A BFS approach to partitioning problems on random graphs

C.R. Subramanian
Max-Planck Institut für Informatik
Saarbrücken, Germany.

k-Coloring Problem :

Given $G = (V, E)$.

Properly color V using $\leq k$ colors.

Partition V into $V_1, \ldots, V_k$, each V_i independent.

Worst-case complexity :

For general k, hard to approximate within $O(n^{1-\epsilon})$ for any fixed $\epsilon > 0$.

NP-Hard for every fixed $k \geq 3$.

NP-Hard to 4-color 3-colorable graphs.

For $k = 3$, requires $\tilde{O}(n^{3/14})$ colors in poly. time.

For $k \geq 4$, requires $\tilde{O}(n^{1-3/(k+1)})$ colors in poly. time.

Average-case complexity :

Random and semi-random k-colorable graphs :

k-colorable *a.s.* in poly. time.

k-colorable in *p.av.t.*

Comparison :

Worst-case intractability results are pessimistic when compared to average-case analysis.

Need to study average-case behavior over larger classes of distributions.

Random models :

$\mathcal{GUC}(n, p, k)$ **:**

Each u into one of $V_1, \ldots, V_k$ uniformly randomly.

Each allowed $\{u, v\}$ chosen with independent prob. p.

Almost surely, every $|V_i| \approx n/k$.

$\mathcal{G}(n, p, k)$ **:**

Form $V_1, \ldots, V_k$ arbitrarily with "roughly" equal sizes.

Each allowed $\{u, v\}$ chosen with independent prob. p.

$\mathcal{G}(n, m, k)$ **:**

Form $V_1, \ldots, V_k$ arbitrarily with "roughly" equal sizes.

Uniformly randomly choose m allowed edges.

Semi-random models :

Plant a k-coloring and choose edges with prob. p.

Arbitrarily add some allowed edges.

$p = 0 \Rightarrow$ worst-case scenario

No new edges added $\Rightarrow$ random graph

Quality of algorithms :

Running time T

Range of distributions (the range of p)

Failure probability F_p

Types of algorithms :

a.s. algorithms :

succeed with prob. $1 - o(1)$.

$o(1)$ can be any of $(\log n)^{-1}$, n^{-2}, $n^{-(\log n)}$, $e^{-n^{0.5}}$.

p.av.t. algorithms :

always succeed, time T polynomial on average.

provide a trade-off, T vs. F_p.

any desired poly. low F_p at a cost of poly. $\uparrow$ in T.

Difficulty : failed instances may have "large" F_p and large T.

Previous work : *a.s.* algorithms :

[BS] : $\mathcal{GUC}()$ model. $p \geq n^{-1+\epsilon}$, fixed $\epsilon > 0$.

[AK] : $|V_i| = n$ for $i = 1, 2, , \ldots, k$, $p \geq c/n$.

[BS], [SFV] : semi-random graphs, $p \geq n^{-\frac{2k}{k^2+k-2}}$.

[FK] : semi-random graphs, $p = \Omega(n^{-1}(\log n))$.

BS : Blum and Spencer

AK : Alon and Kahale

SFV : Subramanian, Furer and Veni Madhavan

FK : Feige and Kilian

p.av.t. algorithms :

[DF] : for slowly decreasing p.

[S] : for $p \geq n^{-1+\epsilon}$, fixed $\epsilon > 1/4$.

[SFV] : semi-random graphs, $p \geq n^{-\frac{2k}{k^2+k-2}}$.

DF : Dyer and Frieze

S : Subramanian

Current work :

a.s. and *p.av.t.* algorithms

for $p \geq n^{-1+\epsilon}, \epsilon \geq \frac{X}{\sqrt{\log n}}$, X large constant.

simple and based on growing BFS trees.

very small F_p, $F_p = o(n^{-D})$ for any fixed $D > 0$.

adaptable to yield even smaller F_p at a cost of $\uparrow$ in T.

A technique for *p.av.t.* coloring by lowering F_p.

Assumptions :

Color class sizes $n_1, \ldots, n_k$ - known to the algorithm.

V_1 is the largest or smallest color class.

General idea :

Separate (in $\uparrow$ or $\downarrow$ order of their sizes) the color classes one by one.

To separate V_1, compute $n(x, y, G)$ for each $y \neq x$.

wanted : For some $x \in V_1$,

for each $y \in V_1$, $z \notin V_1$, $n(x, y, G) > n(x, z, G)$.

For each $x \neq y$, compute $n(x, y, G)$:

For suitable l, starting at x, grow the BFS tree in $G(V - y)$ till the l-th level is computed. Return the number of y's neighbors in the last level as $n(x, y, G)$.

Intuition :

Assume l is even.

In every even-numbered level till l, the contribution of V_1 is significantly more than the contribution of any other V_j.

Hence, y has less neighbors in the l-th level if y is in V_1 than it will have if y belongs to V_j, $j > 2$. This separates V_1 from other classes.

Performance :

polynomial running time.

$F_p \leq e^{-\Omega(\alpha^2 \cdot \min(np, n^l p^{l+1} C^{-l}))} = o(n^{-D})$ for any fixed $D > 0$.

Outline of p.av.t. coloring alg. B **:**

1. Apply an algorithm A with poly. time T_A and f.p. f_A. Exit if A succeeds.

2. Apply algorithms $A_1, \ldots, A_r$ with increasing running times $T_1, \ldots, T_r$ and decreasing failure probabilities $f_1, \ldots, f_r$. Exit immediately if any of them succeeds.

3. Apply brute-force method to find a k-coloring.

Analysis of B **:**

B always produces a k-coloring.

$$\begin{aligned} E(T_B) &\leq T_A + T_1 f_A + \ldots \\ &\qquad + (T_1 + \ldots + T_r) f_{r-1} \\ &\qquad + (T_1 + \ldots + T_r + n^2 k^n) f_r \\ \\ &= poly(n) \text{ for proper choice of } A_1, \ldots, A_r \end{aligned}$$

Intermediate algorithm FindColor4(m) :

Works for $m = o(1/p)$.

Choose a suitable integer $l(m)$ depending on m.

Based on the same intuition : As the BFS tree grows from $A \subset V_1$, $|A| = m$, in each even-numbered level, contribution of V_1 is significantly more than other contributions.

In every level L_i, there are at most m vertices $z \notin V_1$ which have no neighbors in $V_1 \cap L_{i+1}$.

This also works by separating the color classes in $\uparrow$ or $\downarrow$ order of their sizes.

Time $= O(n^{m(\log n)})$.

$F_p \leq e^{-\Omega(mg(n))}$ where $g(n) \geq (\log n)2^{\sqrt{X(\log n)/2}}$.

p.av.t.k-coloring $B(G)$:

[SFV] : algorithm Color2(G, n, k, m).

when $F_p = o(p)$, apply *Color2*. F_p(Color2) $\leq e^{-nk}$.

if *Color2* also fails, apply brute-force coloring.

$B(G)$ has polynomial average running time.

The technique works for any polynomial algorithm A with suitably low F_p.

A can be polynomial (on average) algorithm with stated failure probability.

Conclusions :

can p be lowered by a careful analysis ?

p.av.t. k-coloring for $p \geq c/n$?

BFS approach applies to Graph Bisection and other partitioning problems where parts have bounds (lower/upper) on their edge densities .