Beyond Baker’s Technique

Yi-Kai Wang

Abstract

Baker’s technique, which was created over three decades ago, is a powerful tool for designing polynomial time approximation schemes (PTAS) for NP-hard optimization problems on planar graphs and their generalizations. In this paper, we propose a unified framework to formulate the optimization problems where the local constraints of these problems are encoded by functions attached on the vertices. This framework has much stronger ability for modelling real world problems than classic combinatorial optimization problems. We prove that when the function $f_i$ attached on $i \in V$ is a liberal function ($f_i \geq 0$) for all $i \in V$, then there is a PTAS for computing the max-sum of $f_i$ on planar graphs. We also prove that computing the min-sum of $f_i$ does not admit PTAS even on planar graphs unless P = NP. But if the set of liberal functions satisfies the balance property, we have a PTAS for computing the min-sum. The approximation algorithms for computing max-sum and min-sum imply that the max-product and min-product can also be well approximated in many cases. These results are further generalized to graphs with bounded local treewidth, H-minor-free graphs, $d$-dimensional geometric graphs with bounded density and graphs with bounded number of crossings per edge. Our results lead to PTASs for MAX-CUT, MAX-DICUT, MAX-$k$-CUT on these graphs. We also prove that if the corresponding factor graph of a CNF formula $\varphi$ can be transformed into these graphs through $X - F$ contractions, then computing the MAX-SAT of $\varphi$ has a PTAS. This result can be extended to MAX-CSP problems in a similar way. Our technique generalizes Baker’s technique and many existing methods of graph decomposition.

Our results make contributions to many important computation problems in various fields such as communication scheduling in wireless networks, task allocation for large-scale distributed database systems, MAP inference on graphical models, energy minimization in statistical physics and many applications in computer vision.

*State Key Laboratory for Novel Software Technology, Nanjing University, China. Email: wangyikai1029@gmail.com
†This work was proposed and finished when I was a full-time research intern at Microsoft Research Asia.
1 Introduction

1.1 Background

We start the story from planar graphs. A graph $G$ is a planar graph if it can be embedded into the two-dimensional plane that no pair of edges will cross with each other. Many NP-hard optimization problems are hard to approximate in general graphs but have efficient approximation algorithms on planar graphs. For instance, the maximum independent set problem cannot be approximate within approximation ratio $n^{1-\epsilon}$ for any $\epsilon > 0$ in general graphs unless $P = ZPP$ [56, 42]. Minimum dominating set, whose equivalence to set cover in terms of approximability is well known due to [87], cannot be approximated within $\epsilon \log n$ for some $\epsilon > 0$ unless $P = NP$ [79, 40]. Minimum vertex cover cannot be approximated with approximation ratio $\frac{7}{6} - \epsilon$ for any $\epsilon > 0$ [57, 35] and do not have polynomial time approximation scheme (PTAS) unless $P = NP$ [4]. However, all of them have PTASs on planar graphs.

There are two general approaches for designing PTASs for NP-hard optimization problems on planar graphs: the planar separator approach and Baker’s technique. The planar separator approach is introduced by Lipton and Tarjan in [77, 78]. The planar separator approach has given PTASs for problems such as maximum independent set. Alon et al. [1, 2] generalize the planar separator approach to graphs with an excluded minor. The planar separator approach has some limitations. This approach is only applicable to problems where the size of the optimal solutions was at least a constant fraction of $n$. Furthermore, minimum dominating set does not have separator-based approximation algorithms even on planar graphs [60, 52]. Compared to the planar separator approach, Baker’s technique can achieve better tradeoff between the running time and approximation ratio.

Baker’s technique was introduced in [7, 8], is a method for designing PTAS for NP-hard optimization problems on planar graphs based on layerwise decomposition. It is also independently discovered by Hochbaum and Maass [58], called Shifting Strategy, for designing PTASs for covering and packing problems in the plane. Given a planar graph, its planar embedding $G$ can be generated in linear time [59]. A planar graph is an outerplanar graph if it has a planar embedding where all the nodes are on the exterior face. Given a planar embedding of a planar graph, a node is at level 1 if it is on the exterior face. When all the level-1 nodes are deleted from the planar embedding, the nodes on the exterior face are called level-2 nodes. By this induction the level-$k$ nodes can be defined. A planar graph is a $k$-outerplanar graph if it has a planar embedding with no nodes of level more than $k$. Using computing maximum independent set as an example, for each $0 \leq \ell \leq k$, Baker’s technique removes all the nodes at levels congruent to $\ell \pmod{k+1}$. The remaining graph is composed of several $k$-outerplanar embeddings. The maximum independent set on each $k$-outerplanar embedding can be computed by dynamic programming in $O(2^{O(k)n})$ time. The union of these maximum independent sets is a valid maximum independent set of $G$. Suppose $S_{OPT}$ is the maximum independent set of $G$, then by the pigeonhole principle, for at least one $\ell$, there are at most $1/(k+1)$ of the nodes in $S_{OPT}$ are at levels congruent to $\ell \pmod{k+1}$. Therefore the solution obtained is at least $k/(k+1)$ optimal. This gives a PTAS for computing maximum independent set on planar graph. The problems mentioned in [8] that achieve PTASs by Baker’s technique include maximum independent set, minimum vertex cover, minimum dominating set, minimum edge dominating set, maximum H-matching and maximum tile salvage. In [47], Barker’s technique is used to give a PTAS for subgraph isomorphism, connectivity and shortest path problems on planar graphs and bounded genus graphs. Khanna et al. [66] explore Barker’s technique using logic and proves that when the corresponding incidence graphs of MPSAT, TMAX and TMIN formulas are planar, then there are PTASs for computing their MAX-SAT.

The dynamic programming algorithm of Baker’s technique on each $k$-outerplanar component is based on the outerplanar decomposition. A similar technique for other classes of graphs is tree decomposition.
The dynamic programming on tree decomposition \cite{11} has numerous algorithmic applications. A graph with bounded local treewidth if for each \( v \in V \), the treewidth of the subgraph induced by its \( r \)-neighbors is bounded by some function \( f(r) \) independent of \( n \). Eppstein \cite{38} generalizes Baker’s technique from planar graphs to graphs with bounded local treewidth. The vertices in the graph are labeled into levels by breadth-first search. And the graph is decomposed into several disjoint subgraphs in the way similar to Baker’s technique. The final solution is the combination of optimal solutions on all subgraphs computed by dynamic programming on their tree decompositions. This approach gives PTASs for those problems solved by Baker’s technique on graphs with bounded local treewidth. Frick and Grohe \cite{45} also show that for each property \( \varphi \) definable in first-order logic and any structure that is locally tree-decomposable, there is a linear time algorithm deciding whether the structure has property \( \varphi \).

Eppstein \cite{38} further characterizes that a minor-closed class of graphs has bounded local treewidth if and only if it does not contain all apex graphs where a apex graph is a graph contains a vertex whose removal results in a planar graph. The bound of local treewidth for apex-minor-free graphs in \cite{38} is \( 2^{O(r)} \), and \( O(r) \) for bounded-genus graphs which are also apex-minor-free graphs. This bound is improved in \cite{33} to \( 2^{O(r)} \) and further improved in \cite{26} to the optimal linear bound \( O(r) \). Therefore all apex-minor-free graph have linear local treewidth. The linear bound of local treewidth for apex-minor-free graphs generalizes PTASs given by Baker’s technique to apex-minor-free graphs with the same time complexity \( 2^{O(1/\epsilon)n^{O(1)}} \).

Demaine and Hajiaghayi \cite{27} build a connection between fixed parameter algorithm and PTAS using bidimensionality theory. Bidimensionality theory is a general approach for obtaining treewidth-parameter bounds and subexponential fixed parameterized algorithms which is developed in a series of papers \cite{25, 23, 32, 24}. In \cite{27}, both separator approach and Baker’s technique are generalized. In particular, Baker’s technique is generalized to approximate non-local problems like minimum connected dominating set on graphs with bounded local treewidth. In \cite{71}, Baker’s technique is applied to the travelling salesman problems (TSP) on undirected planar graphs and achieves a linear time approximation scheme. Fomin et al. \cite{43} generalize the separator approach in \cite{27} to obtain EPTAS for more problems on apex-minor-free graphs.

Baker’s technique has also been applied to obtain PTASs for optimization problems on \( H \)-minor-free graphs. This generalization relies heavily on a deep structural theorem \cite{92} of Robertson and Seymour’s graph minor theory which states that any \( H \)-minor-free graph has a tree decomposition into pieces that are \( h \)-almost embeddable in a surface of bounded genus where \( h \) is a constant depending on \( H \). Therefore each piece has bounded local treewidth after removing a set of at most \( h \) vertices. Such a tree decomposition is called graph minor decomposition. Grohe \cite{52} proves that there is a PTAS for computing minimum vertex cover, minimum dominating set and maximum independent set on \( H \)-minor-free graphs based on Baker’s technique. However, this algorithm is existential because constructing the tree decomposition relies on the polynomial time membership test of minor-closed properties \cite{91}, which is non-constructive. Baker’s technique decomposes the input graph into \( k + 1 \) pieces such that removing any one of them results in a graph of bounded treewidth where the treewidth depends on the number \( k \). DeVos et al. \cite{34} show that such a decomposition also exists for \( H \)-minor-free graphs. Demaine et al. \cite{28} provide an algorithm to construct a decomposition with additional properties proved in \cite{34} in \( O(n^c) \) time where \( c \) is a constant depending on the size of the excluded minor \( H \). This makes Robertson and Seymour’s result constructive. Based on the decomposition constructed, they gives PTASs for many maximization and minimization problems. Grohe et al. \cite{53} give a quadratic time algorithm for computing graph minor decompositions. Demaine et al. \cite{30} design a new method called contraction decomposition by which PTASs are given for contraction closed problems on bounded-genus graphs. In \cite{29}, this method is generalized to \( H \)-minor-free graphs.

On geometric graphs, Baker’s technique has also achieved many significant results. Hunt et al. \cite{60} give PTASs for computing several optimization problems such as maximum independent set and minimum dom-
inating set on unit disk graphs and $\lambda$-precision disk graphs. Erlebach et al. [39] and Chan [18] give PTASs for computing maximum independent set and minimum vertex cover on disk graphs and intersection graphs of fat objects. Leeuwen [100] shows that for unit disk graphs with bounded thickness, maximum independent set, minimum vertex cover and minimum dominating set can be computed in linear time. And for unit disk graphs with bounded density, these problems admit PTASs. Furthermore, it also gives asymptotic FP-TASs for minimum connected dominating set on unit disk graphs with bounded density and planar graphs. Chen [19] and Demaine et al. [23] achieve PTASs for maximum independent set and minimum $r$-dominating set on map graphs. Marx shows in [80] that independent set and dominating set are W[1]-hard on general unit disk graphs. This implies that neither of them has EPTAS or FPT algorithm unless FPT = W[1]. Fomin et al. [44] generalize bidimensionality theory to geometric graphs that give EPTAS and subexponential time parameterized algorithms for many problems. Specifically, they also show that feedback vertex set on unit ball graphs in $\mathbb{R}^3$ neither admits PTAS unless P = NP, nor subexponential time parameterized algorithms unless the Exponential Time Hypothesis fails. Grigoriev et al. [51] generalize Baker’s technique to graphs embeddable with bounded number of crossings per edge on a surface of bounded genus. Dawar et al. [21] extend the results of [66] and show that for each class of graphs with excluded minors, the first-order definable optimization problems are contained in PTAS.

1.2 Motivation and results

Baker’s technique has been generalized to many graph classes that are much more general than planar graphs and a much larger set of problems than that is originally proposed in [8]. Therefore, a natural question is how much the Baker’s technique can be further extended.

The classic NP-hard optimization problems are widely used for modelling problems in computer science, physics, economics and many other fields, but they oversimplify the situations in the real world. In this paper, a unified framework is proposed to formulate the combinatorial optimization problems and more complicated situations. We formulate the optimization problems by attaching each vertex $i \in V$ a locally-defined function $f_i$ where $f_i$ encodes the local constraints of the problems. We prove that if $f_i$ are so-called liberal functions, that is, $f_i \geq 0$ for all $i \in V$, then there exists a PTAS for computing the max-sum of all $f_i$ on planar graphs. This result implies that the max-product of $f_i$ can also be approximate very well in many cases. These results are consequently generalized to graphs with bounded local treewidth, H-minor-free graphs, $d$-dimensional geometric graphs with bounded density, and graphs with bounded number of crossings per edge.

**Theorem 1.1.** There are PTASs for computing the max-sum of liberal functions on planar graphs, graphs with bounded local treewidth, geometric graphs with bounded density, H-minor-free graphs and graphs with bounded number of crossings per edge.

Theorem 1.1 also implies that the max-product can be well approximated on these graphs in many situations. When the constraints can be encoded by liberal functions, our technique can be served as an alternative of the classic maximum a posterior estimation algorithm such as max-product and min-sum belief propagation [88].

**Theorem 1.2.** There does not exist PTAS for computing the min-sum of liberal functions even on planar graphs unless P = NP. Therefore the PTAS also does not exist on graphs with bounded local treewidth, geometric graphs, H-minor-free graphs and graphs with bounded number of crossings per edge unless P = NP.
The proof of Theorem 1.2 is a reduction from the computing the chromatic number of the input planar graph to computing the min-sum of liberal functions. By the reduction, the NP-hardness of computing the chromatic number for planar graphs implies the hardness of approximating the min-sum of liberal functions.

Theorem 1.2 shows that there is no PTAS for computing the min-sum of liberal functions in general unless P = NP. However, if the liberal functions satisfy the balance property, computing the min-sum still has PTASs on these graphs, described by the following theorem.

**Theorem 1.3.** There are PTASs for computing the min-sum of balanced liberal functions on planar graphs, graphs with bounded local treewidth, geometric graphs with bounded density, H-minor-free graphs and graphs with bounded number of crossings per edge.

The byproducts of our technique are the PTASs for computing MAX-SAT, MAX-CSP, MAX-CUT, MAX-DICUT and MAX-$k$-CUT on these graphs.

The structures and syntax of the boolean formulas can be expressed directly by graphical models like factor graphs [76]. But they cannot be directly formulated by our model. Therefore we need a method to to transform the corresponding factor graph of the input CNF formula into our model, which is called $X - F$ contraction.

**Theorem 1.4.** Given a boolean formula $\varphi$, there is a PTAS for computing the MAX-SAT if its corresponding factor graph can be transformed into planar graphs, graphs with bounded local treewidth, geometric graphs with bounded density, H-minor-free graphs or graphs with bounded number of crossings per edge through $X - F$ contractions.

Theorem 1.4 strongly extends the results of [66], which restricts the corresponding factor graph of the input boolean formula to be planar. This result can be further extended to MAX-CSP problems.

**Theorem 1.5.** Given a CSP, there is a PTAS for computing its MAX-CSP if its corresponding factor graph can be transformed into planar graphs, graphs with bounded local treewidth, geometric graphs with bounded density, H-minor-free graphs or graphs with bounded number of crossings per edge through $X - F$ contractions.

We also have a similar result for maximum graph-cut problems.

**Theorem 1.6.** There are PTASs for computing MAX-CUT, MAX-DICUT, MAX-$k$-CUT on planar graphs, graphs with bounded local treewidth, geometric graphs with bounded density, H-minor-free graphs and graphs with bounded number of crossings per edge.

The MAX-CUT is polynomial-time solvable on planar graphs [54]. Its PTAS on $H$-minor-free graphs has already been known in [28]. We give a simpler proof through encoding the MAX-CUT problem as computing the max-sum of liberal functions. MAX-$k$-CUT is a natural generalization of the MAX-CUT problem. When $k$ is larger than the maximum degree $\Delta$ of the input graph, the MAX-$k$-CUT is exactly all the edges. Therefore, MAX-$k$-CUT is only interesting when $k \leq \Delta$. Furthermore, in a more general setting where the weight of edge $(i, j) \in E$ is given by a function defined on the edge $w_{ij} : [k] \times [k] \to \mathbb{R}^+$ which takes the values of its endpoints as inputs and returns a positive number as the weight, our technique can still achieve PTASs for these graph cut problems.

Our technique also has algorithmic implications on important computation problems of many fields. Some of them are long-standing open problems in these fields, listed informally as follows.
1. Consider a wireless network consisting of heterogeneous computing nodes. Their interference correlations can be modelled as an interference graph. If the interference graph falls into the class of planar graphs, graphs with bounded local treewidth, geometric graphs with bounded density, $H$-minor-free graphs or graphs with bounded number of crossings per edge, our technique gives a PTAS for computing the channel assignment which maximizes the throughput of the network. This problem is significant for communication scheduling in wireless networks and has been extensively studied.

2. Consider a data centers consisting of thousands of servers. Each dataset $t$ is replicated to several servers for achieving both performance and reliability. Servers maintaining copies of $t$ can share the workloads of $t$ when the workload of $t$ is huge that a single server cannot hold. The correlations between servers can be modelled by a storage graph. If the storage graph falls into the graph classes above, our technique gives a PTAS for computing the optimal real-time task scheduling. Furthermore, if we can control the procedure of storing data into the servers, then the storage graph can be constructed to a $k$-tree or a partial $k$-tree, which implies there is a linear time algorithm for computing the optimal real-time scheduling.

3. The Edwards-Anderson model is a widely accepted mathematical abstraction of the spin glasses in statistical physics. Computing the ground state of Edwards-Anderson model is a long-standing open problem. Our technique shows that there is a PTAS for minimizing the energy for the ferromagnetic Edwards-Anderson model without external magnetic field on $d$-dimensional lattice graphs, which resolves a long-standing open problem in statistical physics.

4. For computer vision problems modeled by energy minimization of the pairwise Markov random fields, that is, minimizing the energy function in the form of

$$E(\sigma) = \sum_{i \in V} \psi_i(\sigma_i) + \sum_{(i,j) \in E} \psi_{i,j}(\sigma_i, \sigma_j),$$

we have PTASs for minimizing the energy if the minimum value of $\psi_i(\cdot)$ is larger than zero for all $i \in V$. For energy minimization of high-order Markov random fields, we also have similar results.

1.3 Organization

This paper is organized as follows. In Section 2, we introduce some concepts and methods which will be used throughout this paper. In Section 3, we propose a unified framework to formulate the optimization problems. In Section 4, we give our approximation algorithms on planar graphs, graphs with bounded local treewidth, $H$-minor-free graphs, $d$-dimensional geometric graphs with bounded density and graphs with bounded crossing number per edge. In Section 5, we describe the details of some typical applications of our algorithms including communication scheduling in wireless networks, distributed database system management, computing the ground state of Edwards-Anderson model in statistical physics and energy minimization for many important applications of computer vision. In Section 6, we will conclude the contributions of this paper and propose some further research directions.

2 Definitions and Notations

In this section we introduce some notations and concepts that will be used throughout this paper. Let $G$ be a graph whose vertex set is $V(G)$ and edge set is $E(G)$. A graph $G'$ is a subgraph of $G$ if $V(G') \subseteq V(G)$.
Given a graph \( G \), the treewidth for graph \( G \) is denoted by \( tw(G) \) and runs in time \( n^{O(f(1/\epsilon))} \) for all \( \epsilon \in (0, 1) \). A PTAS with running time \( f(1/\epsilon) \cdot n^{O(1)} \) is called an efficient polynomial time approximation scheme (EPTAS). An EPTAS where \( f(1/\epsilon) \) is polynomial in \( 1/\epsilon \) is called a fully polynomial time approximation scheme (FPTAS).

**Graph minor.** Given a graph \( G \), if graph \( H \) can be reduced from a subgraph of \( G \) by a sequence of edge contractions, then \( H \) is a minor of \( G \), denoted by \( H \preceq_m G \). We can see that \( H \preceq_m G \) if and only if there is a mapping \( h : V_H \rightarrow 2^V \) such that \( G[h(x)] \) is a connected subgraph of \( G \) for all \( x \in V_H \), \( h(x) \cap h(y) = \emptyset \) for all \( x \neq y \in V_H \) and, for every \( (x, y) \in E_H \) there exists an edge \( (u, v) \in E_G \) such that \( u \in h(x) \) and \( v \in h(y) \). The mapping \( h \) is called a witness of \( H \preceq_m G \).

A class \( C \) of graphs is minor-closed if and only if for all \( G \in C \) and \( H \preceq_m G \) we have \( H \in C \). We say \( C \) is nontrivial if \( C \) does not contain all the graphs. A class \( C \) of graphs is \( H \)-minor-free if \( H \not\preceq_m G \) for all \( G \in C \). Then we call \( H \) an excluded minor of \( C \). Robertson and Seymour’s Graph Minor Theorem [93], which proves Wagner’s conjecture, demonstrates that the undirected graphs partially ordered by the graph minor relationship form a well-quasi-ordering. This implies that every minor-closed class of graphs can be characterized by a finite set of forbidden minors.

**Tree decomposition and path decomposition.** The concept of tree decomposition is introduced by Robertson and Seymour [90], on which the treewidth can be defined to measure the similarity between a graph and a tree, has turned out to be significant for structural graph theory and graph algorithms.

**Definition 2.1.** A tree decomposition of an undirected graph \( G(V, E) \) is a pair

\[
(X_i | i \in I), T = (I, F)
\]

with \( \{X_i | i \in I\} \) a family of subsets of \( V \), one for each node of \( T \). \( T \) is a tree such that

1. \( \bigcup_{i \in I} X_i = V \),
2. for all edges \( \{v, w\} \in E \), there exists an \( i \in I \) with \( v \in X_i \) and \( w \in X_i \),
3. for all \( i, j, k \in I \): if \( j \) is on the path from \( i \) to \( k \) in \( T \), then \( X_i \cap X_k \subseteq X_j \).

Each node of the tree decomposition \( T \) is called a bag. The third property of tree decomposition guarantees that for every \( v \in V \), \( \{X_i : v \in X_i, i \in I\} \) induces a connected subtree of \( T \).

**Definition 2.2.** The treewidth of a tree decomposition \( (\{X_i | i \in I\}, T = (I, F)) \) is \( \max_{i \in I} |X_i| - 1 \). The treewidth of a graph \( G \), denoted by \( tw(G) \), is the minimum treewidth over all tree decompositions of \( G \).

A tree decomposition of width equal to the treewidth is called an optimal tree decomposition. Computing the treewidth for graph \( G \) is NP-complete. But given a graph \( G \), deciding whether the treewidth of \( G \) is at most a fixed constant \( k \) can be decided in linear time by Bodlaender’s algorithm [12]. If the answer is yes, then an optimal tree decomposition of \( G \) can be constructed in linear time (but exponential in \( k \)). The following lemma includes some well-known facts about treewidth.
Lemma 2.3. Let \((\{X_i| i \in I\}, T = (I, F))\) be a tree decomposition of graph \(G\). Then the following holds:

1. If \(X \subseteq V(G)\) is a clique, then there is an \(i \in I\) that \(X \subseteq X_i\).

2. Let \(G, H\) be graphs such that \(V(G) \cap V(H)\) is a clique in both \(G\) and \(H\). Then it holds that \(tw(G \cup H) = \max\{tw(G), tw(H)\}\).

3. For any \(X \subseteq V(G)\). Then \(tw(G) \leq tw(G \setminus X) + |X|\).

4. Let \(G, H\) be graphs such that \(H \leq_m G\). Then \(tw(H) \leq tw(G)\).

Definition 2.4. A tree decomposition is called a path decomposition if \(T = (I, F)\) is a path. The pathwidth of a graph \(G\), denoted by \(pw(G)\), is the minimum width over all path decompositions of \(G\).

Local treewidth. The concept of local treewidth is first introduced by Eppstein in [38] as a generalization of treewidth. The local treewidth of graph \(G\) is a function that maps an integer \(r \in \mathbb{N}\) to the maximum treewidth of the subgraph of \(G\) induced by the \(r\)-neighborhood of any vertex in \(G\), formally defined as follows.

Definition 2.5. The local treewidth of graph \(G(V, E)\) is a function defined as

\[
ltw^G(r) = \max \left\{ tw(G[N_r(i)]) : i \in V \right\}
\]

where \(G[N_r(i)]\) is the subgraph of \(G\) induced by \(N_r(i)\).

Definition 2.6. A class \(C\) of graphs has bounded local treewidth if there is a function \(f : \mathbb{N} \rightarrow \mathbb{N}\) such that \(ltw^G(r) \leq f(r)\) for all \(G \in C\), \(r \in \mathbb{N}\). \(C\) has linear local treewidth if there is a \(\lambda \in \mathbb{R}\) such that \(ltw^G(r) \leq \lambda \cdot r\) for all \(G \in C\), \(r \in \mathbb{N}\).

It is well known that the treewidth of a \(k \times k\) grid is \(k\), so planar graphs do not have bounded treewidth. But for each \(i \in V\) of any planar graph \(G\), the subgraph induced by its \(r\)-neighborhood \(N_r(v)\) has bounded treewidth. Moreover, the \(k\)-outerplanar graph has treewidth at most \(3k-1\) [10]. Therefore, Baker’s technique on planar graphs [8] is a special case of its applications on graphs with bounded local treewidth.

Graphs on surfaces. A surface \(\Sigma\) is a compact and connected 2-manifold without boundary. An embedding refers to a 2-cell embedding. An embedding of a graph \(G\) on a surface \(\Sigma\) is a representation of \(G\) on surface \(\Sigma\) that maps vertices and edges of \(G\) to points and arcs on \(\Sigma\) while preserving their correlations in \(G\). Every face bounded by arcs is homomorphic to an open disk, that is, \(\{(x, y)| x^2 + y^2 < 1\}\). Therefore, an embedding of graph on a surface is drawing the graph on the surface such that the edges only intersect at their endpoints. A graph has bounded genus if it can be embedded on a bounded-genus surface. The genus of a graph is the minimum genus of all surfaces on which it can be embedded.

Clique sum. The clique sum operation is a significant technical tool of graph minor theory, which originates from characterizations of \(K_{3,3}\)-minor-free and \(K_5\)-minor-free graphs by Wagner [102]. The clique-sum operation is a way of combining two graphs by identifying their cliques. Suppose \(G_1\) and \(G_2\) are two graphs, \(W_1 \subseteq V(G_1)\) and \(W_2 \subseteq V(G_2)\) are two cliques of the same size. The clique sum of \(G_1\) and \(G_2\), denoted by \(G_1 \oplus G_2\), is a graph by identifying \(W_1\) and \(W_2\) through a bijection \(h : W_1 \rightarrow W_2\), and then possibly deleting some of the clique edges. The subgraph induced by the clique vertices in \(G_1 \oplus G_2\) is called
the join set. The clique is called a \( k \)-sum if \(|W_1| = |W_2| = k\), denoted by \( G_1 \oplus_k G_2 \). Since there are many possible bijections between vertices of \( W_1 \) and \( W_2 \), there are also many possible results for \( G_1 \oplus G_2 \).

The clique-sum operation plays an important role in the core of Robertson and Seymour’s graph minor theory. The deepest structural theorem \([92]\) of graph minor theory states that any \( H \)-minor-free graph can be decomposed into a collection of graphs each of which can be embedded into a bounded-genus surface by deleting a bounded number of apex vertices where the number only depends on \( V(H) \). These so-called \( h \)-almost embedded graphs are combined in a tree structure by clique-sum operations. The clique-sum decomposition \([28, 53]\) is the building block by which the Baker’s technique can be used to achieve approximation algorithms for \( H \)-minor-free graphs.

**Graphical models.** We introduce two widely-used undirected graphical models: factor graph \([76]\) and Markov random field (MRF) \([70]\).

A factor graph \( G = (\mathcal{X}, \mathcal{F}, \mathcal{E}) \) is a bipartite graph where \( \mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\} \) is the set of variable nodes, \( \mathcal{F} = \{F_1, \ldots, F_m\} \) is the set of function nodes and \( \mathcal{E} \) is the set of undirected edges between \( \mathcal{X} \) and \( \mathcal{F} \). If variable node \( \mathcal{X}_i \) is linked to function node \( F_j \), then \( \mathcal{X}_i \) is an input variable of \( F_j \). Let \( S_j = \{\mathcal{X}_i : (i, j) \in \mathcal{E}\} \). Given a factor graph, a probability distribution is defined by

\[
P(\mathcal{X} = x) = \frac{1}{Z} \prod_{j=1}^{m} F_j(S_j)
\]

where \( x \in \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \) is a joint configuration of all the variable nodes and \( Z \) is the normalizing factor that \( Z = \sum_x \prod_{j=1}^{m} F_j \). The marginal probability of variable node \( \mathcal{X}_i \) taking the value \( x_i \) is

\[
P(\mathcal{X}_i = x_i) = \frac{1}{Z} \sum_{x_i=x_i} \prod_{j=1}^{m} F_j.
\]

When the factor graph is acyclic, the belief propagation algorithm can compute the marginal probability exactly in linear time. In general factor graphs, exact inference is \#P-complete.

A Markov random field consists of two components: an undirected graph \( G(V, E) \) and a set of so-called clique functions. Each vertex in \( G \) represents a random variable and the edges define the independence correlations between the random variables. The independence between random variables satisfies the Markov property: a random variable is independent of all its non-neighbor variables when the values of its neighbors are fixed. For each clique \( c \) in \( G \), there is a corresponding clique function \( \phi_c \) taking the values of vertices in \( c \) as inputs and returns a non-negative real number as output. For such a MRF, a factor graph can be constructed where each vertex in \( G \) corresponds to a variable node and each clique function \( \phi_c \) corresponds to a function node \( F_c \). The function node \( F_c \) is linked to all the variable nodes corresponding to the vertices in clique \( c \). Then a probability distribution can be similarly defined as that of a factor graph.

### 3 The Model

We first build a unified framework to formalize the optimization problems. Consider a graph \( G(V, E) \), directed or undirected. For directed graphs, the neighbors of vertex \( i \in V \) is defined as \( N^-(i) = \{j : (j, i) \in E\} \) and \( N^+(i) = \{j : (i, j) \in E\} \). For undirected graphs, \( N^-(i) = N^+(i) = N(i) \). Each vertex can take one of \( q \) values from the input set \( |q| = \{0, \ldots, q-1\} \). A joint input \( \sigma \in |q|^V \) is called a configuration. Each vertex \( i \in V \) is attached a locally defined function \( f_i \). If \( G \) is undirected, then for each
\( i \in V, f_i : [q]^{N(i)} \times [q] \to \mathbb{R} \) take the configurations of \( N(i) \cup i \) as inputs and return a real number as output. If \( G \) is directed, then for each \( i \in V \), then \( f_i : [q]^{N^-(i)} \times [q] \to \mathbb{R} \) take the configurations of \( N^-(i) \cup i \) as inputs and return a real number as output. We can define various optimization objectives on \( G \) such as maximization and minimization, where the maximization objectives include max-sum and max-product typically which can be formulated by

\[
\max_{\sigma \in [q]^V} \sum_{i \in V} f_i \quad \text{and} \quad \max_{\sigma \in [q]^V} \prod_{i \in V} f_i.
\]

And the minimization objectives include min-sum and min-product typically which can be formulated by

\[
\min_{\sigma \in [q]^V} \sum_{i \in V} f_i \quad \text{and} \quad \min_{\sigma \in [q]^V} \prod_{i \in V} f_i.
\]

Computing max-product and min-sum are known as a pair of dual problems for maximum a posterior estimation in probabilistic inference by the following transformation:

\[
\min_{\sigma \in [q]^V} \sum_{i \in V} f_i \iff \max_{\sigma \in [q]^V} \prod_{i \in V} \exp \left( -f_i \right) = \max_{\sigma \in [q]^V} \prod_{i \in V} F_i
\]

where \( F_i = \exp(-f_i) \). The optimal solution of max-product and min-sum on acyclic graphs can be computed in linear time by corresponding message passing algorithm max-product and min-sum belief propagations [88].

If the function \( f_i \) can be arbitrarily defined, this model is equivalent to a factor graph [76] with the same number of variable nodes and function nodes that each variable node \( i \) has exactly one corresponding function node \( f_i \). The function node \( f_i \) has edges linked to vertex \( i \) and \( N^-(i) \). Other similar models include incidence graph proposed in [66] and holographical reduction proposed in [99] which serves as a technique for representing counting problems by sum-product formulas. More general models such as factor graph and Markov random field [70] for describing the dependence relationship between random variables plays an important role in artificial intelligence and statistical physics.

The definition of this framework is also quite similar to graphical game [65] where each vertex \( i \in V \) in the graph represents a player of the game and the function \( f_i \) attached on vertex \( i \) represents the utility function of player \( i \). But if we do not set any constraint on \( f_i \), this model is more general than graphical game. This is because for graphical games, no matter what the joint configuration of \( N(i) \) is, there is always a local optimal configuration \( \sigma_i \in [q] \) called best response that can be chosen by for \( i \in V \). In general, such a local optimal configuration may not exist. Special cases of graphical games include independent set and coloring where the number of colors \( q > \Delta \) that \( \Delta \) is the maximum degree of \( G \).

The local combinatorial optimization problems can be formulated by this framework. For instance, computing the maximum independent set of \( G \) is equivalent to computing \( \max_{\sigma \in [q]^V} \sum_{i \in V} f_i \) by setting \( f_i \) as

\[
f_i = \begin{cases} 
1, & \text{if } \sigma_i = 1, \text{ and } \sigma_j = 0 \text{ for } \forall j \in N(i), \\
0, & \text{if } \sigma_i = 0, \\
-\infty, & \text{if } \sigma_i = 1, \text{ and } \sigma_j = 1 \text{ for } \exists j \in N(i),
\end{cases}
\]

where \( \sigma_i = 1 \) means vertex \( i \) is in the independent set and \( \sigma_i = 0 \) otherwise. If there is an edge \( (i, j) \in E \) and \( \sigma_i = \sigma_j = 1 \), then \( f_i = f_j = -\infty \), which means the configuration is invalid. Note that \( f_i = -\infty \) can also be replaced by \( f_i = -n \) since the size of the independent set is at most \( n \). Therefore when \( \sum_{i \in V} f_i \) is
maximized, we have the maximum independent set of \( G \). Other combinatorial optimization problems like vertex cover and dominating set have similar encodings.

From the above way of encoding the local constraints of combinatorial optimization problems as \( f_i \), we can see that if \( f_i \geq 0 \) for all \( i \in V \) and all \( \sigma \in [q]^V \), then every configuration \( \sigma \in [q]^V \) of graph \( G \) can be treated as a valid configuration. Such a set of \( f_i \) is called a set of \textit{liberal functions}.

**Definition 3.1.** The function \( f_i \) attached on \( i \in V \) is a liberal function if for all \( \sigma_{N^-(i)} \in [q]^{N^-(i)} \), choosing any \( \sigma_i \in [q] \) will make \( f_i \geq 0 \).

The problems considered in this paper are those optimization problems defined by liberal functions. Although our algorithm is a generalization of Baker’s technique so that it can be used to solve the maximum independent set and other NP-hard problems mentioned in [3], the set of liberal functions does not cover the functions which encode the local constraints of these problems. Furthermore, it is shown in [104] that computing maximum independent set is a special case of computing the optimal pure strategy Nash equilibrium of graphical games, by constructing a reduction such that each maximal independent set corresponds to a pure strategy Nash equilibria and each maximum independent set corresponds to an optimal pure strategy Nash equilibria. Therefore the liberal functions also do not cover the game-theoretical utility functions. These combinatorial or game-theoretical optimization problems need their optimal solutions also to be local optimal. These constraints cannot be encoded by liberal functions.

Although our model is well-defined on directed graphs, we only need to consider undirected graphs. For a directed graph \( G(V, E) \), its corresponding undirected graph \( G' = (V, E') \) is constructed by replacing all the directed edges by undirected edges and then replacing multiple undirected edges between pairs of vertices if exist to a single undirected edge. Each function \( f_i \) is replaced by \( f'_i \) which takes configurations of \( N^-(i) \cup N^+(i) \) as inputs but only the configurations of \( N^-(i) \) can affect the output of \( f'_i \) which is equal to the output of \( f_i \) taking the configurations of \( N^-(i) \) as inputs.

**\( \mathcal{X} - \mathcal{F} \) contraction.** Given a factor graph \( G = (\mathcal{X}, \mathcal{F}, \mathcal{E}) \), \( \mathcal{X} - \mathcal{F} \) contraction is \( m \) edge contraction operations between variable nodes and function nodes where \( m \) is the number of function nodes. At each step, we choose a function node \( \mathcal{F}_j \) and one variable node \( \mathcal{X}_i \in S_j \), then we contract the edge \( (i, j) \in \mathcal{E} \) between variable node \( \mathcal{X}_i \) and function node \( \mathcal{F}_j \). Let \( \mathcal{F}(i) = \{ j : \mathcal{F}_j \text{ is contracted to } \mathcal{X}_i \} \) denote the set of function nodes contracted to \( \mathcal{X}_i \). The contraction operations terminate when all the function nodes are contracted to variable nodes. At last, we replace multiple undirected edges if exist between pairs of variable nodes by a single edge. The obtained graph \( G \) is a graph with only variable nodes and functions \( f_i \) attached on the variable nodes. The function \( f_i \) is set as

\[
f_i = \log \left( \prod_{j \in \mathcal{F}(i)} \mathcal{F}_j \right) = \sum_{j \in \mathcal{F}(i)} \log \mathcal{F}_j.
\]

If \( f_i \) is a liberal function, then it is required that \( \mathcal{F}_j \geq 1 \) for all \( j \in \mathcal{F}(i) \). Then it holds that

\[
\max_{\sigma} \prod_{j=1}^{m} \mathcal{F}_j \Leftrightarrow \max_{\sigma} \sum_{i \in V} f_i,
\]

which transforms the maximum a posterior estimation in factor graph \( G \) to computing the max-sum of liberal functions on \( G \). The \( \mathcal{X} - \mathcal{F} \) contractions build a connection between the factor graph and our model.

Throughout this paper, we ignore the computation time of \( f_i \) when analyzing the computational complexity. We assume that they are all \( O(1) \)-time computable.
4 Approximation Schemes

4.1 Planar Graphs

The following result gives a PTAS for computing the max-sum of liberal functions on planar graph $G$.

**Theorem 4.1.** If $f_i$ is a liberal function for all $i \in V$ where $|V| = n$, then for fixed $k$, there is an $O(q^{O(k)}kn)$-time algorithm for computing $\max_{\sigma \in [q]^V} \sum_{i \in V} f_i$ that achieves a solution of value at least $k/(k+2)$ optimal for general planar graphs.

The proof of Theorem 4.1 is based on the following lemma.

**Lemma 4.2.** Given a graph $G(V, E)$ with treewidth bounded by $k$, for any $U \subseteq V$, any set of liberal functions defined on $G$, $\max_{\sigma \in [q]^V} \sum_{i \in U} f_i$ and $\min_{\sigma \in [q]^V} \sum_{i \in U} f_i$ can be computed in $O(q^{O(k)n})$ time.

**Proof.** Since the input graph $G$ has treewidth bounded by $k$, we construct a tree decomposition $T = (I, F)$ rooted at $r \in I$ with treewidth $k$ for $G$ in linear time (but exponential in $k$ [12]).

For each $i \in I$, the subtree of $T$ rooted at $i$ is denoted by $T_i$. The set of vertices in $T_i$ is denoted by $V_{T_i}$. The configurations of bag $X_i$ where $i \in I$ is denoted by $\sigma_{X_i}$. Suppose the child nodes of $i \in I$ are $i_1, \ldots, i_d \in I$ and the parent node of $i \in I$ is $p_i \in I$.

Our algorithm is a dynamic programming from the leaves to the top. Compared to the classic dynamic programming on tree decomposition, we enumerate the all the possible configurations of $X_i - X_{p_i}$, rather than $X_i$ for each bag $X_i$. For root $r$, $X_r - X_{p_r} = X_r$. By the definition of tree decomposition, $X_i \cap X_{i_j} \subseteq X_i$ for $1 \leq x \neq y \leq d$. Therefore, $X_i - X_j$ for $1 \leq t \leq d$ are pairwise disjoint.

Let $S_{i \setminus p_i}^U(\sigma_{i \setminus p_i})$ denote the max-sum of the liberal functions attached on vertices in $(U \cap V_{T_i}) - (X_{p_i} \cup \partial X_{p_i})$ with the configurations of vertices in $X_i - X_{p_i}$ being fixed to $\sigma_{i \setminus p_i}$. The set $\partial X_{p_i}$ denotes the vertices adjacent to vertices in $X_{p_i}$ but not in $X_{p_i}$. Note that $S_{i \setminus p_i}^U(\sigma_{i \setminus p_i})$ does not include the sum of liberal functions attached on vertices in $\partial X_{p_i}$. This is because their values are not fixed when the configurations of $X_{p_i}$ are not given. The value of $S_{i \setminus p_i}^U(\sigma_{i \setminus p_i})$ can be computed by the following recurrence:

$$S_{i \setminus p_i}^U(\sigma_{i \setminus p_i}) = \Gamma_{X_i - X_{p_i}}^{\sigma_{i \setminus p_i}} + \sum_{t=1}^d \max_{\sigma_{i \setminus t}} \left\{ S_{i \setminus t}^U(\sigma_{i \setminus t}^{\sigma_{i \setminus t}}) \right\}$$

where $\Gamma_{X_i - X_{p_i}}^{\sigma_{i \setminus p_i}}$ is the sum of liberal functions attached on vertices in $(U \cap (X_i \cup \partial X_i)) - (X_{p_i} \cup \partial X_{p_i})$ when the configuration of $X_i - X_{p_i}$ is fixed to $\sigma_{i \setminus p_i}$. Then we have

$$\max_{\sigma \in [q]^V} \sum_{i \in U} f_i = \max_{\sigma \in [q]^V} S_r^U(\sigma_{r \setminus p_r}).$$

If we let $S_{i \setminus p_i}^U(\sigma_{i \setminus p_i})$ denote the min-sum of the liberal functions attached on vertices in $(U \cap V_{T_i}) - (X_{p_i} \cup \partial X_{p_i})$ with the configurations of vertices in $X_i - X_{p_i}$ being fixed to $\sigma_{i \setminus p_i}$, the value of $S_{i \setminus p_i}^U(\sigma_{i \setminus p_i})$ can be computed by the following recurrence similarly.

$$S_{i \setminus p_i}^U(\sigma_{i \setminus p_i}) = \Gamma_{X_i - X_{p_i}}^{\sigma_{i \setminus p_i}} + \sum_{t=1}^d \min_{\sigma_{i \setminus t}} \left\{ S_{i \setminus t}^U(\sigma_{i \setminus t}^{\sigma_{i \setminus t}}) \right\}.$$
For each bag $X_i$, since $|X_i| \leq k + 1$ for all $i \in I$, $\sigma_{i_1,p_i}$ has at most $q^{k+1}$ possible values. We compute $S^U_{i_1,p_i}(\sigma_{i_1,p_i})$ for each $\sigma_{i_1,p_i}$ at most once. To compute $S^U_{i_1,p_i}(\sigma_{i_1,p_i})$, we also need to know $X_i \cap X_{p_i}$ for each $X_i$. These can be computed in $O(kn)$ time if the vertices in each $X_i$ are stored in order or using data structures such as hash tables. Furthermore, we need to decide whether a vertex is in the set $X_i \cap X_{p_i}$ or $X_i \cap \partial X_{p_i}$ when computing $S^U_{i_1,p_i}(\sigma_{i_1,p_i})$. The time complexity of the deciding whether a vertex is in $X_i \cap X_{p_i}$ is $O(1)$ if using hash tables. Whether a vertex $i \in V$ is in $X_i \cap \partial X_{p_i}$ can be decided by checking whether all the inputs of $f_i$ are prepared. Since we have assumed that the computation time of $f_i$ is $O(1)$, the time complexity is also $O(1)$. Therefore, the total time complexity of our dynamic programming algorithm is $O(q^{O(k)n})$.

This algorithm is denoted by $DPT_{G,F}(U)$ where $F$ is the set of liberal functions attached on $G$. Since the dynamic programming algorithm is based on enumerating all the possibilities of each bag, many other optimization objectives can also be solved by this algorithm. This dynamic programming algorithm is a key building block of the proof of Theorem 4.1 given as follows.

**Proof of Theorem 4.1** Given a planar embedding of a planar graph $G$, we decompose it into several disjoint $(k + 2)$-outerplanar subgraphs by deleting all the edges between levels congruent to $\ell \pmod{k + 2}$ and $\ell + 1 \pmod{k + 2}$ for some integer $\ell$ that $0 \leq \ell \leq k + 1$. An example of an $8$-outerplanar graph is shown below in Figure 1. Let a possible partition of $G$ with $k = 3$ and $\ell = 4$, the $8$-outerplanar graph is decomposed into two disjoint $4$-outerplanar graphs. The deleted edges are represented by red dash lines and the boundary nodes (level 4 and level 5) and boundary lines are colored by orange in the figure.

The disjoint $(k + 2)$-outerplanar subgraphs are denoted by $G_1, G_2, \ldots, G_t$ from outside to inside. The corresponding vertex sets are denoted by $V_1, V_2, \ldots, V_t$. Each $G_i$ where $1 < i < t$ consists of exactly $k + 2$ levels of nodes and is surrounded by vertices of the inside boundary and the outside boundary. The $G_1$ and $G_t$ may have levels less than $k + 2$. $G_1$ may have only inside boundary vertices and $G_t$ may have only outside boundary vertices. To unify the notations, each $G_i$ is a $k_i$-outerplanar graph where $1 \leq k_i \leq k + 2$.

The set of outside boundary nodes of $G_i$ is denoted by $V_{i,1}$ and the set of inside boundary nodes of $G_i$ is denoted by $V_{i,k_i}$. The set of remaining nodes is denoted by $A_i$ where

$$A_i = \bigcup_{j=2}^{k_i-1} V_{i,j}.$$  

The set of boundary vertices of $G_i$ is denoted by $B_i = V_i - A_i$. We further let

$$A = \bigcup_{i=1}^{t} A_i \quad \text{and} \quad B = \bigcup_{i=1}^{t} B_i.$$  

Each $G_i$ is a $k_i$-outerplanar graph where $k_i \leq k + 2$, thus the treewidth is bounded by $O(k)$. We construct an optimal tree decomposition $T_i$ for each $G_i$ in linear time using Bodlaender’s algorithm [12].

Let $F_i$ denote the set of liberal functions attached on vertices of $G_i$. For each $G_i$, we use the dynamic programming algorithm $DPT_{G_i,F_i}(A_i)$ to maximize the sum of liberal functions attached on the vertices of $A_i$ while ignoring the values of liberal functions attached on $B_i$. By Lemma 4.2 the dynamic programming on $G_i$ for all $1 \leq i \leq t$ can be computed in parallel with time complexity $O(q^{O(k)n})$.

Suppose the optimal solution is $\sigma_{OPT}$ and the maximum sum of liberal functions is $S_{OPT}$. The sum of liberal functions attached on $A_i$ computed by the dynamic programming is denoted by $S_{A_i}$. The sum of liberal functions of $A_1, \ldots, A_t$ computed by dynamic programming is $S_A$. The sum of liberal functions on
Figure 1: An 8-outerplanar graph with a possible partition

$A_i$ is denoted by $S_i$ when the configurations of $A_i$ are fixed to $\sigma_{OPT}$. We have $S_{A_i} \geq S_i$. Therefore it holds that

$$S_A = \sum_{1 \leq i \leq t} S_{A_i} \geq \sum_{1 \leq i \leq t} S_i.$$ 

By pigeon hole principle, for at least one $\ell$, the sum of the liberal functions attached on boundary nodes $B$ is at most $\frac{2}{k+2}$ of $S_{OPT}$. Since $f_i \geq 0$ for all $i \in V$, whatever the configurations of the boundary nodes are, the sum of the liberal functions on boundary nodes is non-negative, denoted by $S_B$. Therefore it holds that

$$S_A + S_B \geq \sum_{1 \leq i \leq t} S_i \geq \frac{k}{k+2} \cdot S_{OPT}.$$ 

Therefore our algorithm gives a solution that is at least $\frac{k}{k+2}$ optimal. For each $0 \leq \ell \leq k + 1$, we need to repeat the dynamic programming algorithm $\mathcal{DPT}_{G_i,F_i}(A_i)$ for all $1 \leq i \leq t$. The total time complexity is $O(q^{O(k)}kn)$. This completes the proof.

This gives an algorithm that can achieve a solution with accuracy $1 - O(1/k)$ of the optimal solution. Therefore, we have a PTAS for approximating $\max_{\sigma \in [q]^V} \sum_{i \in V} f_i$ for any set of liberal functions. From the proof, we can also see why we need $f_i \geq 0$. If the value of a liberal function can be negative, the sum of the liberal functions attached on the boundary nodes may be negative. If $S_B$ is a negative number, then we cannot achieve such a bound.

We can also observe that why the optimization problems such as maximum independent set can be approximated to $\frac{k}{k+1}$. This is because for maximum independent set, the configurations of the boundary
vertices which maximizes the sum of liberal functions attached on the non-boundary vertices are always all zeros. Therefore adjacent subgraphs can share a common boundary so that the approximation ratio can be improved from $\frac{k}{k+2}$ to $\frac{k}{k+1}$.

Another interesting question is how to approximate the max-product $\max_{\sigma \in [q]^V} \prod_{i \in V} f_i$. As mentioned in [104], by the inequality of arithmetic and geometric means
\[
\frac{f_1 + \ldots + f_n}{n} \geq \sqrt[n]{f_1 \cdot \ldots \cdot f_n},
\]
the value $n(\prod_{i \in V} f_i)^{1/n}$ provides a lower bound for $\sum_{i \in V} f_i$ and the equality holds if and only if $f_1 = \ldots = f_n$. This provides a tradeoff between the utility maximization and fairness in some distributed computing problems. Based on the transformation
\[
\max_{\sigma \in [q]^V} \prod_{i \in V} f_i = \max_{\sigma \in [q]^V} \prod_{i \in V} \exp \left( \log f_i \right) \Leftrightarrow \max_{\sigma \in [q]^V} \sum_{i \in V} \log f_i,
\]
we have the following result.

**Corollary 4.3.** If $f_i$ is a liberal function and $f_i \geq 1$ for all $i \in V$ where $|V| = n$, then for fixed $k$, there is an $O(q^{O(k)kn})$-time algorithm for computing $\max_{\sigma \in [q]^V} \prod_{i \in V} f_i$ that achieves a solution of value which is at least
\[
\left( \max_{\sigma \in [q]^V} \prod_{i \in V} f_i \right)^{k/(k+2)}
\]
for general planar graphs.

This algorithm for approximating the max-product of $f_i$ is not a PTAS. Its performance depends on the value of the maximum product and $k$. When $k = 2$, it is equivalent to computing a solution better than the square root of the optimal solution, which can not be expected to achieve a good approximation unless the value of optimal solution is very close to 1. The approximation ratio of our algorithm when $k = 18$ is shown in Figure 2. By this setting, we can see that the approximation ratio is only 0.5 when the value of optimal solution is close to 1000. However, the approximation ratio is very close to 1 when the value of optimal solution is close to 1.

The approximation ratio shown in Figure 2 ignores the product of all the liberal functions attached on the boundary vertices. Therefore, it is reasonable to expect that the approximation ratio achieved by our algorithm will be much better than that shown in Figure 2. Moreover, as we have mentioned, the factor graph can be transformed into our model through $X - F$ contractions. We have shown such a bound for computing the max-product of liberal functions on planar graphs and we will also show similar bounds on other classes of graphs. If a factor graph $G = (X, F, E)$ satisfies $F_j \geq 1$ for all $1 \leq j \leq m$ and can be transformed into any of these classes of graphs on which we give the bound for computing the max-product, then our technique can serve as an alternative of classic methods such as belief propagation for computing the maximum a posterior configuration. Furthermore, to our knowledge, such provable bounds have not been known for belief propagation [88] and its generalized versions [107, 106, 103].

Baker’s technique has also been successfully applied to minimization problems such as minimum vertex cover and minimum dominating set. Unfortunately we do not have such result for minimization problems defined by liberal functions. We prove that if the liberal functions are balanced, we still have a PTAS for computing the min-sum on planar graphs. The concept of balanced liberal function is defined as follows.
Definition 4.4. The liberal functions \( f_i \) defined on graph \( G(V, E) \) are \( \alpha \)-balanced if for each edge \( i \in V \), we have \( b_i \leq f_i \leq \alpha \cdot b_i \) where \( \alpha \geq 1 \) is a constant. The number \( b_i \geq 0 \) is called the balancer of vertex \( i \in V \).

If the liberal functions are balanced, we trivially achieve a \( \frac{1}{\alpha} \)-approximation algorithm for maximization and an \( \alpha \)-approximation algorithm for minimization. When the liberal functions of \( G \) are balanced, we have the following result for computing the min-sum.

Theorem 4.5. If \( f_i \) is a \( \alpha \)-balanced liberal function for all \( i \in V \) where \( |V| = n \), then for fixed \( k \), there is an \( O(q^{O(k)}kn) \)-time algorithm for computing \( \min_{\sigma \in [q]^{V}} \sum_{i \in V} f_i \) that achieves a solution of value at most \( 1 + \frac{2(\alpha - 1)}{k+2} \) \( \cdot \) \( \text{optimal} \) for general planar graphs.

Proof. We use a similar algorithm in the proof of Theorem 4.1 but in the reverse way. Given a planar embedding of planar graph \( G \), we decompose it into several disjoint subgraphs by deleting the edges between levels congruent to \( \ell \) (mod \( k + 2 \)) and \( \ell + 1 \) (mod \( k + 2 \)) for some integer \( \ell \) that \( 0 \leq \ell \leq k + 1 \). As in the proof of Theorem 4.1, the disjoint subgraphs are denoted by \( G_1, \ldots, G_t \) where their vertex sets are denoted by \( V_1, \ldots, V_t \). The sets of vertices of levels from outside to inside are denoted by \( V_{i,1}, \ldots, V_{i,k} \). And we let

\[
A_i = \bigcup_{j=2}^{k-1} V_{i,j} \quad \text{and} \quad B_i = V_i - A_i
\]

Furthermore, we let

\[
A = \bigcup_{1 \leq i \leq t} A_i \quad \text{and} \quad B = V - A.
\]

That is, \( A \) is the set of all the non-boundary vertices and \( B \) is the set of all the boundary vertices of a possible decomposition of the graph \( G \). We construct a tree decomposition \( T_i \) for \( G_i \) in linear time. Let \( F_i \) denote
the set of liberal functions attached on vertices in $G_i$. Then we use $DPT_{G_i,F_i}(A_i)$ to minimize the sum of liberal functions of vertices in $A_i$ while ignoring the values of liberal functions of vertices in $B_i$ for each $G_i$ in parallel. The min-sum of liberal functions of $A_1, \ldots, A_t$ computed by dynamic programming is $S_A$ and the sum of liberal functions attached on vertices in $B$ is $S_B$. We have

$$S_A + S_B = \sum_{i \in A} f_i + \sum_{i \in B} f_i$$

Suppose the optimal solution is $S_{OPT}$, by pigeon hole principle, for at least one $\ell$ that $0 \leq \ell \leq k + 1$, at most \(2^k\) of $S_{OPT}$ comes from $f_i$ where $i \in B$. Then it holds that

$$S_A \leq \frac{k}{k + 2} \cdot S_{OPT}$$

and

$$S_B \leq \frac{2 \cdot \alpha}{k + 2} \cdot S_{OPT}.$$ 

Therefore we have

$$S_A + S_B \leq \left(1 + \frac{2 \cdot (\alpha - 1)}{k + 2}\right) \cdot S_{OPT}.$$ 

The dynamic programming algorithm $DPT_{G_i,F_i}(A_i)$ takes time $O(q^{O(k)n})$. For each $0 \leq \ell \leq k + 1$, we need to repeat the dynamic programming, thus the whole time complexity is $O(q^{O(k)n})$. This completes the proof.

We can see that the approximability of minimization does not depend on the specific value of $b_i$ but only on the parameter $\alpha$. When $\alpha$ is a constant, this algorithm gives a PTAS for computing the min-sum of liberal functions. For computing the min-product of liberal functions, we have a result similar to Corollary 4.3.

**Corollary 4.6.** If $f_i$ is an $\alpha$-balanced liberal function for all $i \in V$ where $|V| = n$, then for fixed $k$, there is an $O(q^{O(k)n})$-time algorithm for computing $\min_{\sigma \in [q]^V} \prod_{i \in V} f_i$ that achieves a solution of value which is at most

$$\left(\min_{\sigma \in [q]^V} \prod_{i \in V} f_i\right)^{1 + \frac{2(\alpha - 1)}{k + 2}}$$

for general planar graphs.

However, we cannot say that for every set of unbalanced liberal functions, its min-sum is hard to approximate. For classic combinatorial minimization problems, computing the minimum vertex cover can be formulated as computing $\min_{\sigma \in [q]^V} \sum_{i \in V} f_i$ by setting $f_i$ as

$$f_i = \begin{cases} 1 & \text{if } \sigma_i = 1, \\ 0 & \text{if } \sigma_i = 0 \text{ and } \sigma_j = 1 \text{ for } \forall j \in N(i) \\ +\infty & \text{if } \sigma_i = 0 \text{ and } \sigma_j = 0 \text{ for } \exists j \in N(i). \end{cases}$$

Computing the minimum dominating set can be formulated as computing $\min_{\sigma \in [q]^V} \sum_{i \in V} f_i$ by setting $f_i$ as

$$f_i = \begin{cases} 1, & \text{if } \sigma_i = 1, \\ 0, & \text{if } \sigma_i = 0, \text{ and } \sigma_j = 1 \text{ for } \exists j \in N(i), \\ +\infty, & \text{if } \sigma_i = 0, \text{ and } \sigma_j = 0 \text{ for } \forall j \in N(i). \end{cases}$$
For both minimum vertex cover and minimum dominating set, \( f_i = +\infty \) means current configurations are not valid. Hence the \( f_i = +\infty \) can also be replaced by \( f_i = n \) when the configurations are invalid since the sizes of any dominating set and vertex cover will not exceed \( n \). Both the functions \( f_i \) for minimum vertex cover and minimum dominating set are unbalanced, but their minimizations admit PTAS on planar graphs [8]. The following theorem shows that there does not exist PTAS for computing the min-sum of arbitrarily defined liberal functions even on planar graphs.

**Theorem 4.7.** There does not exist PTAS for computing the min-sum of unbalanced liberal functions even on planar graphs unless P = NP.

**Proof.** We simply construct a reduction from computing the chromatic number to computing the min-sum of liberal functions. We set \( f_i \) as

\[
f_i = \begin{cases} w_x & \text{if } \sigma_i = x \in [q] \text{ and } \sigma_i \neq \sigma_j \text{ for } \forall (i, j) \in E \\ +\infty & \text{if } \sigma_i = \sigma_j \text{ for any } (i, j) \in E \end{cases}
\]

where \( w_x \) satisfies

\[
w_x > n \cdot w_{x-1}.
\]

This is allowed because both \( G \) and the set of liberal functions serve as inputs. By this setting, the liberal functions are not balanced. If the graph \( G \) is \( x \)-colorable, then the value of the minimum sum \( S_{\min} = \min_\sigma \sum_{i \in V} f_i \) must fall into the interval \([w_x, w_x \cdot n]\). For any \( x, y \in [q] \), \([w_x, w_x \cdot n]\) and \([w_y, w_y \cdot n]\) are pairwise disjoint. By four color theorem, any planar graph is 4-colorable. Furthermore, it is known that 3-coloring problem remains NP-complete even on planar graph of degree 4 [20]. It implies that it is NP-hard to approximate the chromatic number within approximation ratio \( 4/3 \) even on planar graphs. Therefore if we have a PTAS for computing the min-sum of liberal functions on planar graphs, then we have a polynomial time algorithm for computing the chromatic number of planar graphs, which leads to a contradiction.

By enlarging the gaps between the disjoint intervals, we can achieve stronger inapproximability results by this reduction.

**Corollary 4.8.** There does not exist PTAS for computing the min-sum of unbalanced liberal functions on graphs with bounded local treewidth, \( H \)-minor-free graphs, geometric graphs and graphs with bounded crossing number per edge and more general graphs unless P = NP.

The impossibility result also holds on geometric graphs because by Koebe-Andreev-Thurston theorem [72], for every connected simple planar graph \( G \) there is a circle packing in the plane whose intersection graph is isomorphic to \( G \). Therefore, the class of disk graphs properly covers the class of planar graphs.

This impossibility result only shows that computing the min-sum for some sets of unbalanced liberal functions does not have PTAS. But there are many sets of unbalanced liberal functions such as minimum vertex cover and minimum dominating set for which computing the min-sums has a PTAS. In our reduction, the values of liberal functions depend on the number of vertices \( n \) of the input graph, which is global information for the locally-defined liberal function. A clearer characterization for the approximability of computing the min-sum of unbalanced liberal functions can be left as further research.

### 4.2 Graphs with bounded local treewidth

The following theorem gives a precise characterization of graphs with bounded local treewidth.
Theorem 4.9. (Eppstein [38]) Let \( \mathcal{F} \) be a minor closed family of graphs. Then \( \mathcal{F} \) has bounded local treewidth if and only if \( \mathcal{F} \) does not contain all apex graphs.

A graph is an apex graph if it has a vertex whose removal results in a planar graph. For example, \( K_5 \) and \( K_{3,3} \) are apex graphs. Theorem 4.9 shows that a graph \( G \) has bounded local treewidth if and only if it is apex-minor-free.

Theorem 4.10. (Demaine et al. [26]) Any apex-minor-free graph has linear local treewidth.

The linearity of local treewidth has been proved for some subclasses of apex-minor-free graphs before [26], such as bounded-genus graphs [38] and single-crossing-minor-free graphs [31].

Theorem 4.11. If \( f_i \) is a liberal function for all \( i \in V \) where \( |V| = n \), then for fixed \( k \), there is an \( O(q^{O(k)}kn) \)-time algorithm for computing \( \max_{\sigma \in [q]^V} \sum_{i \in V} f_i \) that achieves a solution of value at least \( k/(k + 2) \) optimal for graphs with bounded local treewidth.

Proof. Choosing any vertex \( v \in V \) as root, construct a BFS tree \( T \) rooted at \( v \). The layer of vertices is defined as its distance to \( v \). Moreover, the set of vertices from layer \( i \) to layer \( j \) is denoted by

\[ L_v^G[i, j] = \{ u \in V | i \leq \text{dist}(u, v) \leq j \}. \]

If \( i > j \), \( L_v^G[i, j] = \emptyset \). For any \( i \leq j \), \( L_v^G[i, j] \) has bounded local treewidth. This is because if we obtain a minor \( H \) of \( G \) by contracting the subgraph of \( G \) induced by \( L_v^G[0, i - 1] \) to a single vertex \( v' \), \( L_v^G[i, j] \subseteq L_{v'}^H[1, j - i + 1] \). Since \( G \) is apex-minor-free, \( H \) is also apex-minor-free. Therefore, \( H \) has bounded local treewidth. Then we have \( tw(L_v^G[i, j]) = O(j - i + 1) \), which implies any subgraph induced by consecutive \( k \) levels of vertices in the BFS tree has treewidth bounded by \( O(k) \).

We delete all the edges between levels congruent to \( \ell \) \((\text{mod } k + 2)\) and \( \ell + 1 \) \((\text{mod } k + 2)\) for some integer \( \ell \) that \( 0 \leq \ell \leq k + 1 \). Then \( G \) is decomposed into several disjoint subgraphs \( G_1, \ldots, G_t \). For each \( G_i \) where \( 1 \leq i \leq t \), we construct a tree decomposition \( T_i \). We still use the dynamic programming algorithm on \( T_i \) to maximize the sum of liberal functions attached on non-boundary vertices while ignoring the values of liberal functions on the boundary of \( G_i \). Since \( tw(G_i) = O(k) \), the time complexity of the dynamic programming is \( O(q^{O(k)}n) \). By a similar argument of Theorem 4.1, there is at least one \( \ell \) that the combination of the solutions on the disjoint subgraphs is at least \( \frac{k}{k + 2} \) optimal of the optimal solution.

Therefore, we have a similar result as Corollary 4.3 on graphs of bounded local treewidth.

Corollary 4.12. If \( f_i \) is a liberal function and \( f_i \geq 1 \) for all \( i \in V \) where \( |V| = n \), then for fixed \( k \), there is an \( O(q^{O(k)}kn) \)-time algorithm for computing \( \max_{\sigma \in [q]^V} \prod_{i \in V} f_i \) that achieves a solution of value which is at least

\[ \left( \max_{\sigma \in [q]^V} \prod_{i \in V} f_i \right)^{k/(k + 2)} \]

for graphs with bounded local treewidth.

Theorem 4.13. If \( f_i \) is a \( \alpha \)-balanced liberal function for all \( i \in V \) where \( |V| = n \), then for fixed \( k \), there is an \( O(q^{O(k)}kn) \)-time algorithm for computing \( \min_{\sigma \in [q]^V} \sum_{i \in V} f_i \) that achieves a solution of value at most

\[ 1 + \frac{2^{2(\alpha - 1)}}{k + 2} \]

optimal for graphs with bounded local treewidth.
Proof. Use the same technique of Theorem 4.11 we decompose the input graph \( G \) into several disjoint subgraphs \( G_1, \ldots, G_t \). For each \( G_i \) where \( 1 \leq i \leq t \), we construct a tree decomposition \( T_i \) of \( G_i \). Then we use the dynamic programming algorithm to minimize the sum of liberal functions attached on the non-boundary vertices of \( G_i \) while ignoring the values of liberal functions attached on the boundary of \( G_i \). Then by a similar argument of Theorem 4.5 there is at least one partition which makes the final solution at most \( 1 + \frac{2(\alpha - 1)}{k+2} \) of the optimal solution.

Also, we have a similar corollary as Corollary 4.6.

Corollary 4.14. If \( f_i \) is a \( \alpha \)-balanced liberal function for all \( i \in V \) where \( |V| = n \), then for fixed \( k \), there is an \( O(q^{O(k)} kn) \)-time algorithm for computing \( \min_{\sigma \in [q]^V} \prod_{i \in V} f_i \) that achieves a solution of value which is at most

\[
\left( \min_{\sigma \in [q]^V} \prod_{i \in V} f_i \right)^{1 + \frac{2(\alpha - 1)}{k+2}}
\]

for graphs with bounded local treewidth.

4.3 H-minor-free graphs

Robertson and Seymour’s graph-minor theorem shows that any nontrivial minor-closed graph class can be characterized by a finite set of forbidden minors. Therefore, \( H \)-minor-free graphs are quite a general graph class. By Theorem 4.9 we know that graphs with bounded local treewidth are exactly apex-minor-free graphs, thus \( H \)-minor-free graphs may not have bounded local treewidth.

A graph \( H \) is a \( k \)-apex graph of a graph \( G \) if \( G = H \setminus A \) for some subset \( A \) of at most \( k \) vertices which is called apices. The definition of almost-embeddable graph is given as follows.

Definition 4.15. A graph \( G \) is almost-embeddable on a surface \( \Sigma \) if \( G \) can be written as the union of \( k + 1 \) graphs \( G_0 \cup G_1 \cup \ldots \cup G_k \), satisfying the following conditions:

- \( G_0 \) has an embedding on \( \Sigma \).
- The graphs \( G_1, G_2, \ldots, G_k \) are pairwise disjoint, called vortices.
- For each index \( i \geq 1 \), there is a disk \( D_i \) inside some face \( F_i \) of \( G_0 \), such that \( U_i = V(G_0) \cap V(G_i) = V(G_0) \cap D_i \). Moreover, the disks \( D_i \) are pairwise disjoint.
- For each index \( i \geq 1 \), the subgraph \( G_i \) has pathwidth less than \( k \). Moreover, \( G_i \) has a path decomposition \( \langle X_i^1, X_i^2, \ldots, X_i^{r_i} \rangle \) with \( r_i \leq k \), such that \( v_i^j \in X_i^j \) for \( 1 \leq j \leq r_i \), where \( v_i^1, v_i^2, \ldots, v_i^{r_i} \) are the vertices of \( U_i \) indexed in cyclic order around the face \( F_i \), clockwise or anti-clockwise.

Lemma 4.16. (Grohe [52]) The class of all graphs almost embeddable in a fixed surface \( S \) has linear local treewidth.

Definition 4.17. A graph \( G \) is \( h \)-almost-embeddable on a surface \( \Sigma \) if \( H \) is a \( h \)-apex graph of a graph that is almost embeddable on \( \Sigma \).

Theorem 4.18. (Roberston and Seymour [92]) For any graph \( H \), there is an integer \( h \geq 0 \) depending only on \( |V(H)| \) such that any \( H \)-minor-free graph is a \( h \)-clique sum of a finite number of graphs that are \( h \)-almost-embeddable on some surfaces on which \( H \) cannot be embedded.
Theorem 4.18 says that any $H$-minor-free graph $G$ can be expressed as a “tree structure” of pieces, where each piece can be embedded on a surface on which $H$ cannot be embedded after deleting at most $h$ apex vertices.

Theorem 4.19. (DeVos et al. [34]) For the clique-sum decomposition of a $H$-minor-free graphs, written as $G_1 \oplus \ldots \oplus G_t$, the join set of each clique-sum operation between $G_1 \oplus \ldots \oplus G_{i-1}$ and $G_i$ is a subset of the apices of $G_i$. Moreover, each join set of the clique-sum decomposition involving $G_j$ contains at most three vertices of the bounded-genus part of $G_j$.

The following theorem gives a polynomial-time algorithm for computing the clique-sum decomposition with the additional properties guaranteed by Theorem 4.19.

Theorem 4.20. (Demaine et al. [28]) For a fixed graph $H$, there is a constant $c_H$ such that, for any integer $k \geq 1$ and for every $H$-minor-free graph $G$, the vertices of $G$ (or the edges of $G$) can be partitioned into $k + 1$ such that any $k$ of the sets induce a graph of treewidth at most $c_H k$. Furthermore, such a partition can be found in polynomial time.

Grohe et al. [53] give a quadratic time algorithm that is faster for computing the clique-sum decomposition of $H$-minor-free graphs. When we describe our approximation algorithm, we always assume that such a clique-sum decomposition has already been given.

Definition 4.21. Graph class $\mathcal{G}$ has truly sublinear treewidth with parameter $\lambda$ where $0 < \lambda < 1$, if for every $\eta > 0$, there exists $\beta > 0$ such that for any graph $G \in \mathcal{G}$ and $X \subseteq V(G)$ the condition $tw(G \setminus X) \leq \eta$ yields that $tw(G) \leq \eta + \beta |X|^\lambda$.

Lemma 4.22. (Fomin et al. [43]) Let $\mathcal{G}_H$ be a class of graphs excluding a fixed graph $H$ as a minor, then $\mathcal{G}_H$ has truly sublinear treewidth with $\lambda = \frac{1}{2}$.

Theorem 4.23. If $f_i$ is a liberal function for all $i \in V$ where $|V| = n$, then for fixed $k$ and the given clique-sum decomposition, there is a $O(q^{O(k)})$-time algorithm for computing $\max_{\sigma \in [\lambda]^V} \sum_{i \in V} f_i$ that achieves a solution of value which is at least $k / (k + 2)$ optimal for $H$-minor-free graphs.

Proof. Our technique is similar to the proof in [28] of Theorem 4.20. Suppose the clique-sum decomposition of the input $H$-minor-free graph $G$ is $G_1 \oplus G_2 \oplus \ldots \oplus G_t$ where each $G_i$ (1 $\leq i \leq t$) is an $h$-almost embeddable graph. The join set $J_i$ of the $i$-th clique-sum operation $(G_1 \oplus \ldots \oplus G_i) \oplus G_{i+1}$ is a subset of the apex set $X_{i+1}$ of $G_{i+1}$. Our approximation algorithm takes the clique-sum decomposition as input, the apex set $X_i$ of each $G_i$ is given as part of the input clique-sum decomposition.

By the definition of the $h$-almost embeddable graphs, $G_i - X_i$ is almost embeddable on a bounded-genus surface where $X_i$ contains at most $h$ vertices. By lemma 4.16 $G_i - X_i$ has bounded local treewidth. From $i = 1$ to $t$, we choose a vertex $v_i \in G_i - X_i$ and construct a BFS tree $T_i$ of $G_i - X_i$ rooted at $v_i$. Each vertex in $u \in G_i - X_i$ is labeled by the distance between $u$ and $v_i$ modulo $k + 2$. After this step, we delete all the edges between levels labeled by $\ell \pmod{k + 2}$ and the adjacent levels labeled by $\ell + 1 \pmod{k + 2}$. Then the $G_i - X_i$ part is decomposed into several disjoint subgraphs with treewidth at most $c(k + 2)$ for some constant $c > 0$. Since $J_{i-1} \subseteq X_i$, the vertices in $J_{i-1}$ has already been labelled in $G_1 \oplus G_2 \oplus \ldots \oplus G_{i-1}$. We label the vertices in $X_i \setminus J_{i-1}$ arbitrarily by the integers from 0 to $k + 1$. After the edge deletions, the obtained graphs $G'_i$ are still $H$-minor-free for $1 \leq i \leq t$. By Lemma 4.22 the treewidth of $G'_i$ is at most $ck + \beta |X_i|^{1/2} \leq ck + \beta h^{1/2} = O(k)$. It is known that $tw(G \oplus H) \leq \max\{tw(G), tw(H)\}$ [13, 31], thus we have
\[
tw(G'_1 \oplus G'_2 \oplus \ldots \oplus G'_t) \leq \max\{tw(G'_1), \ldots, tw(G'_t)\}.
\]
This shows that any $H$-minor-free graph can be transformed into a graph with treewidth bounded by $O(k)$ by deleting at most $|E|/k$ edges. Given a clique-sum decomposition, the vertex labelling and edge deletions can be finished in linear time.

Let $A_i$ denote the vertices with labels other than $\ell \pmod{k+2}$ and $\ell + 1 \pmod{k+2}$ in $G'_i$ and $B_i = V_i \setminus A_i$. Furthermore, we let

$$A = \bigcup_{i=1}^{t} A_i \text{ and } B = \bigcup_{i=1}^{t} B_i.$$

Since we only delete the edges between vertices by $\ell \pmod{k+2}$ and the adjacent levels labeled by $\ell + 1 \pmod{k+2}$, the liberal functions attached on vertices in $A$ will not be undefined. The treewidth of the input graph is bounded by $O(k)$ after the edge deletions. We use the dynamic programming algorithm to compute the max-sum of liberal functions attached on vertices in $A$ while ignoring the values of the liberal functions attached on vertices in $B$. The running time of the dynamic programming algorithm is $O(q^{O(k)n})$.

Suppose the optimal solution is $S_{\text{OPT}}$, then there is at least one $\ell$ such that there is at most $\frac{2}{k+2}$ of the optimal solution falls into the vertices in $B$. If we repeat this computation for all $0 \leq \ell \leq k+1$, the optimal solution among these results is at least $\frac{1}{k+2}$ optimal. The time complexity of our algorithm is $O(q^{O(k)n})$.

\begin{corollary}
If $f_i$ is a liberal function and $f_i \geq 1$ for all $i \in V$ where $|V| = n$, then for fixed $k$ and the given clique-sum decomposition, there is a $O(q^{O(k)n})$-time algorithm for computing $\max_{\sigma \in [q]^V} \prod_{i \in V} f_i$ that achieves a solution of value which is at least

$$\left( \max_{\sigma \in [q]^V} \prod_{i \in V} f_i \right)^{k/(k+2)}$$

for $H$-minor-free graphs.
\end{corollary}

\begin{theorem}
If $f_i$ is a $\alpha$-balanced liberal function for all $i \in V$ where $|V| = n$, then for fixed $k$ and the given clique-sum decomposition, there is a $O(q^{O(k)n})$-time algorithm for computing $\min_{\sigma \in [q]^V} \sum_{i \in V} f_i$ that achieves a solution of value which is at most $1 + \frac{2(\alpha-1)}{k+2}$ optimal for $H$-minor-free graphs.
\end{theorem}

\begin{proof}
After the edge deletion, the input $H$-minor-free graph $G'_1 \oplus G'_2 \oplus \ldots \oplus G'_t$ has treewidth bounded by $O(k)$. We also let $A_i$ denote the vertices with labels other than $\ell \pmod{(k+2)}$ and $\ell + 1 \pmod{(k+2)}$ in $G'_i$ and $B_i = V_i \setminus A_i$. Moreover, we let

$$A = \bigcup_{i=1}^{t} A_i \text{ and } B = \bigcup_{i=1}^{t} B_i.$$

We use the dynamic programming algorithm to compute the min-sum of liberal functions attached on vertices in $A$ while ignoring the values of the liberal functions attached on vertices in $B$. The running time of the dynamic programming algorithm is $O(q^{O(k)n})$. Also, there is at least one $\ell$ such that there is at most $\frac{2}{k+2}$ of the optimal solution falls into the vertices in $B$. Therefore the minimum solution can be found in $O(q^{O(k)n})$ time, which is at most $1 + \frac{2(\alpha-1)}{k+2}$ of the optimal solution.
\end{proof}
Corollary 4.26. If \( f_i \) is a \( \alpha \)-balanced liberal function for all \( i \in V \) where \( |V| = n \), then for fixed \( k \) and the given clique-sum decomposition, there is an \( O(q^{O(k)}kn) \)-time algorithm for computing \( \min_{\sigma \in [q]^V} \prod_{i \in V} f_i \) that achieves a solution of value which is at most
\[
\left( \min_{\sigma \in [q]^V} \prod_{i \in V} f_i \right)^{1+2(\alpha-1)/k+2/k+2}
\]
for \( H \)-minor-free graphs.

4.4 Geometric graphs

We consider the \( d \)-dimensional Euclidean space \( \mathbb{R}^d \). The distance between point \( i \) and point \( j \) is defined by the Euclidean distance, denoted by \( \text{dist}(i,j) = ||i - j||_d \). Among various geometric objects, we only consider \( d \)-dimensional balls. Our results also hold for other geometric objects no matter it is convex or not. To do this, we choose the midpoint of the line determined by two points of the longest distance on the object as the center and the longest distance from the center to any point of the object as the radius. Then the terminologies of \( d \)-dimensional balls can also be used for these objects. Their geometric graphs can be constructed in a similar way. The definitions of some typical geometric graphs are listed as follows.

**Intersection Graph.** Let \( d_i \) represent the diameter of the \( d \)-dimensional ball centered at point \( i \) for \( i \in V \). An intersection graph \( G(V,E) \) can be constructed by adding an edge between each pair of point \( i \) and point \( j \) where \( \text{dist}(i,j) \leq (d_i + d_j)/2 \).

**Interference Graph.** The model of interference graph is only defined for \( d \)-dimensional balls which has been extensively studied by problems in wireless networks. It is a theoretical simplification of the realistic SINR model. The difference between interference graph and intersection graphs is that an interference graph is a directed graph while an intersection graph is undirected. There is an edge from the the center of ball \( i \) to the center of ball \( j \) if \( \text{dist}(i,j) \leq d_i/2 \), as shown in Figure 3.

![Intersection Graph](image1.png) ![Interference Graph](image2.png)

**Figure 3:** Geometric graphs

The recognition of unit disk graphs is NP-hard [17]. For this reason, we assume that a geometric representation of the graph is always given as input. This assumption is reasonable in many applications. We will not distinguish between the geometric graphs and its geometric representation later.
A geometric graph has several important parameters related to the performances of our approximation algorithms, listed as follows. The concept of $\lambda$-precision is presented in [60]. The concepts of thickness and density are presented in [100]. The concept of $\lambda$-precision is defined as follows.

**Definition 4.27.** For $\lambda > 0$, a geometric graph is a $\lambda$-precision geometric graph if the centers of any two $d$-dimensional balls are at least $\lambda$ apart.

To defined the concepts of thickness and density, we first introduce the concepts of slab decomposition and grid decomposition in $d$-dimensional Euclidean space which are introduced in [100] for unit disk graphs. A slab decomposition is partitioning the $d$-dimensional Euclidean space using parallel $(d - 1)$-dimensional hyperplanes where the distances between neighboring hyperplanes are equal. A grid decomposition is recursively partitioning the $d$-dimensional Euclidean space using orthogonal slab decompositions where all the distances between neighboring hyperplanes are equal, until the space is partitioned into $d$-dimensional hypercubes of the same size. The distance between neighboring hyperplanes are set to $d_{\text{max}}$ and $d_{\text{max}}/2$ for intersection graphs and interference graphs respectively where $d_{\text{max}}$ is the diameter of the maximum $d$-dimensional ball. By this decomposition, any vertex has only edges linked to vertices in the neighboring hypercubes.

**Definition 4.28.** Given a slab decomposition, the thickness $\text{th}(G)$ of a geometric graph is determined by the maximum number of centers between any two neighboring hyperplanes. If a center falls precisely on a hyperplane $g(x_1, \ldots, x_d) = 0$, it belongs to the subspace divided by the hyperplane where $g(x_1, \ldots, x_d) < 0$.

**Definition 4.29.** Given a grid decomposition, the density $d_{\text{s}}(G)$ of a geometric graph is determined by the maximum number of centers in any $d$-dimensional hypercube. If a center falls precisely on several hyperplanes $g_1, \ldots, g_m$, it belongs to the hypercube where $g_1 < 0, \ldots, g_m < 0$.

A $\lambda$-precision geometric graph has density $\Theta(\frac{1}{\lambda})$ in $d$-dimensional Euclidean space. When the number of dimensions $d$ is fixed, a slab decomposition or grid decomposition which gives the minimum thickness or density can be found in polynomial time. Later we always assume such a decomposition is given.

**Theorem 4.30.** Given the grid decomposition with density $d_{\text{s}}(G)$, if $f_i$ is a liberal function for all $i \in V$ where $|V| = n$, then for fixed $k$ and $d$, there is an $O(q^{O((k+2)d-1) \cdot \max_{\sigma \in [q]^V} \sum_{i \in V} f_i})$-time algorithm for computing $\max_{\sigma \in [q]^V} \sum_{i \in V} f_i$ that achieves a solution of value which is at least

$$\left(\frac{k}{k+2}\right)^{d-1} \cdot \left(\max_{\sigma \in [q]^V} \sum_{i \in V} f_i\right)$$

for $d$-dimensional geometric graphs with bounded density.

**Proof.** For the first dimension of the grid decomposition, we remove all the edges between the centers in hypercubes of levels congruent to $\ell_1 \pmod{k+2}$ and the centers in hypercubes of levels congruent to $\ell_1 + 1 \pmod{k+2}$ for some $0 \leq \ell_1 \leq k+1$. Then $G$ is decomposed into several disjoint subgraphs with $k+2$ levels of hypercubes in the first dimension, all the levels of hypercubes in the second dimension, ..., all the levels of hypercubes in the $d$-th dimension, denoted by $G_1^1, \ldots, G_1^{t_1}$.

For the second dimension of $G_i^1$ for some $1 \leq i \leq t_1$, we remove all the edges between the centers in hypercubes of levels congruent to $\ell_2 \pmod{k+2}$ and the centers in hypercubes of levels congruent to $\ell_2 + 1 \pmod{k+2}$ for some $0 \leq \ell_2 \leq k+1$. Then $G_i^1$ is decomposed into several disjoint subgraphs with...
$k + 2$ levels of hypercubes in the first dimension, $k + 2$ levels of hypercubes in the second dimension, all the levels of hypercubes in the third dimension, ..., all the levels of hypercubes in the $d$-th dimension, denoted by $G_1^d, \ldots, G_{t_d}^d$.

We repeat this procedure until for the $(d - 1)$-th dimension of $G_i^{d-2}$ for some $1 \leq i \leq t_{d-2}$, we remove all the edges between the centers in hypercubes of levels congruent to $\ell_{d-1} \mod k + 2$ and the centers in hypercubes of levels congruent to $\ell_{d-1} + 1 \mod k + 2$ for some $0 \leq \ell_{d-1} \leq k + 1$. Then $G_i^{d-2}$ is decomposed into several disjoint subgraphs each of which consists of $k + 2$ levels of hypercubes in the first dimension, $k + 2$ levels of hypercubes in the second dimension, ..., $k + 2$ levels of hypercubes in the $(d - 1)$-th dimension, all the levels of hypercubes in the $d$-th dimension, denoted by $G_1^{d-1}, \ldots, G_{t_{d-1}}^{d-1}$.

For each subgraph $G_i^{d-1}$, it satisfies that

$$\text{th}(G_i^{d-1}) \leq (k + 2)^{d-1} \cdot ds(G),$$

for all $1 \leq i \leq t_{d-1}$. Moreover, the pathwidth of $G_i^{d-1}$ is at most $2 \cdot \text{th}(G_i^{d-1}) - 1$ where each bag of the path decomposition contains two neighboring hypercubes.

For each $G_i^{d-1}$, we use the dynamic programming algorithm of Lemma 4.2 to maximize the sum of liberal functions attached on vertices in hypercubes that are in level 2 to level $k + 1$ in the first dimension, level 2 to level $k + 1$ in the second dimension, ..., level 2 to level $k + 1$ in the $(d - 1)$-th dimension and all levels in the $d$-th dimension while ignoring the values of liberal functions attached on all other vertices. The time complexity of the dynamic programming on each $G_i^{d-1}$ is $O(q^O((k+2)^{d-1 \cdot ds(G)}) \cdot n)$.

By the pigeon hole principle, there is at least one tuple $(\ell_1, \ldots, \ell_{d-1})$ that the sum of the results of dynamic programming on each $G_i^{d-1}$ is at least $(\frac{k}{k+2})^{d-1}$ of the optimal solution. For each possible $(\ell_1, \ldots, \ell_{d-1})$, we need to compute the dynamic programming for all the $G_i^{d-1}$. Therefore, the time complexity of our algorithm is $O(q^O((k+2)^{d-1 \cdot ds(G)}) \cdot k^{d-1 \cdot n}).$

This result gives us a PTAS for computing the max-sum of liberal functions on geometric graphs. Given an error $\epsilon > 0$, we choose $k$ as

$$k = \left\lceil \frac{2(1-\epsilon)^{\frac{1}{d-1}}}{{1 - (1-\epsilon)^{\frac{1}{d-1}}} \right\rceil,$$

then the solution achieved by our algorithm is at least $(1 - \epsilon)$ optimal.

**Corollary 4.31.** Given the grid decomposition with density $ds(G)$, if $f_i$ is a liberal function and $f_i \geq 1$ for all $i \in V$ where $|V| = n$, then for fixed $k$ and $d$, there is an $O(q^O((k+2)^{d-1 \cdot ds(G)}) \cdot k^{d-1 \cdot n})$-time algorithm for computing $\max_{\sigma \in [q]^V} \prod_{i \in V} f_i$ that achieves a solution of value which is at least

$$\left(\max_{\sigma \in [q]^V} \prod_{i \in V} f_i\right)^{(\frac{k}{k+2})^{d-1}}$$

for $d$-dimensional geometric graphs with bounded density.

**Theorem 4.32.** Given the grid decomposition with density $ds(G)$, if $f_i$ is a $\alpha$-balanced liberal function for all $i \in V$ where $|V| = n$, then for fixed $k$ and $d$, there is an $O(q^O((k+2)^{d-1 \cdot ds(G)}) \cdot k^{d-1 \cdot n})$-time algorithm for computing $\min_{\sigma \in [q]^V} \sum_{i \in V} f_i$ that achieves a solution of value which is at most

$$\left(1 + \frac{2 \cdot (\alpha - 1)}{k + 2}\right)^{d-1} \cdot \left(\min_{\sigma \in [q]^V} \sum_{i \in V} f_i\right)^{(\frac{k}{k+2})^{d-1}}$$

24
for \(d\)-dimensional geometric graphs with bounded density.

**Proof.** For each \(G_i^{d-1}\), we use the dynamic programming algorithm of Lemma \(4.2\) to minimize the sum of liberal functions attached on vertices in hypercubes that are in level 2 to level \(k+1\) in the first dimension, level 2 to level \(k+1\) in the second dimension, ..., level 2 to level \(k+1\) in the \((d-1)\)-th dimension and all levels in the \(d\)-th dimension while ignoring the values of liberal functions attached on all other vertices. The time complexity of the dynamic programming on each \(G_i^{d-1}\) is \(O(q^{O((k+2)^{d-1}ds(G))} n)\).

This result gives us a PTAS for computing the min-sum of \(\alpha\)-balanced liberal functions on geometric graphs. Given an error \(\epsilon > 0\), we choose \(k\) as

\[
k = \left\lceil \frac{2(\alpha - (1 + \epsilon)\frac{1}{d+1})}{(1 + \epsilon)\frac{1}{d+1} - 1} \right\rceil,\]

then the solution achieved by our algorithm is at most \((1 + \epsilon)\) optimal.

**Corollary 4.33.** Given the grid decomposition with density \(ds(G)\), if \(f_i\) is a \(\alpha\)-balanced liberal function for all \(i \in V\) where \(|V| = n\), then for fixed \(k\) and \(d\), there is an \(O(q^{O((k+2)^{d-1}ds(G))} \cdot k^{d-1} \cdot n)\)-time algorithm for computing the min-product that achieves a solution of value which is at most

\[
\left( \min_{\sigma \in [q]^V} \prod_{i \in V} f_i \right)^{(1 + 2(\alpha - 1))^{d-1}}
\]

for \(d\)-dimensional geometric graphs with bounded density.

### 4.5 Graphs with bounded crossings per edge

We follow the definition of graphs with bounded crossings per edge in [51].

**Definition 4.34.** An embedding of a graph \(G\) on a surface \(S\) of genus \(g\) is a good embedding if it satisfies the following conditions:

- all vertices of the graph are given as distinct points on \(S\),
- no two edge crossings locate at the same point on \(S\),
- for any edge, no vertex except the endpoints of the edge locate on the edge.

**Definition 4.35.** The crossing parameter \(\varphi\) of a graph \(G\) embedded on a surface \(S\) is the minimum over all good embeddings on \(S\) of the maximum over all edges \(e\) of the number of edge crossings of \(e\).

Several natural classes of graphs are graphs with bounded crossing parameter, such as geometric graphs with bounded density discussed above. It also includes graphs with bounded degree. Moreover, by the observation of [51], the class of graphs with bounded crossing parameter is not minor-closed. Therefore it generalizes the discussions about \(H\)-minor-free graphs and geometric graphs.

**Theorem 4.36.** If \(f_i\) is a liberal function for all \(i \in V\) where \(|V| = n\), then for fixed \(k\), there is an \(O(q^{O(kn)})\)-time algorithm for computing \(\max_{\sigma \in [q]^V} \sum_{i \in V} f_i\) that achieves a solution of value at least \((k - \varphi - 2)/k\) optimal for graphs with crossing parameter \(\varphi\).
Proof. Using a similar technique of [51], we obtain a planar graph $G' = (V', E')$ by replacing each edge crossing of $G$ by a new vertex. Construct a breadth first search tree $T$ of $G$, rooted at $v \in V$. The level of a vertex is defined as the distance from the vertex to the root $v$ on $T$. Suppose $\varphi < k$. Different from [51], we remove every $k$-th level in $T$ from $G'$ together with its $\varphi - 1$ successive levels. Then $G'$ is decomposed into several subgraphs $H = \{H_1, \ldots, H_t\}$, where each $H_i = (N_i, E_i)$ that $1 \leq i \leq t$ is induced by $k - \varphi$ consecutive levels of vertices of in $T$ of $G'$.

Let $V_i = N_i \cap V$ and $G_i = G[V_i]$, which is the subgraph of $G$ induced by $V_i$. Since the number of crossings per edge is at most $\varphi$ and $\varphi$ consecutive levels of vertices are removed from $G'$ for every $k$ levels, thus after the removal all the subgraphs $G_i$ are disjoint. For each $G_i$ which consists of $k_i \in [k - \varphi, k]$ levels of vertices in $T$, denoted by $V_{i,1}, \ldots, V_{i,k_i}$. Note that only the vertices in $V_{i,1}$ are possible to be adjacent to vertices in $V_{i-1,k_{i-1}}$ and only the vertices in $V_{i,k_i}$ are possible to be adjacent to the vertices $V_{i+1,1}$ in $G$.

It is proved in [51] that each $G_i$ has treewidth $O(k)$. Let

$$A_i = \bigcup_{j=2}^{k_i-1} V_{i,j} \text{ and } B_i = V_i - A_i.$$ 

For each $G_i$ we construct a tree decomposition $T_i$ of treewidth $O(k)$. And we use a dynamic programming on $T_i$ to maximize $\sum_{j \in A_i} f_j$ while ignoring the values of liberal functions defined on the vertices in $B_i$. After the computation, we combine the solutions on each $G_i$ to form the final solution. By similar arguments, there is at least one partition can achieve a solution which is at least $(k - \varphi - 2)/k$ optimal.

We naturally have the following results.

**Corollary 4.37.** If $f_i$ is a liberal function and $f_i \geq 1$ for all $i \in V$ where $|V| = n$, then for fixed $k$, there is an $O(q^{O(k)kn})$-time algorithm for computing $\max_{\sigma \in [q]^V} \prod_{i \in V} f_i$ that achieves a solution of value which is at least

$$\left(\max_{\sigma \in [q]^V} \prod_{i \in V} f_i\right)^{(k-\varphi-2)/k}$$

for graphs with crossing parameter $\varphi$.

**Theorem 4.38.** If $f_i$ is a $\alpha$-balanced liberal function for all $i \in V$ where $|V| = n$, then for fixed $k$, there is an $O(q^{O(k)kn})$-time algorithm for computing $\min_{\sigma \in [q]^V} \sum_{i \in V} f_i$ that achieves a solution of value at most

$$1 + \frac{\varphi + 2 - \alpha}{k}$$

optimal for graphs with crossing parameter $\varphi$.

**Proof.** We use the same technique of Theorem 4.36 to decompose the input graph $G$ into several disjoint subgraphs $G_1, \ldots, G_t$. Each $G_i$ consists of $k_i \in [k - \varphi, k]$ levels of vertices in $T$, denoted by $V_{i,1}, \ldots, V_{i,k_i}$. We also let

$$A_i = \bigcup_{j=2}^{k_i-1} V_{i,j} \text{ and } B_i = V_i - A_i.$$ 

For each $G_i$ we construct a tree decomposition $T_i$ of treewidth $O(k)$. And we use a dynamic programming on $T_i$ to minimize $\sum_{j \in A_i} f_j$ while ignoring the values of liberal functions attached on the vertices in $B_i$. After the computation, we combine the solutions on each $G_i$ to form the final solution. There are at most $\varphi$ levels of vertices of $T$ between $G_i$ and $G_{i+1}$. The values of the liberal functions attached on these levels of vertices and the vertices in $V_{i,k_i}$ and $V_{i+1,1}$ (totally at most $\varphi + 2$ levels of vertices in $T$) will increase by at most $\alpha$ times. Then by a similar argument of Theorem 4.36 the theorem follows. \qed
Corollary 4.39. If $f_i$ is a $\alpha$-balanced liberal function for all $i \in V$ where $|V| = n$, then for fixed $k$, there is an $O(q^{O(k)kn})$-time algorithm for computing $\min_{\sigma \in [q]^V} \prod_{i \in V} f_i$ that achieves a solution of value which is at most

$$\left( \min_{\sigma \in [q]} \prod_{i \in V} f_i \right)^{1 + \frac{k}{(\varphi+2)/(\alpha-1)}}$$

for graphs with crossing parameter $\varphi$.

5 MAX-SAT, MAX-CSP and Graph Cuts

MAX-SAT is a significant problem for studying approximation algorithms which has been extensively investigated in the literature. Given a CNF boolean formula $\phi$ over $n$ variables $X_1, \ldots, X_n$, the objective of MAX-SAT is finding a truth assignment of the variables of $\phi$ such that the number of satisfied clauses is maximum. In its weighted version, each clause $\varphi_j$ is assigned a weight $w_j$, the objective is finding a truth assignment of the variables of $\phi$ such that the sum of the weights of those satisfied clauses is maximum.

MAX-SAT is APX-hard so that there is no PTAS in general unless P = NP. Moreover, although the decision problem 2-SAT can be decided in polynomial time, MAX-2SAT is APX-hard [6]. Feige et al. [41] provide a 0.931-approximation algorithm for MAX-2SAT, improving the approximation ratio of 0.87856 in [49]. Trevisan et al. [98] give a 0.801-approximation algorithm for MAX-3SAT. If the formula is satisfiable, then there is a 7/8-approximation algorithm for MAX-3SAT [64]. Asano [5] gives a 0.770-approximation algorithm for MAX-SAT. Hastad [57] shows that MAX-3SAT is hard to approximate within 7/8 and MAX-2SAT is hard to approximate within $73/74 \approx 0.986$.

Suppose $\phi = \varphi_1 \land \varphi_2 \land \ldots \land \varphi_m$ where $\varphi_j$ is a disjunction of literals for each $1 \leq j \leq m$, then the corresponding factor graph $G = (\mathcal{X}, \mathcal{F}, E)$ of $\phi$ consists of $n$ variable nodes and $m$ function nodes where $\mathcal{X} = \{X_1, \ldots, X_n\}$, $\mathcal{F} = \{F_1, \ldots, F_m\}$. If $X_i$ is one of the inputs of $\varphi_j$, then there is an edge connecting $X_i$ and $F_j$.

In [66], it is shown that if the MAX-SAT problems are expressible by MPSAT, TMAX, TMIN languages, then their optimization have PTAS when the corresponding factor graph of the instance is planar. However, the planarity restriction of factor graph is too strong. A simple example for clarification is a triangle graph $G(V, E)$ where $V = \{v_1, v_2, v_3\}$ and $E = \{(v_1, v_2), (v_2, v_3), (v_1, v_3)\}$. Its corresponding factor graph is the complete bipartite graph $K_{3,3}$, which is known as a forbidden minor of planar graph class. However, $G$ is a planar graph. As we have mentioned, the maximization problem which can be formulated by computing the max-sum of liberal functions admits PTAS on planar graphs. Furthermore, we will show that computing the MAX-SAT can be formulated by computing the max-sum of liberal functions. The following theorem extends the results about maximization of [66] to a much larger set of MAX-SAT problems.

Theorem 5.1. If the corresponding factor graph of a SAT formula $\phi$ can be transformed into any graph class we have discussed above through $\mathcal{X} - \mathcal{F}$ contractions, then computing MAX-SAT of $\phi$ admits a PTAS.

Proof. Without loss of generality, we assume that $\phi$ is a CNF formula. Therefore $\phi$ can be written as

$$\phi = \varphi_1 \land \varphi_2 \land \ldots \land \varphi_m,$$

where $\varphi_j$ is a disjunction of literals for each $1 \leq j \leq m$. 

27
We make an $\mathcal{X} - \mathcal{F}$ contraction of its corresponding factor graph. The graph obtained is denoted by $G$. Let $\mathcal{F}(i) = \{ j : \mathcal{F}_j \text{ is contracted to } \mathcal{X}_i \}$. For each $i \in V(G)$, we set the function $f_i$ as

$$f_i = \begin{cases} |\mathcal{F}(i)|, & \text{if } \bigwedge_{j \in \mathcal{F}(i)} \varphi_j \text{ is satisfied} \\ 0, & \text{if } \bigwedge_{j \in \mathcal{F}(i)} \varphi_j \text{ is not satisfied} \end{cases}$$

In the weighted version, we set the function $f_i$ as

$$f_i = \begin{cases} \sum_{j \in \mathcal{F}(i)} w_j, & \text{if } \bigwedge_{j \in \mathcal{F}(i)} \varphi_j \text{ is satisfied} \\ 0, & \text{if } \bigwedge_{j \in \mathcal{F}(i)} \varphi_j \text{ is not satisfied} \end{cases}$$

Computing the MAX-SAT configuration of $\phi$ is equivalent to computing the max-sum of $f_i$. We can see that $f_i$ is a liberal function for all $i \in V$. Therefore, if $G$ falls into any graph class we have discussed above, we have a PTAS for computing the MAX-SAT of $\phi$. \qed

A constraint satisfaction problem (CSP) is defined as a triple $\langle X, Q, C \rangle$ where $X = \{ X_1, \ldots, X_n \}$ is a set of variables, $Q = \{ Q_1, \ldots, Q_n \}$ is the set of respective domains of values of $X$, $C = \{ C_1, \ldots, C_m \}$ is a set of constraints. The variable $X_i$ can take the values in domain $Q_i$. Each constraint $C_j$ is a function which takes a subset $S_j \subseteq X$ of variables as inputs and returns a boolean value as output representing whether it is satisfied. A CSP asks whether there is a configuration of variables such that all the constraints are satisfied. MAX-CSP generalizes CSP analogous to the way in which MAX-SAT generalizes SAT, whose objective is to find a configuration of the variables such that the number of satisfied constraints are maximum. In its weighted version, each constraint $C_j$ is assigned a positive weight $w_j$. The objective is to find a configuration of the variables such that the sum of the weights of satisfied constraints are maximum. Each CSP can be modelled by a factor graph where each variable node $X_i$ represents the variable $X_i$, each function node $\mathcal{F}_j$ represents the constraint $C_j$. There is an edge between $X_i$ and $\mathcal{F}_j$ if the constraint $C_j$ takes the variable $X_i$ as input. The following result is based on the assumptions that all the domains in $Q$ has bounded size where $q = \max_{1 \leq i \leq n} |Q_i|$ and all the constraints are polynomial-time verifiable.

**Theorem 5.2.** If the corresponding factor graph of a CSP $\langle X, Q, C \rangle$ can be transformed into any graph class we have discussed above through $\mathcal{X} - \mathcal{F}$ contractions, then computing its MAX-CSP admits a PTAS.

**Proof.** We make an $\mathcal{X} - \mathcal{F}$ contraction of its corresponding factor graph. The graph obtained is denoted by $G$. Let $\mathcal{F}(i) = \{ j : \mathcal{F}_j \text{ is contracted to } \mathcal{X}_i \}$. For each $i \in V(G)$, we set the function $f_i$ as

$$f_i = \begin{cases} |\mathcal{F}(i)|, & \text{if } \bigwedge_{j \in \mathcal{F}(i)} C_j \text{ is satisfied} \\ 0, & \text{if } \bigwedge_{j \in \mathcal{F}(i)} C_j \text{ is not satisfied} \end{cases}$$

In the weighted version, we set the function $f_i$ as

$$f_i = \begin{cases} \sum_{j \in \mathcal{F}(i)} w_j, & \text{if } \bigwedge_{j \in \mathcal{F}(i)} C_j \text{ is satisfied} \\ 0, & \text{if } \bigwedge_{j \in \mathcal{F}(i)} C_j \text{ is not satisfied} \end{cases}$$

Computing the configuration of variables that maximize the total weight of satisfied constraints is equivalent to computing the max-sum of $f_i$. We can see that $f_i$ is a liberal function for all $i \in V$. Therefore, if $G$ falls into any graph class we have discussed above, we have a PTAS for computing the MAX-CSP configuration. \qed
Given an undirected graph $G(V, E, w)$ where $w : E \to \mathbb{R}^+$ assign a nonnegative weight $w_{ij}$ to each edge $(i, j) \in E$. The goal of MAX-CUT problem is dividing the vertices of $G$ into two sets $S$ and $\bar{S}$ such that the value $\sum_{(i, j) \in C(S, \bar{S})} w_{ij}$ is maximum where $C(S, \bar{S})$ is the set of cut edges between $S$ and $\bar{S}$. On planar graphs, MAX-CUT is dual to the route inspection problem so that it is polynomial-time solvable [54]. In general case, this problem is APX-hard [86], thus it does not have PTAS unless P = NP. A simple randomized $\frac{1}{2}$-approximation algorithm for MAX-CUT can be achieved by randomly assigning a configuration for each vertex. This algorithm can be derandomized by the method of conditional probabilities to obtain a deterministic $\frac{1}{2}$-approximation algorithm [85, 84]. The best known approximation ratio achieved by polynomial time algorithm for computing MAX-CUT is $\alpha = \frac{2}{\pi} \min_{0 \leq \theta \leq \pi} \frac{\theta}{1 - \cos \theta} \approx 0.878$ discovered by Goemans and Williamson [49] using semidefinite programming and randomized rounding. Moreover, Khot et al. [68] show that this is the best possible approximation ratio for MAX-CUT if the unique game conjecture [67, 69] is true. Without such unproved assumptions, it is shown in [98, 57] that it is NP-hard to approximate MAX-CUT within $\frac{44}{45} \approx 0.941$ optimal. Theorem 5.3 has already been known for $H$-minor-free graphs in [28]. Here we give a simpler proof of PTAS for MAX-CUT on all graph classes we have discussed.

**Theorem 5.3.** Computing MAX-CUT admits PTASs on $H$-minor-free graphs, $d$-dimensional geometric graphs with bounded density and graphs with bounded crossing number per edge.

**Proof.** Computing MAX-CUT problem can be formulated by computing $\max_{\sigma \in [2]^V} \sum_{i \in V} f_i$ by setting $f_i$ as

$$f_i = \frac{1}{2} \sum_{j \in N(i)} |\sigma_j - \sigma_i|$$

where $\sigma_i \in \{0, 1\}$ for all $i \in V$. In the weighted version, we set $f_i$ as

$$f_i = \frac{1}{2} \sum_{j \in N(i)} w_{ij} \cdot |\sigma_j - \sigma_i|.$$

Since $f_i \geq 0$ so that $f_i$ is a liberal function for all $i \in V$, the theorem follows. 

We have similar results for MAX-DICUT and Max-$k$-CUT. Given a directed graph $G(V, E, w)$ where $w : E \to \mathbb{R}^+$ assigns a nonnegative weight $w_{ij}$ to each directed edge $(i, j) \in E$. The goal of MAX-DICUT problem is dividing the vertices of $G$ into two sets $S$ and $\bar{S}$ such that the total weight of the directed cut $\sum_{i \in S, j \in \bar{S}, (i, j) \in E} w_{ij}$ is maximum. Goemans et al. [49] obtain a 0.796-approximation algorithm for the MAX-DICUT problem by semidefinite programming and relaxation. Feige et al. [41] improve this result to 0.859. Matuura et al. [82] give a 0.863-approximation algorithm for the MAX-DICUT problem. This approximation ratio is later improved to 0.866 in [81].

**Theorem 5.4.** Computing MAX-DICUT admits PTASs on $H$-minor-free graphs, $d$-dimensional geometric graphs with bounded density and graphs with bounded crossing number per edge.

**Proof.** Computing MAX-DICUT can be formulated by computing $\max_{\sigma \in [2]^V} \sum_{i \in V} f_i$ by setting $f_i$ as

$$f_i = \begin{cases} \sum_{j \in N^+(i)} |\sigma_i - \sigma_j|, & \text{if } \sigma_i = 1 \\ 0, & \text{if } \sigma_i = 0 \end{cases}$$

In the weighted version, we set $f_i$ as

$$f_i = \begin{cases} \sum_{j \in N^+(i)} w_{ij} \cdot |\sigma_i - \sigma_j|, & \text{if } \sigma_i = 1 \\ 0, & \text{if } \sigma_i = 0 \end{cases}$$
Since $f_i \geq 0$ so that $f_i$ is a liberal function for all $i \in V$, the theorem follows.

The MAX-$k$-CUT problem is a natural generalization of MAX-CUT problem. Given a connected undirected graph $G(V, E, w)$ where $w : E \to \mathbb{R}^+$ assign a nonnegative weight $w_{ij}$ to each edge $(i, j) \in E$, a $k$-cut is a set of edges $E' \subseteq E$ whose removal decomposes the input graph into $k$ disjoint subgraphs. The goal of MAX-$k$-CUT problem is computing such a set of edges $E'$ that $\sum_{e \in E'} w_e$ is maximum. When $k = 2$, the MAX-$k$-CUT problem is precisely the MAX-CUT problem. When $k > \Delta$ where $\Delta$ is the maximum degree of $G$, the optimal solution is precisely the sum of all the weights of the edges. Thus this problem is only interesting when $k \leq \Delta$. Frieze et al. [46] prove that the randomized approximation ratio $\alpha_k$ of MAX $k$-CUT satisfies $\alpha_k \approx \frac{k - 1}{k} + 2k^2 \ln k$ when $k \to \infty$. This implies $\alpha_2 \geq 0.878567$, $\alpha_3 \geq 0.800217$, $\alpha_4 \geq 0.850304$ and so on. Goemans et al. [48] use complex semidefinite programming and an extension of the random hyperplane technique introduced by [49] to obtain an approximation algorithm for MAX-3-DICUT with approximation ratio 0.79373 and MAX-3-CUT with approximation ratio $\frac{7}{12} + \frac{3}{4\pi} \arccos^2(-1/4) \approx 0.83601$. This approximation ratio matches the ratio 0.836008 given by [22]. Andersson et al. [3] show that it is NP-hard to approximate Max 2-Lin-Mod-3 with $\alpha > 17/18 \approx 0.944$ unless P = NP, which implies that MAX-3-CUT and MAX-DICUT also cannot be approximated within 17/18 unless P = NP.

**Theorem 5.5.** Computing MAX-$k$-CUT admits PTASs on $H$-minor-free graphs, $d$-dimensional geometric graphs with bounded density and graphs with bounded crossing number per edge.

**Proof.** Choosing a set of terminals $S = \{s_1, \ldots, s_k\} \subseteq V$, we assign the configuration $\sigma_t$ to $s_t$ where $1 \leq t \leq k$. Consider the graph $G' = G \setminus S$, we set the function $f_i$ attached on each vertex $i \in V(G')$ as

$$f_i = \frac{1}{2} \sum_{(i, j) \in E(G')} w_{ij} \mathbb{1}(\sigma_i \neq \sigma_j) + \sum_{j : j \in S, (i, j) \in E} w_{ij} \mathbb{1}(\sigma_i \neq \sigma_j)$$

where $\mathbb{1}(\sigma_i \neq \sigma_j) = 1$ if $\sigma_i \neq \sigma_j$ and 0 otherwise. Then computing the MAX-$k$-CUT is equivalent to computing $\max_{\sigma \in \{0, 1\}^V(G')} \sum_{i \in V(G')} f_i$. Since $f_i \geq 0$ so that $f_i$ is a liberal function for all $i \in V$, we have a PTAS for fixed $S$. There are at most $P(n, k) = \frac{n!}{(n-k)!}$ possibilities for $S$, which is polynomial if $k$ is fixed. By enumerating all these cases, we have a PTAS for computing the MAX-$k$-CUT.

Furthermore, in a more general setting where the weight of edge $(i, j) \in E$ is given by a function defined on the edge $w_{ij} : [k] \times [k] \to \mathbb{R}^+$ which takes the values of its endpoints as inputs and returns a positive number as the weight, we can still achieve the PTAS for MAX-$k$-CUT.

An interesting problem that can be left as further research is whether our technique can be used to solve graph-cut problems of minimization such as Multiway-Cut and $k$-Cut or non-local graph-cut problem such as sparsest cut.

### 6 Applications

The model defined by liberal functions is very general. It can model a much larger set of problems than that modeled by classic combinatorial optimization problems. We list some typical motivating examples among the numerous problems that can be solved by our technique.
6.1 Wireless network

Consider a wireless network consisting of \( n \) heterogeneous computing nodes. Each computing node has an unique identifier \( i \in \{1, \ldots, n\} \). The diameter of interference range of computing node \( i \) is denoted by \( r_i \). The distance between computing node \( i \) and computing node \( j \) is \( d_{ij} \). The interference correlation of the wireless network can be described by an interference graph \( G(V, E) \) where \( V \) is the set of the computing nodes. If \( d_{ij} < r_i \), then the directed edge \((i, j)\) from \( i \) to \( j \) belongs to the edge set \( E \). By this construction, the graph \( G(V, E) \) is a directed graph which satisfies some topological constraints given by Proposition 1 in [104]. For each computing node \( i \), the set of its in-neighbors is defined as \( N^-(i) = \{ j : (j, i) \in E \} \) and the set of its out-neighbors is defined as \( N^+(i) = \{ j : (i, j) \in E \} \). For undirected graphs, \( N^-(i) = N^+(i) = N(i) \). The generalization of interference graphs from traditional undirected graphs to directed graphs is first proposed in [9]. Note that the each edge in the interference graph only describes the interference correlation between two endpoints of the edge, which is actually independent on the way of constructing the interference graph. Therefore, if we use the realistic SINR model rather than the geometric measure, an interference graph can still be constructed.

The computing nodes transmit data through \( p \) orthogonal channels, denoted by \( C = \{c_1, \ldots, c_p\} \). The transmission rate of channel \( j \) is denoted by \( \theta_j \). Each computing node can choose a subset of the channels to transmit data. Thus \( \sigma_i \) can be encoded as a boolean vector \( \sigma_i = \{\sigma_{i,1}, \ldots, \sigma_{i,p}\} \) where \( \sigma_{i,j} = 1 \) means computing node \( i \) chooses channel \( j \) and \( \sigma_{i,j} = 0 \) otherwise. It satisfies \( \sum_{j \in C} \sigma_{i,j} \leq \delta_i \) where \( \delta_i \) is the number of transmitters of node \( i \). If each computing node has only one transmitter, then \( q = p \) and \( \sum_{j \in C} \sigma_{i,j} = 1 \), which is the simplest case that has been extensively studied in the literature.

Each computing node \( i \in V \) has an utility function \( f_i \), which takes \( \sigma_i \) and \( \sigma_{N^-(i)} \) as inputs and maps the inputs to a nonnegative real number which represents the utility of computing node \( i \). Then computing a channel assignment which maximizes the total utility of the network can be formulated by computing \( \max_{\sigma} \sum_{i \in V} f_i \). When \( \sum_{i \in V} f_i \) is maximized, the total utility of the network reaches its optimal point. In most situations, \( f_i \) is simply defined by the transmission rate of node \( i \). Some utility functions for typical communication protocols are listed as follows.

For FDMA, TDMA and CDMA, a channel is divided equally into frequency bands, time slots or orthogonal codes. The utility function \( f_i \) can be defined as

\[
f_i = \sum_{j \in C} \theta_j \frac{\sigma_{i,j}}{\sigma_{i,j} + \sum_{k \in N^-(i)} \sigma_{k,j}},
\]

where \( \theta_j \) is assumed to be shared equally by the computing nodes using channel \( j \). This function is undefined when \( \sigma_{i,j} \) and all \( \sigma_{k,j} \) where \( k \in N^-(i) \) are all zeros. In this case, we let \( f_i = 0 \).

For (slotted) Aloha protocol, the strategy of user \( i \in V \) is \( \sigma_i = (\sigma_{i,1}, \ldots, \sigma_{i,p}) \), where \( \sigma_{i,j} \) denotes the probability of user \( i \) using channel \( j \) in a time slot. If node \( i \) has only one transmitter and has data to send, then \( \sum_{j \in C} \sigma_{i,j} = 1 \). Node \( i \) can use channel \( j \) if and only if all the nodes \( k \in N^-(i) \) are not using channel \( j \). Therefore the utility function \( f_i \) can be defined as

\[
f_i = \sum_{j \in C} \theta_j \cdot \sigma_{i,j} \prod_{k \in N^-(i)} (1 - \sigma_{k,j}).
\]

The values of \( \sigma_{i,j} \) is discretized so that \( q \) is a finite number.

For CSMA/CA protocol, when a channel \( j \) is sensed busy, user \( i \) makes a random delay of time \( t_{i,j} \) uniformly generated from \([0, \tau_j]\). Node \( i \) can use channel \( j \) if and only if \( t_{k,j} \) is larger than \( t_{i,j} \) for all
where $n_{i,j} = \sum_{k \in N^-(i)} \sigma_{k,j}$ is the number of nodes adjacent to node $i$ trying to access channel $j$.

For all the considered protocols, we can see that $f_i$ is a liberal function for all $i \in V$ since the transmission rates of computing nodes cannot be negative. Note that we give these functions only for clarification, there may be some other modelling methods for these protocols, but $f_i$ should still be a liberal function for all $i \in V$. In practical situations, $f_i$ can be very complex. Each computing node may be built on different hardwares and operating systems. The communication requests may come from different softwares. Different softwares may have distinct requirements for transmission rates. Sometimes, if the transmission rate is higher than a threshold, the marginal utility will become very low. But these properties can still be expressed by liberal functions.

Our results of $d$-dimensional geometric graphs can be directly applied to this problem by setting $d = 2$ or $d = 3$ and obtain a PTAS for maximizing the total throughput of the network. This resolves a long-standing open problem in wireless networks.

### 6.2 Distributed computing

Consider a large-scale distributed database system which may consists of thousands of servers. Many datasets are stored on these servers. A data stream of computation tasks (e.g. queries) keep flowing to the system. It is required that the computation tasks in the data stream are processed efficiently. Here we assume that the computation tasks are some simple operations such as table scans. Different computation tasks can be treated as identical in their consumption of computation resources.

Let $T$ denote the set of datasets stored in the distributed database system and $T(i)$ denote the datasets stored on server $i$. Each dataset $t \in T$ is replicated to several servers mainly for two reasons.

1. Performance: If the workload on dataset $t$ is very large that a single server cannot hold, then all the servers maintaining a copy of dataset $t$ can share the workload of $t$.

2. Reliability: Some servers may crash during the computation. If a dataset $t \in T$ has only one copy, when the server maintaining this copy of $t$ crashes, all the computation tasks running on $t$ will be blocked, which causes low quality of service.

The system can also be modeled by a graph $G(V,E)$. Each server represents a vertex $i \in V$ of $G$. If $T(i) \cap T(j) \neq \emptyset$, then $(i,j) \in E$. The attribute of the edge $(i,j) \in E$ is the set $T(i) \cap T(j)$. If a data set has exactly $d$ copies on $v_1, \ldots, v_d$, then the subgraph of $G$ induced by $v_1, \ldots, v_d$ is a complete graph. The number $d$ is usually a small integer that $d \ll n$. We can easily store the datasets to make $G$ have some structural properties. For example, we can store the datasets to make $G$ a planar graph. Some typical storage plans with regular structures are shown in Figure 4. In the storage plan represented by Figure 4(a) where $G$ is a square tiling, each server maintains 4 datasets and each dataset has 2 copies. In the storage plan represented by Figure 4(b) where $G$ is a triangle tiling, each server maintains 6 datasets and each dataset has 3 copies. Note that triangle tiling may also represent the case that each server maintains 3 datasets and each dataset has 3 copies. Other regular structures include 3-dimensional grid and hexagon tiling in the 2-dimensional plane and so on.
The servers in the cluster may be heterogeneous in their computation resources such as the number of cores, CPU frequency, memory size and I/O bandwidth. We attach a function \( f_i \) to each server \( i \) in the cluster to formulate these properties. Typically, the function \( f_i \) maps the number of computation tasks allocated to server \( i \) and return a positive number proportional to the reciprocal of average response time of the computation tasks. Therefore, \( f_i \) is a liberal function for all \( i \in V \) so that this problem can be formulated by our framework.

The graphs constructed by such storage plans are usually planar or have bounded local treewidth. Therefore, we can decompose the graph into several disjoint subgraphs with bounded treewidth. On such storage graphs, our technique gives a PTAS (usually linear-time approximation schemes) for computing the optimal scheduling.

Another advantage of our technique is that our algorithm is intrinsically a parallel algorithm. We divide the input graph into several disjoint subgraphs and compute the optimal solution by dynamic programming on each of their tree decompositions. The computation on each subgraph can be processed in parallel to speed up the running time. Moreover, the tree decomposition of each subgraph can be further divided into several subtrees and the computation on each subtree can be processed in parallel. The optimal solution on each subgraph can be achieved by combining the solutions on the subtrees of its tree decomposition. Our results on planar graphs and graphs with bounded local treewidth can be directly applied to this problem to compute the optimal scheduling of computation tasks in the data stream.

Suppose we have made a real-time statistics of the data stream such as a histogram of the workloads on each datasets \( t \in \mathcal{T} \). If the workload of \( t \in \mathcal{T} \) is \( h \) and it has \( d \) copies maintained on \( d \) servers, the data stream made by computation tasks on dataset \( t \) should be partitioned into \( d \) parts, one for each server maintaining a copy of \( t \). Let \( p(h, d) \) denote the number of partitioning \( h \) identical computation tasks into \( d \) distinct servers where the number of computation tasks allocated to a server is allowed to be zero. Then it satisfies the recursion

\[
p(h, d) = \sum_{i=0}^{h} p(i, d - 1)
\]

where \( p(x, 1) = p(0, x) = 1 \) for integer \( x \geq 0 \).

In general, the optimal scheduling can be computed by enumerating all the \( \prod_{i=1}^{\vert \mathcal{T} \vert} p(h_i, d) \) cases where \( h_i \) is the workload of \( i \in \mathcal{T} \). By Lemma 2.3 each clique induced by servers maintaining copies of the same dataset is wholly contained in a bag of the tree decomposition. Thus the optimal scheduling can be computed by a dynamic programming on the tree decomposition of each subgraph in \( O(p(h, d)^{O(k^3)} \cdot n) \) time where \( h = \max_{i=1}^{\vert \mathcal{T} \vert} h_i \), \( k \) is the treewidth of the tree decomposition, \( n \) is the number of servers, \( c \) is the number
of datasets stored on each server. The number $\lceil \frac{ck}{d} \rceil$ denotes the upper bound of the number of different datasets $t \in T$ whose induced cliques are wholly contained in the same bag of the tree decomposition. Although it is a linear time algorithm, $p(h, d)$ is a huge number which makes this algorithm impractical. In practical case, we can simply partition the workload $h$ into $m$ identical parts and allocate these $m$ parts to $d$ servers where $m$ is a small integer and is the same for all the $t \in T$. Then the time complexity becomes $O(p(m, d)O(\lceil \frac{ck}{d} \rceil) \cdot n)$ for each subgraph.

However, unlike scheduling in wireless networks, approximation is not always necessary for this problem if we can control the procedure of storing datasets into the servers and then control the topology the storage graph. The controlling method is based on the recursive definition of $k$-trees. The graph class of $k$-trees is defined recursively as follows:

1. The complete graph of $k + 1$ vertices is a $k$-tree.
2. A $k$-tree with $n + 1$ vertices can be constructed from a $k$-tree with $n$ vertices by adding a vertex $v$ and connecting $v$ to exactly $k$ vertices of a clique of the original $k$-tree.

A graph is a partial $k$-tree [94] if it is a subgraph of a $k$-tree, whose treewidth is bounded by $k$ [101]. For instance, if we have a collection of datasets $T$ and each dataset is required to be stored on $d$ different servers, an available storage plan is described as follows.

1. For the first dataset $t_1 \in T$, we replicate it to $d$ servers. The storage graph is a clique of $d$ vertices, denoted by $G_1$.
2. $G_1$ has $d$ different cliques of $d - 1$ vertices as subgraphs, denoted by $K_2, \ldots, K_{d+1}$. For each $t_i \in T$ where $2 \leq i \leq d + 1$, add a server $v_i$ to $G_1$. We store $t_i$ on $v_i$ and replicate it to all the servers represented by vertices in $G_{i-1}$. Then in the storage graph, $v_i$ is linked to all the vertices in $K_i$. The clique induced by $v_i$ and $K_i$ is denoted by $G_i$.
3. For each clique $G_i$ where $2 \leq i \leq d + 1$, we repeat this procedure recursively for all its sub-cliques of $d - 1$ vertices except $K_i$ until all the datasets are stored into the cluster. The obtained storage graph is a $(d - 1)$-tree where the root of its tree decomposition has $d$ child nodes and each of other non-leaf nodes has at most $d - 1$ child nodes. For this storage graph, the optimal real-time scheduling can be computed in $O(p(h, d) \cdot n)$ time.

When a new server is added into the cluster, the new tree decomposition can be constructed in $O(1)$ time if the tree decomposition for the original cluster is given. Therefore the total time complexity of constructing the tree decomposition is $O(n)$ (NOT exponential in $k$).

Our results show that when a storage plan has some structural properties, we can compute an optimal or near-optimal scheduling efficiently. This does not eradicate the possibilities that some other storage plans may have better performances of resource utilization and average response time even they do not have the structural properties which are required for our technique. The graph structure of the system is usually very similar to the crystal structure of materials studied in crystallography and statistical physics. The scheduling is similar to heat transfer in these materials. To this problem, even an experimental study will be very interesting.

6.3 Statistical physics

Spin system is a model which has been extensively studied in statistical physics. For example, the Ising model [61] is a spin system where each vertex has exactly 2 possible states, which is a powerful tool for
studying phase transition in statistical physics. Another important theoretical model is the Potts model [89], a generalization of the Ising model, is used to study interacting spins of a crystalline structure.

There are two long-standing computation problems for these models in statistical physics. One is computing the partition function $Z(\beta)$, which is equivalent to computing the marginal probability in the corresponding graphical model through Jerrum, Valiant and Vazirani’s well-known reduction [63]. The other one is computing the configuration $\sigma$ which maximizes the Gibbs measure. This is equivalent to maximum a posterior estimation in the corresponding graphical model. Many results with provable bounds for computing the partition function has been known where the technique varies from the classic Markov chain Monte Carlo (MCMC) method [62] to recent developed spatial mixing technique [105]. However, few results with provable bounds have been known for computing the ground state of the Gibbs measure.

Formally, a spin system can be described by a graph $G$. Each vertex of $G$ represents a particle and the edges describe the local interactions between particles. In statistical physics, $G$ is usually a 2-dimensional or 3-dimensional lattice graph corresponding to the crystalline structure of materials.

The Edwards-Anderson model is a widely accepted mathematical abstraction of the spin systems. The input graph is a $d$-dimensional lattice graph $L^d$ where each vertex $i$ in the lattice is a Ising spin $\sigma_i \in \{-1, +1\}$. The energy functions [36, 83] is

$$E(\sigma) = \sum_{(i,j) \in E} J_{ij} \sigma_i \sigma_j + B \sum_{i \in V} \sigma_i,$$

where the parameters $J_{ij}$ are exchange couplings and $B \sum_{i \in V} \sigma_i$ is the external magnetic field. The interactions between spins $\sigma_i$ and $\sigma_j$ is ferromagnetic if $J_{ij} > 0$ and antiferromagnetic if $J_{ij} < 0$. The Gibbs measure of a given configuration $\sigma \in [q]^V$ is defined as

$$P(\sigma) = \frac{1}{Z(\beta)} \exp \left( -\beta E(\sigma) \right)$$

where the normalizing constant $Z$ is the partition function that

$$Z(\beta) = \sum_{\sigma \in [q]^V} \exp \left( -\beta E(\sigma) \right).$$

For the ferromagnetic Edwards-Anderson model without external magnetic field that $J_{ij} > 0$ and $B = 0$, the energy function of the Edwards-Anderson model is

$$E(\sigma) = \sum_{(i,j) \in E} J_{ij} \sigma_i \sigma_j = C - 2 \sum_{(i,j) \in E, \sigma_i \neq \sigma_j} J_{ij},$$

where $C = \sum_{(i,j) \in E} J_{ij}$. In this case, the computation task is equivalent to computing the MAX-CUT of the input graph, which is polynomial-time solvable on planar graphs. This problem is NP-hard even on 3-dimensional lattice graphs. Other optimization techniques such as branch and bound methods, belief propagation have also been applied to this problem but no provable bound has been achieved. Our results for computing the MAX-CUT on $d$-dimensional geometric graphs can be directly applied to this problem to achieve a PTAS for the energy minimization. The density of a $d$-dimensional lattice graph is 1. Therefore, the time complexity of the approximation algorithm is $O(q^{O((k+2)^d-1)} \cdot k^{d-1} \cdot n)$. This resolves a long-standing open problem in statistical physics.
6.4 Computer vision

The theoretical models of statistical physics are widely used in computer vision. The analogy between images and statistical mechanics systems are discussed in [47], where the pixel values are viewed as states of atoms and molecules in a lattice-like spin system. The energy function which describes the interactions between particles determines its Gibbs measure. Due to the intrinsic equivalence between Gibbs measures and Markov random fields, the image restoration problem is solved by maximum a posterior estimation, equivalently, energy minimization on the corresponding Markov random field. This method is then applied to other problems of visual perception like image segmentation [96, 14] and stereo vision [75, 97]. Typical lattice graphs for energy minimization are shown in Figure 5.

![Lattice graphs](a) 4-neighbor system  (b) 6-neighbor system  (c) 8-neighbor system

Figure 5: Typical lattice graph for energy minimization

The computation task is finding a label configuration \( \sigma \in [q]^V \) that minimizes the energy function, which usually takes the form

\[
E(\sigma) = \sum_{i \in V} \psi_i(\sigma_i) + \sum_{(i,j) \in E} \psi_{i,j}(\sigma_i, \sigma_j),
\]

where \( \psi_i(\cdot) \) measures the disagreement between the label and observed data at pixel \( i \) and \( \psi_{i,j}(\cdot) \) measures the pairwise smoothness between pixel \( i \) and pixel \( j \). The \( \psi_i(\cdot) \) corresponds to the external magnetic field in statistical physics. The likelihood of the label configuration \( \sigma \) is measured by the probability

\[
P(\sigma) = \frac{1}{Z(\beta)} \exp\left(-\beta E(\sigma)\right)
\]

where \( \beta \) is the inverse temperature and \( Z(\beta) = \sum_{\sigma \in [q]^V} \exp(-\beta E(\sigma)) \) is the normalizing factor, also called partition function in statistical physics. Therefore, when the energy function is minimized, we have the label configuration of maximum likelihood. The typical form of \( \psi_i(\sigma_i) \) [16] usually takes the form

\[
\psi_i(\sigma_i) = (\sigma_i - p_i)^2
\]

where \( p_i \) is the pixel value of point \( i \). The typical forms [89, 15] of \( \psi_{i,j}(\sigma_i, \sigma_j) \) are

\[
\psi(\sigma_i, \sigma_j) = \begin{cases} 
  w_{i,j} \cdot (1 - \delta(\sigma_i, \sigma_j)) & \text{(Generalized Potts model)} \\
  \min(k_{i,j}, |\sigma_i - \sigma_j|) & \text{(Truncated absolute distance)} \\
  \min(k_{i,j}, (\sigma_i - \sigma_j)^2) & \text{(Truncated quadratic distance)}
\end{cases}
\]
where \( w_{i,j} \geq 0 \) is a weight coefficient and \( \delta(x) = 1 \) when \( x = 0 \), and \( \delta(x) = 1 \) otherwise. \( k_{i,j} > 0 \) is a coefficient represents the maximum allowed penalty in the truncated model. When \( q = 2 \), the energy minimization problem can be solved in polynomial time by maximum flow \([50]\) or graph cuts \([16]\). However, in general this problem is still NP-hard. It is easy to see that the energy minimization problem can be included in our model by setting

\[
f_i = \psi_i(\sigma_i) + \frac{1}{2} \sum_{(i,j) \in E} \psi_{i,j}(\sigma_i, \sigma_j).
\]

\( \psi_i(\sigma_i) \) and \( \psi_{i,j}(\sigma_i, \sigma_j) \) are all liberal functions, therefore \( f_i \) is a liberal function for all \( i \in V \). Furthermore, the lattice graphs \( G \) are usually planar (for 4-neighbor system or 6-neighbor system) or have bounded local treewidth (for 8-neighbor system). To our knowledge, no PTAS has been known for this problem.

All the models investigated above can be described by pairwise MRFs. In our model, \( f_i \) is attached on vertices rather than edges, which can be used to formulate higher-order clique functions in MRFs. Recently, high-order MRFs \([95, 73, 74]\) have been used to achieve a better characterization of interactions among random variables and improve the performances of MRF-based algorithms for visual perception. For example, the \( P^n \) Potts model \([73]\) is defined as

\[
\psi_C(x_C) = \begin{cases} 
\gamma_k & \text{if } x_i = p_k \text{ for } \forall i \in C \\
\gamma_{\text{max}} & \text{otherwise.}
\end{cases}
\]

where \( C \) is a clique on which the clique function is defined, \( \gamma_k \) and \( \gamma_{\text{max}} \) are nonnegative constants satisfying \( \gamma_{\text{max}} \geq \gamma_k \).

Typical high-order MRFs are shown in Figure 6. The clique edges are not drawn in the figures. The corresponding factor graph of such MRFs satisfies Hall’s theorem \([55]\) for the function nodes. Thus for any one of their factor graphs, there exists a matching that all the function nodes are matched. Then it can be contracted to a lattice graph of variable nodes through \( \mathcal{X} - \mathcal{F} \) contractions without combining any two of function nodes. Such a high-order clique function can be formulated by a single \( f_i \) attached on a vertex in the clique of the function. This shows our model is more general than the widely used pairwise models.

For the energy minimization problem of pairwise MRFs, whether its PTAS can be achieved by our technique depends on the minimum value of \( \psi_i(\sigma_i) \). The maximum value of pairwise clique function \( \psi(\sigma_i, \sigma_j) \) is bounded by a positive constant depending on the degree of vertices in the lattice graph. If the minimum value of \( \psi_i(\sigma_i) \) is larger than zero, then the liberal functions are balanced. Then our results of planar graphs and graphs with bounded local treewidth can be directly applied to this problem to obtain a PTAS.
For energy minimization problem of high-order MRFs, the existence of PTAS depends on the minimum value of high-order clique functions. If the minimum value is larger than zero, then the liberal functions are usually balanced. For example, in the $P^n$ Potts model, if $\gamma_k > 0$ for all $k$, since $\psi_C \leq \gamma_k$, the liberal functions are $\alpha$-balanced where $\alpha = \frac{\max_k \gamma_k}{\min_k \gamma_k}$. Our results directly give a PTAS for energy minimization.

7 Conclusion

In this paper, we propose a unified theoretical framework to formulate the optimization problems. The local constraints of the optimization problems are encoded by functions attached on the vertices of the input graph. We prove that when the function $f_i$ is a liberal function for all $i \in V$, then there is a PTAS for computing the max-sum of $f_i$ on planar graphs, which implies that the maximum product of $f_i$ can also be well approximated in many cases. We also prove that computing the min-sum of $f_i$ does not admit PTAS even on planar graphs unless P = NP. But if the set of $f_i$ satisfies the balance property, we have a PTAS for computing the min-sum. This implies the min-product of balanced liberal functions can also be approximated well in many cases. These results are further generalized to graphs with bounded local treewidth, H-minor-free graphs, $d$-dimensional geometric graphs with bounded density and graphs with bounded number of crossings per edge. Our results lead to PTASs for MAX-SAT, MAX-CSP, MAX-CUT, MAX-DICUT, MAX-$k$-CUT on these graphs.

Our results have great algorithmic implications on important computation problems in many fields such as communication scheduling in wireless networks, distributed database system management, maximum a posterior inference in graphical models, energy minimization in statistical physics and many important applications in computer vision.

There are also some interesting further research directions. First, we show that there is no PTAS for computing the min-sum of liberal functions in general. But if the liberal functions are balanced, computing the min-sum has a PTAS. However, the property of balance does not give a clear characterization of the approximability of computing the min-sum of liberal functions. Therefore, a clearer characterization can be left as further research. Second, we show that if the corresponding factor graph of a CNF formula $\varphi$ can be transformed into these graphs through $X - F$ contractions, then computing the MAX-SAT of $\varphi$ have a PTAS. But this result is existential rather than constructive. A specified algorithm for $X - F$ contraction is very interesting. Third, Baker’s technique has been applied to non-local optimization problems such as connected dominating set and TSP. It is also very interesting to generalize our algorithm to non-local problems formulated by liberal functions.

Acknowledgement

The author is very grateful to Prof. Anuj Dawar, Prof. Erik Demaine and Prof. Martin Grohe for patiently answering questions about their papers and helpful suggestions. The author also would like to thank Xue-Zhi Hong, Yao-Yu Wang and Li-Ke Xue for downloading papers that cannot be accessed at Microsoft Research Asia.

References

[1] Alon, N., Seymour, P., and Thomas, R. A separator theorem for graphs with an excluded minor and its applications. In Proceedings of the twenty-second annual ACM symposium on Theory
of computing (1990), ACM, pp. 293–299.

[2] Alon, N., Seymour, P., and Thomas, R. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society* 3, 4 (1990), 801–808.

[3] Andersson, G., Engbergens, E., and Hästad, J. A new way of using semidefinite programming with applications to linear equations mod p. *Journal of Algorithms* 39, 2 (2001), 162–204.

[4] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)* 45, 3 (1998), 501–555.

[5] Asano, T. Approximation algorithms for max sat: Yannakakis vs. goemans-williamson. In *Theory of Computing and Systems, 1997., Proceedings of the Fifth Israeli Symposium on* (1997), IEEE, pp. 24–37.

[6] Ausiello, G. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer, 1999.

[7] Baker, B. S. Approximation algorithms for np-complete problems on planar graphs. In *FOCS* (1983), IEEE, pp. 265–273.

[8] Baker, B. S. Approximation algorithms for np-complete problems on planar graphs. *Journal of the ACM (JACM)* 41, 1 (1994), 153–180.

[9] Bilò, V., Fanelli, A., Flammini, M., and Moscardelli, L. Graphical congestion games. *Algorithmica* 61, 2 (2011), 274–297.

[10] Bodlaender, H. L. *Classes of graphs with bounded tree-width*. Department of Computer Science, University of Utrecht, 1986.

[11] Bodlaender, H. L. *Dynamic programming on graphs with bounded treewidth*. Springer, 1988.

[12] Bodlaender, H. L. A linear time algorithm for finding tree-decompositions of small treewidth. In *STOC* (1993), ACM, pp. 226–234.

[13] Bodlaender, H. L., Van Leeuwen, J., Tan, R., and Thilikos, D. M. On interval routing schemes and treewidth. *Information and Computation* 139, 1 (1997), 92–109.

[14] Boykov, Y., and Funka-Lea, G. Graph cuts and efficient nd image segmentation. *International journal of computer vision* 70, 2 (2006), 109–131.

[15] Boykov, Y., Veksler, O., and Zabih, R. Markov random fields with efficient approximations. In *IEEE computer society conference on Computer vision and pattern recognition* (1998), IEEE, pp. 648–655.

[16] Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 11 (2001), 1222–1239.

[17] Breu, H., and Kirkpatrick, D. G. Unit disk graph recognition is np-hard. *Computational Geometry* 9, 1 (1998), 3–24.
[18] CHAN, T. M. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms* 46, 2 (2003), 178–189.

[19] CHEN, Z.-Z. Approximation algorithms for independent sets in map graphs. *Journal of Algorithms* 41, 1 (2001), 20–40.

[20] DAILEY, D. P. Uniqueness of colorability and colorability of planar 4-regular graphs are np-complete. *Discrete Mathematics* 30, 3 (1980), 289–293.

[21] DAWAR, A., GROHE, M., KREUTZER, S., AND SCHWEIKARDT, N. Approximation schemes for first-order definable optimisation problems. In *21st Annual IEEE Symposium on Logic in Computer Science* (2006), IEEE, pp. 411–420.

[22] DE KLERK, E., PASECHNIK, D. V., AND WARNERS, J. P. On approximate graph colouring and max-k-cut algorithms based on the θ-function. *Journal of Combinatorial Optimization* 8, 3 (2004), 267–294.

[23] DEMaine, E. D., Fomin, F. V., HAJIAGHAYI, M., AND THILIKOS, D. M. Fixed-parameter algorithms for (k, r)-center in planar graphs and map graphs. *ACM Transactions on Algorithms (TALG)* 1, 1 (2005), 33–47.

[24] DEMaine, E. D., Fomin, F. V., Hajiaghayi, M. T., AND THILIKOS, D. M. Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. *Journal of the ACM (JACM)* 52, 6 (2005), 866–893.

[25] DEMaine, E. D., Fomin, F. V., Hajiaghayi, M. T., AND Thilikos, D. M. Bidimensional parameters and local treewidth. *SIAM Journal on Discrete Mathematics* 18, 3 (2004), 501–511.

[26] DEMaine, E. D., AND HAJIAGHAYI, M. Equivalence of local treewidth and linear local treewidth and its algorithmic applications. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms* (2004), SIAM, pp. 840–849.

[27] DEMaine, E. D., AND HAJIAGHAYI, M. Bidimensionality: new connections between fpt algorithms and ptass. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms* (2005), SIAM, pp. 590–601.

[28] DEMaine, E. D., HAJIAGHAYI, M., AND KAWARABAYASHI, K.-I. Algorithmic graph minor theory: Decomposition, approximation, and coloring. In *46th Annual IEEE Symposium on Foundations of Computer Science* (2005), IEEE, pp. 637–646.

[29] DEMaine, E. D., HAJIAGHAYI, M., AND KAWARABAYASHI, K.-I. Contraction decomposition in h-minor-free graphs and algorithmic applications. In *Proceedings of the forty-third annual ACM symposium on Theory of computing* (2011), ACM, pp. 441–450.

[30] DEMaine, E. D., HAJIAGHAYI, M., AND MOHAR, B. Approximation algorithms via contraction decomposition. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), Society for Industrial and Applied Mathematics, pp. 278–287.

[31] DEMaine, E. D., HAJIAGHAYI, M., NISHIMURA, N., RAGDE, P., AND Thilikos, D. M. Approximation algorithms for classes of graphs excluding single-crossing graphs as minors. *Journal of Computer and System Sciences* 69, 2 (2004), 166–195.
[32] Demaine, E. D., Hajiaghayi, M., and Thilikos, D. M. Exponential speedup of fixed-parameter algorithms for classes of graphs excluding single-crossing graphs as minors. *Algorithmica* 41, 4 (2005), 245–267.

[33] Demaine, E. D., and Hajiaghayi, M. T. Diameter and treewidth in minor-closed graph families, revisited. *Algorithmica* 40, 3 (2004), 211–215.

[34] DeVos, M., Ding, G., Oporowski, B., Sanders, D. P., Reed, B., Seymour, P., and Vertigan, D. Excluding any graph as a minor allows a low tree-width 2-coloring. *Journal of Combinatorial Theory, Series B* 91, 1 (2004), 25–41.

[35] Dinur, I., and Safra, S. On the hardness of approximating minimum vertex cover. *Annals of Mathematics* (2005), 439–485.

[36] Edwards, S. F., and Anderson, P. W. Theory of spin glasses. *Journal of Physics F: Metal Physics* 5, 5 (1975), 965.

[37] Eppstein, D. Subgraph isomorphism in planar graphs and related problems. In *SODA* (1995), Society for Industrial and Applied Mathematics, pp. 632–640.

[38] Eppstein, D. Diameter and treewidth in minor-closed graph families. *Algorithmica* 27, 3-4 (2000), 275–291.

[39] Erlebach, T., Jansen, K., and Seidel, E. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing* 34, 6 (2005), 1302–1323.

[40] Feige, U. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)* 45, 4 (1998), 634–652.

[41] Feige, U., and Goemans, M. Approximating the value of two power proof systems, with applications to max 2sat and max dicut. In *Third Israel Symposium on the Theory of Computing and Systems* (1995), IEEE, pp. 182–189.

[42] Feige, U., Goldwasser, S., Lovász, L., Safra, S., and Szegedy, M. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM (JACM)* 43, 2 (1996), 268–292.

[43] Fomin, F. V., Lokshtanov, D., Raman, V., and Saurabh, S. Bidimensionality and eptas. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms* (2011), SIAM, pp. 748–759.

[44] Fomin, F. V., Lokshtanov, D., and Saurabh, S. Bidimensionality and geometric graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms* (2012), SIAM, pp. 1563–1575.

[45] Frick, M., and Grohe, M. Deciding first-order properties of locally tree-decomposable structures. *Journal of the ACM (JACM)* 48, 6 (2001), 1184–1206.

[46] Frieze, A., and Jerrum, M. Improved approximation algorithms for max k-cut and max bisection. *Algorithmica* 18, 1 (1997), 67–81.
[47] Geman, S., and Geman, D. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6 (1984), 721–741.

[48] Goemans, M. X., and Williamson, D. Approximation algorithms for max-3-cut and other problems via complex semidefinite programming. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing* (2001), ACM, pp. 443–452.

[49] Goemans, M. X., and Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)* 42, 6 (1995), 1115–1145.

[50] Greig, D., Porteous, B., and Seheult, A. H. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)* (1989), 271–279.

[51] Grigoriev, A., and Bodlaender, H. L. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica* 49, 1 (2007), 1–11.

[52] Grohe, M. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica* 23, 4 (2003), 613–632.

[53] Grohe, M., Ichi Kawarabayashi, K., and Reed, B. A. A simple algorithm for the graph minor decomposition - logic meets structural graph theory. In *SODA* (2013), SIAM, pp. 414–431.

[54] Hadlock, F. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing* 4, 3 (1975), 221–225.

[55] Hall, P. On representatives of subsets. *J. London Math. Soc* 10, 1 (1935), 26–30.

[56] Hastad, J. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science* (1996), IEEE Computer Society, p. 627.

[57] Håstad, J. Some optimal inapproximability results. *Journal of the ACM (JACM)* 48, 4 (2001), 798–859.

[58] Hochbaum, D. S., and Maass, W. Approximation schemes for covering and packing problems in image processing and vlsi. *Journal of the ACM (JACM)* 32, 1 (1985), 130–136.

[59] Hopcroft, J., and Tarjan, R. Efficient planarity testing. *Journal of the ACM (JACM)* 21, 4 (1974), 549–568.

[60] Hunt III, H. B., Marathe, M. V., Radhakrishnan, V., Ravi, S. S., Rosenkrantz, D. J., and Stearns, R. E. Nc-approximation schemes for np-and pspace-hard problems for geometric graphs. *Journal of Algorithms* 26, 2 (1998), 238–274.

[61] Ising, E. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik A Hadrons and Nuclei* 31, 1 (1925), 253–258.

[62] Jerrum, M., and Sinclair, A. Polynomial-time approximation algorithms for the ising model. *SIAM Journal on computing* 22, 5 (1993), 1087–1116.

[63] Jerrum, M. R., Valiant, L. G., and Vazirani, V. V. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science* 43 (1986), 169–188.
[64] Karloff, H., and Zwick, U. A 7/8-approximation algorithm for max 3sat? In Proceedings of 38th Annual Symposium on Foundations of Computer Science (1997), IEEE, pp. 406–415.

[65] Kearns, M., Littman, M. L., and Singh, S. Graphical models for game theory. In Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence (2001), Morgan Kaufmann Publishers Inc., pp. 253–260.

[66] Khanna, S., and Motwani, R. Towards a syntactic characterization of ptas. In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing (1996), ACM, pp. 329–337.

[67] Khot, S. On the power of unique 2-prover 1-round games. In Proceedings of the thirty-fourth annual ACM symposium on Theory of computing (2002), ACM, pp. 767–775.

[68] Khot, S., Kindler, G., Mossel, E., and O’Donnell, R. Optimal inapproximability results for max-cut and other 2-variable csps? SIAM Journal on Computing 37, 1 (2007), 319–357.

[69] Khot, S., and Vishnoi, N. K. On the unique games conjecture. In FOCS (2005), vol. 5, p. 3.

[70] Kindermann, R., Snell, J. L., et al. Markov random fields and their applications, vol. 1. American Mathematical Society Providence, RI, 1980.

[71] Klein, P. N. A linear-time approximation scheme for tsp in undirected planar graphs with edge-weights. SIAM Journal on Computing 37, 6 (2008), 1926–1952.

[72] Koebe, P. Kontaktprobleme der konformen Abbildung. Hirzel, 1936.

[73] Kohli, P., Kumar, M. P., and Torr, P. H. P3 & beyond: Solving energies with higher order cliques. In IEEE Conference on Computer Vision and Pattern Recognition (2007), IEEE, pp. 1–8.

[74] Kohli, P., Torr, P. H., et al. Robust higher order potentials for enforcing label consistency. International Journal of Computer Vision 82, 3 (2009), 302–324.

[75] Kolmogorov, V., and Zabih, R. Multi-camera scene reconstruction via graph cuts. In ECCV 2002. Springer, 2002, pp. 82–96.

[76] Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47, 2 (2001), 498–519.

[77] Lipton, R. J., and Tarjan, R. E. A separator theorem for planar graphs. SIAM Journal on Applied Mathematics 36, 2 (1979), 177–189.

[78] Lipton, R. J., and Tarjan, R. E. Applications of a planar separator theorem. SIAM journal on computing 9, 3 (1980), 615–627.

[79] Lund, C., and Yannakakis, M. On the hardness of approximating minimization problems. Journal of the ACM (JACM) 41, 5 (1994), 960–981.

[80] Marx, D. Efficient approximation schemes for geometric problems? In ESA 2005. Springer, 2005, pp. 448–459.

[81] Matuura, S., and Matsui, T. 0.935-approximation randomized algorithm for max 2sat and its derandomization.
[82] Matuura, S., and Matsui, T. 0.863 approximation algorithm for max dicut. *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques* Lecture Notes in Computer Science 2129 (2001), 138–146.

[83] Mezard, M., and Montanari, A. *Information, physics, and computation*. Oxford University Press, 2009.

[84] Mitzenmacher, M., and Upfal, E. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[85] Motwani, R., and Raghavan, P. Randomized algorithms.

[86] Papadimitriou, C. H., and Yannakakis, M. Optimization, approximation, and complexity classes. *Journal of computer and system sciences* 43, 3 (1991), 425–440.

[87] Paz, A., and Moran, S. Non deterministic polynomial optimization problems and their approximations. *Theoretical Computer Science* 15, 3 (1981), 251–277.

[88] Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[89] Potts, R. B. Some generalized order-disorder transformations. In *Mathematical Proceedings of the Cambridge Philosophical Society* (1952), vol. 48, Cambridge Univ Press, pp. 106–109.

[90] Robertson, N., and Seymour, P. D. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B* 36, 1 (1984), 49–64.

[91] Robertson, N., and Seymour, P. D. Graph minors. xiii. the disjoint paths problem. *Journal of combinatorial theory, Series B* 63, 1 (1995), 65–110.

[92] Robertson, N., and Seymour, P. D. Graph minors. xvi. excluding a non-planar graph. *Journal of Combinatorial Theory, Series B* 89, 1 (2003), 43–76.

[93] Robertson, N., and Seymour, P. D. Graph minors. xx. wagner’s conjecture. *Journal of Combinatorial Theory, Series B* 92, 2 (2004), 325–357.

[94] Rose, D. J. On simple characterizations of k-trees. *Discrete Mathematics* 7, 3 (1974), 317–322.

[95] Roth, S., and Black, M. J. Fields of experts: A framework for learning image priors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2005), vol. 2, IEEE, pp. 860–867.

[96] Rother, C., Kolmogorov, V., and Blake, A. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 309–314.

[97] Sun, J., Zheng, N.-N., and Shum, H.-Y. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 7 (2003), 787–800.

[98] Trevisan, L., Sorkin, G. B., Sudan, M., and Williamson, D. P. Gadgets, approximation, and linear programming. *SIAM Journal on Computing* 29, 6 (2000), 2074–2097.
[99] Valiant, L. Holographic algorithms. In Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science (2004), IEEE, pp. 306–315.

[100] Van Leeuwen, E. J. Approximation algorithms for unit disk graphs. In Graph-theoretic concepts in computer science (2005), Springer, pp. 351–361.

[101] Van Leeuwen, J. Graph algorithms. In Handbook of theoretical computer science (vol. A) (1991), MIT Press, pp. 525–631.

[102] Wagner, K. Über eine eigenschaft der ebenen komplexe. Mathematische Annalen 114, 1 (1937), 570–590.

[103] Wainwright, M. J., and Jordan, M. I. Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning 1, 1-2 (2008), 1–305.

[104] Wang, Y.-K., Yin, Y., and Zhong, S. Belief propagation for spatial spectrum access games. In MobiHoc (2014), ACM, pp. 225–234.

[105] Weitz, D. Counting independent sets up to the tree threshold. In Proceedings of the thirty-eighth annual ACM symposium on Theory of computing (2006), ACM, pp. 140–149.

[106] Yedidia, J. S., Freeman, W. T., and Weiss, Y. Understanding belief propagation and its generalizations. Exploring artificial intelligence in the new millennium 8 (2003), 236–239.

[107] Yedidia, J. S., Freeman, W. T., Weiss, Y., et al. Generalized belief propagation. In NIPS (2000), vol. 13, pp. 689–695.