Exact Exponential Algorithms for Coloring

Arpon Basu

Indian Institute of Technology, Bombay

November 26, 2023

Introduction

We will talk about **exact exponential algorithms** (EEA): Consider a NP-complete problem $\mathcal{P}$, for which there exists a c^n time solution, where n is the input size.
We want to minimize c.

Motivation and Background

- If $\mathcal{P}$ is a NP-complete problem with a linear-sized reduction to SAT, then under *Exponential Time Hypothesis*, we can't do better than c^n for some $c > 1$.
- Small c might actually make algorithm feasible for moderately large input sizes! For example, $c^n \leq n^3$ for $n \leq 100$ if $c \leq 1.14$.
- See [3] for further details.

EEA for Coloring: Literature Review

We'll see exact exponential algorithms for coloring in this presentation.

- **Lawler's Algorithm** [4]: k-coloring can be decided in $(1 + \sqrt[3]{3})^n$ time, where n is the number of vertices of the graph.

Brief sketch of algorithm: Note that

$$\chi(N) = 1 + \min_{\substack{S \subseteq N \\ S \text{ independent}}} \chi(N \setminus S)$$

We now use fact that a r-vertex graph can have atmost $\sqrt[3]{3}^r$ maximal independent sets. Thus runtime of algorithm $\sim \sum_{r=0}^{n} \binom{n}{r} \sqrt[3]{3}^r = (1 + \sqrt[3]{3})^n$, as desired.

Literature Review Contd.

Björklund, Husfeldt, and Koivisto [2] showed that k-colorability can be decided in 2^n time, using Yates' algorithm, which we shall see later.
Open Question: Can k-colorability be decided faster than 2^n?
Beigel and Eppstein [1] showed that 3-colorability and 4-colorability can be decided in 1.3289^n and 1.8072^n time respectively.
We shall see this later.

Zamir's Algorithm

The decidability of k-colorability for $k \geq 5$, in time c^n, for $c < 2$ was open for a long time.
Zamir finally resolved it (partially) in 2021 [5], when he showed that 5-colorability can be decided faster than 2^n time, and 6-colorability can also be decided by a randomized algorithm faster than 2^n time.
Problem still open for $k \geq 7$.
We shall explore Zamir's algorithm, and Beigel and Eppstein's algorithm in this presentation, and see some standard tools in the field of EEAs en route.

Basic Skeleton of Zamir's Argument

- Graphs are divided into 2 categories: (α, Δ) bounded graphs, and graphs which are not (α, Δ) bounded.
- A graph is called (α, Δ) bounded, if $\geq \alpha$ fraction of vertices have degree $\leq \Delta$.

These two classes of graphs are then dealt with separately.

Unbounded Graphs

- If G is **not** (α, Δ) bounded, then it is "very dense". In particular, G has a small *dominating set*.
- A set $S \subseteq V(G)$ is called dominating if for every $v \in V(G) \setminus S$, v is adjacent to some vertex in S.
- In other words, one BFS iteration from a dominating set covers the whole graph.
- A small dominating set allows us to brute-force over all colorings in faster than 2^n time.
- **This brute force requires $(k-1)$-list colorability results**.
- More details later.

Bounded Graphs and concluding the argument

- For bounded graphs, we use **Yates' algorithm**, alluded to earlier.
- The k-colorability of bounded graphs follows **without** any dependence on results about $(k-1)$-list colorability.
- Finally, we choose the parameters α, Δ suitably, so that both the algorithms, for the bounded and the unbounded case, terminate faster than 2^n time.

Colorability of Unbounded Graphs

We first deal with unbounded graphs, for they are easier to reason about.

Small Dominating Sets

Theorem

Let G be a graph such that there exists a subset $V' \subseteq V(G)$, $|V'| \geq (1-\alpha)|V(G)|$, where for every $v \in V'$ we have $\deg(v) \geq \Delta - 1$. Then G has a dominating set R of size atmost $((1-\alpha) \cdot \frac{1+\ln(1+\Delta)}{1+\Delta} + \alpha) \cdot |V(G)|$. Furthermore, R can be found in deterministic polynomial time.

Proof Sketch.

Let R_0 be a random subset of $V(G)$, where each vertex is chosen with probability p. Let $R_1 \subseteq V' \setminus R_0$ be the set of vertices which don't have any neighbor in R_0. Let $R_2 \subseteq (V(G) \setminus V') \setminus R_0$ be the set of vertices which don't have any neighbor in R_0.
Then $R := R_0 \cup R_1 \cup R_2$ is a dominating set. Also, □

Proof.

$$\mathbb{E}[|R|] = \mathbb{E}[|R_0|] + \mathbb{E}[|R_1|] + \mathbb{E}[|R_2|]$$
$$\leq p \cdot |V(G)| + (1-p)^{\Delta} \cdot |V'| + (1-p)^{\delta(G)+1} \cdot |V'|$$
$$\leq \left(p + (1-\alpha) \cdot (1-p)^{\Delta} + \alpha \cdot (1-p)^2\right) \cdot |V(G)|$$

Putting $p = \frac{\ln \Delta}{\Delta}$ in the above expression yields the desired result. □

Note that even though the dominating set was constructed randomly, it can be derandomized completely through the method of conditional expectations.

Brute Force Coloring

- Suppose we are given a k-coloring c of a dominating set R. Then for every vertex $v \in V(G) \setminus R$, the list of available colors at v is $\text{list}_c(v) := [k] \setminus \{c(r) : r \in R,\ r \sim v\}$.
- Thus, checking if c can be extended to $V(G)$ is equivalent to solving a $(k-1)$-list coloring problem on $V(G) \setminus R$.
- Thus, checking if G is k-colorable or not takes time $k^{|R|} \cdot$ (time taken to check $(k-1)-$ list colorability of $V(G) \setminus R$)
- Along with list colorability results of Beigel-Eppstein, the constants in the above theorem ensure that checking 5-colorability of dense graphs can be done faster than 2^n.

Coloring Bounded Graphs: Toolkit

Before we present results on how to color (α, Δ)-bounded graphs, we first introduce **Yates' algorithm**, as promised earlier.

Definition (Zeta Transform)

Consider a function $f : 2^{[n]} \mapsto \mathbb{R}$. We define it's "zeta-transform" $\widehat{f} : 2^{[n]} \mapsto \mathbb{R}$ to be the function defined as

$$\widehat{f}(X) := \sum_{Y \subseteq X} f(Y)$$

for every $X \subseteq [n]$.

Yates' Algorithm

Note that a naïve way of computing the zeta transform takes $\sum_{r=0}^{n} \binom{n}{r} 2^r = 3^n$ time. However, in Yates' algorithm, we use *dynamic programming* to reduce the time taken to 2^n. Indeed, we define the functions $f = f_0, f_1, \ldots, f_n = \widehat{f}$, where

$$f_i(X) := \begin{cases} f_{i-1}(X) + f_{i-1}(X \setminus i), \text{if } i \in X \\ f_{i-1}(X), \text{otherwise} \end{cases}$$

One can show by induction that $f_i(X) := \sum_{Y \in S_i(X)} f(Y)$, where $S_i(X) := \{Y \subseteq X : \{j \in Y : j > i\} = \{j \in X : j > i\}\}$. Clearly, $f_n = \widehat{f}$. Furthermore, (assuming we store the functions $f_1, \ldots, f_{n-1}$ in memory), $f_n(X)$ can be calculated in polynomial time through the recursion above, and thus calculating $\widehat{f}$ takes 2^n time.

Connecting Yates' to Independent Sets

- Write $i(H)$ to be the number of independent sets in some graph H.
- Observe that $i(G[V']) = i(G[V' \setminus \{v\}]) + i(G[V' \setminus N[v]])$, where $N[v] := N(v) \cup \{v\}$.
- Thus, exactly as in Yates', we can calculate $i(G[V'])$ for all $V' \subseteq V(G)$ in 2^n time.

Inclusion-Exclusion Principle

- Note that a graph is k-colorable iff it can be covered by k independent sets.
- Thus, a graph is k-colorable if $F(G) > 0$, where

$$F(G) := \left| \left\{ (I_0, \ldots, I_{k-1}) : I_*\text{'s are independent}, \bigcup_{i=0}^{k-1} I_i = V(G) \right\} \right|$$

- Note that $F(G)$ is computable in 2^n time.

Theorem (Inclusion-Exclusion)

$$F(G) = \sum_{V' \subseteq V(G)} (-1)^{|V(G)|-|V'|} \cdot i(G[V'])^k$$

Proof.

$i(G[V'])^k$ counts the number of k-tuples of independent sets contained in $G[V']$. Now, note that a particular tuple $(I_0, \ldots, I_{k-1})$ occurs in the above summation for all supersets of I, where $I = \bigcup_{i=0}^{k-1} I_i$.
It is easy to see that if $I \neq V(G)$, then the minus signs cancel of the contribution of that tuple, and the result follows. □

So far

- The argument described above, calculates $F(G)$ in 2^n time, and thus determines if G is k-colorable. (This is exactly the argument in [2]). Note that we haven't used (α, Δ)-boundedness so far.
- Note that if graph is (α, Δ)-bounded, then it has an independent set of size $\geq \frac{\alpha n}{1+\Delta}$ (standard greedy argument).
- Thus, let S be a large independent set in our graph.
- Let c be a coloring of $V(G) \setminus S$. c is extendable to S iff $|c(N(s))| < k$ for every $s \in S$.

So far (Contd.)

- We then apply a variant of the inclusion-exclusion argument outlined above to $V(G) \setminus S$ to check if some particular coloring of $V(G) \setminus S$ is extendable to S.
- Managing constants appropriately, checking colorability of G can be done faster than 2^n.
- The essence of the proof is similar to the proof in [2]. The inclusion-exclusion predicate though, is considerably more complicated here.
- Check out my thesis, or Zamir's paper [5] for further details.

Interlude

- We have thus seen a very brief sketch of Zamir's proof that 5-colorability can be decided faster than 2^n time.
- The proof *crucially* relies on the fact that 4-list colorability can be decided faster than 2^n. Recall that this dependence arises in the "dense", unbounded régime [1].
- We shall thus spend some time analyzing Beigel-Eppstein's 4-list colorability algorithm.

[1] It is not surprising that deciding chromatic number of (α, Δ)-unbounded graphs should be harder than their sparser counterparts: We have lots of edges, but not enough to rule out low chromatic numbers either.

Work Factor: Master theorem for EEAs

Consider the recursion

$$T(n) = T(n - r_1) + T(n - r_2) + \ldots + T(n - r_\ell) + \text{poly}(n)$$

The solution to this recursion is $\mathcal{O}(\lambda^n \cdot \text{poly}(n)) = \mathcal{O}^*(\lambda^n)$, where $\lambda = \lambda(r_1, \ldots, r_\ell)$ is known as the *work factor* of the algorithm. λ is in fact the smallest positive root of the equation $\sum x^{-r_i} = 1$.
The above connection implies $r_1 \geq r'_1, \ldots, r_\ell \geq r'_\ell$, then $\lambda(r_1, \ldots, r_\ell) \leq \lambda(r'_1, \ldots, r'_\ell)$.

Definition ((a, b)-Constraint Satisfaction Problems)

Suppose we have n variables $x_1, \ldots, x_n$, each of which can be assigned a color from the *set of colors* $\mathcal{C}$, where $|\mathcal{C}| = a$. We also have $m = \text{poly}(n)$ constraints, where each constraint involves $r \leq b$ variables, say $x_{\ell_1}, \ldots, x_{\ell_r}$, and that constraint dictates that $(x_{\ell_1}, \ldots, x_{\ell_r}) \neq (c_1, \ldots, c_r)$, for some $c_1, \ldots, c_r \in \mathcal{C}$.
The constraint satisfaction problem (CSP) then asks if all of these constraints can be satisfied simultaneously.

An Alternative Viewpoint of $(a, 2)$-CSPs

- We present a "graphical" presentation of $(a, 2)$-CSPs.
- Any constraint of size 2 in a $(a, 2)$-CSP is of the form $\{(x_i, c_i), (x_j, c_j)\}$, where $c_i, c_j \in \mathcal{C}$.
- Consequently, we can create a graph with n vertices, with the i^{th} vertex representing a "bag of colors" corresponding to the possible assignments of x_i.
- To represent the constraint $\{(x_i, c_i), (x_j, c_j)\}$, we construct an edge between the color c_i in the i^{th} vertex and the color c_j in the j^{th} vertex.

Preliminaries

- We can express k-list-colorability as a $(k, 2)$-CSP.
- We will thus focus on $(3, 2)$, and $(4, 2)$-CSPs.
- In fact, we will "reduce" $(4, 2)$-CSPs to $(3, 2)$-CSPs, i.e. a $(4, 2)$-CSP of input size n will become a $(3, 2)$-CSP of input size $n' \geq n$.
- Thus, WLOG lets focus on $(3, 2)$-CSPs.

Preliminaries (Contd.)

We first "pre-process" our CSP so that it becomes amenable to later analysis. Some examples of this pre-processing are as follows:

Lemma

Let v be a variable in an $(a, 2)$-CSP, and suppose only two of the a colors are allowed at v. Then we can obtain an equivalent $(a, 2)$-CSP with one fewer variable.

Lemma

Consider a $(3, 2)$-CSP instance, and let (v, d) be a variable-color pair such that there exists another variable w for which we have all 3 constraints $\{(v, d), (w, c)\}, \{(v, d), (w, d)\}, \{(v, d), (w, e)\}$, where the color set is $\{c, d, e\}$. Then we can find an equivalent $(3, 2)$-CSP with one fewer variable.

The proofs are routine, so we skip them. Now, before we jump into Beigel-Eppstein's algorithm for deciding $(3,2)$-CSPs, lets see a randomized algorithm for the same.

Randomized Algorithm to decide $(3, 2)$-CSPs

Theorem

Given a satisfiable $(3, 2)$*-CSP instance, there exists a randomized algorithm that finds the solution to the instance in expected time* $\mathcal{O}^*\left(\sqrt{2}^n\right)$.

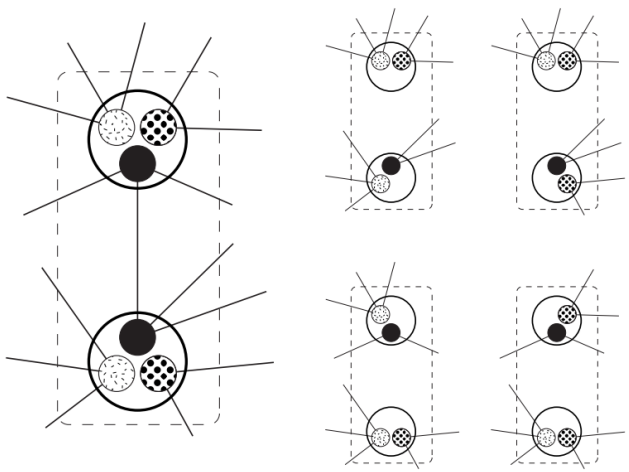

Figure: Reduction of a $(3, 2)$-CSP

Proof.

Consider any constraint of the form $\{(x_i, c_i), (x_j, c_j)\}$. WLOG assume $c_i = c_j$, as shown in Fig. 1. Note that any valid coloring of the two vertices shown in the figure is retained in exactly two of the four small configurations drawn on the right-hand side of Fig. 1. □

Proof.

Thus, our randomized algorithm is as follows: Given a $(3, 2)$-CSP instance, pick any constraint (which consists of two variables, both of which are 3-color), and *randomly* choose one of the 4 reduced configurations as depicted in Fig. 1. In the reduced configuration, both variables become 2-color and thus can be eliminated by Theorem 5. Thus, in every reduction, we can remove two variables. Furthermore, the probability that satisfiability is maintained in any reduction is $\geq \frac{1}{2}$. Consequently, after $n/2$ reductions, with probability $2^{-n/2}$ we'll be left with a satisfying assignment.

Thus, if we're given a satisfiable $(3, 2)$-instance, we will need to carry out the above reduction process $2^{n/2}$ times to find a satisfying assignment. □

Although this randomized algorithm is quite slick, it is possible to do even better deterministically. We shall see the deterministic algorithm now.

Conversion of $(4, 2)$-CSPs to $(3, 2)$-CSPs

We shall now not only elucidate a deterministic algorithm for $(3, 2)$-CSPs, but also show how a $(4, 2)$-CSP instance can be reduced to a $(3, 2)$-CSP instance.
We first estimate the "size" of a $(4, 2)$-CSPs.
Firstly, let our $(4, 2)$-CSP instance have n_i i-color variables, where $i \in \{3, 4\}$. In order to obtain a fast algorithm, we shall perform a book-keeping trick: We shall instead declare the "size" of a $(4, 2)$-CSP instance to be $n = n_3 + (2 - \varepsilon)n_4$, where $\varepsilon < \frac{1}{2}$ is a small constant we shall optimize to obtain a low work factor for our algorithm.

To solve CSPs we make structural cases on the underlying graph, and for each structural case we calculate the work factor.
The final work factor of the algorithm is the maximum of all the cases.

Structural Lemmata

We present some examples of "structural lemmata" below.

Proposition

Let $\eta = \{(v, R), (w, R)\}$ be an isolated constraint in a $(4, 2)$-CSP instance. Then the instance can be reduced to smaller instances with work factor $\leq \lambda(2 - \varepsilon, 3 - \varepsilon)$.

Proposition

Let $\eta = \{(v, R), (w, R)\}$ be an dangling constraint (we assume (w, R) is present in other constraints too, while (v, R) is only present in η) in a $(4, 2)$-CSP instance. Then the instance can be reduced to smaller instances with work factor $\leq \lambda(2 - \varepsilon, 3 - \varepsilon)$.

Structural Lemmata (Contd.)

Proposition

Suppose we have a $(4,2)$-CSP instance that has a variable v and a color R such that at least one of the following conditions is satisfied:

1. *(v,R) is involved in ≥ 3 constraints, and v is a 4-color variable.*
2. *(v,R) is involved in ≥ 4 constraints, and v is a 3-color variable.*

Then the instance can be reduced with a work factor $\leq \lambda(1-\varepsilon, 5-4\varepsilon)$.

Conclusions

In the original Beigel-Eppstein paper [1], even more lemmata of the above nature were proved. We don't include those lemmata here as they don't add any significantly new idea to the problem: Indeed, Beigel and Eppstein make increasingly fine-grained structural assumptions, and calculate the corresponding work factors, until the structural assumptions exhaust all possibilities for a $(4, 2)$-CSP.

Conclusions

Thus, after applying all the relevant lemmata, the work factor of the entire algorithm is of the form $f(\varepsilon) := \max(\lambda(1-\varepsilon, 5-4\varepsilon), \lambda(2-\varepsilon, 3-2\varepsilon), \lambda(2-\varepsilon, 3-\varepsilon), \ldots)$. We optimize $f(\varepsilon)$ to obtain that $f(\varepsilon)$ attains its minima at $\varepsilon \approx 0.095543$, and the minimum value of $f(\varepsilon)$ equals $\Lambda := \lambda(4,4,5,5) \approx 1.36443$, which becomes the work factor for our algorithm. As promised, this is better than the $\mathcal{O}^*\left(\sqrt{2}^n\right)$ randomized algorithm for $(3,2)$-CSPs.

Conclusions

Some immediate consequences of the above result are as follows:

Corollary

3-coloring and 3-list coloring on a graph with n vertices and m edges can be solved in $\mathcal{O}^*(\Lambda^n)$ time, and 3-edge coloring can be solved in $\mathcal{O}^*(\Lambda^m)$ time.

Proof.

All of the problems mentioned here can be translated to $(3, 2)$-CSPs. □

Another corollary goes as follows:

Corollary

There exists a randomized algorithm that finds the solution to any solvable $(d, 2)$-CSP in expected time $\mathcal{O}^*\left((0.4518d)^n\right)$, where $d \geq 4$.

Conclusions

- The SOTA today is that we can decide, faster than 2^n, $\leq$ 5-colorability.
- We can even decide 6-colorability faster than 2^n if we're allowed to use randomization.
- The primary obstruction in Zamir's argument, which doesn't allow us to extend it to 7-colorability and beyond, is the fact that we proved that 5-colorability was $o^*(2^n)$ decidable using that fact that 4-list-colorability was $o^*(2^n)$ decidable. Since we don't know anything about 5-list-colorability for example, Zamir's arguments don't work, as it is.

Conclusions

- The obstruction in Beigel-Eppstein's arguments which prevent extension to 5-list-colorability, is that the case-work in their arguments is very specific to $(4, 2)$-CSPs. Once again, it seems that significantly new ideas will be needed to extend their work for higher CSPs.
- Extending Zamir's and Beigel-Eppstein's arguments to answer whether k-colorability is $o^*(2^n)$ decidable for every $k \in \mathbb{N}$ is a natural avenue to pursue.

References

[1] Richard Beigel and David Eppstein. "3-coloring in time $O(1.3289^n)$". In: *Journal of Algorithms* 54.2 (2005), pp. 168–204. ISSN: 0196-6774.

[2] Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. "Set Partitioning via Inclusion-Exclusion". In: *SIAM Journal on Computing* 39.2 (2009), pp. 546–563.

[3] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 364216532X.

[4] Eugene L. Lawler. "A Note on the Complexity of the Chromatic Number Problem". In: *Inf. Process. Lett.* 5 (1976), pp. 66–67.

[5] Or Zamir. *Breaking the 2^n barrier for 5-coloring and 6-coloring*. 2021. arXiv: 2007.10790 [cs.DS].

Acknowledgements

I thank Prof. Sundar Vishwanathan for suggesting the topic of this presentation to me, and very patiently fielding and clearing all my queries regarding the same.

The End

Questions? Comments?