Fine-Grained Completeness for Optimization in P

Karl Bringmann
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Alejandro Cassis
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Nick Fischer
Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Marvin Künnemann
Institute for Theoretical Studies, ETH Zürich, Switzerland

Abstract

We initiate the study of fine-grained completeness theorems for exact and approximate optimization in the polynomial-time regime.

Inspired by the first completeness results for decision problems in P (Gao, Impagliazzo, Kolokolova, Williams, TALG 2019) as well as the classic class MaxSNP and MaxSNP-completeness for NP optimization problems (Papadimitriou, Yannakakis, JCSS 1991), we define polynomial-time analogues MaxSP and MinSP, which contain a number of natural optimization problems in P, including Maximum Inner Product, general forms of nearest neighbor search and optimization variants of the k-XOR problem. Specifically, we define MaxSP as the class of problems definable as $\max_{x_1,\dots,x_k} \#\{(y_1,\dots,y_\ell) : \phi(x_1,\dots,x_k,y_1,\dots,y_\ell)\}$, where ϕ is a quantifier-free first-order property over a given relational structure (with MinSP defined analogously). On m-sized structures, we can solve each such problem in time $O(m^{k+\ell-1})$. Our results are:

- We determine (a sparse variant of) the Maximum/Minimum Inner Product problem as complete under *deterministic* fine-grained reductions: A strongly subquadratic algorithm for Maximum/Minimum Inner Product would beat the baseline running time of $O(m^{k+\ell-1})$ for *all* problems in MaxSP/MinSP by a polynomial factor.
- This completeness transfers to approximation: Maximum/Minimum Inner Product is also complete in the sense that a strongly subquadratic *c-approximation* would give a $(c+\varepsilon)$-*approximation* for all MaxSP/MinSP problems in time $O(m^{k+\ell-1-\delta})$, where $\varepsilon > 0$ can be chosen arbitrarily small. Combining our completeness with (Chen, Williams, SODA 2019), we obtain the perhaps surprising consequence that refuting the OV Hypothesis is *equivalent* to giving a $O(1)$-approximation for all MinSP problems in faster-than-$O(m^{k+\ell-1})$ time.
- By fine-tuning our reductions, we obtain mild algorithmic improvements for solving and approximating all problems in MaxSP and MinSP, using the fastest known algorithms for Maximum/Minimum Inner Product.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Fine-grained Complexity & Algorithm Design, Completeness, Hardness of Approximation in P, Dimensionality Reductions

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.9

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2107.01721

Funding *Karl Bringmann, Alejandro Cassis, Nick Fischer:* This work is part of the project TIPEA that has received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 850979). *Marvin Künnemann:* Research supported by Dr. Max Rössler, by the Walter Haefner Foundation, and by the ETH Zürich Foundation. Part of this research was performed while the author was employed at MPI Informatics.

1 Introduction

For decades, increasingly strong hardness of approximation techniques have been developed to pinpoint the best approximation guarantees achievable in polynomial time. Among the early successes of the field, we find the MaxSNP completeness theorems by Papadimitriou and Yannakakis [24], giving the first strong evidence against PTASes for Max-SAT and related problems. Such completeness theorems constitute valuable tools in complexity theory: Generally speaking, proving a problem A to be complete for a class $\mathcal{C}$ shows that A is *the* representing problem for $\mathcal{C}$. The precise notion of completeness is typically chosen such that a certain algorithm for A would yield unexpected algorithms for the whole class $\mathcal{C}$ – thus establishing that A is unlikely to admit such an algorithm. However, a completeness result may also open up algorithmic uses. Namely, since any problem in $\mathcal{C}$ can be reduced to its complete problem A, we may find (possibly mildly) improved algorithms for all problems in $\mathcal{C}$ by making algorithmic progress on the single problem A.

Given this usefulness, it may be surprising that there are currently no completeness results for studying optimization barriers *within the polynomial-time regime*, e.g., for approximability in strongly subquadratic time (in fact, even for studying decision problems, completeness results are an exception rather than the norm, see [29] for a recent survey of the field). Thus, this work sets out to initiate the quest for completeness results for optimization in P, which corresponds to studying the (in-)approximability of problems on large data sets.

Previous Completeness Results in P

The essentially only known completeness result in fine-grained complexity theory in P is a recent result by Gao, Impagliazzo, Kolokolova, and Williams [18]: The orthogonal vectors problem (OV)[1] is established as complete problem for the class of model-checking first-order properties[2] under fine-grained reductions[3]. From this completeness, they derive in particular:

- **Hardness:** If there are $\gamma, \delta > 0$ such that OV with moderate dimension $d = n^\gamma$ can be solved in time $O(n^{2-\delta})$, then there is some $\delta' > 0$ such that all $(k+1)$-quantifier first-order properties can be model-checked in time $O(m^{k-\delta'})$ for $k \geq 2$. The negation of this statement's premise is known as the moderate-dimensional OV Hypothesis; the consequence would be very surprising, as model-checking first-order properties is a very general class of problems for which no $O(m^{k-\delta})$-time algorithm is known. This result can be seen as support for the moderate-dimensional OV Hypothesis.
- **Algorithms:** Using a stronger notion than fine-grained reductions, Gao et al. also prove that mildly subquadratic algorithms for OV have algorithmic consequences for model-checking first-order properties. Specifically, by combining their reductions with the fastest known algorithm for OV [4, 12], they obtain an $m^k/2^{\Omega(\sqrt{\log m})}$-time algorithm for model-checking any $(k+1)$-quantifier first-order property.

[1] Given two sets of n vectors in $\{0,1\}^d$, determine whether there exists a pair of vectors, one of each set, that are orthogonal.

[2] Let ϕ be a first-order property (in prenex normal form) over a relational structure of size m. Given the structure, determine whether ϕ holds. See Section 2 for details.

[3] For a formal definition of fine-grained reductions, see [11, 18]. For this paper, the reader may think of the following slightly simpler notion: A fine-grained reduction from a problem P_1 with presumed time complexity T_1 to a problem P_2 with presumed time complexity T_2 is an algorithm A for P_1 that has oracle access to P_2 and whenever we use an $O(T_2(n)^{1-\delta})$ algorithm for the calls to the P_2-oracle (for some $\delta > 0$), there is a $\delta' > 0$ such that A runs in time $O(T_1(n)^{1-\delta'})$.

No comparable fine-grained completeness results are known for polynomial-time optimization problems, raising the question: Can we give completeness theorems also for a general class of optimization problems in P, both for exact and approximate computation?

Hardness of Approximation in P

Studying the fine-grained approximability of polynomial-time optimization problems (hardness of approximation in P), is a recent and influential trend: After a breakthrough result by Abboud, Rubinstein, and Williams [3] establishing the *Distributed PCP in P* framework, a number of works gave strong conditional lower bounds, including results for nearest neighbor search [28] or a tight characterization of the approximability of maximum inner product [13, 15]. Further results include work on approximating graph problems [25, 6, 10, 22], the Fréchet distance [7], LCS [1, 2], monochromatic inner product [23], earth mover distance [26], as well as equivalences for fine-grained approximation in P [15, 14, 10]. Related work studies the inapproximability of parameterized problems, ruling out certain approximation guarantees within running time $f(k)n^{g(k)}$ under parameter k (such as FPT time $f(k)\operatorname{poly}(n)$, or $n^{o(k)}$), see [17] for a recent survey.[4]

An Optimization Class: Polynomial-Time Analogues of MaxSNP

We define a natural and interesting class of polynomial-time optimization problems, inspired by the approach of Gao et al. [18] as well as the classic class MaxSNP introduced by Papadimitriou and Yannakakis [24] to study the approximability of NP optimization problems.

The definition of MaxSNP is motivated by Fagin's theorem (see, e.g., [20, 19]), which characterizes NP as the family of problems expressible as $\exists S\, \forall \bar{y}\, \exists \bar{z}\, \phi(\bar{y}, \bar{z}, G, S)$ where G is a given relational structure, $\exists S$ ranges over a relational structure S and $\forall \bar{y}\, \exists \bar{z}\, \phi(\bar{y}, \bar{z}, G, S)$ is a $\forall^*\exists^*$-quantified first-order property. A subclass of this is SNP, which consists of those problems expressible without the $\exists \bar{z}$-part. Its natural optimization variant is MaxSNP, defined as the set of problems expressible as $\max_S \#\{\bar{y} : \phi(\bar{y}, G, S)\}$. Notably, this class of problems contains central optimization problems (Max-3-SAT, Max-Cut, etc.), all of which admit a constant-factor approximation in polynomial time. Using a notion of MaxSNP-completeness, Papadimitriou and Yannakakis identified several problems (including Max-3-SAT and Max-Cut) as hardest-to-approximate in this class, giving a justification for the lack of a PTAS for these problems.[5]

To study the same type of questions in the polynomial-time regime, the perhaps most natural approach is to restrict the syntax defining MaxSNP problems such that it solely contains polynomial-time problems. Specifically, we replace $\max_S$ by a maximization over a bounded number of k variables $x_1, \dots, x_k$ and restrict the counting operator to tuples $\bar{y} = (y_1, \dots, y_\ell)$ of bounded length ℓ. The resulting formula $\max_{x_1,\dots,x_k} \#\{(y_1, \dots, y_\ell) : \phi(x_1, \dots, x_k, y_1, \dots, y_\ell)\}$ can be easily seen (see the full version of the paper [8, Appendix A]) to be solvable in time $O(m^{k+\ell-1})$, where m denotes the problem size. We define $\text{MaxSP}_{k,\ell}$ to denote the class of these optimization problems and let $\text{MaxSP} = \bigcup_{k\geq 2,\ell\geq 1} \text{MaxSP}_{k,\ell}$. Note that here, "SP" stands for "strict P" in analogy to the name "strict NP" of SNP. We refer to Section 2 for more details.

[4] Note that these parameterized inapproximability results do not necessarily apply to the case of a fixed parameter k, which would correspond to our setting. See [22] for an interesting exception.

[5] A stronger justification was later given by the PCP theorem, establishing inapproximability even under P $\neq$ NP. In general, these two approaches (approximation-preserving completeness theorems as well as proving inapproximability under established assumptions on exact computation) can result in incomparable hardness of approximation results.

We obtain an analogous minimization class MinSP by replacing max by min everywhere. These classes include interesting problems:

- Vector-definable problems: Let $\Sigma = \{0, \dots, c\}$ be a fixed alphabet and $f : \Sigma^k \to \{0, 1\}$ be an arbitrary Boolean function. Then we can express the following problem: Given sets $X_1, \dots, X_k$ in Σ^d of vectors, maximize (or minimize) $\sum_{i=1}^{d} f(x_1[i], \dots, x_k[i])$ over all $x_1 \in X_1, \dots, x_k \in X_k$. Each such problem is definable in $\text{MaxSP}_{k,1}$/$\text{MinSP}_{k,1}$, e.g.:
  - Maximum Inner Product (MaxIP): Given sets $X_1, X_2 \subseteq \{0, 1\}^d$, maximize the inner product $x_1 \cdot x_2$ over $x_1 \in X_1, x_2 \in X_2$. To see that this problem is in $\text{MaxSP}_{2,1}$, consider the formula $\max_{x_1 \in X_1, x_2 \in X_2} \#\{y \in Y : E(x_1, y) \wedge E(x_2, y)\}$, where $E(x, y)$ indicates that the y-th coordinate of x is equal to 1.
  - Consider minimization with $k = 2$ and view $f : \Sigma^2 \to \{0, 1\}$ as classifying pairs of characters as *similar* (0) or *dissimilar* (1). This expresses the following problem that generalizes the nearest-neighbor problem over the Hamming metric: Given a set of length-d strings over Σ, determine the most similar pair of strings by minimizing the number of dissimilar characters.
  - View Σ as the finite field $\mathbb{F}_q$ and let $f(z_1, \dots, z_k) = 1$ iff $\sum_{i=1}^{k} z_i \equiv 0 \pmod q$. This gives optimization variants of the k-XOR problem [21, 16], generalized to arbitrary finite fields.
- Beyond vector-definable problems, in $\text{MaxSP}_{2,\ell-2}$ we can express the graph problem of computing, over all edges e, the maximum number of length-ℓ circuits containing e:

$$\max_{x_1, x_2} \#\{(y_1, \dots, y_{\ell-2}) : E(x_1, x_2) \wedge E(x_2, y_1) \wedge \dots \wedge E(y_{\ell-3}, y_{\ell-2}) \wedge E(y_{\ell-2}, x_1)\}.$$

  In fact, MaxSP also contains generalizations of this problem to other pattern graphs than length-ℓ circuits (e.g., length-ℓ cycles or ℓ-cliques), even arbitrary fixed patterns in *hypergraphs*.

We let m denote the size of the relational structure, that is, the number of tuples in an explicit representation of all relations. For vector-definable examples, the input can be represented as a relational structure of size $m = O(nd \log |\Sigma|)$, which is the natural input size. Note, however, that the relational structure also allows us to succinctly encode *sparse* vectors in very large dimension (such as $d = \Theta(n)$), which is why we often refer to MaxSP and MinSP as describing a *sparse setting*. It is easy to see that each MaxSP or MinSP formula ψ can be solved in time $O(m^{k+\ell-1})$ (see [8, Appendix A] in the full version); note that for a fixed ψ, k and ℓ always denote the number of maximization/minimization and counting variables, respectively. Can we obtain completeness results with respect to improvements over this baseline running time?

(Sparse) Maximum Inner Product

Our results prove the Maximum Inner Product problem (MaxIP) as representative for the class MaxSP. We will formally introduce two important variants of this problem.

▶ **Problem 1** (MaxIP). *Given two sets of n vectors $X_1, X_2 \subseteq \{0, 1\}^d$, the task is to compute the maximum inner product $\langle x_1, x_2 \rangle = \sum_j x_1[j] \cdot x_2[j]$ for $x_1 \in X_1, x_2 \in X_2$.*

When $d = n^\gamma$ for some (small) $\gamma > 0$, we speak of the *moderate-dimensional* MaxIP problem. In this paper, we also use MaxIP in another context, depending on the input format. To make the distinction explicit, let us formally introduce the *Sparse* Maximum Inner Product problem (Sparse MaxIP):

▶ **Problem 2** (Sparse MaxIP). *Given two sets of n vectors $X_1, X_2 \subseteq \{0, 1\}^d$, sparsely represented as a list of pairs (x_i, j) which represent the one-coordinates $x_i[j] = 1$, the task is to compute the maximum inner product $\langle x_1, x_2 \rangle$ for $x_1 \in X_1, x_2 \in X_2$.*

For moderate-dimensional MaxIP we measure the complexity in n and for Sparse MaxIP we measure the complexity in m, the total number of one-coordinates. We note that Sparse MaxIP is also special in our setting as this problem can be seen as a member of $\text{MaxSP}_{2,1}$. Indeed, Sparse MaxIP is the same problem as maximizing the formula

$$\psi = \max_{x_1 \in X_1, x_2 \in X_2} \#\{y \in [d] : E(x_1, y) \wedge E(x_2, y)\},$$

where $E(x_i, y)$ indicates that the y-th coordinate of x_i is equal to 1. We also define the (Sparse) Minimum Inner Product problems (MinIP, Sparse MinIP) as the analogous problems with the task to minimize $\langle x_1, x_2 \rangle$.

1.1 Our Results

Our first main result is a completeness theorem for exact optimization, establishing Maximum Inner Product as complete for MaxSP (and Minimum Inner Product for MinSP).

▶ **Theorem 3** (Sparse MaxIP is MaxSP-complete). *Sparse MaxIP is complete for the class MaxSP under fine-grained reductions: If there is some $\delta > 0$ such that Sparse MaxIP can be solved in time $O(m^{2-\delta})$, then for every $\text{MaxSP}_{k,\ell}$ formula ψ, there is some $\delta' > 0$ such that ψ can be solved in time $O(m^{k+\ell-1-\delta'})$.*

The analogous statement holds for minimization, if we replace Sparse MaxIP and MaxSP by Sparse MinIP and MinSP, respectively.

Turning to the approximability of MaxSP and MinSP, we show how to obtain a fine-grained completeness that even preserves approximation factors (up to an arbitrarily small blow-up). Here and throughout the paper, we say that an algorithm gives a c-approximation for a maximization problem if it outputs a value in the interval $[c^{-1} \cdot \text{OPT}, \text{OPT}]$, where OPT is the optimal value. For minimization, the algorithm computes a value in the interval $[\text{OPT}, c \cdot \text{OPT}]$.

▶ **Theorem 4** (Sparse MaxIP is MaxSP-complete, (almost) approximation preserving). *Let $c \geq 1$ and $\varepsilon > 0$. If there is some $\delta > 0$ such that Sparse MaxIP can be c-approximated in time $O(m^{2-\delta})$, then for every $\text{MaxSP}_{k,\ell}$ formula ψ, there is some $\delta' > 0$ such that ψ can be $(c+\varepsilon)$-approximated in time $O(m^{k+\ell-1-\delta'})$.*

The analogous statement for minimization holds for Sparse MinIP and MinSP.

As a key technical step to obtain Theorems 3 and 4, we prove a *universe reduction* for MaxSP/MinSP formulas (detailed in Sections 3 and 4.3). Along the way, this universe reduction establishes the following fine-grained equivalence between the sparse and moderate-dimensional settings of MaxIP/MinIP.

▶ **Theorem 5** (Equivalence between MaxIP and Sparse MaxIP).

- *There are some $\gamma, \delta > 0$ such that MaxIP with dimension $d = n^\gamma$ can be solved in time $O(n^{2-\delta})$ if and only if there is some $\delta' > 0$ such that Sparse MaxIP can be solved in time $O(m^{2-\delta'})$.*
- *Let $c > 1$ and $\varepsilon > 0$. If there are some $\gamma, \delta > 0$ such that MaxIP with dimension $d = n^\gamma$ can be c-approximated in time $O(n^{2-\delta})$ then there is some $\delta' > 0$ such that Sparse MaxIP can be $(c+\varepsilon)$-approximated in time $O(m^{2-\delta'})$. Conversely, if there is some $\delta > 0$ such that Sparse MaxIP can be c-approximated in time $O(m^{2-\delta})$ then there are some $\gamma, \delta' > 0$ such that MaxIP with dimension $d = n^\gamma$ can be c-approximated in time $O(n^{2-\delta'})$.*

The analogous statements for minimization hold for MinIP.

We prove Theorems 3, 4 and 5 in Section 4.1.

Consequences for Hardness of Approximation

As a consequence of the above completeness results and dimension reduction, we obtain the following statements.

- Since Maximum Inner Product and Minimum Inner Product are subquadratic equivalent in moderate dimensions [15, Theorem 1.6], we obtain from Theorems 3 and 5 that a strongly subquadratic algorithm solving moderate-dimensional Maximum Inner Product *exactly* would give a polynomial-factor improvement over the $O(m^{k+\ell-1})$ running time for all MaxSP *and* MinSP formulas. This adds an additional surprising consequence of fast Maximum Inner Product algorithms, besides refuting the Orthogonal Vectors Hypothesis.
- There is a $O(1)$-approximation beating the quadratic baseline for moderate-dimensional Maximum Inner Product *if and only if* there is a $O(1)$-approximation beating the $O(m^{k+\ell-1})$ time baseline for all MaxSP formulas. To obtain this result combine the fine-grained equivalence of $O(1)$-approximation of moderate-dimensional MaxIP and Sparse MaxIP (Theorem 5) with the completeness of Sparse MaxIP (Theorem 4). This adds an additional consequence of fast Maximum Inner Product approximation, besides refuting SETH [3, 13].
- In the minimization world, we obtain a tight connection between approximating MinSP formulas and OV: The (moderate-dimensional) OV hypothesis is *equivalent* to the non-existence of a $O(1)$-approximation for all MinSP formulas in time $O(m^{k+\ell-1})$. To obtain this result, combine the equivalence of moderate-dimensional OV Hypothesis and non-existence of a $O(1)$-approximation for moderate-dimensional MinIP [15, Theorem 1.5] with the equivalence of $O(1)$-approximation algorithms for moderate-dimensional MinIP and MinSP (Theorem 4 and Theorem 5). Interestingly, this can be seen as additional support for the Orthogonal Vectors Hypothesis.

Algorithms: Lower-Order Improvements

Since Maximum Inner Product has received significant interest for improved algorithms (see particularly [13, 15]), we turn to the question whether our completeness result also yields lower-order algorithmic improvements for all problems in the class. Indeed, by combining the best known Maximum/Minimum Inner Product algorithms with our reductions, we obtain the following general results for MaxSP and MinSP. We give the proofs for both theorems in Section 4.1.

▶ **Theorem 6** (Lower-Order Improvement for Exact MaxSP and MinSP)**.** *We can exactly optimize any $\mathit{MaxSP}_{k,\ell}$ and $\mathit{MinSP}_{k,\ell}$ formula in randomized time $m^{k+\ell-1}/\log^{\Omega(1)} m$.*

Interestingly, for constant-factor approximations, a complete shave of logarithmic factors is possible.

▶ **Theorem 7** (Lower-Order Improvement for Approximate MaxSP and MinSP)**.** *For every constant $c > 1$, we can c-approximate every $\mathit{MaxSP}_{k,\ell}$ and $\mathit{MinSP}_{k,\ell}$ formula in time $m^{k+\ell-1}/2^{\Omega(\sqrt{\log m})}$. For $\mathit{MaxSP}_{k,\ell}$ the algorithm is deterministic; for $\mathit{MinSP}_{k,\ell}$ it uses randomization.*

2 Preliminaries

For an integer $k \geq 1$, we set $[k] = \{1, \ldots, k\}$. Moreover, we write $\widetilde{O}(T) = T \log^{O(1)} T$.

First-Order Model-Checking

A *relational structure* $(X, R_1, \ldots, R_r)$ consists of n objects X and relations $R_j \subseteq X^{a_j}$ (of arbitrary arities a_j) between these objects. A *first-order formula* is a quantified formula of the form

$$\psi = (Q_1 x_1) \ \ldots \ (Q_k x_k)\, \phi(x_1, \ldots, x_k),$$

where $Q_i \in \{\exists, \forall\}$ and ϕ is a Boolean formula over the predicates $R_j(x_{i_1}, \ldots, x_{i_{a_j}})$. Given a relational structure, the *model-checking problem* (or *query evaluation problem*) is to check whether ψ holds on the given structure, that is, for $x_1, \ldots, x_k$ ranging over X and by instantiating the predicates $R_j(x_{i_1}, \ldots, x_{i_{a_j}})$ in ϕ according to the structure, ψ is valid.

Following previous work in this line of research [18, 9], we assume that the input is represented sparsely – that is, we assume that the relational structure is written down as an exhaustive enumeration of all records in all relations; let m denote the total number of such entries. This convention is reasonable as this data format is common in the context of database theory and also for the representation of graphs (where it is called the *adjacency list* representation). By ignoring objects not occurring in any relation, we may always assume that $n \leq O(m)$.

It is often convenient to assume that each variable x_i ranges over a separate set X_i. We can make this assumption without loss generality, by introducing some additional unary predicates.

MaxSP$_{k,\ell}$ and MinSP$_{k,\ell}$

In analogy to first-order properties with quantifier structure $\exists^k \forall^\ell$ (with maximization instead of $\exists$ and counting instead of $\forall$), we now define a class of optimization problems: Let MaxSP$_{k,\ell}$ be the class containing all formulas of the form

$$\psi = \max_{x_1,\ldots,x_k} \ \#_{y_1,\ldots,y_\ell} \ \phi(x_1, \ldots, x_k, y_1, \ldots, y_\ell), \tag{1}$$

where, as before, ϕ is a Boolean formula over some predicates of arbitrary arities. We similarly define MinSP$_{k,\ell}$ with "min" in place of "max". Occasionally, we write OptSP$_{k,\ell}$ to refer to both of these classes simultaneously, and we write "opt" as a placeholder for either "max" or "min". In analogy to the model-checking problem for first-order properties, we associate to each formula $\psi \in$ OptSP$_{k,\ell}$ an algorithmic problem:

▶ **Definition 8** (Max(ψ) and Min(ψ)). *Let $\psi \in$ MaxSP$_{k,\ell}$ be as in* (1)*. Given a relational structure on objects X, the* Max(ψ) *problem is to compute*

$$\mathrm{OPT} = \max_{x_1,\ldots,x_k \in X} \ \#_{y_1,\ldots,y_\ell \in X} \ \phi(x_1, \ldots, x_k, y_1, \ldots, y_\ell).$$

We similarly define Min(ψ) *for $\psi \in$ MinSP$_{k,\ell}$. Occasionally, for $\psi \in$ OptSP$_{k,\ell}$, we write* Opt(ψ) *to refer to both problems simultaneously.*

As before, we usually assume (without loss of generality) that each variable ranges over a separate set: $x_i \in X_i$, $y_i \in Y_i$. In particular, as claimed before we can express the Sparse MaxIP formula

$$\psi = \max_{x_1 \in X_1, x_2 \in X_2} \#\{y \in [d] : E(x_1, y) \wedge E(x_2, y)\}$$

in a way which is consistent with Definition 8 by introducing three unary predicates for X_1, X_2 and $[d]$. For convenience, we introduce some further notation: For objects $x_1 \in X_1, \ldots, x_k \in X_k$, we denote by $\mathrm{Val}(x_1, \ldots, x_k) = \#_{y_1,\ldots,y_\ell} \phi(x_1, \ldots, x_k, y_1, \ldots, y_\ell)$ the *value* of $(x_1, \ldots, x_k)$.

Definition 8 introduces $\textsc{Max}(\psi)$ and $\textsc{Min}(\psi)$ as *exact* optimization problems (i.e., OPT is required to be computed exactly). We say that an algorithm computes a *c-approximation* for $\textsc{Max}(\psi)$ if it computes any value in the interval $[c^{-1} \cdot \mathrm{OPT}, \mathrm{OPT}]$. Similarly, a c-approximation for $\textsc{Min}(\psi)$ computes any value in $[\mathrm{OPT}, c \cdot \mathrm{OPT}]$.

The problem $\textsc{Opt}(\psi)$ can be solved in time $O(m^{k+\ell-1})$ for all $\textsc{OptSP}_{k,\ell}$ formulas ψ, by a straightforward extension of the model-checking baseline algorithm; see the full version of our paper [8, Appendix A] for details. As this is clearly optimal for $k + \ell = 2$, we will often implicitly assume that $k + \ell \geq 3$ in the following.

As we show in the full version of the paper [8, Appendix B], we can *exactly* solve $\textsc{OptSP}_{k,\ell}$ in time $O(m^{k+\ell-3/2})$ when $\ell \geq 2$. Thus, in the remaining sections we will be working with the *hardest* case $\ell = 1$. For convenience we write $\textsc{MaxSP}_k := \textsc{MaxSP}_{k,1}$, and similarly for $\textsc{MinSP}_k$ and $\textsc{OptSP}_k$. Since for a fixed formula $\psi \in \textsc{OptSP}$, k and ℓ are constants, $f(k, \ell)$-factors are hidden in the O-notation throughout the paper.

3 Technical Overview

In this section we give an overview of the main technical ideas used to give our completeness result (Theorem 3). Let ψ be a $\textsc{MaxSP}_{k,\ell}$ formula. We will outline the reduction from $\textsc{Max}(\psi)$ to Sparse MaxIP. Since for $\ell \geq 2$ we can solve $\textsc{Max}(\psi)$ in time $O(m^{k+\ell-3/2})$ (see the full version of our paper [8, Appendix B]) we focus on the case of $\ell = 1$. The reduction consists of two phases. In the first phase (Appendix A), we reduce ψ to an intermediate problem called the *Hybrid Problem* which captures the core hardness, but is more restricted. For now, the reader can think of the Hybrid Problem as a vector-definable problem (as introduced in the introduction) $\max_{x_1 \in X_1, \ldots, x_k \in X_k} \sum_{i=1}^{d} f(x_1[i], ..., x_k[i])$ with $X_1, \ldots, X_k \subseteq \{0,1\}^d$; we define it formally in Section 4.2. Since a Hybrid Problem is more restricted than the general problem $\textsc{Max}(\psi)$, the first phase consists of the following 4 steps in which we progressively restrict the shape of ψ:

1. Remove all *hyperedges*, that is, ψ no longer contains predicates of arity ≥ 3 so an instance of $\textsc{Max}(\psi)$ can be thought of as a graph with parallel (or alternatively, colored) edges.
2. Remove all edges between vertices x_i and x_j that we maximize over. We will call these *cross edges*. After this step the only remaining edges are between vertices x_i and the counting variable y.
3. Remove all *parallel edges* (or alternatively, colored edges), that is, we combine parallel edges into simple edges.
4. Remove unary predicates, finally turning the $\textsc{Max}(\psi)$ instances into graphs. At this point it becomes simple to rewrite $\textsc{Max}(\psi)$ as a Hybrid Problem.

The second phase of the reduction is to reduce the Hybrid Problem to a Sparse MaxIP instance (Section 4.3). The general idea of this step seems straightforward: For simplicity again let us focus on a vector-definable problem $\max_{x_1 \in X_1, \ldots, x_k \in X_k} \sum_{i=1}^{d} f(x_1[i], ..., x_k[i])$ with $X_1, \ldots, X_k \subseteq \{0,1\}^d$. We can precisely "cover" each $f(x_1[i], \ldots, x_k[i])$ by at most 2^k summands expressing

$$\sum_{\substack{\alpha_1,\ldots,\alpha_k \in \{0,1\} \\ f(\alpha_1,\ldots,\alpha_k)=1}} [(x_1[i], \ldots, x_k[i]) = (\alpha_1, \ldots, \alpha_k)],$$

where the outer $[\cdot]$ denotes the Iverson brackets. Observe that each such summand is equivalent to the MaxIP function, *up to complementing some $x_j[i]$'s* (i.e. each summand can be expressed as MaxIP by setting $x_j[i] := 1 - x_j[i]$ whenever $\alpha_j = 0$). The issue, however, is that complementing $x_j[i]$'s means complementing a binary relation of size $O(m)$ (between n vectors and d coordinates). Since complementing a sparse relation generally produces a dense relation (here: of size $\Omega(nd)$), this will produce a prohibitively large problem size for the Sparse MaxIP formulation if d is large.

The natural approach to overcome this issue is to reduce the dimension of the Hybrid Problem, so that we can afford the complementation step. One challenge in this is that MaxSP formula might have its optimal objective value anywhere in $\{0, \dots, m^\ell\}$, but reducing the dimension from $d \leq m^\ell$ to, say, $d = m^\gamma$ also reduces the range of possible objective values to $\{0, \dots, m^\gamma\}$. It appears counter-intuitive that such a "compression" of objective values should be possible while allowing us to reconstruct the optimum value *exactly*. Perhaps surprisingly, we are able to achieve this by a simple deterministic dimension reduction.

The idea of our dimension reduction is as follows. For concreteness, focus on the Sparse MaxIP problem. Starting from a Sparse MaxIP instance $X_1, X_2 \subseteq \{0,1\}^d$, we construct a hash function $h : \{0,1\}^d \mapsto \{0,1\}^{d'}$ with $d' \ll d$, which maps every one-entry to t coordinates in $[d']$. More precisely, for every coordinate $i \in [d]$, we deterministically choose an auxiliary vector $w_i \in \{0,1\}^{d'}$ with exactly t one-entries for some parameter t. Then, the hash function is defined as $h(x) = \bigvee_{i:x[i]=1} w_i$ (here the OR is applied coordinate-wise).

We say that there is a collision between two vectors x_1, x_2 if there are distinct $i, j \in [d]$ such that $x_1[i] = x_2[j] = 1$ and the auxiliary vectors w_i and w_j share a common one-entry. Ideally, every pair of vectors $x_1 \in X_1, x_2 \in X_2$ is hashed *perfectly*, meaning that no collision takes place. In that case, it holds that $\langle h(x_1), h(x_2)\rangle = t \cdot \langle x_1, x_2\rangle$ and thus also $\mathrm{OPT}' = t \cdot \mathrm{OPT}$, where OPT and OPT' are the objective values of the original and the hashed instance, respectively. However, in reality we cannot expect the hashing to be perfect. Note that nevertheless the difference $|\langle h(x_1), h(x_2)\rangle - t \cdot \langle x_1, x_2\rangle|$ is at most the number of collisions between x_1 and x_2.

We will construct h in such a way that for all pairs x_1, x_2, the number of collisions is small, say at most C. Then by setting $t > 2C$, we ensure that $|t \cdot \mathrm{OPT} - \mathrm{OPT}'| < t/2$ so we can recover $t \cdot \mathrm{OPT}$ by computing OPT' and rounding to the closest multiple of t. In particular, the optimal pair of vectors in the hashed instance correspond to the pair with maximum inner product in the original instance. Note that we crucially use the fact that MaxIP is expressive enough to *compute the value* of the inner product, which allows us to get rid of the small additive error introduced by the hashing (after rounding).

In Section 4.3 we show that the desired hash function exists and is in fact deterministic: Pick any t primes $p_1, \dots, p_t$ of size $\Theta(t \log t)$ and let $d' = p_1 + \cdots + p_t$. We identify $[d']$ with $\{(i, p_j) : 1 \leq j \leq t, 0 \leq i < p_j\}$ and assign the auxiliary vector w_i to have one-entries exactly at all coordinates $(i \bmod p_j, p_j)$, $1 \leq j \leq t$. A simple calculation shows that with this construction the number of collisions between x_1 and x_2 is at most $\|x_1\|_1 \cdot \|x_2\|_1 \cdot \log d$, see Lemma 15. With some additional tricks, we can control this quantity.

Our analysis allows us to even maintain c-approximate solutions, albeit with an arbitrarily small blow-up due to the small error introduced by rounding. Finding a fully approximation-preserving reduction remains a challenge for future work. Additionally, we need to take great care that our reductions are efficient enough to even transfer $\log^{0.1} n$-improvements, to obtain our speed-up for exact optimization (Theorem 6).

Comparison to Gao et al.'s Work

Our reduction is similar to the work of Gao, Impagliazzo, Kolokolova and Williams [18], showing that the sparse version of Orthogonal Vectors is complete for model checking first-order properties. Here we discuss the key differences.

This first phase of our reduction follows the same structure as in Gao et al., but we simplify the proof significantly: One major difference is that they define a more complicated version of the Hybrid Problem including cross predicates [18, Section 5.2]. Borrowing ideas from [9], we remove the cross predicates at an earlier stage of the reduction (Step 2), which simplifies the remaining Steps 3 and 4. The absence of cross predicates also simplifies the baseline algorithm (see [8, Appendix A] in the full version). More generally, by splitting the reduction into a chain of four steps we cleanly separate the main technical ideas used in the first phase; see Appendix A for more details. In the same spirit we simplify Gao et al.'s improved algorithm [18, Section 9.2] for all problems with more than 1 counting quantifier avoiding their case distinction of 9 different cases by using a simple basis to represent all Boolean functions $\phi : \{0,1\}^3 \to \{0,1\}$; see [8, Appendix B] in the full version.

In the second phase of the reduction, their work faces the same main challenge as ours. Specifically, reducing their Hybrid Problem to OV naively requires complementing a sparse binary relation, possibly resulting in a large dense complement. They solve this issue by designing a similar dimension reduction as ours using a Bloom filter. Naturally their dimension reduction is randomized, but they also provide a derandomization. However, note that there is a crucial difference: They reduce to OV which is a *decision* problem, while we reduce to the *optimization* problem MaxIP. For this reason, the dimension reductions differ in nature: One the one hand, we exploit that MaxIP is more expressive than OV – namely that MaxIP can handle a small number of errors if we round the result, while for OV any introduced error would result in vectors that are not orthogonal anymore. On the other hand, by reducing to OV, Gao et al. do not have to worry about "compressing" the range of possible optimal values, or making the reduction approximation-preserving. For these reasons, their dimension reduction would be unsuitable in our work, and ours would be unsuitable in their work.

4 The Reduction

In this section we give the proofs of our main results. The following lemma captures our reduction in all generality. Let k-MaxIP denote the generalization of the MaxIP problem with the objective to compute $\max_{x_1 \in X_1, \ldots, x_k \in X_k} \langle x_1, \ldots, x_k \rangle$, where $\langle x_1, \ldots, x_k \rangle = \sum_y x_1[y] \cdot \ldots \cdot x_k[y]$. We define k-MinIP analogously.

▶ **Lemma 9.** *Let $s(n) \leq n^{1/6}$ be a nondecreasing function and let $c \geq 1$ be constant. Assume that k-MaxIP in dimension $d = \widetilde{O}(s(n)^4 \log^2 n)$ can be c-approximated in time $O(n^k/s(n))$, and let ψ be an arbitrary MaxSP$_k$ formula.*

- *If $c = 1$ (i.e., we are in the case of exact computation), then Max(ψ) can be exactly solved in time $O(m^k/s(\sqrt[k+1]{m}))$.*
- *If $c > 1$, then Max(ψ) can be $(c+\varepsilon)$-approximated in time $O(m^k/s(\sqrt[k+1]{m}))$, for any constant $\varepsilon > 0$.*

The analogous statement holds for k-MinIP and MinSP$_k$.

The outline for this section is as follows. First we show how to derive the completeness result (Theorems 3 and 4) and the lower-order improvements (Theorems 6 and 7) from Lemma 9 in Section 4.1. Then we present the proof of Lemma 9, which is carried out in

two phases as explained in the technical overview. In Section 4.2 we formally introduce the intermediate problem called the *Hybrid Problem*. In Section 4.3 we give a fine-grained reduction from the Hybrid Problem to Maximum or Minimum Inner Product (Lemma 14). Finally, in Appendix A we reduce any $\text{OPT}_{k,\ell}$ formula to the Hybrid Problem (Lemma 17), thus finishing the proof of Lemma 9. We will pay particularly close attention to the exact savings s in every step.

4.1 Consequences

First we derive the completeness Theorems 3 and 4 from Lemma 9.

Proof of Theorems 3 and 4. Let $c \geq 1$ denote the approximation ratio (that is, $c = 1$ for Theorem 3 and $c \geq 1$ for Theorem 4). Assuming that Sparse MaxIP can be c-approximated in time $O(m^{2-\delta})$ for some $\delta > 0$, we obtain an algorithm for c-approximating MaxIP in dimension $d = n^{4\delta/9}$ in time $O((nd)^{2-\delta}) = O(n^{2-\delta}d^2) = O(n^{2-\delta/9})$. We also obtain an algorithm for c-approximating k-MaxIP in the same dimension in time $O(n^{k-\delta/9})$ (brute-force all options for the first $k-2$ vectors, then use the 2-MaxIP algorithm). We can now plug this improved algorithm into our reduction: Setting $s(n) = n^{\delta/9}/\operatorname{polylog}(n)$ we have that k-MaxIP in dimension $d = \widetilde{O}(s(n)^4 \log^2 n)$ can be c-approximated in time $O(n^k/s(n))$. Thus, if $c = 1$ we obtain by Lemma 9 that $\text{OPT}(\psi)$ can be exactly solved in time $O(m^k/s(\sqrt[k+1]{m})) = O(m^{k-\beta})$ for $\beta = \frac{\delta}{9(k+1)} > 0$. If $c > 1$, we obtain that $\text{OPT}(\psi)$ can be $(c+\varepsilon)$-approximated in the same running time, for an arbitrarily small constant $\varepsilon > 0$. ◀

Next, we prove Theorem 5.

Proof of Theorem 5. The reductions from Sparse MaxIP to MaxIP and from Sparse MinIP to MinIP for both the exact and approximate settings are a direct consequence of Lemma 9.

For the other direction, assume there exists some $\delta > 0$ such that Sparse MaxIP can be c-approximated in time $O(m^{2-\delta})$. Set $\gamma := \delta/2$ and observe that any MaxIP instance with $d = n^\gamma$ yields a Sparse MaxIP instance of size $m = O(nd) = O(n^{1+\gamma})$. Since we can solve this instance in time $O(m^{2-\delta}) = O(n^{(1+\gamma)(2-\delta)}) = O(n^{(1+\delta/2)(2-\delta)}) = O(n^{2-\delta^2/2})$, we obtain a $O(n^{2-\delta'})$-algorithm for MaxIP with $d = n^\gamma$ and $\delta' = \delta^2/2$. Note that this works for both the exact ($c = 1$) and approximate ($c > 1$) settings. The proof for the minimization case is analogous. ◀

To prove Theorems 6 and 7, we make use of the following state-of-the-art algorithms for MaxIP and MinIP, established in three previous papers [5, 13, 15].

▶ **Theorem 10** (Improved Algorithms for MaxIP and MinIP [5, 13, 15]).

- *k-MaxIP and k-MinIP in dimension $d = O(\log^{2.9} n)$ can be exactly solved in randomized time $O(n^k/\log^{100} n)$* [5].
- *For any constant $c > 1$, k-MaxIP in dimension $d = 2^{O(\sqrt{\log n})}$ can be c-approximated in deterministic time $n^k/2^{\Omega(\sqrt{\log n})}$* [13, Theorem 1.5].
- *For any constant $c > 1$, k-MinIP in dimension $d = 2^{O(\sqrt{\log n})}$ can be c-approximated in randomized time $n^k/2^{\Omega(\sqrt{\log n})}$* [15, Theorem 1.7].

Proof of Theorems 6 and 7. To prove Theorem 6, we plug in the first algorithm from Theorem 10 into Lemma 9 and choose $s(n) = \log^{0.1} n$. We obtain an exact OPTSP_k algorithm in time $m^k/\log^{\Omega(1)} m$.

For Theorem 7, we plug the second and third algorithms from Theorem 10 into Lemma 9 and choose $s(n) = 2^{O(\sqrt{\log n})}$. We get a c-approximation for OPTSP_k in time $m^k/2^{\Omega(\sqrt{\log m})}$, for any constant $c > 1$. ◀

Note that only one of these algorithms is deterministic; other known deterministic algorithms are not efficient enough for our reduction[6].

4.2 The Hybrid Problem

We start with another problem definition.

▶ **Definition 11** (Basic Problem). *Given set families $\mathcal{S}_1, \ldots, \mathcal{S}_k$ over a universe U, the* Basic Maximization Problem of type $\tau \in \{0,1\}^k$ *is to to compute*

$$\text{OPT} = \max_{S_1 \in \mathcal{S}_1, \ldots, S_k \in \mathcal{S}_k} \left| \left(\bigcap_{i:\tau[i]=1} S_i \right) \setminus \left(\bigcup_{i:\tau[i]=0} S_i \right) \right|.$$

For example, the Basic Problem of type $\tau = 11$ is to maximize the common intersection of two sets S_1 and S_2, the Basic Problem of type $\tau = 10$ is to maximize the number of elements in S_1 not contained in S_2 and the Basic Problem of type $\tau = 00$ is to maximize the number of universe elements contained in neither S_1 nor S_2.

Note that every Basic Problem can be seen as an OptSP_k formula: We introduce objects for all sets S_i and all universe elements u, and connect S_i to u via an edge $E(S_i, u)$ if and only if $u \in S_i$. Consistent with this analogy, we define n as the total number of sets S_i and m as the total cardinality of all sets S_i and, as before, study the Basic Problem with respect to the sparsity m.

▶ **Definition 12** (Hybrid Problem). *Given set families $\mathcal{S}_1, \ldots, \mathcal{S}_k$ over a universe U, which is partitioned into 2^k parts $U = \bigcup_{\tau \in \{0,1\}^k} U_\tau$, the* Hybrid Maximization Problem *is to compute*

$$\text{OPT} = \max_{S_1 \in \mathcal{S}_1, \ldots, S_k \in \mathcal{S}_k} \sum_{\tau \in \{0,1\}^k} \left| U_\tau \cap \left(\bigcap_{i:\tau[i]=1} S_i \right) \setminus \left(\bigcup_{i:\tau[i]=0} S_i \right) \right|.$$

We similarly define Basic Minimization Problems and define c-approximations of Basic Problems in the obvious way. For any $S_1, \ldots, S_k$ and $\tau \in \{0,1\}^k$ we denote by $\text{Val}_\tau(S_1, \ldots, S_k)$ the *value* of the Basic Problem constraint of type τ:

$$\text{Val}_\tau(S_1, \ldots, S_k) := \left| U_\tau \cap \left(\bigcap_{i:\tau[i]=1} S_i \right) \setminus \left(\bigcup_{i:\tau[i]=0} S_i \right) \right|.$$

And we use $\text{Val}(S_1, \ldots, S_k) := \sum_\tau \text{Val}_\tau(S_1, \ldots, S_k)$ to denote the total value of the sets $S_1, \ldots, S_k$ in a Hybrid Problem instance.

Intuitively, the Hybrid Problem simultaneously optimizes Basic Problem constraints of different types. If we could afford to complement (parts of) the sets S_i, then there is a straightforward reduction from the Hybrid Problem to a Basic Problem of arbitrary type τ: For each constraint of type $\tau' \neq \tau$, we simply complement all sets S_i with $\tau[i] \neq \tau'[i]$ (more precisely, construct sets S'_i such that $U_{\tau'} \cap S'_i = U_{\tau'} \setminus S_i$) and reinterpret the τ'-constraint as type τ. In summary:

[6] Focus on exact MaxSP_k for illustration: To obtain the same savings as in Theorem 6, we would need a deterministic algorithm for MaxIP in dimension $d = O(\log^{2.9} n)$ running in time $O(n^2/\log^{100} n)$. However, for this speed-up the current best algorithm [5] requires $d = O(\log^{1.9} n)$, so one needs to either improve the algorithm or improve our dimension reduction (Lemma 9) to dimension $d = \text{poly}(s(n)) \log n$, say.

▶ **Observation 13.** *In time $O(n|U|)$, any Hybrid Problem instance can be converted into an equivalent Basic Problem instance of arbitrary type τ. The sparsity of the constructed instance is up to $n|U|$.*

However, being in the sparse setup we cannot tolerate the blow-up in the sparsity. Therefore, in order to efficiently apply Observation 13, we first have to control the universe size $|U|$.

4.3 Universe Reduction

The goal of this section is to reduce the Hybrid Problem to k-MaxIP. We give a reduction which closely preserves the savings $s(n)$ achieved by exact or approximate k-MaxIP algorithms (losing only polynomial factors in $s(n)$). As a drawback, the reduction slightly worsens the approximation factor, turning a c-approximation into a $(c+\varepsilon)$-approximation.

▶ **Lemma 14.** *Let $s(n) \leq n^{1/6}$ be a nondecreasing function and assume that k-MaxIP in dimension $d = \widetilde{O}(s(n)^4 \log^2 n)$ can be c-approximated in time $O(n^k/s(n))$.*

- *If $c = 1$ (i.e., we are in the case of exact computation), then the Hybrid Problem can be exactly solved in time $O(m^k/s(m))$.*
- *If $c > 1$, then the Hybrid Problem can be $(c+\varepsilon)$-approximated in time $O(m^k/s(m))$, for any constant $\varepsilon > 0$.*

The analogous statement holds for k-MinIP and MinSP$_k$.

On a high level, we prove Lemma 14 by first using a deterministic construction to reduce the universe size, and then reducing further to k-MaxIP as in Observation 13. The following lemma provides our universe reduction in the form of a hash-like function h.

▶ **Lemma 15.** *Let U be a universe and let t be a parameter. There exists a universe U' of size at most $4t^2 \log t$ and a function h mapping elements in U to size-t subsets of U', such that the following properties hold. By abuse of notation, we write $h(S) = \bigcup_{u \in S} h(u)$ for sets $S \subseteq U$.*

1. *(Hashing.) For all sets $S \subseteq U$, it holds that $|h(S)| \geq t|S| - |S|^2 \log |U|$.*
2. *(Efficiency.) Evaluating $h(u)$ takes time $\widetilde{O}(t)$.*

Proof. We start with the construction of h. By the Prime Number Theorem, there exist t primes $p_1, \ldots, p_t$ in the interval $[2t \log t, 4t \log t]$ (for large enough t, see [27, Corollary 3] for the quantitative version). Let $U' = \{(i,j) : 1 \leq i \leq t, 0 \leq j < p_i\}$, then $|U'| \leq 4t^2 \log t$. We identify U with $[|U|]$ in an arbitrary way and define $h(u) = \{(i, u \bmod p_i) : 1 \leq i \leq t\}$ for $u \in U$.

In order to prove the first property, let us define the *collision number* of two distinct elements $u, u' \in U$ as $|h(u) \cap h(u')|$. It is easy to see that the collision number of any such pair is at most $\log |U|$: For any prime p_i, we have that $u \bmod p_i = u' \bmod p_i$ if and only if p_i divides $u - u'$. Since $u - u'$ has absolute value at most U, there can be at most $\log |U|$ distinct prime factors p_i of $u - u'$. It follows that $t|S| - |h(S)| \leq \sum_{u,u' \in S} |h(u) \cap h(u')| \leq |S|^2 \log |U|$.

Finally, the function can be efficiently evaluated: Computing the primes $p_1, \ldots, p_t$ takes time $O(t \log t \log \log t)$ using Eratosthenes' sieve, for example. After this precomputation, evaluating $h(u)$ in time $O(t)$ is straightforward. ◀

▶ **Lemma 16** (Universe Reduction). *Let $\mathcal{S}_1, \ldots, \mathcal{S}_k$ over the universe $U = \bigcup_\tau U_\tau$ be a Hybrid Problem instance of maximum set size $s = \max_{S_i \in \mathcal{S}_i} |S_i|$, and let t be a parameter. In time $\widetilde{O}(mt)$ we can compute a number $\Delta \geq 0$ and a new Hybrid Problem instance $\mathcal{S}'_1, \ldots, \mathcal{S}'_k$ over a small universe $U' = \bigcup_\tau U'_\tau$ of size $|U'| = O(t^2 \log t)$ such that:*

1. *The sets $S_i \in \mathcal{S}_i$ and the sets $S'_i \in \mathcal{S}'_i$ stand in one-to-one correspondence.*
2. *For all $S_1 \in \mathcal{S}_1, \ldots, S_k \in \mathcal{S}_k$, it holds that:*

$$|t \cdot \mathrm{Val}(S_1, \ldots, S_k) - \mathrm{Val}(S'_1, \ldots, S'_k) - \Delta| = O(s^2 \log |U|).$$

Proof. We first describe how to construct the new instance. The first goal is to design individual universe reductions for all subuniverses U_τ, that is, we construct new universes U'_τ and functions h_τ mapping U_τ to size-t subsets of U_τ. We distinguish two cases:

- If $|U_\tau| \leq 4t \log t$, then we simply take U'_τ as t copies of U_τ and let h_τ be the function which maps any element to its t copies in U_τ. It holds that $|U'_\tau| = t \cdot |U_\tau| \leq 4t^2 \log t$.
- If $|U_\tau| > 4t \log t$, then we apply Lemma 15 with parameter t to obtain U'_τ and h_τ. The lemma guarantees that $|U'_\tau| \leq 4t^2 \log t$.

Next, we assemble these individual reductions into one. Set $U' = \bigcup_\tau U'_\tau$, where we treat the sets U'_τ as disjoint. Since in both of the previous two cases we have $|U_\tau| = O(t^2 \log t)$ it follows that $|U| = \sum_\tau |U_\tau| = O(t^2 \log t)$. Let h be the function which is piece-wise defined by the h_τ's, that is, h returns $h_\tau(u)$ on input $u \in U_\tau$. Recall the notation $h(S) = \bigcup_{u \in S} h(u)$. The new Hybrid Problem instance is constructed by hashing every set $S_i \in \mathcal{S}_i$ into the smaller universe, that is, we set $S'_i := h(S_i) \in \mathcal{S}'_i$. Property 1 is immediate from this construction, and the computation takes time $\widetilde{O}(mt)$.

It remains to prove Property 2. For the remainder of the proof fix some sets $S_1, \ldots, S_k$ and let $S = S_1 \cup \cdots \cup S_k$ (clearly, S has size $O(s)$). We start with the (unrealistic) assumption that S is hashed *perfectly*, that is, $|h(S)| = t|S|$. In this case we claim that:

- $t \cdot \mathrm{Val}_\tau(S_1, \ldots, S_k) = \mathrm{Val}_\tau(h(S_1), \ldots, h(S_k))$ for all $\tau \neq 0^k$,
- $t \cdot \mathrm{Val}_\tau(S_1, \ldots, S_k) = \mathrm{Val}_\tau(h(S_1), \ldots, h(S_k)) + \Delta$ for $\tau = 0^k$, where $\Delta := t \cdot |U_{0^k}| - |U'_{0^k}|$.

Indeed, if S is hashed perfectly then we exactly scale the number of satisfying elements by a factor of t for every type $\tau \neq 0^k$. This holds because a satisfying assignment for $\tau \neq 0^k$ corresponds to some element of the universe $u \in U_\tau$ for which $u \in S_i$ for all i's such that $\tau[i] = 1$. The perfect hashing implies that the element u in these sets S_i gets mapped to t different elements in the new universe, and since there are no collisions these form t satisfying assignments in the hashed instance. The type $\tau = 0^k$ is exceptional because each satisfying assignment does not correspond to any $u \in U_{0^k}$. Instead, the hashing scales the number of *falsifying* elements of type 0^k, $|U_{0^k} \cap S|$. The number of *satisfying* elements of type 0^k, $|U_{0^k} \setminus S|$, is preserved up to an additive error of exactly Δ.

We will now remove the unrealistic assumption that h is hashed perfectly. The strategy is to define another function h^* obtained from h by artificially making the hashing with S perfect. To that end, we list the elements in S in an arbitrary order $s_1, \ldots, s_{|S|}$, and start with the assignment $h^*(s_j) = h(s_j)$. As long as there exist indices $i < j$ such that $h^*(s_i)$ and $h^*(s_j)$ share a common element z, we reassign $h^*(s_j) := h^*(s_j) \setminus \{z\} \cup \{z'\}$ for some unused universe element $z' \in U'$. The function h^* obtained in this way also maps elements of U to size-t subsets of U' and hashes S perfectly. Let Z be the set of all pairs of elements z and z' that occurred in the process; since there are exactly $t|S| - |h(S)|$ iterations we have $|Z| \leq 2t|S| - 2|h(S)|$ and by Lemma 15 it follows that $|Z| = O(s^2 \log |U|)$. By the definition of h^*, it is clear that $|\mathrm{Val}(h(S_1), \ldots, h(S_k)) - \mathrm{Val}(h^*(S_1), \ldots, h^*(S_k))| \leq |Z|$. Therefore, by the previous paragraph (applied with h^*) and by an application of the triangle inequality, we obtain:

- $|t \cdot \mathrm{Val}_\tau(S_1, \ldots, S_k) - \mathrm{Val}_\tau(h(S_1), \ldots, h(S_k))| = O(s^2 \log |U|)$ for all $\tau \neq 0^k$,
- $|t \cdot \mathrm{Val}_\tau(S_1, \ldots, S_k) - \mathrm{Val}_\tau(h(S_1), \ldots, h(S_k)) - \Delta| = O(s^2 \log |U|)$ for $\tau = 0^k$.

The claimed Property 2 is now immediate by summing over all types τ and by another application of the triangle inequality.

Finally, it remains to prove that $\Delta \geq 0$. There are two cases depending on how the set U'_{0^k} was constructed: In the first case of the construction we have $t \cdot |U_{0^k}| = |U'_{0^k}|$ and thus $\Delta = 0$. In the second case we have $|U'_\tau| \leq 4t^2 \log t < t \cdot |U_\tau|$ and thus $\Delta = t \cdot |U_{0^k}| - |U'_{0^k}| > 0$. ◀

Having established the universe reduction, we can finally prove Lemma 14.

Proof of Lemma 14. The algorithm consists of three steps, which are implemented in the same way for all combinations of maximization versus minimization and exact versus approximate computation.

1. *(Eliminating heavy sets.)* We say that a set $S_i \in \mathcal{S}_i$ is *heavy* if $|S_i| > s(m)$, and *light* otherwise. Our first goal is to eliminate all heavy sets. Since the total cardinality of all sets S_i is bounded by m, there can be at most $O(m/s(m))$ heavy sets. Therefore, we can brute-force over every such set S_i and solve the remaining Hybrid Problem on $k-1$ set families using the baseline algorithm in time $O(m^{k-1})$. Afterwards, we can safely remove all heavy sets. Overall, this step takes time $O(m^k/s(m))$.
2. *(Reduction to k-MaxIP or k-MinIP.)* In the remaining instance we have that $|S_i| \leq s(m)$ for all sets S_i. Therefore, we can apply the universe reduction from Lemma 16 (with some parameter t to be specified in the next step) to obtain an instance $\mathcal{S}'_1, \ldots, \mathcal{S}'_k$ over a smaller universe $U' = \bigcup_\tau U'_\tau$ of size $O(t^2 \log t)$, and an offset $\Delta \geq 0$.
   The Hybrid Maximization Problem instance $\mathcal{S}'_1, \ldots, \mathcal{S}'_k$ reduces to k-MaxIP in the natural way: Recall that k-MaxIP is the same as the Basic Problem of type $\tau = 1^k$. Hence, we can apply Observation 13 to reduce to an instance of k-MaxIP with $n = O(m)$ vectors in dimension $O(t^2 \log t)$ in time $O(n|U'|) = O(nt^2 \log t)$. An analogous reduction works for Hybrid Minimization Problems and k-MinIP.
3. *(Recovering the optimal value.)* Solve (or approximate) the constructed k-MaxIP instance and let ALG' denote the output. Then compute $\mathrm{ALG} := (\mathrm{ALG}' + \Delta)/t$ and return ALG rounded to an integer. The precise way of rounding depends on maximization versus minimization and exact versus approximate, see the following analysis.

Let $\varepsilon > 0$ be a constant which we will specify later, and set $t = Cs(m)^2 \log m$ for some sufficiently large constant $C = C(\varepsilon)$. Then by Property 2 of Lemma 16 we have

$$\left| \mathrm{Val}(S_1, \ldots, S_k) - \frac{\mathrm{Val}(S'_1, \ldots, S'_k) + \Delta}{t} \right| = O\left(\frac{s(m)^2 \log m}{t} \right) < \varepsilon.$$

In particular, it holds that

$$\left| \mathrm{OPT} - \frac{\mathrm{OPT}' + \Delta}{t} \right| < \varepsilon, \tag{2}$$

where OPT and OPT' are the optimal values of the original and the reduced instance, respectively. As the new universe has size $O(t^2 \log t) = \widetilde{O}(s(m)^4 \log m^2)$ as claimed, we can indeed use the efficient $O(m^k/s(m))$-time k-MaxIP or k-MinIP algorithm in the third step. The total running time is as stated: Recall that $s(m) \leq m^{1/6}$ and thus all previous steps run in time $O(m^k/s(m))$. It remains to argue about the guarantees of the reduction; we need to consider three cases:

- *(Exact maximization or minimization: $c = 1$.)* It suffices to set $\varepsilon < \frac{1}{2}$. Since we can exactly compute $\mathrm{ALG}' = \mathrm{OPT}'$, by rounding $\mathrm{ALG} = (\mathrm{ALG}' + \Delta)/t$ to the nearest integer, we obtain the only integer in the interval $((\mathrm{OPT}' + \Delta)/t - \frac{1}{2}, (\mathrm{OPT}' + \Delta)/t + \frac{1}{2})$, and thus we output OPT.

- *(Approximate maximization: $c > 1$.)* We have $c^{-1}\mathrm{OPT}' \leq \mathrm{ALG}' \leq \mathrm{OPT}'$ and therefore

$$\mathrm{ALG} = \frac{\mathrm{ALG}' + \Delta}{t} \leq \frac{\mathrm{OPT}' + \Delta}{t} \overset{(2)}{\leq} \mathrm{OPT} + \varepsilon,$$
$$\mathrm{ALG} = \frac{\mathrm{ALG}' + \Delta}{t} \geq \frac{c^{-1}(\mathrm{OPT}' + \Delta)}{t} \overset{(2)}{\geq} c^{-1}(\mathrm{OPT} - \varepsilon) \geq c^{-1}\mathrm{OPT} - \varepsilon,$$

  where in the first inequality of the second line we used both $\mathrm{ALG}' \geq c^{-1}\mathrm{OPT}'$ and $c > 1$. From these bounds we derive that the algorithm should return $\lceil \mathrm{ALG} - \varepsilon \rceil$. Indeed, as $\lceil \mathrm{ALG} - \varepsilon \rceil \leq \mathrm{OPT}$ this is always a feasible solution. Moreover, the solution is $\frac{c}{1-2\varepsilon}$-approximate: If $\mathrm{OPT} = 0$, then $\lceil \mathrm{ALG} - \varepsilon \rceil = 0$ (if we set $\varepsilon < \frac{1}{2}$). If $\mathrm{OPT} \geq 1$, then $\lceil \mathrm{ALG} - \varepsilon \rceil \geq \frac{1}{c}\mathrm{OPT} - 2\varepsilon \geq \frac{1-2\varepsilon}{c}\mathrm{OPT}$. Setting ε small enough yields approximation ratio $c + \varepsilon'$, for any $\varepsilon' > 0$.
- *(Approximate minimization: $c > 1$.)* We have $\mathrm{OPT}' \leq \mathrm{ALG}' \leq c \cdot \mathrm{OPT}'$ and therefore

$$\mathrm{ALG} = \frac{\mathrm{ALG}' + \Delta}{t} \leq \frac{c \cdot \mathrm{OPT}' + \Delta}{t} \leq \frac{c \cdot (\mathrm{OPT} + \Delta)}{t} \overset{(2)}{\leq} c \cdot \mathrm{OPT} + c \cdot \varepsilon,$$
$$\mathrm{ALG} = \frac{\mathrm{ALG}' + \Delta}{t} \geq \frac{\mathrm{OPT}' + \Delta}{t} \overset{(2)}{\geq} \mathrm{OPT} - \varepsilon.$$

  In this case the algorithm should return $\lfloor \mathrm{ALG} + \varepsilon \rfloor$. This solution is always feasible as $\mathrm{OPT} \leq \lfloor \mathrm{ALG} + \varepsilon \rfloor$. Moreover, the solution is $c(1 + 2\varepsilon)$-approximate: If $\mathrm{OPT} = 0$, then $\lfloor \mathrm{ALG} + \varepsilon \rfloor = 0$ (if we set $\varepsilon < \frac{1}{2}$). If $\mathrm{OPT} \geq 1$, then $\lfloor \mathrm{ALG} + \varepsilon \rfloor \leq c \cdot \mathrm{OPT} + (c+1)\varepsilon \leq c(1 + 2\varepsilon)\mathrm{OPT}$. We may again set ε small enough to obtain approximation ratio $c + \varepsilon'$, for any $\varepsilon' > 0$. ◀

References

1. Amir Abboud and Arturs Backurs. Towards hardness of approximation for polynomial time problems. In *Proceedings of the 8th Conference on Innovations in Theoretical Computer Science*, volume 67 of *ITCS '17*, pages 11:1–11:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
2. Amir Abboud and Aviad Rubinstein. Fast and deterministic constant factor approximation algorithms for LCS imply new circuit lower bounds. In *Proceedings of the 9th Conference on Innovations in Theoretical Computer Science*, volume 94 of *ITCS '18*, pages 35:1–35:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
3. Amir Abboud, Aviad Rubinstein, and Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In *Proceedings of the 58th IEEE Annual Symposium on Foundations of Computer Science*, FOCS '17, pages 25–36. IEEE Computer Society, 2017.
4. Amir Abboud, Ryan Williams, and Huacheng Yu. More applications of the polynomial method to algorithm design. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 218–230. SIAM, 2015.
5. Josh Alman, Timothy Chan, and Ryan Williams. Polynomial representations of threshold functions and algorithmic applications. In *Proceedings of the 57th IEEE Annual Symposium on Foundations of Computer Science*, FOCS '16, pages 467–476. IEEE Computer Society, 2016.
6. Arturs Backurs, Liam Roditty, Gilad Segal, Virginia Vassilevska Williams, and Nicole Wein. Towards tight approximation bounds for graph diameter and eccentricities. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, STOC '18, pages 267–280. ACM, 2018.
7. Karl Bringmann. Why walking the dog takes time: Fréchet distance has no strongly subquadratic algorithms unless SETH fails. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science*, FOCS '15, pages 661–670. IEEE Computer Society, 2014.

8 Karl Bringmann, Alejandro Cassis, Nick Fischer, and Marvin Künnemann. Fine-grained completeness for optimization in P. *CoRR*, abs/2107.01721, 2021. URL: `http://arxiv.org/abs/2107.01721`.

9 Karl Bringmann, Nick Fischer, and Marvin Künnemann. A fine-grained analogue of Schaefer's theorem in P: Dichotomy of $\exists^k\forall$-quantified first-order graph properties. In *Proceedings of the 34th Computational Complexity Conference*, volume 137 of *CCC '19*, pages 31:1–31:27. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019.

10 Karl Bringmann, Marvin Künnemann, and Karol Wegrzycki. Approximating APSP without scaling: Equivalence of approximate min-plus and exact min-max. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, STOC '19, pages 943–954. ACM, 2019.

11 Marco L. Carmosino, Jiawei Gao, Russell Impagliazzo, Ivan Mihajlin, Ramamohan Paturi, and Stefan Schneider. Nondeterministic extensions of the strong exponential time hypothesis and consequences for non-reducibility. In *Proceedings of the 7th ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, pages 261–270. ACM, 2016.

12 Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1246–1255. SIAM, 2016.

13 Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Theory of Computing*, 16(4):1–50, 2020. `doi:10.4086/toc.2020.v016a004`.

14 Lijie Chen, Shafi Goldwasser, Kaifeng Lyu, Guy N. Rothblum, and Aviad Rubinstein. Fine-grained complexity meets IP = PSPACE. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 1–20. SIAM, 2019.

15 Lijie Chen and Ryan Williams. An equivalence class for orthogonal vectors. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, pages 21–40. SIAM, 2019.

16 Martin Dietzfelbinger, Philipp Schlag, and Stefan Walzer. A subquadratic algorithm for 3XOR. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science*, volume 117 of *MFCS '18*, pages 59:1–59:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

17 Andreas Emil Feldmann, Karthik C. S., Euiwoong Lee, and Pasin Manurangsi. A survey on approximation in parameterized complexity: Hardness and algorithms. *Algorithms*, 13(6):146, 2020. `doi:10.3390/a13060146`.

18 Jiawei Gao, Russell Impagliazzo, Antonina Kolokolova, and Ryan Williams. Completeness for first-order properties on sparse structures with algorithmic applications. *ACM Trans. Algorithms*, 15(2):23:1–23:35, 2019.

19 Erich Grädel, Phokion G. Kolaitis, Leonid Libkin, Maarten Marx, Joel Spencer, Moshe Y. Vardi, Yde Venema, and Scott Weinstein. *Finite Model Theory and Its Applications*. Springer Berlin Heidelberg, 2007.

20 Neil Immerman. *Descriptive Complexity*. Springer New York, 1999.

21 Zahra Jafargholi and Emanuele Viola. 3SUM, 3XOR, triangles. *Algorithmica*, 74(1):326–343, 2016.

22 C. S. Karthik, Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. *J. ACM*, 66(5):33:1–33:38, 2019.

23 Karthik C. S. and Pasin Manurangsi. On closest pair in euclidean metric: Monochromatic is as hard as bichromatic. In *Proceedings of the 10th Conference on Innovations in Theoretical Computer Science*, volume 124 of *ITCS '19*, pages 17:1–17:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

24 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

25 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, STOC '13, pages 515–524. ACM, 2013.

26 Dhruv Rohatgi. Conditional hardness of earth mover distance. In Dimitris Achlioptas and László A. Végh, editors, *Proceedings of Approximation, Randomization, and Combinatorial Optimization (APPROX/RANDOM'19)*, volume 145 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.APPROX-RANDOM.2019.12.

27 J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math.*, 6(1):64–94, March 1962. doi:10.1215/ijm/1255631807.

28 Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, STOC '18, pages 1260–1268. ACM, 2018.

29 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians*, ICM '18, pages 3447–3487, 2018.

A Reducing OptSP$_k$ Formulas to the Hybrid Problem

In this section we give the first phase of the reduction, where we reduce $\textsc{Opt}(\psi)$ to the Hybrid Problem. The main lemma is the following. As before, let $s(m) \leq m^{1/6}$ be a nondecreasing function and let $c \geq 1$ be constant.

▶ **Lemma 17.** *Let $k \geq 2$. If the Hybrid Problem can be c-approximated in time $O(m^k/s(m))$, then $\textsc{Opt}(\psi)$ can be c-approximated in time $O(m^k/s(\sqrt[k+1]{m}))$, for any $\textsc{OptSP}_k$ formula ψ.*

Recall that we only have to deal with $\textsc{OptSP}_k = \textsc{OptSP}_{k,1}$ formulas, as any $\textsc{OptSP}_{k,\ell}$ problem with $\ell > 1$ directly admits an improved algorithm; see [8, Appendix B] in the full version. As explained in Section 3, we prove Lemma 17 by progressively simplifying $\textsc{Opt}(\psi)$ in four steps:

1. Remove all *hyperedges*, that is, ψ no longer contains predicates of arity ≥ 3 so an instance of $\textsc{Opt}(\psi)$ can be thought of as a (colored) graph.
2. Remove all *cross edges*, that is, edges between vertices x_i and x_j that we maximize over.
3. Remove all *parallel edges* (or alternatively, colored edges), that is, we combine parallel edges into simple edges.
4. Remove unary predicates, finally turning the $\textsc{Opt}(\psi)$ instances into graphs. At this point it becomes simple to rewrite $\textsc{Opt}(\psi)$ as a Hybrid Problem.

Step 1: Removing Hyperedges

As a first step, we eliminate all *hyperpredicates*, that is, predicates of arity ≥ 3. Formally, we prove the following lemma.

▶ **Lemma 18.** *Suppose that, for any $\textsc{OptSP}_k$ formula ψ not containing hyperpredicates, $\textsc{Opt}(\psi)$ can be c-approximated in time $O(m^k/s(m))$. Then $\textsc{Opt}(\psi)$ can be c-approximated in time $O(m^k/s(m))$ for any $\textsc{OptSP}_k$ formula ψ.*

The proof is quite similar to [18, Section 7]. We start with a technical lemma:

▶ **Lemma 19.** *Let*

$$\psi = \operatorname*{opt}_{x_1,\ldots,x_k} \#_{y} \Big(E(x_i, x_j) \wedge \phi(x_1, \ldots, x_k, y) \Big),$$

for some $i, j \in [k], i \neq j$ and arbitrary ϕ. Then $\textsc{Opt}(\psi)$ can be solved exactly in time $O(m^{k-1/2})$.

Proof. Let us begin with the simplest case $k = 2$. For a vertex x in the given instance, let $\deg(x)$ denote the total number of records containing x over all relations. We distinguish the following three cases:

Case 1: $\deg(x_1) \geq \sqrt{m}$. We explicitly list all vertices x_1 with $\deg(x_1) \geq \sqrt{m}$; note that there can be at most $O(\sqrt{m})$ such elements since the sparsity of the $\text{Max}(\psi)$ instance is bounded by $O(m)$. The remaining MaxSP_1 formula can be solved in time $O(m)$ using the baseline algorithm. In total, this step takes time $O(\sqrt{m}m) = O(m^{3/2})$.

Case 2: $\deg(x_2) \geq \sqrt{m}$. By exchanging the roles of x_1 and x_2, we deal with this case in the same way as case 1.

Case 3: $\deg(x_1) < \sqrt{m}$ and $\deg(x_2) < \sqrt{m}$. Assuming that the previous two cases were executed, we can assume that $\deg(x_1) < \sqrt{m}$ and $\deg(x_2) < \sqrt{m}$ for all remaining objects x_1, x_2. We exploit that any non-zero solution (x_1, x_2) of $\text{Max}(\psi)$ satisfies $E(x_1, x_2)$: It suffices to maximize over all $O(m)$ edges $E(x_1, x_2)$, counting the number of y's satisfying ϕ. Since $\deg(x_1) < \sqrt{m}$ and $\deg(x_2) < \sqrt{m}$, we can enumerate and test all objects y which are connected to either x_1 or x_2 by some relation in time $O(\sqrt{m})$. What remains are objects y not connected to either x_1 or x_2 by any relation. To account for these missing objects, we can substitute *false* for all non-unary predicates in ϕ; what remains is a Boolean function over unary predicates over y. We can precompute the number of y's satisfying that function in linear time, so again the total time is $O(m+m\sqrt{m}) = O(m^{3/2})$.

It remains to lift this proof to the general case $k > 2$. We brute-force over all x-variables except for x_i and x_j. This amounts for a factor $O(m^{k-2})$ in the running time. What remains is a MaxSP_2 formula in the shape as before which can be solved exactly in time $O(m^{3/2})$ by the previous case analysis. In total this takes time $O(m^{k-2}m^{3/2}) = O(m^{k-1/2})$. The proof works in exactly the same way for minimization problems. ◀

Proof of Lemma 18. Let $\psi = \max_{x_1,\dots,x_k} \#_y \phi(x_1, \dots, x_k, y)$ be a MaxSP_k formula possibly containing some hyperpredicates. We introduce a new binary relation $N(x_i, x_j)$ defined as follows: For any $x_i, x_j \in V$ it holds that $N(x_i, x_j) = \mathit{true}$ if and only if x_i and x_j are connected by some (hyper-)edge. Observe that any (hyper-)edge contributes to at most a constant number of records $N(x_i, x_j)$, so we can construct N in time $O(m)$ and the sparsity blows up only by a constant factor. We can now rewrite ψ via

$$\psi_0 = \max_{x_1,\dots,x_k} \#_y \left(\left(\bigwedge_{i \neq j} \overline{N}(x_i, x_j) \right) \wedge \phi_0(x_1, \dots, x_k, y) \right)$$

and

$$\psi_{i,j} = \max_{x_1,\dots,x_k} \#_y \Big(N(x_i, x_j) \wedge \phi(x_1, \dots, x_k, y) \Big),$$

where ϕ_0 is obtained from ϕ by replacing all occurrences of hyperpredicates by *false*. It follows that we can express

$$\text{OPT} = \max\{\text{OPT}_0, \max_{i \neq j} \text{OPT}_{i,j}\},$$

where OPT_0 is the optimal value of ψ_0, and $\text{OPT}_{i,j}$ is the optimal value of $\psi_{i,j}$. Observe that ψ_0 is a MaxSP_k formula not involving any hyperpredicates, so we can by assumption c-approximate OPT_0 in time $T(m)$. Moreover, the formulas $\psi_{i,j}$ are precisely in the shape to apply Lemma 19, so we can compute $\text{OPT}_{i,j}$ exactly in time $O(m^{k-1/2})$. ◀

Step 2: Removing Cross Edges

Next, the goal is to remove all binary predicates $E(x_i, x_j)$ between two x-variables. Let us call these predicates $E(x_i, x_j)$ *cross predicates* and the associated entries (x_i, x_j) *cross edges*.

▶ **Lemma 20.** *Suppose that, for any* OPTSP_k *formula* ψ *not containing hyperpredicates and cross predicates,* $\text{OPT}(\psi)$ *can be* c*-approximated in time* $O(m^k/s(m))$*. Then* $\text{OPT}(\psi)$ *can be* c*-approximated in time* $O(m^k/s(\sqrt[k+1]{m}))$ *for any* OPTSP_k *formula* ψ *not containing hyperpredicates.*

Proof. Let $\psi = \max_{x_1,\dots,x_k} \#_y \phi(x_1, \dots, x_k, y)$ and let $E_1, \dots, E_r$ denote the cross predicates in the given instance. We define

$$\psi_0 := \max_{x_1,\dots,x_k} \#_y \left(\left(\bigwedge_{\ell,i,j} \overline{E}_\ell(x_i, x_j) \right) \wedge \phi_0(x_1, \dots, x_k, y) \right)$$

and

$$\psi_{\ell,i,j} := \max_{x_1,\dots,x_k} \#_y \Big(E_\ell(x_i, x_j) \wedge \phi(x_1, \dots, x_k, y) \Big),$$

where $\ell \in [r]$ and $i \neq j \in [k]$ and ϕ_0 is the propositional formula obtained from ϕ by substituting all predicates $E_\ell(x_i, x_j)$ by *false*. It is easy to verify that

$$\text{OPT} = \max\{\text{OPT}_0, \max_{\ell,i,j} \text{OPT}_{\ell,i,j}\},$$

where OPT_0 and $\text{OPT}_{\ell,i,j}$ are the optimal values of $\text{MAX}(\psi_0)$ and $\text{MAX}(\psi_{\ell,i,j})$, respectively. Using Lemma 19, we can compute $\text{OPT}_{\ell,i,j}$ exactly in time $O(m^{k-1/2})$ for all ℓ, i, j. It remains to efficiently solve $\text{MAX}(\psi_0)$ to compute OPT_0.

As described before, we can always assume that each variable ranges over a separate set: $x_i \in X_i$, $y \in Y$. We call a vertex x_i *heavy* if it has degree at least $\sqrt[k+1]{m}$, and *light* otherwise. The first step is to eliminate all heavy vertices; there can exist at most $O(m/\sqrt[k+1]{m})$ many such vertices x_i. Fixing x_i, we can solve the remaining problem in time $O(m^{k-1})$ using the baseline algorithm. We keep track of the optimal solution detected in this way. This precomputation step takes time $O(m^k/\sqrt[k+1]{m})$ and afterwards we can safely remove all heavy vertices.

Next, partition each set X_i into several groups $X_{i,1}, \dots, X_{i,g}$ such that the total degree of all vertices in a group is $O(\sqrt[k+1]{m})$, and the number of groups is $g = O(m/\sqrt[k+1]{m})$. This is implemented by greedily inserting vertices into $X_{i,j}$ until its total degree exceeds $\sqrt[k+1]{m}$. As each vertex inserted in that way is light, we can overshoot by at most $\sqrt[k+1]{m}$.

Let $\psi_1 := \max_{x_1,\dots,x_k} \#_y \phi_0(x_1, \dots, x_k, y)$; note that ψ_1 equals ψ_0 except that it disregards the cross predicates. Therefore, by assumption we can c-approximate $\text{MAX}(\psi_1)$ in time $O(m^k/s(m))$. The algorithm continues as follows:

1. For all combinations $(j_1, \dots, j_k) \in [g]^k$, compute a c-approximation of $\text{MAX}(\psi_1)$ on the input $X_{1,j_1}, \dots, X_{k,j_k}$. We keep track of the $\binom{k}{2}mn^{k-2} + 1$ combinations with largest values (breaking ties arbitrarily).
2. For any of the top-most $\binom{k}{2}mn^{k-2} + 1$ combinations $(j_1, \dots, j_k)$, solve $\text{MAX}(\psi_0)$ exactly on $X_{1,j_1}, \dots, X_{k,j_k}$ using the baseline algorithm. Return the best solution detected in this step or the precomputation phase.

We begin with the correctness of the algorithm. First, the value of any solution $(x_1, \dots, x_k)$ in $\text{MAX}(\psi_0)$ is at least as large as its value in $\text{MAX}(\psi_1)$. In particular, the optimal solution of $\text{MAX}(\psi_0)$ has value at least OPT_0 in $\text{MAX}(\psi_1)$. We next establish an upper bound on

the number *false positives*, that is, tuples $(x_1, \dots, x_k)$ of different value in $\textsc{Max}(\psi_0)$ than in $\textsc{Max}(\psi_1)$. Observe that any such false positive contains at least one edge (x_i, x_j) and since there are at most m edges, at most $\binom{k}{2}$ choices of i, j and at most n^{k-2} choices for the remaining vertices x_ℓ, $\ell \neq i, j$, we can indeed bound the number of false positives by $\binom{k}{2}mn^{k-2}$. Thus, if we witness the top-most $\binom{k}{2}mn^{k-2}+1$ solutions of $\textsc{Max}(\psi_1)$ in step 1, among these there exists at least one solution of value $\geq \mathrm{OPT}_0/c$ in $\textsc{Max}(\psi_0)$.

Finally, let us bound the running time of the above algorithm. Recall that removing heavy vertices accounts for $O(m^k/\sqrt[k+1]{m})$ time. In step 1, the $\textsc{Max}(\psi_1)$ algorithm is applied g^k times on instances of size $O(\sqrt[k+1]{m})$, which takes time $O((m/\sqrt[k+1]{m})^k \cdot (\sqrt[k+1]{m})^k/s(\sqrt[k+1]{m})) = O(m^k/s(\sqrt[k+1]{m}))$. Step 2 runs the baseline algorithm $mn^{k-2} = O(m^{k-1})$ times on instances of size $O(\sqrt[k+1]{m})$, which takes time $O(m^{k-1}(\sqrt[k+1]{m})^k) = O(m^k/\sqrt[k+1]{m})$. Thus, the total running time is $O(m^k/\sqrt[k+1]{m} + m^k/s(\sqrt[k+1]{m}))$. As $s(m) \leq m$, this is as claimed. The proof for the maximization variant is complete and there are only minor adaptions necessary for minimization. ◀

Step 3: Removing Parallel Edges

After applying the previous steps we can assume that ψ is an $\textsc{OptSP}_k$ formula not containing hyperedges or cross edges. Let $E_1, \dots, E_r$ be the binary relations featured in ψ. We say that ψ does not have *parallel edges* if $r = 1$. In an instance of $\textsc{Opt}(\psi)$ with parallel edges, any pair of vertices (x_i, y) may be connected by up to r parallel edges, or equivalently by an edge of 2^r possible *colors*. We adopt the second viewpoint for this step: Let $\chi(x_i, y) := (E_1(x_i, y), \dots, E_r(x_i, y)) \in \{0,1\}^r$ be the *color* of the edge (x_i, y) and let $\chi(x_1, \dots, x_k, y) := (\chi(x_1, y), \dots, \chi(x_k, y)) \in (\{0,1\}^r)^k$ be the color of the tuple $(x_1, \dots, x_k, y)$.

▶ **Lemma 21.** *Suppose that, for any $\textsc{OptSP}_k$ formula ψ not containing hyperedges, cross edges and parallel edges, $\textsc{Opt}(\psi)$ can be c-approximated in time $O(m^k/s(m))$. Then $\textsc{Opt}(\psi)$ can be c-approximated in time $O(m^k/s(m))$ for any $\textsc{OptSP}_k$ formula ψ not containing hyperedges and cross edges.*

Proof. Let $E_1, \dots, E_r$ denote the binary relations featured in the given instance; our goal is to construct a new instance with only a single edge predicate E. We leave the vertex sets X_i unchanged and construct $Y' = \{y_\alpha : y \in Y, \alpha \in (\{0,1\}^r)^k\}$, i.e., each vertex $y \in Y$ is copied $2^{rk} = O(1)$ times and each copy y_α is indexed by a k-tuple of colors $\alpha = (\alpha_1, \dots, \alpha_k) \in (\{0,1\}^r)^k$. For every α we also introduce a new unary predicate C_α and assign $C_\alpha(y_{\alpha'})$ if and only if $\alpha = \alpha'$.

Now let $i \in [k]$ and let $x_i \in X_i$ and $y \in Y$ be arbitrary vertices in the original instance. We assign the edges in the constructed instance as follows. If $\chi(x_i, y) = 0 = (0, \dots, 0)$, then x_i and y are not connected and we do not introduce new edges. So suppose that $\chi(x_i, y) \neq 0$. Then we add $2 \cdot 2^{r(k-1)}$ edges

- $E(x_i, y_\beta)$, for all $\beta \in (\{0,1\}^r)^k$ with $\beta_i = \chi(x_i, y)$, and
- $E(x_i, y_\gamma)$, for all $\gamma \in (\{0,1\}^r)^k$ with $\gamma_i = 0$.

Clearly the sparsity of the new instance is bounded by $2 \cdot 2^{r(k-1)}m = O(m)$ plus the contribution of the new unary predicates which is also $O(m)$.

Now let $\psi = \mathrm{opt}_{x_1,\dots,x_k} \#_y \phi(x_1, \dots, x_k)$. To define an equivalent $\textsc{MaxSP}_k$ formula ψ', for any $\alpha \in (\{0,1\}^r)^k$ let ϕ_α denote the formula obtained from ϕ by substituting $E_j(x_i, y)$ by *true* if $\alpha_{i,j} = 1$ and by *false* otherwise. We define $\psi' = \max_{x_1,\dots,x_k} \#_y \phi'(x_1, \dots, x_k, y)$,

where $\phi'(x_1, \dots, x_k, y)$ is

$$\bigvee_{\alpha \in (\{0,1\}^r)^k} \Bigg(\underbrace{C_\alpha(y)}_{\text{(i)}} \wedge \underbrace{\Bigg(\bigwedge_{i \in [k]} (E(x_i, y) \iff \alpha_i \neq 0) \Bigg)}_{\text{(ii)}} \wedge \underbrace{\phi_\alpha(x_1, \dots, x_k, y)}_{\text{(iii)}} \Bigg).$$

As desired, the constructed instance contains only a single binary predicate and no cross or hyperedges. It remains to argue that the value of any tuple $(x_1, \dots, x_k)$ is not changed by the reduction. Indeed, for all $y \in Y$ we prove the following two conditions and thereby the claim.

- $\phi'(x_1, \dots, x_k, y_\alpha) = \phi(x_1, \dots, x_k, y)$ for $\alpha = \chi(x_1, \dots, x_k, y)$,
- $\phi'(x_1, \dots, x_k, y_\alpha) = \mathit{false}$ for all $\alpha \neq \chi(x_1, \dots, x_k, y)$.

The first bullet is simple to verify: In the evaluation of $\phi'(x_1, \dots, x_k, y_\alpha)$ we only have to focus on the α-disjunct by the constraint (i). The constraint (ii) is satisfied by our construction of E and therefore only (iii) is decisive: $\phi'(x_1, \dots, x_k, y_\alpha) = \phi_\alpha(x_1, \dots, x_k, y_\alpha) = \phi(x_1, \dots, x_k, y)$. Next, focus on the second bullet. For $\alpha \neq \chi(x_1, \dots, x_k, y)$ there exists some index i such that $\alpha_i \neq \chi(x_i, y)$. By (i), we again only need to consider the α-disjunct. We now prove that $E(x_i, y) \iff \alpha_i = 0$ which falsifies (ii) and shows $\phi'(x_1, \dots, x_k, y_\alpha) = \mathit{false}$. On the one hand, if $\alpha_i \neq 0$ then there is no edge $E(x_i, y_\alpha)$, since $0 \neq \alpha_i \neq \chi(x_i, y)$. On the other hand, if $\alpha_i = 0$ then we added an edge $E(x_i, y_\alpha)$. ◀

Step 4: Removing Unary Predicates

As the final simplification, we eliminate unary predicates and show that the resulting problem can be reduced to the Hybrid Problem.

Proof of Lemma 17. By applying the reductions in Lemmas 18, 20 and 21, it suffices to show that any $\textsc{OptSP}_k$ property ψ not containing hyperpredicates, cross edge predicates and parallel edge predicates can be reduced to the Hybrid Problem. The shape of ψ is significantly restricted and contains only the following three types of relations: Unary predicates on $X_1, \dots, X_k$, unary predicates on Y and binary predicates of the form $E(x_i, y)$ for $i \in [k]$.

We can assume that there are no unary predicates on $X_1, \dots, X_k$ as follows: By enumerating all possible assignments of these unary predicates, and by restricting the sets $X_1, \dots, X_k$ to those vertices matching the current assignment, we create a constant number of instances each without unary predicates on $X_1, \dots, X_k$.

This leaves only unary predicates on Y and the edge predicates $E(x_i, y)$. Let $\psi = \mathrm{opt}_{x_1,\dots,x_k} \#_y \phi(x_1, \dots, x_k, y)$. Another way to view this problem is associate a Boolean function $\phi_y : \{0,1\}^k \to \{0,1\}$ to every vertex $y \in Y$, which takes as input $E(x_i, y)$ and does no longer depend on the unary predicates of y. In that way, we can rewrite the objective as

$$\mathop{\mathrm{opt}}_{x_1,\dots,x_k} \mathop{\#}_{y} \phi_y(E(x_1, y), \dots, E(x_k, y)).$$

Our goal is now to reinterpret this problem as an instance of the Hybrid Problem. As the universe, we assign

$$U = \Big\{ (y, \tau) : y \in Y, \tau \in \{0,1\}^k \text{ is a satisfying assignment of } \phi_y \Big\},$$

along with the partition $U = \bigcup_{\tau \in \{0,1\}^k} U_\tau$, $U_\tau = U \cap (Y \times \{\tau\})$. For every vertex $x_i \in X_i$, we construct a set $S_i \in \mathcal{S}_i$ as $S_i = \{(y, \tau) : E(x_i, y)\} \cap U$. It is easy to check that the value of every solution is preserved in this way: $\mathrm{Val}(S_1, \dots, S_k) = \mathrm{Val}(x_1, \dots, x_k)$. The overhead of this rewriting step is $O(m)$ and thus negligible in the running time bound. ◀