the system S is nonempty, do the following:

1. Choose an equation $\prod_{i\in I} x_i = b$ and mark a variable x_l such that $l \in I$.
2. Mark this equation and delete it from the system.
3. Replace every equation $\prod_{i\in I'} x_i = b'$ in the system containing x_l by $\prod_{i\in I\Delta I'} x_i = bb'$, where $I\Delta I'$ is the symmetric difference of I and I' (the weight of the equation is unchanged).
4. Apply Reduction Rule 1 to the system.

The *maximum $\mathcal{H}$-excess* of S is the maximum possible total weight of equations marked by $\mathcal{H}$ for S taken over all possible choices in Step 1 of $\mathcal{H}$. The following lemma indicates the potential power of $\mathcal{H}$.

Lemma 2 ([3]) *Let S be an irreducible system. Then the maximum excess of S equals its maximum $\mathcal{H}$-excess.*

Theorem 1 ([1]) *There exists an $O(n^{2k}(nm)^{O(1)})$-time algorithm for* MaxLin2-AA$[k]$ *that returns an assignment of excess of at least $2k$ if one exists, and returns* NO *otherwise.*

In order to prove the above, the authors pick n equations $e_1, \ldots, e_n$ such that their rows in A are linearly independent. An assignment of excess at least $2k$ must either satisfy one of these equations or falsify them all. If they are all falsified, then the value of all variables is completely determined. Thus, by Lemma 2, algorithm $\mathcal{H}$ can mark one of these equations, implying a search tree of depth at most $2k$ and width at most k. This implies the desired time bound.

Theorem 2 below is proved using M-sum-free sets, which are defined as follows (see [3]). Let K and M be sets of vectors in $\mathbb{F}_2^n$ such that $K \subseteq M$. We say K is *M-sum-free* if no sum of two or more distinct vectors in K is equal to a vector in M.

Theorem 2 ([1]) *Let S be an irreducible system of* MaxLin2-AA$[k]$ *and let $k \geq 1$. If $2k \leq m \leq \min\{2^{n/(2k-1)} - 1, 2^n - 2\}$, then the maximum excess of S is at least $2k$. Moreover, we can find an assignment with excess of at least $2k$ in time $O(m^{O(1)})$.*

Using the above, we can solve the problem when $2k \leq m \leq 2^{n/(2k-1)} - 2$ and when $m \geq n^{2k} - 1$ (using Theorem 1). The case when $m < 2k$ immediately gives a kernel and the remaining case when $2^{n/(2k-1)} - 2 \leq m \leq n^{2k} - 2$ can be shown to imply that $n \in O(k^2 \log k)$, thereby giving us the main theorem and corollary of this section.

Theorem 3 ([1]) *The problem* MaxLin2-AA$[k]$ *has a kernel with at most $O(k^2 \log k)$ variables.*

Corollary 1 ([1]) *The problem* MaxLin2-AA$[k]$ *can be solved in time $2^{O(k \log k)}(nm)^{O(1)}$.*

Max-r-Lin2-AA

In [6] it was proved that the problem Max-r-Lin2-AA admits a kernel with at most $O(k^2)$ variables and equations (where r is treated as a constant). The bound on the number of variables can be improved and it was done by Crowston et al. [3] and Kim and Williams [10]. The best known improvement is by Crowston et al. [1].

Theorem 4 ([1]) *The problem* Max-r-Lin2-AA *admits a kernel with at most $(2k - 1)r$ variables.*

Both Theorem 4 and a slightly weaker analogous result of the results in [10] imply the following:

Lemma 3 ([1,10]) *There is an algorithm of runtime $2^{O(k)} + m^{O(1)}$ for* Max-r-Lin2-AA.

Kim and Williams [10] proved that the last result is best possible, in a sense, if the exponential time hypothesis holds.

Theorem 5 ([10]) *If* Max-3-Lin2-AA *can be solved in $O(2^{\epsilon k}2^{\epsilon m})$ time for every $\epsilon > 0$, then 3-SAT can be solved in $O(2^{\delta n})$ time for every $\delta > 0$, where n is the number of variables.*

Open Problems

The kernel for MaxLin2-AA contains at most $O(k^2 \log k)$ variables, but may contain an exponential number of equations. It would be of interest to decide if MaxLin2-AA admits a kernel that has at most a polynomial number of variables and equations.

Cross-References

- Kernelization, Constraint Satisfaction Problems Parameterized above Average
- Kernelization, Permutation CSPs Parameterized above Average

Recommended Reading

1. Crowston R, Fellows M, Gutin G, Jones M, Kim EJ, Rosamond F, Ruzsa IZ, Thomassé S, Yeo A (2014) Satisfying more than half of a system of linear equations over GF(2): a multivariate approach. J Comput Syst Sci 80(4):687–696
2. Crowston R, Gutin G, Jones M (2010) Note on Max Lin-2 above average. Inform Proc Lett 110:451–454
3. Crowston R, Gutin G, Jones M, Kim EJ, Ruzsa I (2010) Systems of linear equations over $\mathbb{F}_2$ and problems parameterized above average. In: SWAT 2010, Bergen. Lecture notes in computer science, vol 6139, pp 164–175
4. Downey RG, Fellows MR (2013) Fundamentals of parameterized complexity. Springer, London/Heidelberg/New York
5. Flum J, Grohe M (2006) Parameterized complexity theory. Springer, Berlin
6. Gutin G, Kim EJ, Szeider S, Yeo A (2011) A probabilistic approach to problems parameterized above or below tight bounds. J Comput Syst Sci 77:422–429
7. Gutin G, Yeo A (2012) Constraint satisfaction problems parameterized above or below tight bounds: a survey. Lect Notes Comput Sci 7370:257–286

8. Håstad J (2001) Some optimal inapproximability results. J ACM 48:798–859
9. Håstad J, Venkatesh S (2004) On the advantage over a random assignment. Random Struct Algorithms 25(2):117–149
10. Kim EJ, Williams R (2012) Improved parameterized algorithms for above average constraint satisfaction. In: IPEC 2011, Saarbrücken. Lecture notes in computer science, vol 7112, pp 118–131
11. Mahajan M, Raman V (1999) Parameterizing above guaranteed values: MaxSat and MaxCut. J Algorithms 31(2):335–354. Preliminary version in Electr. Colloq. Comput. Complex. (ECCC), TR-97-033, 1997

Kernelization, Partially Polynomial Kernels

Christian Komusiewicz
Institute of Software Engineering and Theoretical Computer Science, Technical University of Berlin, Berlin, Germany

Keywords

Data reduction; Fixed-parameter algorithms; Kernelization; NP-hard problems

Years and Authors of Summarized Original Work

2011; Betzler, Guo, Komusiewicz, Niedermeier
2013; Basavaraju, Francis, Ramanujan, Saurabh
2014; Betzler, Bredereck, Niedermeier

Problem Definition

In parameterized complexity, each instance (I, k) of a problem comes with an additional parameter k which describes structural properties of the instance, for example, the maximum degree of an input graph. A problem is called *fixed-parameter tractable* if it can be solved in $f(k)\cdot\text{poly}(n)$ time, that is, the super-polynomial part of the running time depends only on k. Consequently, instances of the problem can be solved efficiently if k is small.

One way to show fixed-parameter tractability of a problem is the design of a polynomial-time data reduction algorithm that reduces any input instance (I, k) to one whose size is bounded in k. This idea is captured by the notion of kernelization.

Definition 1 Let (I, k) be an instance of a parameterized problem P, where $I \in \Sigma^*$ denotes the input instance and $k \in \mathbb{N}$ is a parameter. Problem P admits a *problem kernel* if there is a polynomial-time algorithm, called *problem kernelization*, that computes an instance (I', k') of the same problem P such that:

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance, and
- $|I'| + k' \leq g(k)$

for a function g of k only.

Kernelization gives a performance guarantee for the effectiveness of data reduction: instances (I, k) with $|I| > g(k)$ are provably reduced to smaller instances. Thus, one aim in the design of kernelization algorithms is to make the function g as small as possible. In particular, one wants to obtain kernelizations where g is a polynomial function. These algorithms are called *polynomial problem kernelizations*.

For many parameterized problems, however, the existence of such a polynomial problem kernelization is considered to be unlikely (under a standard complexity-theoretic assumption) [4]. Consequently, alternative models of parameterized data reduction, for example Turing kernelization, have been proposed.

The concept of *partial kernelization* offers a further approach to obtain provably useful data reduction algorithms. Partial kernelizations do not aim for a decrease of the instance *size* but for a decrease of some *part* or *dimension* of the instance. For example, if the problem input is a binary matrix with m rows and n columns, the instance size is $\Theta(n\cdot m)$. A partial kernelization can now aim for reducing one dimension of the input, for example the number of rows n. Of course,

such a reduction is worthwhile only if we can algorithmically exploit the fact that the number of rows n is small. Hence, the aim is to reduce a dimension of the problem for which there are fixed-parameter algorithms. The dimension can thus be viewed as a secondary parameter.

Altogether, this idea is formalized as follows.

Definition 2 Let (I, k) be an instance of a parameterized problem P, where $I \in \Sigma^*$ denotes the input instance and k is a parameter. Let $d : \Sigma^* \to \mathbb{N}$ be a computable function such that P is fixed-parameter tractable with respect to $d(I)$. Problem P admits a *partial problem kernel* if there is a polynomial-time algorithm, called *partial problem kernelization*, that computes an instance (I', k') of the same problem such that:

- (I, k) is a yes-instance if and only if (I', k') is a yes-instance, and
- $d(I') + k' \leq g(k)$

for a computable function g.

Any parameterized problem P which has a partial kernel for some appropriate dimension d is fixed-parameter tractable with respect to k: First, one may reduce the original input instance (I, k) to the partial kernel (I', k'). In this partial kernel, we have $d(I') \leq g(k)$ and, since P can be solved in $f(d(I')) \cdot \text{poly}(n)$ time, it can thus be solved in $f(g(k)) \cdot \text{poly}(n)$ time.

Using partial problem kernelization instead of classic problem kernelization can be motivated by the following two arguments.

First, the function d in the partial problem kernelization gives us a different goal in the design of efficient data reduction rules. For instance, if the main parameter determining the hardness of a graph problem is the maximum degree, then an algorithm that produces instances whose maximum degree is $O(k)$ but whose size is unbounded might be more useful than an algorithm that produces instances whose size is $O(k^4)$ but the maximum degree is $\Omega(k^2)$.

Second, if the problem does not admit a polynomial-size problem kernel, then it might still admit a *partially polynomial kernel*, that is, a partial kernel in which $d(I') + k' \leq \text{poly}(k)$.

We now give two examples for applications of partial kernelizations.

Key Results

The partial kernelization concept was initially developed to obtain data reduction algorithms for consensus problems, where one is given a collection of combinatorial objects and one is asked to find one object that represents this collection [2].

In the KEMENY SCORE problem, these objects are permutations of a set U and the task is to find a permutation that is close to these permutations with respect to what is called *Kendall's Tau distance*, here denoted by τ. The formal definition of the (unparameterized) problem is as follows.

Input: A multiset $\mathcal{P}$ of permutations of a ground set U and an integer ℓ.
Question: Is there a permutation P such that $\sum_{P' \in \mathcal{P}} \tau(P, P') \leq \ell$?

The parameter k under consideration is the average distance between the input partitions, that is,

$$k := \sum_{\{P, P'\} \subseteq \mathcal{P}} \tau(P, P') / \binom{|\mathcal{P}|}{2}.$$

Observe that, since τ can be computed efficiently, KEMENY SCORE is fixed-parameter tractable with respect to $|U|$: try all possible permutations of U and choose the best one. Hence, if U is small, then the problem is easy. Furthermore, the number of input permutations is not such a crucial feature since KEMENY SCORE is already NP-hard for a constant number of permutations; the partial kernelization thus aims for a reduction of $|U|$ and ignores the—less important—number of input permutations.

This reduction is obtained by removing elements in U that are, compared to the other elements, in roughly the same position in many input permutations. The idea is based on a

generalization of the following observation: If some element u is the first element of at least $3|\mathcal{P}|/4$ input permutations, then this element is the first element of an optimal partition. Any instance containing such an element u can thus be reduced to an equivalent with one less element.

By removing such elements, one obtains a sub-instance of the original instance in which every element contributes a value of $16/3$ to the average distance k between the input permutations. This leads to the following result.

Theorem 1 ([3]) KEMENY SCORE *admits a partial kernel with* $|U| \leq 16/3k$.

Further partial kernelizations for consensus problems have been obtained for CONSENSUS CLUSTERING [2, 7] and SWAP MEDIAN PARTITION [2], the partial kernelization for KEMENY SCORE has been experimentally evaluated [3].

Another application of partial kernelization has been proposed for covering problems such as SET COVER [1].

Input: A family $\mathcal{S}$ of subsets of a ground set U.
Question: Is there a subfamily $\mathcal{S}' \subseteq \mathcal{S}$ of size at most ℓ such that every element in U is contained in at least one set of $\mathcal{S}'$?

If $\ell \geq |U|$, then SET COVER has a trivial solution. Thus, a natural parameter is the amount that can be saved compared to this trivial solution, that is, $k := |U| - \ell$. A polynomial problem kernelization for SET COVER parameterized by k is deemed unlikely, again under standard complexity-theoretic assumptions. There is, however, a partially polynomial problem kernel. The dimension d is the universe size $|U|$. SET COVER is fixed-parameter tractable with respect to $|U|$ as it can be solved in $f(|U|) \cdot \text{poly}(n)$ time, for example, by dynamic programming.

The idea behind the partial kernelization is to greedily compute a subfamily $\mathcal{T} \subseteq \mathcal{S}$ of size k. Then, it is observed that either this subfamily has a structure that can be used to efficiently compute a solution of the problem, or $|U| \leq 2k^2 - 2$, or there are elements in U whose removal yields an equivalent instance. Altogether this leads to the following.

Theorem 2 ([1]) SET COVER *admits a partial problem kernel with* $|U| \leq 2k^2 - 2$.

Open Problems

The notion of partial kernelization is quite recent. Hence, the main aim for the near future is to identify further useful applications of the technique. We list some problem areas that contain natural candidates for such applications. Problems that are defined on set families, such as SET COVER, have two obvious dimensions: the number m of sets in the set family and the size n of the universe. Matrix problems also have two obvious dimensions: the number m of rows and the number n of columns. For graph problems, useful dimensions could be identified by examining the so-called parameter hierarchy [6, 8]. Here, the idea is to find dimensions whose value can be much smaller than the number of vertices in the graph. If the size $|I|$ of the instance cannot be reduced to be smaller than $\text{poly}(k)$, then this might be still possible for the smaller dimension $d(I)$. A further interesting research direction could be to study the relationship between partial kernelization and other relaxed notions of kernelization such as Turing kernelization.

For some problems, the existence of partially polynomial kernels has been proven, but it is still unknown whether polynomial kernels exist. One such example is MAXLIN2-AA [5].

Cross-References

- Kernelization, MaxLin Above Average
- Kernelization, Polynomial Lower Bounds
- Kernelization, Turing Kernels

Recommended Reading

1. Basavaraju M, Francis MC, Ramanujan MS, Saurabh S (2013) Partially polynomial kernels for set cover and test cover. In: FSTTCS '13, Guwahati. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, LIPIcs, vol 24, pp 67–78
2. Betzler N, Guo J, Komusiewicz C, Niedermeier R (2011) Average parameterization and partial kernelization for computing medians. J Comput Syst Sci 77(4):774–789

3. Betzler N, Bredereck R, Niedermeier R (2014) Theoretical and empirical evaluation of data reduction for exact Kemeny rank aggregation. Auton Agents Multi-Agent Syst 28(5):721–748
4. Bodlaender HL, Downey RG, Fellows MR, Hermelin D (2009) On problems without polynomial kernels. J Comput Syst Sci 75(8):423–434
5. Crowston R, Fellows M, Gutin G, Jones M, Kim EJ, Rosamond F, Ruzsa IZ, Thomassé S, Yeo A (2014) Satisfying more than half of a system of linear equations over GF(2): a multivariate approach. J Comput Syst Sci 80(4):687–696
6. Fellows MR, Jansen BMP, Rosamond FA (2013) Towards fully multivariate algorithmics: parameter ecology and the deconstruction of computational complexity. Eur J Comb 34(3): 541–566
7. Komusiewicz C (2011) Parameterized algorithmics for network analysis: clustering & querying. PhD thesis, Technische Universität Berlin, Berlin
8. Komusiewicz C, Niedermeier R (2012) New races in parameterized algorithmics. In: MFCS '12, Bratislava. Lecture notes in computer science, vol 7464. Springer, pp 19–30

Kernelization, Permutation CSPs Parameterized above Average

Gregory Gutin
Department of Computer Science, Royal Holloway, University of London , Egham, UK

Keywords

Betweenness; Linear ordering; Parameterization above tight bound

Years and Authors of Summarized Original Work

2012; Gutin, van Iersel, Mnich, Yeo

Problem Definition

Let r be an integer and let V be a set of n variables. An *ordering* α is a bijection from V to $\{1, 2, \ldots, n\}$; a *constraint* is an ordered r-tuple $(v_1, v_2, \ldots, v_r)$ of distinct variables of V; α *satisfies* $(v_1, v_2, \ldots, v_r)$ if $\alpha(v_1) < \alpha(v_2) < \cdots < \alpha(v_r)$. An instance of MAX-$r$-LIN-ORDERING consists of a multiset $\mathcal{C}$ of constraints, and the objective is to find an ordering that satisfies the maximum number of constraints. Note that MAX-2-LIN ORDERING is equivalent to the problem of finding a maximum weight acyclic subgraph in an integer-weighted directed graph. Since the FEEDBACK ARC SET problem is NP-hard, MAX-2-LIN ORDERING is NP-hard, and thus MAX-r-LIN-ORDERING is NP-hard for each $r \geq 2$.

Let α be an ordering chosen randomly and uniformly from all orderings and let $c \in \mathcal{C}$ be a constraint. Then the probability that α satisfies c is $1/r!$. Thus the expected number of constraints in $\mathcal{C}$ satisfied by α equals $|\mathcal{C}|/r!$. This is a lower bound on the maximum number of constraints satisfied by an ordering, and, in fact, it is a tight lower bound. This allows us to consider the following parameterized problem (AA stands for *Above Average*).

MAX-r-LIN-ORDERING-AA

Instance: A multiset $\mathcal{C}$ of constraints and a nonnegative integer k.
Parameter: k.
Question: Is there an ordering satisfying at least $|\mathcal{C}|/r! + k$ constraints?

$(1, 2, \ldots, r)$ is the identity permutation of the symmetric group $\mathcal{S}_r$. We can extend MAX-r-LIN-ORDERING by considering an arbitrary subset of $\mathcal{S}_r$ rather than just $\{(1, 2, \ldots, r)\}$. Instead of describing the extension for each arity $r \geq 2$, we will do it only for $r = 3$, which is our main interest, and leave the general case to the reader.

Let $\Pi \subseteq \mathcal{S}_3 = \{(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)\}$ be arbitrary. For an ordering $\alpha\colon V \to \{1, 2, \ldots, n\}$, a constraint $(v_1, v_2, v_3) \in \mathcal{C}$ is *Π-satisfied by α* if there is a permutation $\pi \in \Pi$ such that $\alpha(v_{\pi(1)}) < \alpha(v_{\pi(2)}) < \alpha(v_{\pi(3)})$. Given Π, the problem Π-CSP is the problem of deciding if there exists an ordering of V that Π-satisfies all the constraints. Every such problem is called a Permutation CSP of arity 3. We will consider the maximization version of these problems, denoted by MAX-Π-

Kernelization, Permutation CSPs Parameterized above Average, Table 1 Permutation CSPs of arity 3 (after symmetry considerations)

$\Pi \subseteq \mathcal{S}_3$	Name	Complexity
$\Pi_0 = \{(123)\}$	3-LIN-ORDERING	Polynomial
$\Pi_1 = \{(123), (132)\}$		Polynomial
$\Pi_2 = \{(123), (213), (231)\}$		Polynomial
$\Pi_3 = \{(132), (231), (312), (321)\}$		Polynomial
$\Pi_4 = \{(123), (231)\}$		NP-comp.
$\Pi_5 = \{(123), (321)\}$	BETWEENNESS	NP-comp.
$\Pi_6 = \{(123), (132), (231)\}$		NP-comp.
$\Pi_7 = \{(123), (231), (312)\}$	CIRCULAR ORDERING	NP-comp.
$\Pi_8 = \mathcal{S}_3 \setminus \{(123), (231)\}$		NP-comp.
$\Pi_9 = \mathcal{S}_3 \setminus \{(123), (321)\}$	NON-BETWEENNESS	NP-comp.
$\Pi_{10} = \mathcal{S}_3 \setminus \{(123)\}$		NP-comp.

CSP, parameterized above the average number of constraints satisfied by a random ordering of V (which can be shown to be a tight bound).

It is easy to see that there is only one distinct Π-CSP of arity 2. Guttmann and Maucher [5] showed that there are in fact only 13 distinct Π-CSPs of arity 3 up to symmetry, of which 11 are nontrivial. They are listed in Table 1 together with their complexity. Some of the problems listed in the table are well known and have special names. For example, the problem for $\Pi = \{(123), (321)\}$ is called the BETWEENNESS problem.

Gutin et al. [4] proved that all 11 nontrivial MAX-Π-CSP problems are NP-hard (even though four of the Π-CSP are polynomial).

Now observe that given a variable set V and a constraint multiset $\mathcal{C}$ over V, for a random ordering α of V, the probability of a constraint in $\mathcal{C}$ being Π-satisfied by α equals $\frac{|\Pi|}{6}$. Hence, the expected number of satisfied constraints from $\mathcal{C}$ is $\frac{|\Pi|}{6}|\mathcal{C}|$, and thus there is an ordering α of V satisfying at least $\frac{|\Pi|}{6}|\mathcal{C}|$ constraints (and this bound is tight). A derandomization argument leads to $\frac{|\Pi|}{6}$-approximation algorithms for the problems MAX-Π-CSP [1]. No better constant factor approximation is possible assuming the Unique Games Conjecture [1].

We will study the parameterization of MAX-Π-CSP above tight lower bound:

Π-ABOVE AVERAGE (Π-AA)

Instance: A finite set V of variables, a multiset $\mathcal{C}$ of ordered triples of distinct variables from V and a nonnegative integer k.

Parameter: k.

Question: Is there an ordering α of V such that at least $\frac{|\Pi|}{6}|\mathcal{C}| + k$ constraints of $\mathcal{C}$ are Π-satisfied by α?

Key Results

The following is a simple but important observation in [4] allowing one to reduce Π-AA to MAX-3-LIN-ORDERING-AA.

Proposition 1 *Let Π be a subset of $\mathcal{S}_3$ such that $\Pi \notin \{\emptyset, \mathcal{S}_3\}$. There is a polynomial time transformation f from Π-AA to* MAX-3-LIN-ORDERING-AA *such that an instance $(V, \mathcal{C}, k)$ of Π-AA is a Yes-instance if and only if $(V, \mathcal{C}', k) = f(V, \mathcal{C}, k)$ is a Yes-instance of* MAX-3-LIN-ORDERING-AA.

Using a nontrivial reduction from MAX-3-LIN-ORDERING-AA to a combination of MAX-2-LIN-ORDERING-AA and BETWEENNESS-AA and the facts that both problems admit kernels with quadratic numbers of variables and constraints (proved in [3] and [2], respectively),

Gutin et al. [4] showed that MAX-3-LIN-ORDERING-AA also admits a kernel with quadratic numbers of variables and constraints. Kim and Williams [6] partially improved this result by showing that MAX-3-LIN-ORDERING-AA admits a kernel with $O(k)$ variables.

The polynomial-size kernel result for MAX-3-LIN-ORDERING-AA and Proposition 1 imply the following (see [4] for details):

Theorem 1 ([4]) *Let Π be a subset of $\mathcal{S}_3$ such that $\Pi \notin \{\emptyset, \mathcal{S}_3\}$. The problem Π-AA admits a polynomial-size kernel with $O(k^2)$ variables.*

Open Problems

Similar to Proposition 1, it is easy to prove that, for each fixed r every Π-AA can be reduced to LIN-r-ORDERING-AA. Gutin et al. [4] conjectured that for each fixed r the problem MAX-r-LIN-ORDERING-AA is fixed-parameter tractable.

Cross-References

- Kernelization, Constraint Satisfaction Problems Parameterized above Average
- Kernelization, MaxLin Above Average

Recommended Reading

1. Charikar M, Guruswami V, Manokaran R (2009) Every permutation CSP of arity 3 is approximation resistant. In: Computational complexity 2009, Paris, pp 62–73
2. Gutin G, Kim EJ, Mnich M, Yeo A (2010) Betweenness parameterized above tight lower bound. J Comput Syst Sci 76:872–878
3. Gutin G, Kim EJ, Szeider S, Yeo A (2011) A probabilistic approach to problems parameterized above tight lower bound. J Comput Syst Sci 77:422–429
4. Gutin G, van Iersel L, Mnich M, Yeo A (2012) All ternary permutation constraint satisfaction problems parameterized above average have Kernels with quadratic number of variables. J Comput Syst Sci 78:151–163
5. Guttmann W, Maucher M (2006) Variations on an ordering theme with constraints. In: 4th IFIP international conference on theoretical computer science-TCS 2006, Santiago. Springer, pp 77–90
6. Kim EJ, Williams R (2012) Improved parameterized algorithms for above average constraint satisfaction. In: IPEC 2011, Saarbrücken. Lecture notes in computer science, vol 7112, pp 118–131

Kernelization, Planar F-Deletion

Neeldhara Misra
Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India

Keywords

Finite-integer index; Meta-theorems; Protrusions; Treewidth

Years and Authors of Summarized Original Work

2012; Fomin, Lokshtanov, Misra, Saurabh
2013; Kim, Langer, Paul, Reidl, Rossmanith, Sau, Sikdar

Problem Definition

Several combinatorial optimization problems on graphs involve identifying a subset of nodes S, of the smallest cardinality, such that the graph obtained after removing S satisfies certain properties. For example, the VERTEX COVER problem asks for a minimum-sized subset of vertices whose removal makes the graph edgeless, while the FEEDBACK VERTEX SET problem involves finding a minimum-sized subset of vertices whose removal makes the graph acyclic. The $\mathcal{F}$-DELETION problem is a generic formulation that encompasses several problems of this flavor.

Let $\mathcal{F}$ be a finite set of graphs. In the $\mathcal{F}$-DELETION problem, the input is an n-vertex graph G and an integer k, and the question is if

G has a subset S of at most k vertices, such that $G-S$ does not contain a graph from $\mathcal{F}$ as a minor. The optimization version of the problem seeks such a subset of the smallest possible size. The PLANAR $\mathcal{F}$-DELETION problem is the version of the problem where $\mathcal{F}$ contains at least one planar graph. The $\mathcal{F}$-DELETION problem was introduced by [3], who gave a non-constructive algorithm running in time $O(f(k) \cdot n^2)$ for some function $f(k)$. This result was improved by [1] to $O(f(k) \cdot n)$, for $f(k) = 2^{2^{O(k \log k)}}$.

For different choices of sets of forbidden minors $\mathcal{F}$, one can obtain various fundamental problems. For example, when $\mathcal{F} = \{K_2\}$, a complete graph on two vertices, this is the VERTEX COVER problem. When $\mathcal{F} = \{C_3\}$, a cycle on three vertices, this is the FEEDBACK VERTEX SET problem. The cases of $\mathcal{F}$ being $\{K_{2,3}, K_4\}$, $\{K_4\}$, $\{\theta_c\}$, and $\{K_3, T_2\}$, correspond to removing vertices to obtain an outerplanar graph, a series-parallel graph, a diamond graph, and a graph of pathwidth one, respectively.

Tools

Most algorithms for the PLANAR $\mathcal{F}$-DELETION problem appeal to the notion of *protrusions* in graphs. An r-protrusion in a graph G is a subgraph H of treewidth at most r such that the number of neighbors of H in $G - H$ is at most r. Intuitively, a protrusion H in a graph G may be thought of as subgraph of small treewidth which is cut off from the rest of the graph by a small separator.

Usually, as a means of preprocessing, protrusions are identified and *replaced* by smaller ones, while maintaining equivalence. The notion of graph replacement in this fashion originates in the work of [4]. The modern notion of protrusion reductions have been employed in various contexts [2, 5, 6, 12]. A widely used method for developing a protrusion replacement algorithm is via the notion of *finite-integer index*. Roughly speaking, this property ensures that graphs can be related under some appropriate notion of equivalence with respect to the problem, and that there are only finitely many equivalence classes. This allows us to identify the class that the protrusion belongs to and replace it with a canonical representative for that class.

Key Results

The algorithms proposed for PLANAR $\mathcal{F}$-DELETION usually have the following ingredients. First, the fact that $\mathcal{F}$ contains a planar graph implies that any YES-instance of the problem must admit a small subset of vertices whose removal leads to a graph of small treewidth. It turns out that such graphs admit a convenient structure from the perspective of the existence of protrusions. In particular, most of the graph can be decomposed into protrusions. From here, there are two distinct themes.

In the first approach, the protrusions are *replaced* by smaller, equivalent graphs. Subsequently, we have a graph that has no large protrusions. For such instances, it can be shown that if there is a solution, there is always one that is incident to a constant fraction of the edges in the graph, and this leads to a randomized algorithm by branching. Notably, the protrusion replacement can be performed by an algorithm that guarantees the removal of a constant fraction of vertices in every application. This helps in ensuring that the overall running time of the algorithm has a linear dependence on the size of the input. This algorithm is limited to the case when all graphs in $\mathcal{F}$ are connected, as is required in demonstrating finite-integer index.

Theorem 1 ([8]) *When every graph in $\mathcal{F}$ is connected, there is a randomized algorithm solving* PLANAR $\mathcal{F}$-DELETION *in time* $2^{O(k)} \cdot n$.

The second approach involves exploring the structure of the instance further. Here, an $O(k)$-sized subset of vertices is identified, with the key property that there is a solution that lives within it. The algorithm then proceeds to exhaustively branch on these vertices. This technique requires a different protrusion decomposition from the previous one. The overall algorithm is implemented using iterative compression. Since the protrusions are not replaced, this algorithm works for all instances of PLANAR $\mathcal{F}$-DELETION, with-

out any further assumptions on the family $\mathcal{F}$. While both approaches lead to algorithms that are single-exponential in k, the latter has a quadratic dependence on the size of the input.

Theorem 2 ([11]) PLANAR-$\mathcal{F}$-DELETION *can be solved in time* $2^{O(k)} \cdot n^2$.

In the context of approximation algorithms, the protrusion replacement is more intricate, because the notion of equivalence is now more demanding. The replacement should preserve not only the exact solutions, but also approximate ones. By appropriately adapting the machinery of replacements with lossless protrusion replacers, the problem admits the following approximation algorithm.

Theorem 3 ([8]) PLANAR $\mathcal{F}$-DELETION *admits a randomized constant ratio approximation algorithm.*

The PLANAR $\mathcal{F}$-DELETION problem also admits efficient preprocessing algorithms. Formally, a kernelization algorithm for the problem takes an instance (G, k) as input and outputs an equivalent instance (H, k') where the size of the output is bounded by a function of k. If the size of the output is bounded by a polynomial function of k, then it is called a polynomial kernel. The reader is referred to the survey [13] for a more detailed introduction to kernelization.

The technique of protrusion replacement was developed and used successfully for kernelization algorithms on sparse graphs [2,5]. These methods were also used for the special case of the PLANAR $\mathcal{F}$-DELETION problem when $\mathcal{F}$ is a graph with two vertices and constant number of parallel edges [6]. In the general setting of PLANAR $\mathcal{F}$-DELETION, kernelization involves anticipating protrusions, that is, identifying subgraphs that become protrusions after the removal of some vertices from an optimal solution. These "near-protrusions" are used to find irrelevant edges, i.e., an edge whose removal does not change the problem, leading to natural reduction rules. The process of finding an irrelevant edge appeals to the well-quasi-ordering of a certain class of graphs as a subroutine.

Theorem 4 ([8]) PLANAR $\mathcal{F}$-DELETION *admits a polynomial kernel.*

Applications

The algorithms for PLANAR $\mathcal{F}$-DELETION apply to any vertex deletion problem that can be described as hitting minor models of some fixed finite family that contains a planar graph.

For a finite set of graphs $\mathcal{F}$, let $\mathcal{G}_{\mathcal{F},k}$ be a class of graphs such that for every $G \in \mathcal{G}_{\mathcal{F},k}$ there is a subset of vertices S of size at most k such that $G \setminus S$ has no minor from $\mathcal{F}$. The following combinatorial result is a consequence of the kernelization algorithm for PLANAR $\mathcal{F}$-DELETION.

Theorem 5 ([8]) *For every set* $\mathcal{F}$ *that contains a planar graph, every minimal obstruction for* $\mathcal{G}_{\mathcal{F},k}$ *is of size polynomial in* k.

Kernelization algorithms on apex-free and H-minor-free graphs for all bidimensional problems from [5] can be implemented in linear time by employing faster protrusion reducers. This leads to randomized linear time, linear kernels for several problems.

In the framework for obtaining EPTAS on H-minor-free graphs in [7], the running time of approximation algorithms for many problems is $f(1/\varepsilon) \cdot n^{O(g(H))}$, where g is some function of H only. The only bottleneck for improving polynomial-time dependence is a constant factor approximation algorithm for TREEWIDTH η-DELETION. Using Theorem 3 instead, each EPTAS from [7] runs in time $O(f(1/\varepsilon) \cdot n^2)$. For the same reason, the PTAS algorithms for many problems on unit disk and map graphs from [9] become EPTAS algorithms.

Open Problems

An interesting direction for further research is to investigate PLANAR $\mathcal{F}$-DELETION when none of the graphs in $\mathcal{F}$ is planar. The most interesting case here is when $\mathcal{F} = \{K_5, K_{3,3}\}$, also known

as the VERTEX PLANARIZATION problem. The work in [10] demonstrates an algorithm with running time $2^{O(k \log k)}n$, which notably has a linear-time dependence on n. It remains open as to whether VERTEX PLANARIZATION can be solved in $2^{O(k)}n$ time. The question of polynomial kernels in the non-planar setting is also open, in particular, even the specific case of $\mathcal{F} = \{K_5\}$ is unresolved.

Cross-References

- Bidimensionality
- Kernelization, Preprocessing for Treewidth
- Treewidth of Graphs

Recommended Reading

1. Bodlaender HL (1997) Treewidth: algorithmic techniques and results. In: 22nd international symposium on mathematical foundations of computer science (MFCS), Bratislava, vol 1295, pp 19–36
2. Bodlaender HL, Fomin FV, Lokshtanov D, Penninkx E, Saurabh S, Thilikos DM (2009) (Meta) kernelization. In: Proceedings of the 50th annual IEEE symposium on foundations of computer science (FOCS), Atlanta, pp 629–638
3. Fellows MR, Langston MA (1988) Nonconstructive tools for proving polynomial-time decidability. J ACM 35(3):727–739
4. Fellows MR, Langston MA (1989) An analogue of the Myhill-Nerode theorem and its use in computing finite-basis characterizations (extended abstract). In: Proceedings of the 30th annual IEEE symposium on foundations of computer science (FOCS), Research Triangle Park, pp 520–525
5. Fomin FV, Lokshtanov D, Saurabh S, Thilikos DM (2010) Bidimensionality and kernels. In: Proceedings of the twenty-first annual ACM-SIAM symposium on discrete algorithms (SODA), Austin, pp 503–510
6. Fomin FV, Lokshtanov D, Misra N, Philip G, Saurabh S (2011) Hitting forbidden minors: approximation and kernelization. In: Proceedings of the 8th international symposium on theoretical aspects of computer science (STACS), LIPIcs, Dortmund, vol 9, pp 189–200
7. Fomin FV, Lokshtanov D, Raman V, Saurabh S (2011) Bidimensionality and EPTAS. In: Proceedings of the 22nd annual ACM-SIAM symposium on discrete algorithms (SODA), San Francisco. SIAM, pp 748–759
8. Fomin FV, Lokshtanov D, Misra N, Saurabh S (2012) Planar F-deletion: approximation, kernelization and optimal FPT algorithms. In: Proceedings of the 2012 IEEE 53rd annual symposium on foundations of computer science, New Brunswick, pp 470–479
9. Fomin FV, Lokshtanov D, Saurabh S (2012) Bidimensionality and geometric graphs. In: Proceedings of the 23rd annual ACM-SIAM symposium on discrete algorithms (SODA), Kyoto. SIAM, pp 1563–1575
10. Jansen BMP, Lokshtanov D, Saurabh S (2014) A near-optimal planarization algorithm. In: Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms, (SODA), Portland, pp 1802–1811
11. Kim EJ, Langer A, Paul C, Reidl F, Rossmanith P, Sau I, Sikdar S (2013) Linear kernels and single-exponential algorithms via protrusion decompositions. In: Proceedings of the 40th international colloquium on automata, languages, and programming (ICALP), Riga, Part I, pp 613–624
12. Langer A, Reidl F, Rossmanith P, Sikdar S (2012) Linear kernels on graphs excluding topological minors. CoRR abs/1201.2780
13. Lokshtanov D, Misra N, Saurabh S (2012) Kernelization – preprocessing with a guarantee. In: Bodlaender HL, Downey R, Fomin FV, Marx D (eds) The multivariate algorithmic revolution and beyond. Bodlaender, HansL. and Downey, Rod and Fomin, FedorV. and Marx, Dániel (eds) Lecture notes in computer science, vol 7370. Springer, Berlin/Heidelberg, pp 129–161

Kernelization, Polynomial Lower Bounds

Stefan Kratsch
Department of Software Engineering and Theoretical Computer Science, Technical University Berlin, Berlin, Germany

Keywords

Kernelization; Parameterized complexity; Satisfiability; Sparsification

Years and Authors of Summarized Original Work

2010; Dell, van Melkebeek

Problem Definition

The work of Dell and van Melkebeek [4] refines the framework for lower bounds for kernelization introduced by Bodlaender et al. [1] and Fortnow and Santhanam [6]. The main contribution is that their results yield a framework for proving polynomial lower bounds for kernelization rather than ruling out all polynomial kernels for a problem; this, for the first time, gives a technique for proving that some polynomial kernelizations are actually best possible, modulo reasonable complexity assumptions. A further important aspect is that, rather than studying kernelization directly, the authors give lower bounds for a far more general oracle communication protocol. In this way, they also obtain strong lower bounds for sparsification, lossy compression (in the sense of Harnik and Naor [7]), and probabilistically checkable proofs (PCPs).

To explain the connection between kernelization and oracle communication protocols, let us first recall the following. A *parameterized problem* is a language $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$; the second component k of instances $(x,k) \in \Sigma^* \times \mathbb{N}$ is called the *parameter*. A *kernelization for $\mathcal{Q}$ with size $h\colon \mathbb{N} \to \mathbb{N}$* is an efficient algorithm that gets as input an instance $(x,k) \in \Sigma^* \times \mathbb{N}$ and returns an equivalent instance (x',k'), i.e., such that $(x,k) \in \mathcal{Q}$ if and only if $(x',k') \in \mathcal{Q}$, with $|x'|, k' \leq h(k)$. If $h(k)$ is polynomially bounded in k, then we also call it a polynomial kernelization.

One way to use a kernelization is to first simplify a given input instance and then solve the reduced instance by any (possibly brute-force) algorithm; together this yields an algorithm for solving the problem in question. If we abstract out the algorithm by saying that the answer for the reduced instance is given by an oracle, then we arrive at a special case of the following communication protocol.

Definition 1 (oracle communication protocol [4]) An *oracle communication protocol* for a language L is a communication protocol for two players. The first player is given the input x and has to run in time polynomial in the length of the input; the second player is computationally unbounded but is not given any part of x. At the end of the protocol, the first player should be able to decide whether $x \in L$. The cost of the protocol is the number of bits of communication from the first player to the second player.

As an example, if $\mathcal{Q}$ has a kernelization with size h, then instances (x,k) can be solved by a protocol of cost $h(k)$. It suffices that the first player can compute a reduced instance (x',k') and send it to the oracle who decides membership of (x',k') in $\mathcal{Q}$; this yields the desired answer for whether $(x,k) \in \mathcal{Q}$. Note that the communication protocol is far more general than kernelization because it makes no assumption about what exactly is sent (or in what encoding). More importantly, it also allows multiple rounds of communication, and the behavior of the oracle could also be active rather than just answering queries for the first player. Thus, the obtained lower bounds for oracle communication protocols are very robust, covering also relaxed forms of kernelization (like bikernels and compressions), and also yield the other mentioned applications.

Key Results

A central result in the work of Dell and van Melkebeek [4] (see also [5]) is the following lemma, called *complementary witness lemma*.

Lemma 1 (complementary witness lemma [4]) *Let L be a language and $t\colon \mathbb{N} \to \mathbb{N} \setminus \{0\}$ be polynomially bounded such that the problem of deciding whether at least one out of $t(s)$ inputs of length at most s belongs to L has an oracle communication protocol of cost $\mathcal{O}(t(s) \log t(s))$, where the first player can be conondeterministic. Then $L \in$ coNP/poly.*

A previous work of Fortnow and Santhanam [6] showed that an efficient algorithm for encoding any t instances $x_1, \ldots, x_t$ of size at most s into one instance y of size poly(s) such that $y \in L$ if and only if at least one x_i is in L implies $L \in$ coNP/poly. (We recall that this

settled the *OR-distillation conjecture* of Bodlaender et al. [1] and allowed their framework to rule out polynomial kernels under the assumption that NP $\not\subseteq$ coNP/poly.) Lemma 1 is obtained by a more detailed analysis of this result and requires an encoding of the OR of $t(s)$ instances into one instance of size $\mathcal{O}(t(s)\log t(s))$ rather than allowing only size poly(s) for all values of t. This focus on the number $t(s)$ of instances in relation to the maximum instance size s is the key for getting polynomial lower bounds for kernelization (and other applications). In this overview, we will not discuss the possibility of conondeterministic behavior of the first player, but the interested reader is directed to [9, 10] for applications thereof.

Before outlining further results of Dell and van Melkebeek [4], let us state a lemma that captures one way of employing the complementary witness lemma for polynomial lower bounds for kernelization. The lemma is already implicit in [4] and is given explicitly in follow-up work of Dell and Marx [3] (it can also be found in the current full version [5] of [4]). We recall that OR(L) refers to the language of all tuples $(x_1, \ldots, x_t)$ such that at least one x_i is contained in L.

Lemma 2 ([3, 5]) *Suppose that a parameterized problem Π has the following property for some constant c: For some* NP*-complete language L, there exists a polynomial-time mapping reduction from* OR(L) *to Π that maps an instance $(x_1, \ldots, x_t)$ of* OR(L) *in which each x_i has size at most s to an instance of Π with parameter $k \leq t^{1/c+o(1)} \cdot \text{poly}(s)$. Then Π does not have a communication protocol of cost $\mathcal{O}(k^{c-\epsilon})$ for any constant $\epsilon > 0$ unless* NP $\subseteq$ coNP/poly*, even when the first player is conondeterministic.*

Intuitively, Lemma 2 follows from Lemma 1 because if the reduction and communication protocol in Lemma 2 both exist (for all t), then we can choose $t(s)$ large enough (but polynomially bounded in s) such that for all s we get an oracle communication protocol of cost $\mathcal{O}(t(s))$ as required for Lemma 1. This implies $L \in$ coNP/poly and, hence, NP $\subseteq$ coNP/poly (since L is NP-complete). As discussed earlier, any kernelization yields an oracle communication protocol with cost equal to the kernel size and, thus, this bound carries over directly to kernelization.

Let us now state the further results of Dell and van Melkebeek [4] using the context of Lemma 2. The central result is the following theorem on lower bounds for vertex cover on d-uniform hypergraphs.

Theorem 1 ([4]) *Let $d \geq 2$ be an integer and ϵ a positive real. If* NP $\not\subseteq$ coNP/poly*, there is no protocol of cost $\mathcal{O}(n^{d-\epsilon})$ to decide whether a d-uniform hypergraph on n vertices has a vertex cover of at most k vertices, even when the first player is conondeterministic.*

To prove Theorem 1, Dell and van Melkebeek devise a reduction from OR(SAT) to CLIQUE on d-uniform hypergraphs parameterized by the number of vertices (fulfilling the assumption of Lemma 2 for $c = d$). This reduction relies on an intricate lemma, the *packing lemma*, that constructs a d-uniform hypergraph with t cliques on s vertices each, but having only about $\mathcal{O}(t^{1/d+o(1)} \cdot s)$ vertices and no further cliques of size s. In follow-up work, Dell and Marx [3] give a simpler proof for Theorem 1 without making use of the packing lemma, but use the lemma for another of their results.

Note that the stated bound for VERTEX COVER in d-uniform hypergraphs follows by complementation. Furthermore, since every nontrivial instance has $k \leq n$, this also rules out kernelization to size $\mathcal{O}(k^{d-\epsilon})$. The following lower bound for SATISFIABILITY is obtained by giving a reduction from VERTEX COVER on d-uniform hypergraphs with parameter n. In the reduction, hyperedges of size d are encoded by positive clauses on d variables (one per vertex), and an additional part of the formula (which requires $d \geq 3$) checks that at most k of these variables are set to true.

Theorem 2 ([4]) *Let $d \geq 3$ be an integer and ϵ a positive real. If* NP $\not\subseteq$ coNP/poly*, there is no*

protocol of cost $\mathcal{O}(n^{d-\epsilon})$ to decide whether an n-variable d-CNF formula is satisfiable, even when the first player is conondeterministic.

Finally, the following theorem proves that several known kernelizations for graph modification problems are already optimal. The theorem is proved by a reduction from VERTEX COVER (on graphs) with parameter k that is similar in spirit to the classical result of Lewis and Yannakakis on NP-completeness of the Π-VERTEX DELETION problem for nontrivial hereditary properties Π. Note that Theorem 3 requires that the property Π is not only hereditary, i.e., inherited by *induced* subgraphs, but inherited by *all subgraphs*.

Theorem 3 ([4]) *Let Π be a graph property that is inherited by subgraphs and is satisfied by infinitely many but not all graphs. Let ϵ be a positive real. If $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$, there is no protocol of cost $\mathcal{O}(k^{2-\epsilon})$ for deciding whether a graph satisfying Π can be obtained from a given graph by removing at most k vertices.*

As an example, the theorem implies that the FEEDBACK VERTEX SET problem does not admit a kernelization with size $\mathcal{O}(k^{2-\epsilon})$. This is in fact tight since a kernelization by Thomassé [12] achieves $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ edges (cf. [4]); improving to $\mathcal{O}(k^{2-\epsilon})$ edges is ruled out since it would yield an encoding in size $\mathcal{O}(k^{2-\epsilon'})$. Similarly, the well-known kernelization for VERTEX COVER to $2k$ vertices is tight and cannot, in general, be expected to yield instances with less than the trivial $\mathcal{O}(k^2)$ edges.

Applications

Several authors have used the present approach to get polynomial lower bounds for kernelizations of certain parameterized problems; see, e.g., [2, 3, 8, 11]. Similarly, some results make use of conondeterminism [9, 10] and the more general setting of lower bounds for oracle communication protocols [11].

Open Problems

Regarding applications it would be interesting to have more lower bounds that use the full generality of the oracle communication protocols. Furthermore, it is an open problem to relax the assumption of $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$ to the minimal $\mathsf{P} \neq \mathsf{NP}$.

Recommended Reading

1. Bodlaender HL, Downey RG, Fellows MR, Hermelin D (2009) On problems without polynomial kernels. J Comput Syst Sci 75(8):423–434
2. Cygan M, Grandoni F, Hermelin D (2013) Tight kernel bounds for problems on graphs with small degeneracy – (extended abstract). In: Bodlaender HL, Italiano GF (eds) ESA. Lecture notes in computer science, vol 8125. Springer, pp 361–372
3. Dell H, Marx D (2012) Kernelization of packing problems. In: SODA, Kyoto. SIAM, pp 68–81
4. Dell H, van Melkebeek D (2010) Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. In: Schulman LJ (ed) STOC, Cambridge. ACM, pp 251–260
5. Dell H, van Melkebeek D (2010) Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. Electron Colloq Comput Complex 17:38
6. Fortnow L, Santhanam R (2011) Infeasibility of instance compression and succinct PCPs for NP. J Comput Syst Sci 77(1):91–106
7. Harnik D, Naor M (2010) On the compressibility of $\mathcal{NP}$ instances and cryptographic applications. SIAM J Comput 39(5):1667–1713
8. Hermelin D, Wu X (2012) Weak compositions and their applications to polynomial lower bounds for kernelization. In: SODA, Kyoto. SIAM, pp 104–113
9. Kratsch S (2012) Co-nondeterminism in compositions: a kernelization lower bound for a Ramsey-type problem. In: SODA, Kyoto. SIAM, pp 114–122
10. Kratsch S, Pilipczuk M, Rai A, Raman V (2012) Kernel lower bounds using co-nondeterminism: finding induced hereditary subgraphs. In: Fomin FV, Kaski P (eds) SWAT, Helsinki. Lecture notes in computer science, vol 7357. Springer, pp 364–375
11. Kratsch S, Philip G, Ray S (2014) Point line cover: the easy kernel is essentially tight. In: SODA, Portland. SIAM, pp 1596–1606
12. Thomassé S (2010) A quadratic kernel for feedback vertex set. ACM Trans Algorithms 6(2), 32:1–32:8

Kernelization, Preprocessing for Treewidth

Stefan Kratsch
Department of Software Engineering and Theoretical Computer Science, Technical University Berlin, Berlin, Germany

Keywords

Kernelization; Parameterized complexity; Preprocessing; Structural parameters; Treewidth

Years and Authors of Summarized Original Work

2013; Bodlaender, Jansen, Kratsch

Problem Definition

This work undertakes a theoretical study of preprocessing for the NP-hard TREEWIDTH problem of finding a *tree decomposition* of width at most k for a given graph G. In other words, given G and $k \in \mathbb{N}$, the question is whether G has *treewidth* at most k. Several efficient reduction rules are known that provably preserve the correct answer, and experimental studies show significant size reductions [3, 5]. The present results study these and further newly introduced rules and obtain upper and lower bounds within the framework of *kernelization* from parameterized complexity.

The general interest in computing tree decompositions is motivated by the well-understood approach of using dynamic programming on tree decompositions that is known to allow fast algorithms on graphs of bounded treewidth (but with runtime exponential in the treewidth). A bottleneck for practical applications is the need for finding, as a first step, a sufficiently good tree decomposition; the best known exact algorithm due to Bodlaender [2] runs in time exponential in k^3 and is thus only of theoretical interest. This motivates the use of heuristics and preprocessing to find a reasonably good tree decomposition quickly.

Tree Decompositions and Treewidth

A *tree decomposition* for a graph $G = (V, E)$ consists of a tree $T = (N, F)$ and a family $\mathcal{X} := \{X_i \mid i \in N, X_i \subseteq V\}$. The sets X_i are also called *bags* and the vertices of T are usually referred to as *nodes* to avoid confusion with G; there is exactly one bag X_i associated with each node $i \in N$. The pair $(T, \mathcal{X})$ must fulfill the following three properties: (1) Every vertex of G is contained in at least one bag; (2) For each edge $\{u, v\} \in E$ there must be a bag X_i containing both u and v; (3) For each vertex v of G the set of nodes i of T with $v \in X_i$ induce a (connected) subtree of T. The *width of a tree decomposition* $(T, \mathcal{X})$ is equal to the size of the largest bag $X_i \in \mathcal{X}$ minus one. The *treewidth* of a graph G, denoted $\mathsf{tw}(G)$, is the smallest width taken over all tree decompositions of G.

Parameters

The framework of parameterized complexity allows the study of the TREEWIDTH problem with respect to different *parameters*. A parameter is simply an integer value associated with each problem instance. The *standard parameter* for an optimization problem like TREEWIDTH is the desired solution quality k and we denote this problem by TREEWIDTH(k). Apart from this, *structural parameters* are considered that capture structural aspects of G. For example, the work considers the behavior of TREEWIDTH when the input graph G has a small vertex cover S, i.e., such that deletion of $\ell = |S|$ vertices yields an independent set, with ℓ being used as the parameter. Similarly, several other parameters are discussed, foremost among them the feedback vertex set number and the vertex deletion distance to a single clique; the corresponding vertex sets are called *modulators*, e.g., a feedback vertex set is a modulator to a forest. We denote the arising parameterized problems by TREEWIDTH(vc), TREEWIDTH(fvs), and TREEWIDTH($\mathsf{vc}(\overline{G})$). To decouple the overhead of finding, e.g., a mini-

mum vertex cover for G, all these variants assume that an appropriate modulator is given along with the input and the obtained guarantees are in terms of the size of this modulator. Since all studied parameters can be efficiently approximated to within a constant factor of the optimum, not providing an (optimal) modulator gives only a constant-factor blowup in the obtained results.

Kernelization

A kernelization for a problem with parameter ℓ is an efficient algorithm that given an instance (x, ℓ) returns an equivalent instance (x', ℓ') of size and parameter value ℓ' bounded by some computable function of ℓ. If the bound is polynomial in ℓ then we have a *polynomial kernelization*. Specialized to, for example, TREEWIDTH(vc) a polynomial kernelization would have the following behavior: It gets as input an instance (G, S, k), asking whether the treewidth of G is at most k, where S is a vertex cover for G. In polynomial time it creates an instance (G', S', k') such that: (1) The size of the instance (G', S', k') and the parameter value $|S'|$ are bounded polynomially in k; (2) The set S' is a vertex cover of G'; (3) The graph G has treewidth at most k if and only if G' has treewidth at most k'.

Key Results

The kernelization lower bound framework of Bodlaender et al. [6] together with recent results of Drucker [9] is known to imply that TREEWIDTH(k) admits no polynomial kernelization unless NP $\subseteq$ coNP/poly and the polynomial hierarchy collapses. The present work takes a more detailed look at polynomial kernelization for TREEWIDTH with respect to structural parameters. The results are as follows.

Theorem 1 TREEWIDTH(vc) *i.e., parameterized by vertex cover number, admits a polynomial kernelization to an equivalent instance with* $\mathcal{O}((\mathsf{vc}(G))^3)$ *vertices.*

An interesting feature of this result is that it uses only three simple reduction rules that are well known and often used (cf. [5]). Two rules address so-called simplicial vertices, whose neighborhood is a clique, and a third rule inserts edges between certain pairs of vertices that have a large number of shared vertices. Analyzing these empirically successful rules with respect to the vertex cover number of the input graph yields a kernelization. A fact that nicely complements the observed experimental success.

Theorem 2 TREEWIDTH(fvs) *i.e., parameterized by feedback vertex set number, admits a polynomial kernelization to an equivalent instance with* $\mathcal{O}((\mathsf{fvs}(G))^4)$ *vertices.*

The feedback vertex set number of a graph is upper bounded by its vertex cover number, and forests have feedback vertex set number zero but arbitrarily large vertex cover number. Thus, for large families of input graphs, this second result is stronger. The result again builds on several known reduction rules (including the above ones), among others, for handling vertices that are *almost simplicial*, i.e., all but one neighboring vertex form a clique. On top of these, several new rules are added. One of them addresses a previously uncovered case of almost simplicial vertex removal, namely, when the vertex has degree exactly $k + 1$, where k is the desired treewidth bound. Furthermore, these reduction rules lead to a structure dubbed *clique-seeing paths*, which takes a series of fairly technical rules and analysis to reduce and bound. Altogether, this combination leads to the above result.

Theorem 3 TREEWIDTH($\mathsf{vc}(\overline{G})$) *i.e., parameterized by deletion distance to a single clique, admits no polynomial kernelization unless* NP $\subseteq$ coNP/poly *and the polynomial hierarchy collapses.*

The proof uses the notion of a *cross-composition* introduced by Bodlaender et al. [8], which builds directly on the kernelization lower bound framework of Bodlaender et al. [6] and Fortnow and Santhanam [10]. The cross-composition builds on the proof of NP-

completeness of TREEWIDTH by Arnborg et al. [1], which uses a Karp reduction from CUTWIDTH to TREEWIDTH. This construction is extended significantly to yield a cross-composition of CUTWIDTH ON SUBCUBIC GRAPHS (i.e., graphs of maximum degree three) into TREEWIDTH, which, roughly, requires an encoding of many CUTWIDTH instances into a single instance of TREEWIDTH with sufficiently small parameter.

Overall, together with previously known results, the obtained upper and lower bounds for TREEWIDTH cover a wide range of natural parameter choices (see the discussion in [7]). If $\mathcal{C}$ is any graph class that contains all cliques, then the vertex deletion distance of a graph G to $\mathcal{C}$ is upper bounded by $\mathsf{vc}(\overline{G})$. Thus, TREEWIDTH parameterized by distance to $\mathcal{C}$ does not admit a polynomial kernelization unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. This includes several well-studied classes like interval graphs, cographs, and perfect graphs. Since TREEWIDTH remains NP-hard on bipartite graphs, the result for parameterization by feedback vertex set number cannot be generalized to vertex deletion to a bipartite graph. It may, however, be possible to generalize this parameter to vertex deletion distance to an outerplanar graph, i.e., planar graphs having an embedding with all vertices appearing on the outer face. Since these graphs generalize forests, this value is upper bounded by the feedback vertex set number.

Theorem 4 WEIGHTED TREEWIDTH(vc) *i.e., parameterized by vertex cover number, admits no polynomial kernelization unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ *and the polynomial hierarchy collapses.*

In the WEIGHTED TREEWIDTH problem, each vertex comes with an integer weight, and the size of a bag in the tree decomposition is defined as the sum of the weights of its vertices. (To note, the present paper uses an extra deduction of one such that treewidth and weighted treewidth coincide for graphs with all vertices having weight one.) The result is proved by a cross-composition from TREEWIDTH (to WEIGHTED TREEWIDTH parameterized by vertex cover number) and complements the polynomial kernelization for the unweighted case. A key idea for the cross-composition is to use a result of Bodlaender and Möhring [4] on the behavior of treewidth under the join operation on graphs. This is combined with replacing all edges (in input graphs and join edges) by using a small number of newly introduced vertices of high weight.

Open Problems

A particular interesting case left open by existing results on polynomial kernelization for structural parameterizations of TREEWIDTH is the vertex deletion distance to outerplanar graphs.

Recommended Reading

1. Arnborg S, Corneil DG, Proskurowski A (1987) Complexity of finding embeddings in a k-tree. SIAM J Algebra Discret 8(2):277–284. doi:10.1137/0608024
2. Bodlaender HL (1996) A linear-time algorithm for finding tree-decom-positions of small treewidth. SIAM J Comput 25(6):1305–1317. doi:10.1145/167088.167161
3. Bodlaender HL, Koster AMCA (2006) Safe separators for treewidth. Discret Math 306(3): 337–350. doi:10.1016/j.disc.2005.12.017
4. Bodlaender HL, Möhring RH (1993) The pathwidth and treewidth of cographs. SIAM J Discret Math 6(2):181–188. doi:10.1137/0406014
5. Bodlaender HL, Koster AMCA, van den Eijkhof F (2005) Preprocessing rules for triangulation of probabilistic networks. Comput Intell 21(3):286–305. doi:10.1111/j.1467-8640.2005.00274.x
6. Bodlaender HL, Downey RG, Fellows MR, Hermelin D (2009) On problems without polynomial kernels. J Comput Syst Sci 75(8):423–434
7. Bodlaender HL, Jansen BMP, Kratsch S (2013) Preprocessing for treewidth: a combinatorial analysis through kernelization. SIAM J Discret Math 27(4):2108–2142
8. Bodlaender HL, Jansen BMP, Kratsch S (2014) Kernelization lower bounds by cross-composition. SIAM J Discret Math 28(1):277–305
9. Drucker A (2012) New limits to classical and quantum instance compression. In: FOCS, New Brunswick. IEEE Computer Society, pp 609–618
10. Fortnow L, Santhanam R (2011) Infeasibility of instance compression and succinct PCPs for NP. J Comput Syst Sci 77(1):91–106

Kernelization, Turing Kernels

Henning Fernau
Fachbereich 4, Abteilung Informatikwissenschaften, Universität Trier, Trier, Germany
Institute for Computer Science, University of Trier, Trier, Germany

Keywords

Karp reduction; Kernelization; Kernel size; Turing kernelization; Turing reduction

Years and Authors of Summarized Original Work

2012; Binkele-Raible, Fernau, Fomin, Lokshtanov, Saurabh
2013; Hermelin, Kratsch, Soltys, Wahlström, Wu
2014; Jansen

Definition and Discussion

The basic definition of the field expresses kernelization as a Karp (many-one) self-reduction. Classical complexity and recursion theory offers quite a lot of alternative and more general notions of reducibilities. The most general notion, that of a Turing reduction, motivates the following definition:

Let (Q, κ) be a parameterized problem over a finite alphabet Σ.

- An *input-bounded oracle* for (Q, κ) is an oracle that, for any given input $x \in \Sigma^*$ of (Q, κ) and any bound t, first checks if $|x|, |\kappa(x)| \leq t$, and if this is certified, it decides in constant time whether the input x is a YES instance of (Q, κ).
- A *Turing kernelization (algorithm)* for (Q, κ) is an algorithm that, provided with access to some input-bounded oracle for (Q, κ), decides on input $x \in \Sigma^*$ in polynomial time whether x is a YES instance of (Q, κ) or not. During its computation, the algorithm can produce (polynomially many) oracle queries x' with bound $t = h(\kappa(x))$, where h is an arbitrary computable function. The function h is referred to as the *size* of the kernel.

If only one oracle access is permitted in a run of the algorithm, we basically get the classical notion of a (many-one or Karp) kernelization.

A more general definition was given in [4], allowing access to a different (auxiliary) problem (Q', κ'). As long as there is a computable reduction from Q' to Q, this does not make much of a difference, as we could translate the queries to Q' into queries of Q. Therefore, we prefer to use the definition given in [1].

Out-Branching: Showing the Difference

In [1], the first example of a natural problem is provided that admits a Turing kernel of polynomial size, but (most likely) no Karp kernel of polynomial size. We provide some details in the following.

Problem Definition

A subdigraph T of a digraph D is an *out-tree* if T is an oriented tree with only one vertex r of indegree zero (called the *root*). The vertices of T of outdegree zero are called *leaves*. If T is a spanning out-tree, i.e., $V(T) = V(D)$, then T is called an *out-branching* of D. The Directed Maximum Leaf Out-Branching problem is to find an out-branching in a given digraph with the maximum number of leaves. The parameterized version of the Directed Maximum Leaf Out-Branching problem is k-Leaf Out-Branching, where for a given digraph D and integer k, it is asked to decide whether D has an out-branching with at least k leaves. If we replace "out-branching" with "out-tree" in the definition of k-Leaf Out-Branching, we get a problem called k-Leaf

OUT-TREE. The parameterization κ is set to k in both problems. As the two problems are easily translatable into each other, we focus on k-LEAF OUT-BRANCHING as the digraph analogue of the well-known MAXIMUM LEAF SPANNING TREE problem.

Key Results

It is shown that the problem variant where an explicit root is given as additional input, called ROOTED k-LEAF OUT-BRANCHING, admits a polynomial Karp kernel. Alternatively, this variant can be seen as a special case of k-LEAF OUT-BRANCHING by adding one vertex of indegree zero and outdegree one, pointing to the designated root of the original graph. By making a call to this oracle for each of the vertices as potential roots, this provides a Turing kernelization of polynomial size for k-LEAF OUT-BRANCHING. This result is complemented by showing that k-LEAF OUT-TREE has no polynomial Karp kernel unless $coNP \subseteq NP/poly$.

We list the reduction rules leading to the polynomial-size kernel for the rooted version in the following.

Reachability Rule:If there exists a vertex u which is disconnected from the root r, then return NO.

Useless Arc Rule:If vertex u disconnects a vertex v from the root r, then remove the arc vu.

Bridge Rule:If an arc uv disconnects at least two vertices from the root r, contract the arc uv.

Avoidable Arc Rule:If a vertex set S, $|S| \leq 2$, disconnects a vertex v from the root r, $vw \in A(D)$ and $xw \in A(D)$ for all $x \in S$, then delete the arc vw.

Two Directional Path Rule:If there is a path $P = p_1 p_2 \ldots p_{l-1} p_l$ with $l = 7$ or $l = 8$ such that

- p_1 and $p_{\text{in}} \in \{p_{l-1}, p_l\}$ are the only vertices with in-arcs from the outside of P
- p_l and $p_{\text{out}} \in \{p_1, p_2\}$ are the only vertices with out-arcs to the outside of P
- The path P is the unique out-branching of $D[V(P)]$ rooted at p_1
- There is a path Q that is the unique out-branching of $D[V(P)]$ rooted at p_{in} and ending in p_{out}
- The vertex after p_{out} on P is not the same as the vertex after p_l on Q

then delete $R = P \setminus \{p_1, p_{\text{in}}, p_{\text{out}}, p_l\}$ and all arcs incident to these vertices from D. Add two vertices u and v and the arc set $\{p_{\text{out}}u, uv, vp_{\text{in}}, p_l v, vu, up_1\}$ to D.

This reduction was simplified and improved in [2] by replacing the rather complicated last reduction rule by a rule that shortens induced bipaths of length four to length two. Here, $P = \{x_1, \ldots, x_l\}$, with $l \geq 3$, is an *induced bipath of length* $l-1$ if the set of arcs neighbored to $\{x_2, \ldots, x_{l-1}\}$ in D is exactly $\{(x_i, x_{i+1}), (x_{i+1}, x_i) \mid i \in \{1, \ldots, l-1\}\}$. This yielded a Karp kernel with a quadratic number of vertices (measured in terms of the parameter k) for the rooted version. For directed acyclic graphs (DAGs), even a Karp kernel with a linear number of vertices is known for the rooted version [3]. Notice that also for DAGs (in fact, for quite restricted DAGs called *willow graphs*), the unrooted problem versions have no polynomial Karp kernel unless $coNP \subseteq NP/poly$, as suggested by the hardness proof in [1]. Another direction of research is to obtain faster kernelization algorithms, often by restricting the use (and power) of reduction rules. For the k-LEAF OUT-BRANCHING, this was done by Kammer [6].

Hierarchies Based on Turing Kernels

Based on the notion of *polynomial parametric transformation*, in [4] an intertwined WK/MK hierarchy was defined, in analogy to the well-known W/M hierarchy of (hard) parameterized problems. The lowest level (MK[1]) corresponds to (NP) problems with polynomial-size Karp kernels. The second-lowest level is WK[1], and this

does not equal MK[1] unless $coNP \subseteq NP/poly$. Typical complete problems for WK[1] are:

- Given a graph G of order n and an integer k, does G contain a clique of size k? Here, the parameterization is $\kappa(G,k) = k \cdot \log(n)$.
- Given a nondeterministic Turing machine M and an integer k, does M stop within k steps? Here, the parameterization is $\kappa(M,k) = k \cdot \log(|M|)$.

As noticed in [4], the CLIQUE problem provides also another (less natural) example of a problem without polynomial-size Karp kernel that has a polynomial-size Turing kernel, taking as parameterization the maximum degree of the input graph.

How Much Oracle Access Is Needed?

The examples we gave so far make use of oracles in a very simple way. More precisely, a very weak notion of truth-table reduction (disjunctive reduction) is applied. The INDEPENDENT SET problem on bull-free graphs [7] seems to provide a first example where the power of Turing reductions is used more extensively, as the oracle input is based on the previous computation of the reduction. Therefore, it could be termed an adaptive kernelization [5]. Yet another way of constructing Turing kernels was described by Jansen [5]. There, in a first step, the instance is decomposed (according to some graph decomposition in that case), and then the fact is used that either a solution is already obtained or it only exists in one of the (small) components of the decomposition. This framework is then applied to deduce polynomial-size Turing kernels, e.g., for the problem of finding a path (or a cycle) of length at least k in a planar graph G, where k is the parameter of the problem.

Open Problems

One of the most simple open questions is whether LONGEST PATH, i.e., the problem of finding a path of length at least k, admits a polynomial-size Turing kernel on general graphs.

Conversely, no tools have been developed so far that allow for ruling out polynomial-size Turing kernels. For the question of practical applications of kernelization, this would be a much stronger statement than ruling out traditional Karp kernels of polynomial size, as a polynomial number of polynomial-size kernels can give a practical solution (see the discussion of k-LEAF OUT-BRANCHING above).

Cross-References

- ▸ Enumeration of Paths, Cycles, and Spanning Trees
- ▸ Kernelization, Exponential Lower Bounds

Recommended Reading

1. Binkele-Raible D, Fernau H, Fomin FV, Lokshtanov D, Saurabh S, Villanger Y (2012) Kernel(s) for problems with no kernel: on out-trees with many leaves. ACM Trans Algorithms 8(4):38
2. Daligault J, Thomassé S (2009) On finding directed trees with many leaves. In: Chen J, Fomin FV (eds) Parameterized and exact computation, 4th international workshop, IWPEC, Copenhagen. LNCS, vol 5917. Springer, pp 86–97
3. Daligault J, Gutin G, Kim EJ, Yeo A (2010) FPT algorithms and kernels for the directed k-leaf problem. J Comput Syst Sci 76(2):144–152
4. Hermelin D, Kratsch S, Soltys K, Wahlström M, Wu X (2013) A completeness theory for polynomial (Turing) kernelization. In: Gutin G, Szeider S (eds) Parameterized and exact computation – 8th international symposium, IPEC, Sophia Antipolis. LNCS, vol 8246. Springer, pp 202–215
5. Jansen BMP (2014) Turing kernelization for finding long paths and cycles in restricted graph classes. Tech. Rep. 1402.4718v1, arXiv.CS.DS
6. Kammer F (2013) A linear-time kernelization for the rooted k-leaf outbranching problem. In: Brandstädt A, Jansen K, Reischuk R (eds) Graph-theoretic concepts in computer science – 39th international workshop, WG, Lübeck. LNCS, vol 8165. Springer, pp 310–320
7. Thomassé S, Trotignon N, Vuskovic K (2013) Parameterized algorithm for weighted independent set problem in bull-free graphs. CoRR abs/1310.6205, a conference version appeared at WG (LNCS volume) 2014

Kinetic Data Structures

Bettina Speckmann
Department of Mathematics and Computer Science, Technical University of Eindhoven, Eindhoven, The Netherlands

Years and Authors of Summarized Original Work

1999; Basch, Guibas, Hershberger

Problem Definition

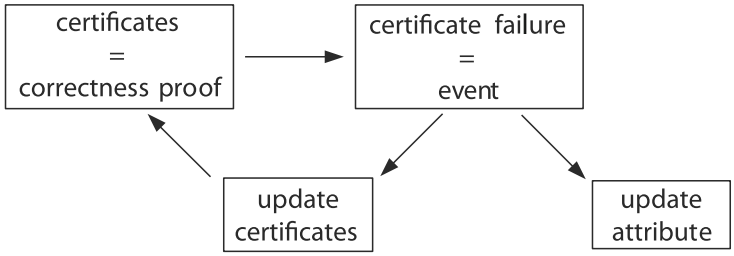
Many application areas of algorithms research involve objects in motion. Virtual reality, simulation, air-traffic control, and mobile communication systems are just some examples. Algorithms that deal with objects in motion traditionally discretize the time axis and compute or update their structures based on the position of the objects at every time step. If all objects move continuously then in general their configuration does not change significantly between time steps – the objects exhibit *spatial* and *temporal coherence*. Although *time-discretization* methods can exploit spatial and temporal coherence they have the disadvantage that it is nearly impossible to choose the perfect time step. If the distance between successive steps is too large, then important interactions might be missed, if it is too small, then unnecessary computations will slow down the simulation. Even if the time step is chosen just right, this is not always a satisfactory solution: some objects may have moved only slightly and in such a way that the overall data structure is not influenced.

One would like to use the temporal coherence to detect precisely those points in time when there is an actual change in the structure. The *kinetic data structure* (KDS) framework, introduced by Basch et al. in their seminal paper [2], does exactly that: by maintaining not only the structure itself, but also some additional information, they can determine when the structure will undergo a "real" (combinatorial) change.

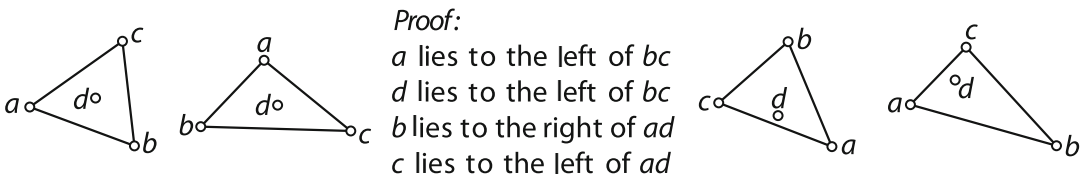
Key Results

A kinetic data structure is designed to maintain or monitor a discrete attribute of a set of moving objects, for example, the convex hull or the closest pair. The basic idea is, that although all objects move continuously, there are only certain discrete moments in time when the combinatorial structure of the attribute changes (in the earlier examples, the ordered set of convex-hull vertices or the pair that is closest, respectively). A KDS therefore contains a set of *certificates* that constitutes a proof of the property of interest. Certificates are generally simple inequalities that assert facts like "point c is on the left of the directed line through points a and b." These certificates are inserted in a priority queue (*event queue*) based on their time of expiration. The KDS then performs an event-driven simulation of the motion of the objects, updating the structure whenever an *event* happens, that is, when a certificate fails (see Fig. 1). It is part of the art of designing efficient kinetic data structures to find a small set of simple and easily updatable certificates that serve as a proof of the property one wishes to maintain.

A KDS assumes that each object has a known motion trajectory or *flight plan*, which may be subject to restrictions to make analysis tractable. Two common restrictions would be translation along paths parametrized by polynomials of fixed degree d, or translation and rotation described by algebraic curves. Furthermore, certificates are generally simple algebraic equations, which implies that the failure time of a certificate can be computed as the next largest root of an algebraic expression. An important aspect of kinetic data structures is their on-line character: although the positions and motions (flight plans) of the objects are known at all times, they are not necessarily known far in advance. In particular, any object can change its flight plan at any time. A good KDS should be able to handle such changes in flight plans efficiently.

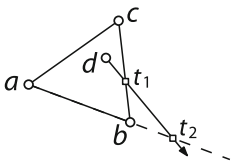


Kinetic Data Structures, Fig. 1 The basic structure of an event based simulation with a KDS



Kinetic Data Structures, Fig. 2 Equivalent convex hull configurations (*left* and *right*), a proof that a, b, and c form the convex hull of S (*center*)

Kinetic Data Structures, Fig. 3 Certificate structure for points a, b, and c being stationary and point d moving along a straight line



Certificate	Failure time
a lies to the left of bc	never
d lies to the left of bc	t_1
b lies to the right of ad	t_2
c lies to the left of ad	never

A detailed introduction to kinetic data structures can be found in Basch's Ph. D. thesis [1] or in the surveys by Guibas [3, 4]. In the following the principles behind kinetic data structures are illustrated by an easy example.

Consider a KDS that maintains the convex hull of a set S of four points a, b, c, and d as depicted in Fig. 2. A set of four simple certificates is sufficient to certify that a, b, and c form indeed the convex hull of S (see Fig. 2 center). This implies, that the convex hull of S will not change under any motion of the points that does not lead to a violation of these certificates. To put it differently, if the points move along trajectories that move them between the configurations depicted in Fig. 2 without the point d ever appearing on the convex hull, then the KDS in principle does not have to process a single event.

Now consider a setting in which the points a, b, and c are stationary and the point d moves along a linear trajectory (Fig. 3 left). Here the KDS has exactly two events to process. At time t_1 the certificate "d is to the left of bc" fails as the point d appears on the convex hull. In this easy setting, only the failed certificate is replaced by "d is to the *right* of bc" with failure time "never", generally processing an event would lead to the scheduling and descheduling of several events from the event queue. Finally at time t_2 the certificates "b is to the right of ad" fails as the point b ceases to be on the convex hull and is replaced by "b is to the *left* of ad" with failure time "never."

Kinetic data structures and their accompanying maintenance algorithms can be evaluated and compared with respect to four desired characteristics.

Responsiveness. One of the most important performance measures for a KDS is the time needed to update the attribute and to repair the certificate set when a certificate fails. A KDS is called *responsive* if this update time is "small", that is, polylogarithmic.

Compactness. A KDS is called *compact* if the number of certificates is near-linear in the

total number of objects. Note that this is not necessarily the same as the amount of storage the entire structure needs.

Locality. A KDS is called *local* if every object is involved in only a small number of certificates (again, "small" translates to polylogarithmic). This is important whenever an object changes its flight plane, because one has to recompute the failure times of all certificates this object is involved in, and update the event queue accordingly. Note that a local KDS is always compact, but that the reverse is not necessarily true.

Efficiency. A certificate failure does not automatically imply a change in the attribute that is being maintained, it can also be an *internal event*, that is, a change in some auxiliary structure that the KDS maintains. A KDS is called *efficient* if the worst-case number of events handled by the data structure for a given motion is small compared to the number of combinatorial changes of the attribute (*external events*) that must be handled for that motion.

Applications

The paper by Basch et al. [2] sparked a large amount of research activities and over the last years kinetic data structures have been used to solve various dynamic computational geometry problems. A number of papers deal foremost with the maintenance of discrete attributes for sets of moving points, like the closest pair, width and diameter, clusters, minimum spanning trees, or the constrained Delaunay triangulation. Motivated by ad hoc mobile networks, there have also been a number of papers that show how to maintain the connected components in a set of moving regions in the plane. Major research efforts have also been seen in the study of kinetic binary space partitions (BSPs) and kinetic kd-trees for various objects. Finally, there are several papers that develop KDSs for collision detection in the plane and in three dimensions. A detailed discussion and an extensive list of references can be found in the survey by Guibas [4].

Cross-References

▸ Fully Dynamic Minimum Spanning Trees
▸ Minimum Geometric Spanning Trees

Recommended Reading

1. Basch J (1999) Kinetic data structures. PhD thesis, Stanford University
2. Basch J, Guibas L, Hershberger J (1999) Data structures for mobile data. J Algorithms 31:1–28
3. Guibas L (1998) Kinetic data structures: a state of the art report. In: Proceedings of 3rd workshop on algorithmic foundations of robotics, pp 191–209
4. Guibas L (2004) Modeling motion. In: Goodman J, O'Rourke J (eds) Handbook of discrete and computational geometry, 2nd edn. CRC

Knapsack

Hans Kellerer
Department of Statistics and Operations Research, University of Graz, Graz, Austria

Keywords

Approximation algorithm; Fully polynomial time approximation scheme (FPTAS)

Years and Authors of Summarized Original Work

2000; Ibarra, Kim

Problem Definition

For a given set of items $N = \{1, \ldots, n\}$ with nonnegative integer weights w_j and profits p_j, $j = 1, \ldots, n$, and a knapsack of capacity c, the *knapsack problem* (KP) is to select a subset of the items such that the total profit of the selected items is maximized and the corresponding total weight does not exceed the knapsack capacity c.

Alternatively, a knapsack problem can be formulated as a solution of the following linear integer programming formulation:

$$(KP) \text{ maximize } \sum_{j=1}^{n} p_i x_j \tag{1}$$

$$\text{subject to } \sum_{j=1}^{n} w_j x_j \leq c, \tag{2}$$

$$x_j \in (0,1), \; j = 1, \ldots, n. \tag{3}$$

The knapsack problem is the simplest nontrivial integer programming model having binary variables, only a single constraint, and only positive coefficients. A large number of theoretical and practical papers have been published on this problem and its extensions. An extensive overview can be found in the books by Kellerer, Pferschy, and Pisinger [4] or Martello and Toth [7].

Adding the integrality condition (3) to the simple linear program (1)–(2) already puts (KP) into the class of $\mathcal{NP}$-hard problems. Thus, (KP) admits no polynomial time algorithms unless $\mathcal{P} = \mathcal{NP}$ holds.

Therefore, this entry will focus on approximation algorithms for (KP). A common method to judge the quality of an approximation algorithm is its worst-case performance. For a given instance I, define by $z^*(I)$ the optimal solution value of (KP) and by $z^H(I)$ the corresponding solution value of a heuristic H. For $\varepsilon \in [0,1[$, a heuristic H is called a $(1-\varepsilon)$-*approximation algorithm* for (KP) if for any instance I

$$z^H(I) \geq (1-\varepsilon) z^*(I)$$

holds. Given a parameter ε, a heuristic H is called a *fully polynomial approximation scheme*, or an FTPAS, if H is a $(1-\varepsilon)$-approximation algorithm for (KP) for any $\varepsilon \in [0,1[$, and its running time is polynomial both in the length of the encoded input n and $1/\varepsilon$. The first FTPAS for (KP) was suggested by Ibarra and Kim [1] in 1975. It was among the early FPTASes for discrete optimization problems. It will be described in detail in the following.

Key Results

(KP) can be solved in pseudopolynomial time by a simple dynamic programming algorithm. One possible variant is the so-called *dynamic programming by profits* (DP-Profits). The main idea of DP-Profits is to reach every possible total profit value with a subset of items of minimal total weight. Clearly, the highest total profit value, which can be reached by a subset of weight not greater than the capacity c, will be an optimal solution.

Let $y_j(q)$ denote the minimal weight of a subset of items from $\{1, \ldots, j\}$ with total profit equal to q. To bound the length of every array y_j, an upper bound u on the optimal solution value has to be computed. An obvious possibility would be to use the upper bound $U_{\mathrm{LP}} = \lfloor z^{\mathrm{LP}} \rfloor$ from the solution z^{LP} of the LP-relaxation of (KP) and set $U := U_{\mathrm{LP}}$. It can be shown that U_{LP} is at most twice as large as the optimal solution value z^*. Initializing $y_0(0) := 0$ and $y_0(q) := c+1$ for $q = 1, \ldots, U$, all other values can be computed for $j = 1, \ldots, n$ and $q = 0, \ldots, U$ by using the recursion

$$y_i(q) := \begin{cases} y_{i-1}(q) & \text{if } q < p_j, \\ \min\left(y_{i-1}(q), \left(y_{i-1}(q)\right)\right) & \text{if } q \geq p_j. \end{cases}$$

The optimal solution value is given by $\max\{q | y_n(q) \leq c\}$ and the running time of DP-Profits is bounded by $O(nU)$.

Theorem 1 (Ibarra, Kim) *There is an FTPAS for (KP) which runs in $O(n \log n + n/\varepsilon^2)$ time.*

Proof The FTPAS is based on appropriate *scaling* of the profit values p_j and then running DP-Profits with the scaled profit values. Scaling means here that the given profit values p_j are replaced by new profits $\tilde{p}_j$ such that $\tilde{p}_j := \left\lfloor \frac{p_j}{K} \right\rfloor$ for an appropriate chosen constant K.

This scaling can be seen as a partitioning of the profit range into intervals of length K with starting points $0, K, 2K, \ldots$. Naturally, for every profit value p_j, there is some integer value $i \geq 0$ such that p_j falls into the interval $[iK, (i+1)K[$. The scaling procedure generates for every p_j the

value $\tilde{p}_j$ as the corresponding index i of the lower interval bound iK.

Running DP-Profits yields a solution set $\tilde{X}$ for the scaled items which will usually be different from the original optimal solution set X^*. Evaluating the original profits of item set $\tilde{X}$ yields the approximate solution value z^H . The difference between z^H and the optimal solution value can be bounded as follows:

$$z^H \geq \sum_{j \in \tilde{X}} K \left\lfloor \frac{p_j}{K} \right\rfloor \geq \sum_{j \in \tilde{X}*} K \left\lfloor \frac{p_j}{K} \right\rfloor$$
$$\geq \sum_{j \in \tilde{X}*} K \left(\frac{p_j}{K} - 1 \right) = z* - |X*|\, K.$$

To get the desired performance guarantee of $1-\varepsilon$, it is sufficient to have

$$\frac{z* - z^H}{z*} \leq \frac{|X*|\, K}{z*} \leq \varepsilon.$$

To ensure this, K has to be chosen such that

$$K \leq \frac{\varepsilon z^*}{|X*|}. \tag{4}$$

Since $n \geq |X^*|$ and $U_{\mathrm{LP}}/2 \leq z^*$, choosing $K := \frac{\varepsilon U_{\mathrm{LP}}}{2n}$ satisfies condition (4) and thus guarantees the performance ratio of $1-\varepsilon$. Substituting U in the $O(nU)$ bound for DP-Profits by U/K yields an overall running time of $O(n^2\varepsilon)$.

A further improvement in the running time is obtained in the following way. Separate the items into *small* items (having profit $\leq \frac{\varepsilon}{2} U_{\mathrm{LP}}$) and *large* items (having profit $> \frac{\varepsilon}{2} U_{\mathrm{LP}}$). Then, perform DP-Profits for the scaled large items only. To each entry q of the obtained dynamic programming array with corresponding weight $y(q)$, the small items are added to a knapsack with residual capacity $c - y(q)$ in a greedy way. The small items shall be sorted in nonincreasing order of their profit to weight ratio. Out of the resulting combined profit values, the highest one is selected. Since every optimal solution contains at most $2/\varepsilon$ large items, $|X^*|$ can be replaced in (4) by $2/\varepsilon$ which results in an overall running time $O(n \log n + n/\varepsilon^2)$. The memory requirement of the algorithm is $O(n + 1/\varepsilon^3)$.

Two important approximation schemes with advanced treatment of items and algorithmic fine-tuning were presented some years later. The classical paper by Lawler [5] gives a refined scaling resp. partitioning of the items and several other algorithmic improvements which results in a running time $O(n \log(1/\varepsilon) + 1/\varepsilon^4)$. A second paper by Magazine and Oguz [6] contains among other features a partitioning and recombination technique to reduce the space requirements of the dynamic programming procedure. The fastest algorithm is due to Kellerer and Pferschy [2, 3] with running time $O(n \min\{\log n, \log(1/\varepsilon)\} + 1/\varepsilon^2 \log(1/\varepsilon) \cdot \min\{n, 1/\varepsilon \log(1/\varepsilon)\})$ and space requirement $O(n + 1/\varepsilon^2)$.

Applications

(KP) is one of the classical problems in combinatorial optimization. Since (KP) has this simple structure and since there are efficient algorithms for solving it, many solution methods of more complex problems employ the knapsack problem (sometimes iteratively) as a subproblem.

A straightforward interpretation of (KP) is an investment problem. A wealthy individual or institutional investor has a certain amount of money c available which he wants to put into profitable business projects. As a basis for his decisions, he compiles a long list of possible investments including for every investment the required amount w_j and the expected net return p_j over a fixed period. The aspect of risk is not explicitly taken into account here. Obviously, the combination of the binary decisions for every investment such that the overall return on investment is as large as possible can be formulated by (KP).

One may also view the (KP) as a "cutting" problem. Assume that a sawmill has to cut a log into shorter pieces. The pieces must however be cut into some predefined standard-lengths w_j , where each length has an associated selling price p_j . In order to maximize the profit of the log, the sawmill can formulate the problem as a (KP) where the length of the log defines the capacity c.

Among the wide range of "real-world" applications shall be mentioned two-dimensional

cutting problems, column generation, separation of cover inequalities, financial decision problems, asset-backed securitization, scheduling problems, knapsack cryptosystems, and most recent combinatorial auctions. For a survey on applications of knapsack problems, the reader is referred to [4].

Recommended Reading

1. Ibarra OH, Kim CE (1975) Fast approximation algorithms for the knapsack and sum of subset problem. J ACM 22:463–468
2. Kellerer H, Pferschy U (1999) A new fully polynomial time approximation scheme for the knapsack problem. J Comb Optim 3:59–71
3. Kellerer H, Pferschy U (2004) Improved dynamic programming in connection with an FPTAS for the knapsack problem. J Comb Optim 8:5–11
4. Kellerer H, Pisinger D, Pferschy U (2004) Knapsack problems. Springer, Berlin
5. Lawler EL (1979) Fast approximation algorithms for knapsack problems. Math Oper Res 4:339–356
6. Magazine MJ, Oguz O (1981) A fully polynomial approximation algorithm for the 0-1 knapsack problem. Eur J Oper Res 8:270–273
7. Martello S, Toth P (1990) Knapsack problems: algorithms and computer implementations. Wiley, Chichester

Knowledge in Distributed Systems

Yoram Moses
Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa, Israel

Keywords

Common knowledge; Coordinated Attack; Distributed computing; Knowledge; Knowledge gain; Message chain; Potential causality

Years and Authors of Summarized Original Work

1984; Halpern, Moses
1986; Chandy, Misra

Problem Definition

What is the role of knowledge in distributed computing?

Actions taken by a process in a distributed system can only be based on its local information or local *knowledge*. Indeed, in reasoning about distributed protocols, people often talk informally about what processes know about the state of the system and about the progress of the computation. Can the informal reasoning about knowledge in distributed and multi-agent systems be given a rigorous mathematical formulation, and what uses can this have?

Key Results

In [4] Halpern and Moses initiated a theory of knowledge in distributed systems. They suggested that states of knowledge ascribed to groups of processes, especially *common knowledge*, have an important role to play. Knowledge-based analysis of distributed protocols has generalized well-known results and enables the discovery of new ones. These include new efficient solutions to basic problems, tools for relating results in different models, and proving lower bounds and impossibility results. For example, the inability to attain common knowledge when communication is unreliable was established in [4] and shown to imply and generalize the Coordinated Attack problem. Chandy and Misra showed in [1] that in asynchronous systems there is a tight connection between the manner in which knowledge is gained or lost and the message chains that underly Lamport's notion of potential causality.

Modeling Knowledge

In philosophy, knowledge is often modeled by so-called *possible-worlds* semantics. Roughly speaking, at a given "world," an agent will know a given fact to be true precisely if this fact holds at all worlds that the agent "considers possible." In a distributed system, the agents are processes or computing elements. A simple language

for reasoning about knowledge is obtained by starting out with a set $\boldsymbol{\Phi} = \{p, q, p', q' \ldots\}$ of propositions, or basic facts. The facts in Φ will depend on the application we wish to study; they may involve statements such as $x = 0$ or $x > y$ concerning values of variables or about other aspects of the computation (e.g., in the analysis of mutual exclusion, a proposition $\mathsf{CS}(i)$ may be used to state that process i is in the critical section). We obtain a logical language $\mathcal{L}_n^K = \mathcal{L}_n^K(\Phi)$ for knowledge, which is a set of formulas, by the following inductive definition. First, $p \in \mathcal{L}_n^K$ for all propositions $p \in \Phi$. Moreover, for all formulas $\varphi, \psi \in \mathcal{L}_n^K$, the language contains the formulas $\neg\varphi$ (standing for "*not* φ"), $\varphi \wedge \psi$ (standing for "φ *and* ψ"), and $K_i\varphi$ ("*process* i *knows* φ"), for every process $i \in \{1, \ldots, n\}$ (Using the operators "$\neg$" and "$\wedge$," we can express all of the Boolean operators. Thus, $\varphi \vee \psi$ ("φ *or* ψ") can be expressed as $\neg(\neg\varphi \wedge \neg\psi)$, while $\varphi \Rightarrow \psi$ ("φ *implies* ψ") is $\neg\varphi \vee \psi$, etc.). The language $\mathcal{L}_n^K$ is the basis of a propositional logic of knowledge. Using it, we can make formulas such as $K_1\mathsf{CS}(1) \wedge K_1K_2\neg\mathsf{CS}(2)$, which states that "*process 1 knows that it is in the critical section, and it knows that process 2 knows that 2 is not in the critical section*." $\mathcal{L}_n^K$ determines the *syntax* of formulas of the logic. A mathematical definition of what the formulas mean is called its *semantics*.

A process will typically know different things in different computations; even within a computation, its knowledge changes over time as a result of communication and of observing various events. We refer to time t in a run (or computation) r by the pair $\boldsymbol{(r, t)}$, which is called a *point*. Formulas are considered to be true or false at a point (r, t), with respect to a set of runs R (we call R a *system*). The set of points of R is denoted by $\boldsymbol{\mathsf{Pts}(R)}$.

The definition of knowledge is based on the idea that at any given point (r, t), each process has a well-defined *view*, which depends on i's history up to time t in r. This view may consist of all events that i has observed, or on a much more restricted amount of information, which is considered to be available to i at (r, t). In the language of [3], this view can be thought of as being process i's *local state* at (r, t), which we denote by $\boldsymbol{r_i(t)}$. Intuitively, a process is assumed to be able to distinguish two points iff its local state at one is different from its state in the other. In a given system R, the meaning of the propositions in a set Φ needs to be defined explicitly. This is done by way of an *interpretation* $\pi : \Phi \times \mathsf{Pts}(R) \rightarrow \{\mathsf{True}, \mathsf{False}\}$. The pair $\boldsymbol{\mathcal{I} = (R, \pi)}$ is called an *interpreted system*. We denote the fact that φ is satisfied, or true, at a point (r, t) in the system $\mathcal{I}$ by $(\mathcal{I}, r, t) \models \varphi$. Semantics of $\mathcal{L}_n^K(\Phi)$ with respect to an interpreted system $\mathcal{I}$ is given by defining the satisfaction relation "$\models$" defined by induction on the structure of the formulas, as follows:

$(\mathcal{I}, r, t) \models p$ *iff* $\pi\big(p, (r, t)\big) = \mathsf{True}$, for a proposition $p \in \Phi$

$(\mathcal{I}, r, t) \models \neg\varphi$ *iff* $(\mathcal{I}, r, t) \not\models \varphi$

$(\mathcal{I}, r, t) \models \varphi \wedge \psi$ *iff* both $(\mathcal{I}, r, t) \models \varphi$ and $(\mathcal{I}, r, t) \models \psi$

$(\mathcal{I}, r, t) \models K_i\varphi$ *iff* $(\mathcal{I}, r', t') \models \varphi$ whenever $r_i'(t') = r_i(t)$ and $(r', t') \in \mathsf{Pts}(R)$

The fourth clause, which defines satisfaction for knowledge formulas, can be applied repeatedly. This gives meaning to formulas involving knowledge about formulas that themselves involve knowledge, such as $K_2\big(\mathsf{CS}(2) \wedge \neg K_1\neg\mathsf{CS}(2)\big)$. Knowledge here is ascribed to processes. The intuition is that the local state captures all of the information available to the process. If there is a scenario leading to another point at which a fact φ is false and the process has the same state as it has not, then the process does not know φ.

This notion of knowledge has fairly strong properties that distinguish it from what one might

consider a reasonable notion of, say, human knowledge. For example, it does not depend on computation, thoughts, or a derivation of what the process knows. It is purely "information based." Indeed, any fact that holds at all elements of $\mathsf{Pts}(R)$ (e.g., the protocol that processes are following) is automatically known to all processes. Moreover, it is not assumed that a process can report its knowledge or that its knowledge is explicitly recorded in the local state. This notion of knowledge can be thought of as being ascribed to the processes by an external observer and is especially useful for analysis by a protocol designer.

Common Knowledge and Coordinated Attack

A classic example of a problem for which the knowledge terminology can provide insight is Jim Gray's *Coordinated Attack* problem. We present it in the style of [4]:

The Coordinated Attack Problem

Two divisions of an army are camped on two hilltops, and the enemy awaits in the valley below. Neither general will decide to attack unless he is sure that the other will attack with him, because only a simultaneous attack guarantees victory. The divisions do not initially have plans to attack, and one of the commanding generals wishes to coordinate a simultaneous attack (at some time the next day). The generals can only communicate by means of a messenger. Normally, it takes the messenger 1 h to get from one encampment to the other. However, the messenger can lose his way or be captured by the enemy. Fortunately, on this particular night, everything goes smoothly. How long will it take them to coordinate an attack?

It is possible to show by induction that k trips of the messenger do not suffice, for all $k \geq 0$, and hence the generals will be unable to attack. Gray used this example to illustrate the impact of unreliable communication on the ability to consistently update distinct sites of a distributed database. A much stronger result that generalizes this and applies directly to practical problems can be obtained based on a notion called *common knowledge*. Given a group $G \subseteq \{1, \ldots, n\}$ of processes, we define two new logical operators E_G and C_G, corresponding to *everyone (in G) knows* and *is common knowledge in G*, respectively. We shall denote $E_G^1\varphi = E_G\varphi$ and inductively define $E_G^{k+1}\varphi = E_G(E_G^k\varphi)$. Satisfaction for the new operators is given by

$$(\mathcal{I}, r, t) \models E_G\varphi \ \textbf{\textit{iff}} \ (\mathcal{I}, r, t) \models K_i\varphi \text{ holds for all } i \in G$$
$$(\mathcal{I}, r, t) \models C_G\varphi \ \textit{iff} \ (\mathcal{I}, r, t) \models E_G^k\varphi \text{ holds for all } k \geq 1$$

Somewhat surprisingly, common knowledge is not uncommon in practice. People shake hands to signal that they attained common knowledge of an agreement, for example. Similarly, a public announcement to a class or to an audience is considered common knowledge. Indeed, as we now discuss, simultaneous actions can lead to common knowledge.

Returning to the Coordinated Attack problem, consider three propositions attack_A, attack_B, and $\mathsf{delivered}$, corresponding, respectively, to "*general A is attacking*," "*general B is attacking*," and "*at least one message has been delivered*." The fact that the generals do not have a plan to attack can be formalized by saying that at least one of them does not attack unless $\mathsf{delivered}$ is true. Consider a set of runs R consisting of all possible interactions of the generals in the above setting. Suppose that the generals follow the specifications, so they only ever attack simultaneously at points of R. Then, roughly speaking, since the generals' actions depend on their local state, general A knows when attack_A is true. But since they only attack simultaneously and attack_B is true whenever attack_B is true, $K_A\mathsf{attack}_B$ will hold whenever general A attacks. Since B similarly knows when A attacks, $K_BK_A\mathsf{attack}_B$ will hold as well. Indeed, it can be shown that when the generals attack in a system that guarantees that attacks are simultaneous,

they must have common knowledge that they are attacking.

Theorem 1 (Halpern and Moses [4]) *Let R be a system with unreliable communication, let $\mathcal{I} = (R, \pi)$, let $(r, t) \in \mathsf{Pts}(R)$, and assume that $|G| > 1$. Then $(\mathcal{I}, r, t) \models \neg C_G \mathsf{delivered}$.*

As in the case of Coordinated Attack, simultaneous actions must be common knowledge when they are performed. Moreover, in cases in which such actions require a minimal amount of communication to be materialize, $C_G \mathsf{delivered}$ must hold when they are performed. Theorem 1 implies that no such actions can be coordinated when communication is unreliable. One immediate consequence is:

Corollary 1 *Under a protocol that satisfies the constraints of the Coordinated Attack problem, the generals never attack.*

The connection between common knowledge and simultaneous actions goes even deeper. It can be shown that when a fact that is not common knowledge to G becomes common knowledge to G, a state transition must occur simultaneously at all sites in G. If simultaneity cannot be coordinated in the system, common knowledge cannot be attained. This raises some philosophical issues: Events and transitions that are viewed as being simultaneous in a system that is modeled at a particular ("coarse") granularity of time will fail to be simultaneous when time is modeled at a finer granularity. As discussed in [4], this is not quite a paradox, since there are many settings in which it is acceptable, and even desirable, to model interactions at a granularity of time in which simultaneous transitions do occur.

A Hierarchy of States of Knowledge and Common Knowledge

Common knowledge is a much stronger state of knowledge than, say, knowledge of an individual process. Indeed, it is best viewed as a state of knowledge of a group. There is an essential difference between E_G^k (everyone knows that everyone knows, for k levels), even for large k, and C_G (common knowledge). Indeed, for every k, there are examples of tasks that can be achieved if $E_G^{k+1}\varphi$ holds but not if $E_G^k\varphi$ does. This suggests the existence of a hierarchy of states of group knowledge, ranging from $E_G\varphi$ to $C_G\varphi$. But it is also possible to define natural states of knowledge for a group that are weaker than these. One is S_G, where $S_G\varphi$ is true if $\bigvee_{i \in G} K_i\varphi$ – *someone in G knows φ*. Even weaker is *distributed knowledge*, denoted by D_G, which is defined by

$$(\mathcal{I}, r, t) \models D_G\varphi \text{ iff } (\mathcal{I}, r', t') \models \varphi \text{ for all } (r', t') \text{ satisfying } r_i'(t') = r_i(t) \text{ for all } i \in G$$

Roughly speaking, the distributed knowledge of a group corresponds to what follows from the combined information of the group at a given instant. Thus, for example, if all processes start out with initial value 1, they will have distributed knowledge of this fact, even if no single process knows this individually. Halpern and Moses propose a hierarchy of states of group knowledge and suggest that communication can often be seen as the process of moving the state of knowledge up the hierarchy:

$$C_G\varphi \Rightarrow E_G^{k+1}\varphi \Rightarrow E_G^k\varphi \Rightarrow \cdots \Rightarrow E_G\varphi \Rightarrow S_G\varphi \Rightarrow D_G\varphi.$$

Knowledge Gain and Loss in Asynchronous Systems

In asynchronous systems there are no guarantees about the pace at which communication is delivered and no guarantees about the relative rates at which processes operate. This motivated Lamport's definition of the happened-before relation among events. It is based on the intuition that in asynchronous systems only information obtained via message chains can affect the activity at a

given site. A crisp formalization of this intuition was discovered by Chandy and Misra in [1]:

Theorem 2 (Chandy and Misra) *Let $\mathcal{I}$ be an asynchronous interpreted system, let $\varphi \in \mathcal{L}_n^K$, and let $t' > t$. Then*

Knowledge Gain: *If $(\mathcal{I}, r, t) \not\models K_j \varphi$ and $(\mathcal{I}, r, t') \models K_{i_m} K_{i_{m-1}} \cdots K_{i_1} \varphi$, then there is a message chain through processes $\langle i_1, i_2, \ldots, i_m \rangle$ in r between times t and t'.*

Knowledge Loss: *If $(\mathcal{I}, r, t) \models K_{i_m} K_{i_{m-1}} \cdots K_{i_1} \varphi$ and $(\mathcal{I}, r, t') \not\models \varphi$, then there is a message chain through processes $\langle i_m, i_{m-1}, \ldots, i_1 \rangle$ in r between times t and t'.*

Note that the second clause implies that sending messages can cause a process to lose knowledge about other sites. Roughly speaking, the only way a process can know a nontrivial fact about a remote site is if this fact can only be changed by explicit permission from the process.

Applications and Extensions

The knowledge framework has been used in several ways. We have already seen its use for proving impossibility results in the discussion of the Coordinated Attack example. One interesting use of the formalism is as a tool for expressing *knowledge-based* protocols, in which programs can contain tests such as *if* K_i(*msg received*) *then*..., Halpern and Zuck, for example, showed that distinct solutions to the sequence transmission problem under different assumptions regarding communication faults were all implementations of the same knowledge-based protocol [5].

A knowledge-based analysis can lead to the design of efficient, sometimes optimal, distributed protocols. Dwork and Moses analyzed when facts become common knowledge when processes can crash and obtained an efficient and optimal solution to simultaneous consensus in which decisions are taken when initial values become common knowledge [2]. Moses and Tuttle showed that in a slightly harsher failure model, similar optimal solutions exist, but they are not computationally efficient, because computing when values are common knowledge is NP-hard [7]. A thorough exposition of reasoning about knowledge in a variety of fields including distributed systems, game theory, and philosophy appears in [3], while a later discussion of the role of knowledge in coordination, with further references, appears in [6].

Cross-References

- Byzantine Agreement
- Causal Order, Logical Clocks, State Machine Replication
- Distributed Snapshots

Recommended Reading

1. Chandy KM, Misra J (1986) How processes learn. Distrib Comput 1(1):40–52
2. Dwork C, Moses Y (1990) Knowledge and common knowledge in a Byzantine environment: crash failures. Inf Comput 88(2):156–186
3. Fagin R, Halpern JY, Moses Y, Vardi MY (2003) Reasoning about knowledge. MIT, Cambridge, MA
4. Halpern JY, Moses Y (1990) Knowledge and common knowledge in a distributed environment. J ACM 37(3):549–587. A preliminary version appeared in Proceeding of the 3rd ACM symposium on principles of distributed computing, 1984
5. Halpern JY, Zuck LD (1992) A little knowledge goes a long way: knowledge-based derivations and correctness proofs for a family of protocols. J ACM 39(3):449–478
6. Moses Y (2015) Relating knowledge and coordinated action: the knowledge of preconditions principle. In: Proceedings of the 15th TARK conference, Pittsburgh, pp 207–215
7. Moses Y, Tuttle MR (1988) Programming simultaneous actions using common knowledge. Algorithmica 3:121–169