Open Problems from Dagstuhl Seminar 16451:
Structure and Hardness in P

Organizers: Moshe Lewenstein, Seth Pettie, Virginia Vassilevska Williams

The Dagstuhl Seminar Structure and Hardness in P took place November 7-11, 2016. The following open problems were contributed by the seminar attendees, and compiled and edited by the organizers.

Open Problem 1: Parameterizing problems in P by treewidth

Background. Let $t$ be the treewidth of an input graph. Many NP-hard problems, particularly those expressible in MSOL, are solvable in $f(t)n$ time and there are lower bounds on the (exponential) function $f$ conditioned on the Strong Exponential Time Hypothesis (SETH) [26]. For problems in P the picture is less clear. Consider your favorite problem $\Pi$ in P solvable in $T_\Pi(n)$ time on a graph with $n$ vertices. Some problems $\Pi$ admit algorithms running in $\text{poly}(t) \cdot o(T_\Pi(n))$ time whereas others do not. For example, Abboud et al. [5] proved that Diameter can be solved in $2^{O(t \log t)}n^{1+o(1)}$ time, yet a $2^{o(t)n^{2-\epsilon}}$ time algorithm would refute SETH. On the other hand, maximum cardinality matching can be solved in randomized $O(t^3 \cdot n \log n)$-time [31].

Question. Classify graph problems in P according to their dependence on treewidth. Which problems admit $f(t) \cdot n^{1+o(1)}$-time algorithms with polynomial $f$, and which require exponential $f$? A specific goal is the determine whether maximum weight perfect matching has an $\tilde{O}(\text{poly}(t)n)$ algorithm, for integer weights from a polynomial range.

Main paper reference: Abboud et al. [5], Fomin et al. [31]

[Contributed by Fedor V. Fomin.]

Open Problem 2: Approximate all-pairs shortest paths

Background. In unweighted, undirected graphs, we can compute All Pairs Shortest Paths (APSP) in $O(n^3)$ time with a fast “combinatorial” algorithm, or in $O(n^\omega)$ time, where $\omega < 2.373$ is the matrix multiplication exponent. It is conjectured that a truly subcubic combinatorial algorithm does not exist, which is equivalent to the combinatorial Boolean matrix multiplication conjecture.

What about approximation algorithms? The best kind of approximation is an additive $+2$, so that for all pairs $u, v$ we return a value that is between $d(u, v)$ and $d(u, v) + 2$. Dor, Halperin, and Zwick [30] presented a combinatorial algorithm with runtime $\tilde{O}(n^{7/3})$. Note that this runtime is currently even better that $O(n^\omega)$, and has the advantage of being practical.
Questions. Is there a conditional lower bound for 2-APSP? Can we show that a combinatorial algorithm must spend $n^{7/3-o(1)}$ time? Would a faster non-combinatorial algorithm require improvements to $\omega$? Alternatively, is there an $\tilde{O}(n^2)$ time algorithm for 2-APSP?

Main paper reference: Dor, Halperin, and Zwick [30].

Open Problem 3: Approximate diameter

Background. Computing the diameter of a sparse graph in truly subquadratic time refutes SETH: Roditty and Vassilevska Williams [62] showed that a $(3/2-\varepsilon)$-approximation to the diameter requires $n^{2-o(1)}$ time, even on a sparse unweighted undirected graph under SETH. On the other hand, there are algorithms [62, 22] that give a (roughly) $3/2$ approximation in $\tilde{O}(m\sqrt{n})$ time on unweighted graphs, or $\tilde{O}(\min\{m^{3/2}, mn^{2/3}\})$ time on weighted graphs. Extending these algorithms further, Cairo et al. [19] showed that for all integers $k \geq 1$, there is an $\tilde{O}(mn^{1/(k+1)})$ time algorithm that approximates the diameter of an undirected unweighted graph within a factor of (roughly) $2 - 1/2^k$.

Question. If we insist on near-linear runtime, what is the best approximation factor we can get? It is easy to see that a 2-approximation can be achieved in linear time, but what about an $\alpha$-approximation, where $3/2 \leq \alpha < 2$?

Main paper reference: Roditty and Vassilevska W. [62]

Open Problem 4: Finding cycles and approximating the girth

Background. Consider an unweighted undirected graph $G = (V, E)$. The girth of $G$ is the length of the shortest cycle. The problem of detecting 3-cycles (and odd cycles of any length) is reducible to matrix multiplication and there are reductions in the reverse direction; see [66]. Yuster and Zwick [67] showed that detecting $2k$-cycles can be computed in $O(f(k)n^2)$ time, where $f$ is exponential.

Question. For any fixed constant $k$, give a conditional lower bound, showing that there does not exist an algorithm deciding whether $G$ contains a $2k$-cycle in time $O(f(k)n^{2-\varepsilon})$ for any $\varepsilon > 0$, or one running in $O(f(k)m^{2k/(k+1)-\varepsilon})$ time, where $m$ is the number of edges.

Main paper reference: Yuster and Zwick [67]
**Question.** Prove or disprove the following conjecture: There exists a truly subquadratic algorithm for finding a 4-cycle in a graph if and only if there exists a truly subquadratic algorithm for finding a multiplicative $(2 - \varepsilon)$-approximation of the girth.

**Question.** Prove or disprove the following conjecture from [63]: the problem of detecting a 3-cycle in a graph $G$ without 4- and 5-cycles requires $n^{2-o(1)}$ time. Note that if there exists a subquadratic $(2 - \varepsilon)$-approximation for the girth, it must be able to detect 3-cycles in graphs without 4- and 5-cycles. See [63] for more details.

**Main paper reference:** Roditty and Vassilevska W. [63]

[Contributed by Mathias Bæk Tejs Knudsen and Liam Roditty.]

**Open Problem 5: Minimum cycle problem in directed graphs**

**Background.** Given an unweighted directed graph $G = (V, E)$ on $n$ vertices, the problem is to find a shortest cycle in $G$. The potentially simpler Girth problem asks to compute just the length of the shortest cycle.

The girth and the minimum cycle can be computed in $O(n^\omega)$ time exactly, as shown by Itai and Rodeh [49], where $\omega < 2.373$. It is easy to see that the minimum cycle problem is at least as hard as finding a triangle in a graph. In fact, even obtaining a $(2 - \delta)$-approximation for the girth for any constant $\delta > 0$ is at least as hard as triangle detection. The fastest algorithm for the Triangle problem in $n$ node graphs runs in $O(n^\omega)$ time.

**Question.** Is there any $O(1)$-approximation algorithm for the girth that runs faster than $O(n^\omega)$ time? In recent work, Pachocki, Roditty, Sidford, Tov, and Vassilevska Williams [59] showed that for any integer $k$, there is an $\tilde{O}(mn^{1/k})$ time $O(k \log n)$ approximation algorithm for the Minimum Cycle problem. Thus, in nearly linear time, one can obtain an $O(\log^2 n)$-approximation. Can one improve the approximation factor further? Can one even obtain a constant factor approximation in linear time?

**Main paper reference:** Pachocki et al. [59]

[Contributed by Virginia Vassilevska Williams.]

**Open Problem 6: Linear Programming**

**Background.** Consider a linear program of the following form: minimize $c^T x$ subject to $Ax \geq b$, where $A$ is an $d$-by-$n$ constraint matrix. Suppose that we could solve any such LP in time

$$\tilde{O} \left( (\text{nnz}(A) + d^2) d^\delta \log L \right),$$

where nnz$(A)$ is the number of non-zero entries of $A$, $L$ is the bound on the bit complexity of the input entries, and $\delta$ is a positive constant.
**Question.** Is there some value of $\delta$ for which the above (hypothetical) running time bound would disprove any of the popular hardness conjectures?

In [57], it is shown that one can achieve the above running time bound for $\delta = \frac{1}{2}$.

**Main paper reference:** Lee and Sidford [57] 
[Contributed by Aleksander Madry.]

---

**Open Problem 7: Fully dynamic APSP**

**Background.** In the fully dynamic all-pairs shortest paths (APSP) problem we are interested in maintaining the distance matrix of a graph under insertions and deletions of nodes. Demetrescu and Italiano [28] showed that the distance matrix can be updated in amortized time $\tilde{O}(n^2)$ after each node update. The current fastest worst case algorithms have update times of $\tilde{O}(n^{2+2/3})$ (randomized Monte Carlo [7]) and $\tilde{O}(n^{2+3/4})$ (deterministic [65]).

**Questions.** Can the worst case update time $\tilde{O}(n^2)$ be achieved? A barrier for current algorithmic approaches is $n^{2.5}$. Is there a conditional lower bound showing this to be a true barrier?

**Main paper reference:** Abraham et al. [7] 
[Contributed by Sebastian Krinninger.]

---

**Open Problem 8: Dynamic reachability in planar graphs**

**Background.** Dynamic reachability in a planar graph $G$ is the problem of maintaining a data structure supporting the following operations: (i) Insert a directed edge $(u, v)$ into $G$, (ii) delete an edge from $G$, and (iii) query whether $v$ is reachable from $u$ in $G$.

An algorithm with update and query time $\tilde{O}(\sqrt{n})$ is known (Diks and Sankowski [29]) for dynamic plane graphs—that is, the graph is dynamic but the plane embedding is fixed.

**Question.** Does an $n^{1/2-\Omega(1)}$ algorithm exist or is there a conditional $n^{1/2-o(1)}$ hardness result? Any polynomial hardness result would be interesting. A good place to start for the latter part would be the recent paper by Abboud and Dahlgaard [3] about hardness for dynamic problems in planar graphs.

**Main paper reference:** Abboud and Dahlgaard [3] 
[Contributed by Søren Dahlgaard.]
Open Problem 9: Static hardness for planar graphs

**Background.** An important direction is to show conditional hardness for important problems, even on restricted (easier) classes of graphs, e.g., planar graphs. Abboud and Dahlgaard [3] recently showed hardness for several dynamic problems in planar graphs, but nothing is known for static problems.

**Question.** On planar graphs, many problems (such as shortest paths, multi-source multi-sink max-flow, etc.) run in near-linear time. Can we show that some problem does not? No hardness results are known for any static problem in \( \mathbf{P} \) on planar graphs. Two candidate problems to consider are diameter and sum of distances. Both require subquadratic time (Cabello [18]), but it may still be possible to show a hardness result, e.g., \( n^{3/2-o(1)} \) hardness.

**Main paper reference:** Cabello [18]

[Contributed by Søren Dahlgaard.]

Open Problem 10: Sparse reductions for graph problems

**Background.** Many graph problems are known to be as hard as APSP on dense graphs [66, 4, 64], in the sense that a subcubic algorithm for any of them implies a subcubic algorithm for all of them. When the graph sparsity is taken into account, these problems currently are no longer in a single class: many have \( \tilde{O}(mn) \)-time algorithms whereas finding minimum weight triangle and related problems have \( \tilde{O}(m^{3/2}) \)-time algorithms. Most known fine-grained reductions between graph problems do not preserve the graph sparsity. Until recently, the only examples of sparseness preserving truly subcubic reductions appeared in [4]. Agarwal and Ramachandran [8] presented several more such reductions, strengthening the connections between problems with \( \tilde{O}(mn) \)-time algorithms.

**Questions.** Is there a sparseness-preserving, \( \tilde{O}(n^2) \) time reduction from undirected weighted All Nodes Shortest Cycles (ANSC) to APSP? Is there a sparseness-preserving, \( \tilde{O}(m + n) \) time reduction from undirected Min-Wt-Cycle to either Radius or Eccentricities? Is it SETH-hard to find a sub-\( mn \) bound for Min-Wt-Cycle or an \( O(n^2 + \text{sub-}mn) \) bound on APSP? Note that the known SETH-hardness results for Diameter and Eccentricities [62] do not apply to APSP, as they address \( O(n^{2-\epsilon}) \) time algorithms in \( \tilde{O}(n) \) node graphs, whereas just the output of APSP is of size \( n^2 \).

**Main paper reference:** Agarwal and Ramachandran [8]

[Contributed by Vijaya Ramachandran.]
Open Problem 11: Hardness for partially dynamic graph problems

**Background.** Many results show hardness for fully-dynamic problems in graphs, but the techniques do not seem to extend well to amortized lower bounds in the incremental and decremental cases. (See Abboud and Vassilevska Williams [2], Henzinger, Krinninger, Nanongkai, and Sarannrak [46], Kopelowitz, Pettie and Porat [55], and Dahlgaard [27] for some initial results on incremental/decremental problems.)

**Question.** Develop general techniques for showing amortized hardness of partially dynamic problems in graphs. One candidate problem is decremental single-source reachability. A result of Chechik, Hansen, Italiano, Lacki, and Parotsidis [23] shows that $\tilde{O}(m\sqrt{n})$ total time is sufficient. Is it necessary?

[Contributed by Søren Dahlgaard.]

Open Problem 12: Hardness of vertex connectivity

**Background.** A connected undirected graph is $k$-vertex (resp. edge) connected if it remains connected after any set of at most $k - 1$ vertices (edges) is removed from the graph. A strongly connected directed graph is $k$-vertex (edge) connected if it remains strongly connected after any set of at most $k - 1$ vertices (edges) is removed from the graph. The vertex (edge) connectivity of a graph is the maximum value of $k$ such that the graph is $k$-vertex (edge) connected.

The edge-connectivity $\lambda$ of an undirected graph can be determined in time $O(m \log^2 n \log \log n)$ [47, 54], and for directed graphs in time $O(\lambda m \log(n^2/m))$ [37]. In contrast, the vertex-connectivity $\kappa$ can only be computed in time $O((n + \min\{\kappa^{5/2}, \kappa n^{3/4}\})m)$ [38], where for undirected graphs $m$ can be replaced by $kn$.

**Question.** To check $k$-vertex connectivity means to either confirm that $\kappa \geq k$ or to find a set of $k - 1$ vertices that disconnects the graph. Even when $k$ is constant, no $o(n^2)$ time (or $o(mn)$ time for directed graphs) algorithms are known for checking $k$-connectivity. Is there a conditional superlinear lower bound?

**Main paper reference:** Gabow [38]

[Contributed by Veronika Loitzenbauer.]
Open Problem 13: Parity and mean-payoff games

**Background.** Parity games, and their generalization mean-payoff games, are among the rare “natural” problems in $\text{NP} \cap \text{co-NP}$ (and in $\text{UP} \cap \text{co-UP}$ [52]) for which no polynomial-time algorithm is known. Both parity games and mean-payoff games are 2-player games played by taking an infinite walk on a directed graph; one of the vertices is designated the start vertex. In parity games each vertex is labeled by an integer in $[0,c]$; in mean payoff games each edge is labeled by an integer in $[-W,W]$. (See [53] for a description of the game.) The algorithmic question is to decide, for each start vertex, which of the two players wins the game and to construct a corresponding winning strategy. Parity games can be reduced to mean-payoff games with $W = n^c$. Very recently, quasi-polynomial $O(n^{\log c})$ time algorithms for parity games were discovered [20, 51]. The best known algorithms for mean-payoff games run in pseudo-polynomial time $O(mnW)$ [16] and randomized sub-exponential time $O(2^{\sqrt{n \log n}} \log W)$ [15].

**Questions.** Is there a polynomial-time algorithm for parity or mean-payoff games? Are there conditional superlinear lower bounds on these problems?

[Contributed by Veronika Loitzenbauer.]

Open Problem 14: Unknotting

**Background.** A knot is a closed, non-self-intersecting polygonal chain in $\mathbb{R}^3$. Two knots are equivalent if one can be continuously deformed into the other without self-intersection. The unknot problem is to decide if a knot is equivalent to one that is embeddable in the plane.

Knots can be represented combinatorially, by projecting the polygonal chain onto $\mathbb{R}^2$, placing a vertex wherever two edges intersect. The result is a 4-regular planar graph (possibly with loops and parallel edges) where each vertex carries a bit indicating which pair of edges is “over” and which pair is “under.” Reidemeister moves (a small set of transformations on the knot diagram) suffice to transform any knot diagram to one of its equivalent representations.

The complexity of unknot and related problems (e.g., are two knots equivalent?, can two knots simultaneously embedded in $\mathbb{R}^3$ be untangled?) are known to be in $\text{NP}$ [45] and solvable in $2^{O(n)}$ time [45, 50].

**Questions.** Given a plane knot diagram with $n$ intersections, can unknot or knot-equivalence be solved in time near-linear in $n$? If not, are there conditional lower bounds that show even some polynomial hardness?

[Contributed by Seth Pettie.]
Open Problem 15: 3-Collinearity (general position testing)

**Background.** A set $S$ of $n$ points in $\mathbb{R}^2$ is said to be in *general position* if there do not exist three points in $S$ that lie on a line. The 3-Collinearity problem is to test whether $S$ is in general position. The 3-Collinearity problem is known to be as hard as 3SUM, and an algorithm that runs in $O(n^2)$ time is known.

**Questions.** The question is whether the $O(n^2)$ algorithm is optimal or whether it can be solved in $o(n^2)$ time. Recent subquadratic algorithms for 3SUM [12, 44, 34, 41] indicate that polylogarithmic improvements should be possible. A related question is whether there is an $O(n^{2-\epsilon})$-depth decision tree for 3-Collinearity; see [44, 13].

**Main paper reference:** Gajentaan and Overmars [39]

[Contributed by Omer Gold.]

Open Problem 16: Element uniqueness in $X + Y$

**Background.** Given two sets $X$ and $Y$, each of $n$ real numbers, determine whether all the elements of $X + Y = \{x + y \mid x \in X, y \in Y\}$ are distinct. A somewhat stronger variant of this problem is to sort $X + Y$.

The decision tree complexity of sorting $X + Y$ and Element Uniqueness in $X + Y$ was shown to be $O(n^2)$ by Fredman [33].

**Question.** Can these problems can be solved in $o(n^2 \log n)$ time, even for the special case $X = Y$?

[Contributed by Omer Gold.]

Open Problem 17: Histogram indexing

**Background.** The histogram $\psi(T)$ of a string $T \in \Sigma^*$ is a $|\Sigma|$-length vector containing the number of occurrences of each letter in $T$. The histogram indexing problem (aka jumbled indexing) is to preprocess a string $T$ to support the following query: given a histogram vector $\psi$, decide whether there is a substring $T'$ of $T$ such that $\psi(T') = \psi$.

The state-of-the-art algorithm for histogram indexing [21] preprocesses a binary text $T$ in $O(n^{1.859})$ time and answers queries in $O(1)$ time. Over a $d$-letter alphabet the preprocessing and query times are $\tilde{O}(n^{2-\delta})$ and $\tilde{O}(n^{2/3+\delta(d+13)/6})$, for any $\delta \geq 0$. On the lower bound side [10, 42], the 3SUM conjecture implies that it is impossible to simultaneously improve $n^{2-\delta}$ preprocessing and $n^{\delta(d/2-1)}$ query time by polynomial factors, where $\delta \leq 2/(d-1)$ and $d \geq 3$.

**Question.** Are there any non-trivial lower bounds on histogram indexing when $d = 2$? Is it possible to close the gap between the lower and upper bounds in general, or to base the hardness off of a different conjecture than 3SUM?
Main paper reference: Chan and Lewenstein [21]

[Contributed by Isaac Goldstein.]

Open Problem 18: Integer programming

Background. The objective of Integer Programming (IP) is to decide, for a given $m \times n$ matrix $A$ and an $m$-vector $b = (b_1, \ldots, b_m)$, whether there is a non-negative integer $n$-vector $x$ such that $Ax = b$. In 1981, Papadimitriou [61] showed that (IP) is solvable in pseudo-polynomial time on instances for which the number of constraints $m$ is constant. The rough estimation of the running time of Papadimitriou’s algorithm is $n^{O(m)} \cdot d^{O(m^2)}$, where $d$ bounds the magnitude of any entry in $A$ and $b$. The best known lower bound is $n^{o(\log m)} \cdot d^{O(m)}$ [32], assuming the Exponential Time Hypothesis (ETH).

Question. Is it possible to narrow the gap between algorithms for IP and the ETH-hardness of IP?

Main paper reference: Fomin et al. [32]

[Contributed by Fedor V. Fomin.]

Open Problem 19: All-pairs min-cut and generalizations

Background. The all-pairs min-cut problem is, given an edge-capacitated undirected graph $G = (V, E, c)$, to compute the minimum $s$-$t$ cut over all pairs $s, t \in V$. Gomory and Hu [43] showed the problem is reducible to $n - 1$ $s$-$t$ min-cut instances, and moreover, all $\binom{n}{2}$ min-cuts can be represented by a capacitated tree $T$ on the vertex set $V$. On unweighted graphs, the construction of $T$ takes time $\tilde{O}(mn)$ [14, 60].

Generalizations of this problem include finding the min-cut separating every triple $(r, s, t) \in V^3$, which is NP-hard, and finding the min-cuts separating all pairs of $k$-sets $\{s_1, \ldots, s_k\}$ from $\{t_1, \ldots, t_k\}$. See [24].

Questions. Are there superlinear conditional lower bounds for all-pairs min-cut/Gomory-Hu tree construction? (Refer to [6] for conditional lower bounds for variants of the problem on directed graphs.) Are there non-trivial conditional lower bounds for all-triplets approximate min-cut, or all-$k$-sets min-cut?

[Contributed by Robert Krauthgamer.]
Open Problem 20: Parameterizing string algorithms by compressibility

**Background.** The broad idea can be illustrated with a lower bound for string edit distance: Given two strings of length \( N \) whose compressed length (say, using Lempel-Ziv compression) is \( n \), it is known that their edit distance can be computed in \( O(n^p N^q) \) time. Is it possible to prove an \( \Omega(n^p N^q) \) conditional lower bound? The known conditional lower bound [11, 1, 17] reduces CNF-SAT (with \( n \) variables) to string edit distance by creating two strings each consisting of \( O(2^{n/2}) \) blocks. To make such a reduction suitable for proving \( \Omega(n^p N^q) \) lower bound, one needs to generate instead two strings whose length is much more than \( 2^{n/2} \) but that compress to much less than \( 2^{n/2} \).

[Contributed by Oren Weimann.]

Open Problem 21: Reductions from low complexity to high complexity

**Background.** We know that improving the runtime of our 10-Clique algorithms improves the runtime of our 100-Clique algorithms. E.g., if 10-Clique can be solved in \( O(n^5) \), then 100-Clique can be solved in \( O(n^{50}) \). In general, we have many examples of reductions showing that a faster algorithm for a problem with best known runtime \( O(n^a) \), implies a faster algorithm for a problem with runtime \( O(n^b) \), where \( a \leq b \).

However, we have no interesting reductions in the other way, showing that improvements over \( n^b \) imply improvements over \( n^a \), where \( a < b \). In particular, we do not know how to use an algorithm that solved 100-Clique in \( O(n^{50}) \) or even \( O(n^{11}) \) time, to speed up the known algorithms for 10-Clique.

Could it be that such reductions, from low complexity to high complexity, do not exist? It is not hard to construct artificial problems where this can be done, but what about the natural problems we typically study: Clique, Orthogonal Vectors, \( k \)-SUM, APSP, LCS, etc. Can we show that a fine-grained reduction from 10-Clique to 100-Clique is unlikely due to some surprising consequences? Another candidate is 3SUM (for which the complexity is \( n^2 \)) vs. APSP (for which the complexity is \( N^{1.5} \), where \( N \) is the input size). We repeatedly ask if faster 3SUM implies faster APSP, but maybe proving such a result (via fine-grained reductions) has unexpected consequences?

On the other hand, it would be of great interest to find examples of such reductions between interesting and natural problems.

[Contributed by Amir Abboud.]

Open Problem 22: Stable matching in the two-list model

**Background.** Gale and Shapley’s stable matching [40] algorithm runs in \( O(n^2) \) time (linear in the input size) and it is known that \( \Omega(n^2) \) is optimal if the preference lists are arbitrary. Künemann, Moeller, Paturi, and Schneider [56] studied the complexity of stable matching when the preference lists are constrained, and encoded in some succinct manner. Many succinct input models nonetheless require \( n^{2-o(1)} \) time, conditioned on SETH.
Question. A problem left open by [56] is two-list stable matching. A matching market in the two-list model consists of two sets $M$ and $W$, both of size $n$, and permutations $\pi_1, \pi_2$ on $M$ and $\sigma_1, \sigma_2$ on $W$. The preference list of each agent $m \in M$ is either $\sigma_1$ or $\sigma_2$ and the preference list of each agent $w \in W$ is either $\pi_1$ or $\pi_2$. The input size is $O(n)$. The goal is to find a stable matching in the resulting matching market. Can this problem be solved in linear time, or is there a superlinear conditional lower bound?

Main paper reference: Kuennemann et al. [56]

Open Problem 23: Boolean vs. real maximum inner product

Background. In the maximum inner product problem we are given two sets of $d$-dimensional vectors $U$ and $V$ of size $n$ as well as a threshold $l$. The problem is to decide if there is a pair $u \in U, v \in V$ such that their inner product $u \cdot v$ is at least $l$. If the vectors are Boolean, then a randomized algorithm by Alman and Williams [9] solves the problem in time $n^{2-1/\Theta(c \log^2 c)}$ where $d = c \log n$. In contrast, if the vectors are real or integer, then using ray-shooting techniques [58] we can solve the problem in time $n^{2-1/\Theta(d)}$. This leaves a large gap between the two problems. In particular, the Boolean case is strongly subquadratic if $d = O(\log n)$, while the real case is only strongly subquadratic for constant $d$. The conditional lower bounds of [9] show that any $n^{2-\epsilon}$ algorithm when $d = \omega(\log n)$ refutes SETH.

Questions. Can the gap between the boolean and integer/real case be closed, with a better maximum inner product algorithm? If the gap is natural, can it be explained with a stronger conditional lower bound on (real or integer) maximum inner product?

[Contributed by Stefan Schneider.]

Open Problem 24: Hardness of Approximating NP-hard Problems

Background. Many approximation algorithms for NP-hard problems run in polynomial time, but not linear time. This is often due to the use of general LP or SDP solvers, but not always. To take two examples, the chromatic index (edge coloring) and minimum degree spanning tree problems are NP-hard, but can both be approximated to within 1 of optimal in $\tilde{O}(m\sqrt{n})$ time [36] and $\tilde{O}(mn)$ time [35], respectively.

Question. Prove superlinear conditional lower bounds on the time complexity of any approximation problem, whose exact version is NP-hard.

[Contributed by Seth Pettie.]
Open Problem 25: Chromatic index/edge coloring

**Background.** The chromatic index of a graph is the least number of colors needed for a proper edge-coloring. Vizing’s theorem implies that the chromatic index is either $\Delta$ or $\Delta + 1$ (where $\Delta$ is the maximum degree), but determining which one is NP-hard. The NP-hardness reduction of Holyer [48] reduces 3SAT to a 3-regular graph on $O(n)$ vertices, so the ETH implies a $2^{\Omega(n)}$ lower bound. There is an $O^*(2^m)$ algorithm for chromatic index, by reduction to vertex coloring, so the hardness is well understood when $m = O(n)$.

**Questions.** Does the ETH rule out a $2^{o(m)}$ algorithm for chromatic index on dense graphs? Is there, for example, an $n^{O(n)}$ or $2^{n^{2-\epsilon}}$-time algorithm?

[Contributed by Marek Cygan.]

Open Problem 26: Communication Complexity of Approximate Hamming Distance

**Background.** Consider strings $P$ of length $n$ and $T$ of length $2n$. Alice has the whole of $P$ and the first half of $T$. That is she has $P$ and $T[0, \ldots, n-1]$. Bob has the second half of $T$, that is $T[n, \ldots, 2n-1]$. Alice sends one message to Bob and Bob has to output a $(1 + \epsilon)$ multiplicative approximation of $\text{HD}(P, T[i, \ldots, i + n])$ for all $i \in [n]$ where $\text{HD}$ is the Hamming Distance.

In [25] a $O(\sqrt{n}\log n/\epsilon^2)$ bit communication protocol was given.

**Question.** Is there a matching lower bound for the randomized one-way communication complexity of this problem?

**Main paper reference.** Clifford and Starikovskaya [25].

[Contributed by Raphaël Clifford.]

**References**


