Coloring Graphs with Intervals

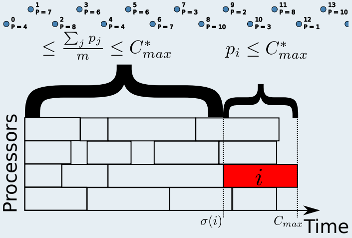
Erik Saule and Dante Durrman

The University of North Carolina at Charlotte

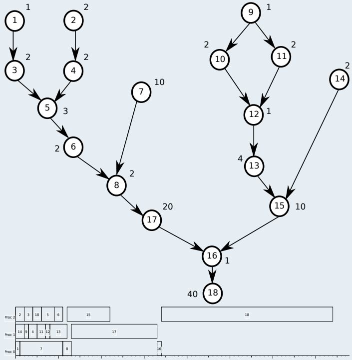
Scheduling for Large Scale Systems Workshop
May 22, 2023

Scheduling, Traditionally

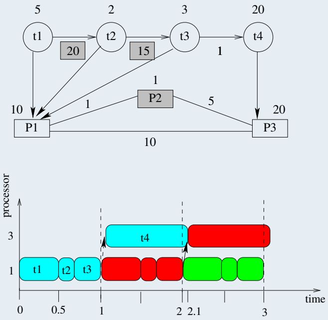
Independent Tasks



Parallel Task Graph



Pipelined Graphs



All these cases have precedence dependencies that come from the application

Conflict Graph Model

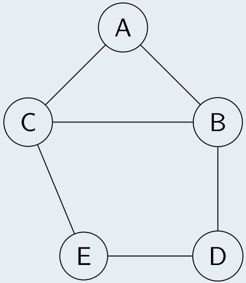
Applications

Process X can't run while Process Y runs.

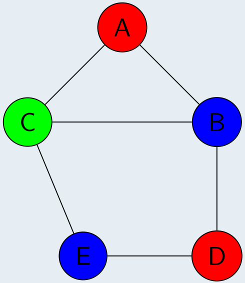
- Transactions in Databases
- Transpose sparse matrix operations
- Anytimes tasks share memory writes

Traditionally solved with mutex and atomics.

A Coloring Model



Solution



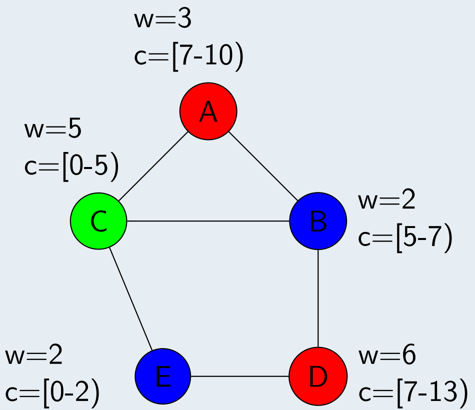
Schedule

In batches of colors with a hard synchronization between colors.

Conflict models do not account for runtime of tasks.

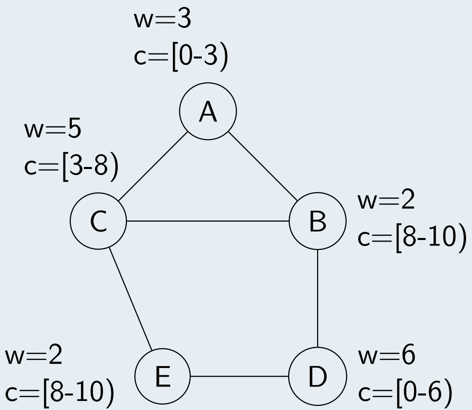
The runtime of tasks matters

Executing colors one at a time



Solution in 13

Optimal schedule



Solution in 10

Formal Definition of the Interval Vertex Coloring Problem

Interval Vertex Coloring Problem (IVC)

Let $G = (V, E)$ be an undirected graph and $w : V \to \mathbb{Z}^+$ be a weight function.
An interval coloring of the vertices of G is a function $start : V \to \mathbb{Z}^+$.
We say that vertex v is colored with the open interval $[start(v), start(v) + w(v))$.

Valid Interval Coloring

For the coloring to be valid, neighboring vertices must have disjoint color intervals:
$\forall (a, b) \in E, [start(a), start(a) + w(a)) \cap [start(b), start(b) + w(b)) = \emptyset$.

Optimal Interval Coloring

A particular coloring of vertices is said to use $maxcolor = \max_{v \in V} start(v) + w(v)$ colors.
The optimization problem is to find a coloring that minimizes $maxcolor$.
We denote the optimal value of $maxcolor$ as $maxcolor^*$.

Coloring vertices with interval is harder than regular graph coloring

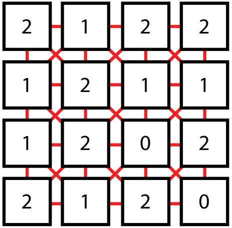
- NP Complete in general
- No general approximation algorithm

Outline

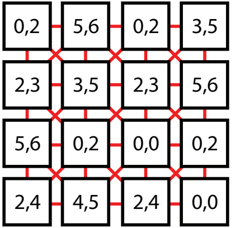
1. Introduction
2. Coloring 9-pt and 27-pt stencils with intervals [IPDPS22]
3. Optimizing Distributed Dataflow Algorithms [PDCO23]
4. Conclusion

The Problem of Coloring Stencils with Intervals

2D Example (4x4)

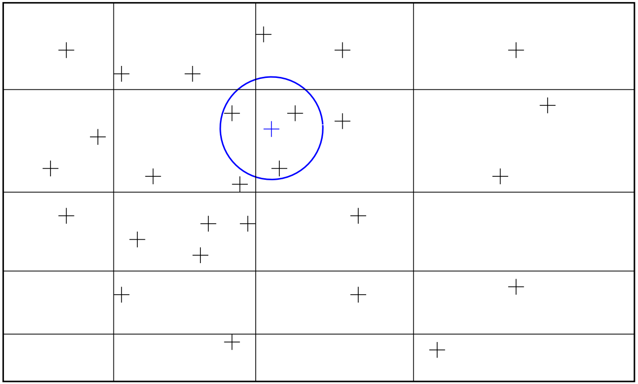


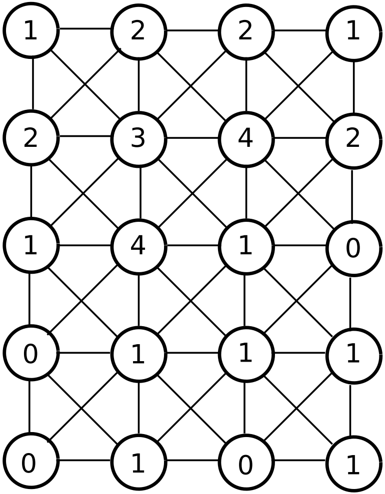
Example Solution



- A graph which is a
  - 2D 9-pt stencil
  - or 3D 27-pt stencil
- Each vertex has a weight $w(v)$
- Color each vertex with an interval larger than its weight
  - Intervals should have the form $[start(v), start(v) + w(v))$
- No adjacent vertices can have overlapping intervals
- *maxcolors* is the largest right endpoint in the set of intervals
- Objective is to minimize *maxcolor*

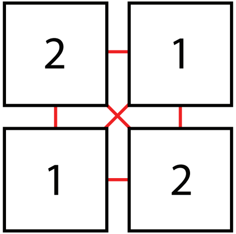
Spatial Applications often Parallelize as Stencils



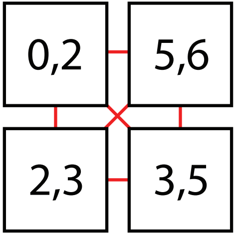


Cliques can be Colored in Linear Time

K4 Example



K4 Solution



Algorithm

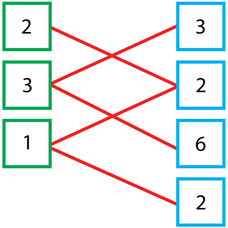
- No vertex can share any color with any other vertex in clique
- We must use at least $\sum_{v \in K} w(v)$ colors
- Greedily color the interval with the lowest available $start(v)$
- Complexity $\Theta(V)$

Implications

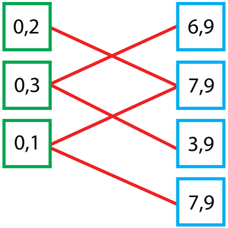
- Each square block of 4 vertices is a K_4
  - Sum of weights of K_4 is a lower bound of 2D 9-pt stencil
- Each square block of 8 vertices is a K_8
  - Sum of weights of K_8 is a lower bound of 3D 27-pt stencil

Bipartite Graphs can be Colored in Linear Time

Bipartite Example



Bipartite Solution



Algorithm

- Partition vertices into A, B, s.t, $(i, j) \in E \implies i \in A, j, \in B$
- Compute $maxcolor = \max_{(i,j)\in E} w(i) + w(j)$
- Color $i \in A$ starting at 0 with $[0; w(i))$
- Color $j \in B$ ending at $maxcolor$ with $[maxcolor - w(j); maxcolor)$
- Complexity $\Theta(E)$

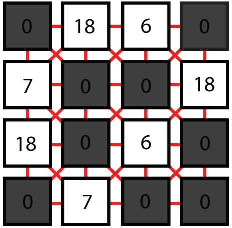
Implications

Many subgraphs of a stencil are bipartite and induce lower bounds:

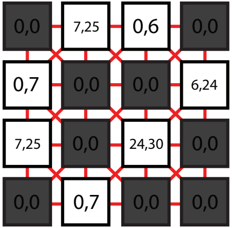
- Each edge in the graph
- 2D 5-pt stencils
- 3D 7-pt stencils
- Many cycles of even length

Odd Cycles can be Colored in Linear Time

Odd Cycle Example



Odd Cycle Solution



Algorithm

- Let *maxpair* be the largest sum of any 2 consecutive vertices
- Let *minchain*3 is the smallest sum of 3 consecutive vertices
- We have $maxcolor = max(maxpair, minchain3)$
- Identify the *minchain*3 triplet: $0, 1, 2$
  - Color 0 with $[0; w(0))$
  - Color 1 with $[w(0); w(0) + w(1))$
  - Color 2 with $[w(0) + w(1); w(2))$
  - Color the other alternatively with $[0; w(v))$
  - or $[maxcolor - w(v); maxcolor)$
- Complexity $\Theta(E)$

Implications

Many odd cycles in 2D 9-pt stencils and 3D 27-pt stencils

27pt-Stencil is NP-Complete by Reduction from NAE-3SAT

NAE-3SAT: Not-All-Equal 3-SAT

- n binary variables in m groups of 3 variables
- Assign true or false to each variable
- The instance is positive if every group has at least one variable that is true and at least one that is false

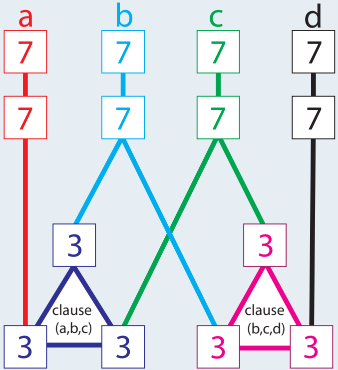
NAE-3SAT is known to be NP-Complete

Solving NAE-3SAT by Coloring a Simple Graph with 14 Colors

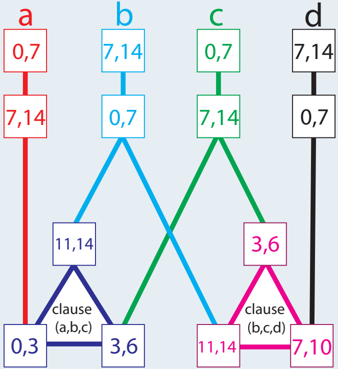
NAE-3SAT Instance

Variables: $\{a, b, c, d\}$; Clauses: $\{(a, b, c), (b, c, d)\}$

Constructed Graph Instance

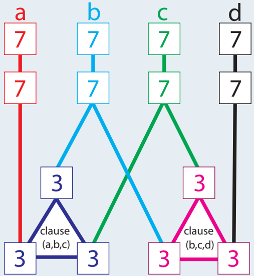


Solution in 14 Colors

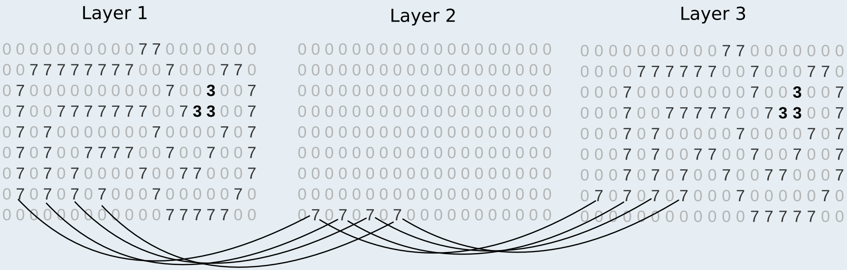


Embedding the Constructed Graph in a 3D 27-pt Stencil

From Graph ...



... To a 27-pt 3D stencil



Greedy Algorithms

Greedy Principles

Any greedy coloring will color vertex v with an interval that ends before: $\sum_{j \in \Gamma(v)} w(j) + (\Gamma(v) + 1)w(v) - \Gamma(v)$

By Vertex

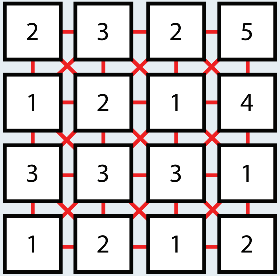
- Greedy Largest First
- Greedy Line by Line
- Greedy Z-Order

By Set

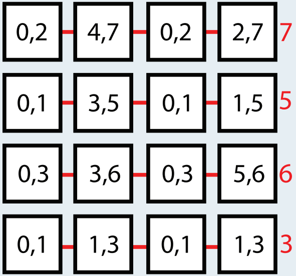
- Greedy Largest Clique First
  - Schedule vertices in the largest clique first; order within clique uses vertex id
- Smart Greedy Largest Clique First
  - Permute each clique and use the order with least *maxcolor*

Bipartite Decomposition is a 2-approx. in 2D (and 4-approx. in 3D)

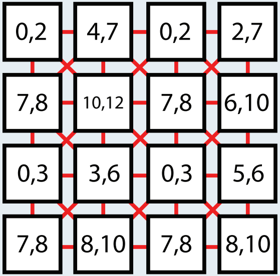
Example



Do Rows Independently



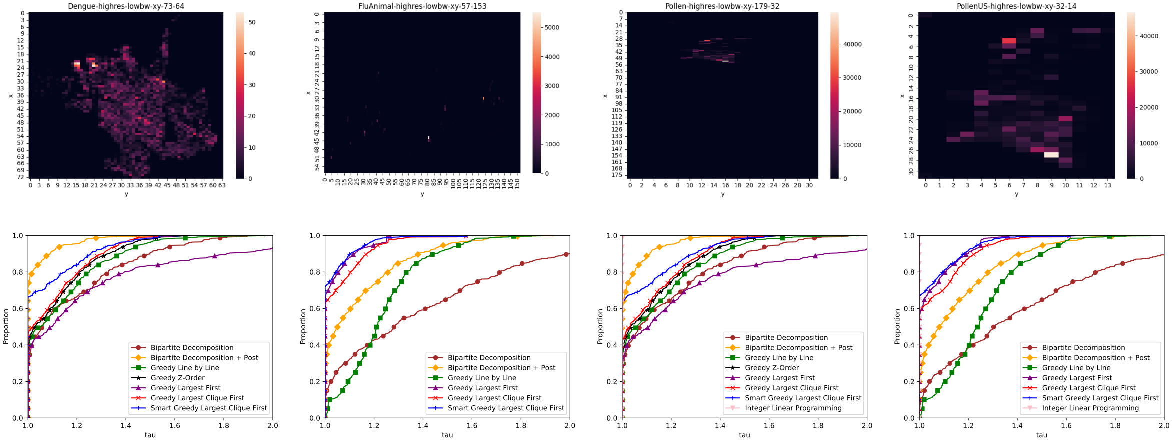
Shift Odd Rows by Max Color



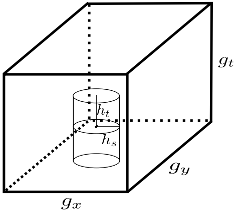
Bipartite Decomposition with Post Optimization

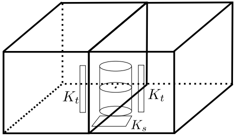
- Sort K_4 or K_8 (in 3D) by non-increasing order by the sum total of their weights
- Sort vertices within K_4 by increasing order of lowest value in their scheduled interval
- Recolor each vertex one at a time using a greedy principle

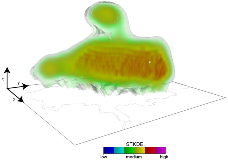
We Ran Simulations. These Methods Work.

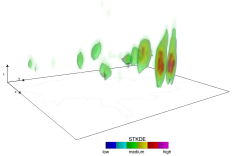


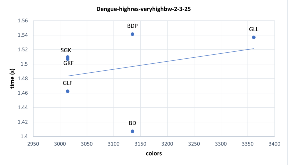
We Integrated that in a Real Application. These Methods Work.

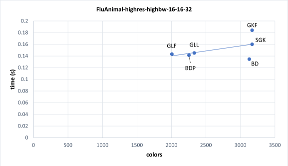


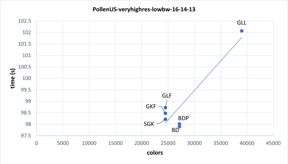


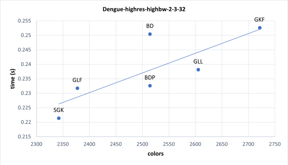


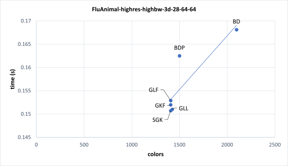


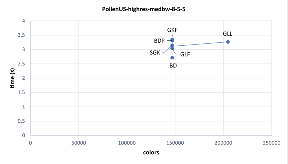












Outline

Luby's Algorithm is an Example of a Dataflow Algorithm

Luby's Algorithm for Maximal Independent Set

- Each vertex v picks unique random number $r(v)$ uniformly in $[0; 1)$
- v sends $r(v)$ to each of its neighbors u
- v notes which neighbors u have the property $r(u) > r(v)$
- v marks its own state as `unknown`
- v awaits a message from each of its neighbors u if $r(u) < r(v)$
- If the state of u is `marked`, the state of v is changed to `unmarked`
- After receiving messages from all neighbors u, if the state of v is `unknown`, the state of v is changed to `marked`
- v sends its state to all neighbors u, such that $r(u) > r(v)$
- All vertices in the `marked` state are a maximal independent set

Introduction to Dataflow Algorithms

Distributed Dataflow Algorithms

- Only use local information
- Processing order of vertices is generated randomly
- Once the order is picked the vertices are processed from low to high in each neighborhood
- Cost to determine other desirable properties is too high
- We are interested in these methods as a model for distributed graph algorithms

Examples

- Luby's Algorithm for Maximal Independent Set
- Jones-Plassmann Algorithm for Graph Coloring

Choice of random order matters to algorithm runtime

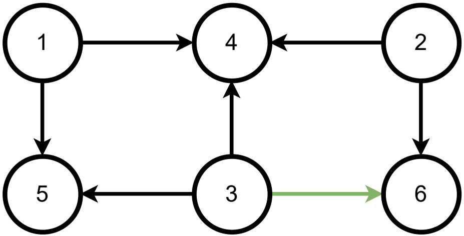


Figure 1: Lucky Draw

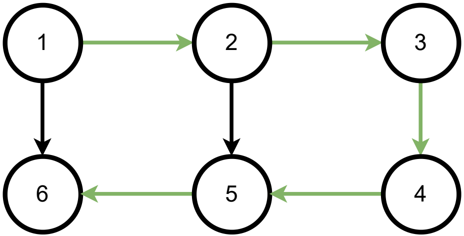


Figure 2: Unlucky Draw

Choice of random order matters to algorithm runtime

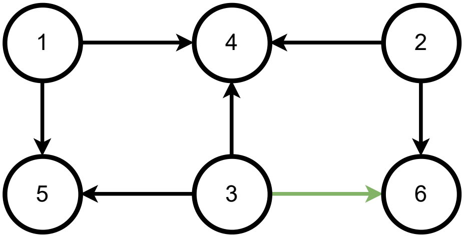


Figure 1: Lucky Draw

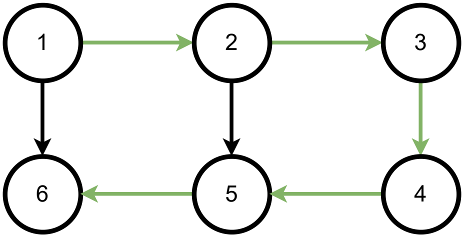


Figure 2: Unlucky Draw

The question

How can we avoid unlucky draws?

Developing a Dataflow Model for Distributed Graph Algorithms

Model

- Let $G = \{V, E\}$ be an undirected graph and $w : V \rightarrow \mathbb{Z}^+$ be a weight function
- Assign $r(v)$ to each vertex v with your algorithm of choice
- Construct directed graph $\bar{G}$ by orienting the existing edges from low $r(v)$ to high $r(v)$
- Calculate length of critical path of $\bar{G}$

Critical Path

Longest weighted path in $\bar{G}$

Objective

Minimizing the length of the critical path (which minimizes the algorithm execution time)

The Weight Function is Non-trivial

Special Case: $w(v) = 1$

- Execution time is dominated by latency.
- The critical path is the number of phases for the graph.
- Critical path is the same as longest path using euclidean distance

Special Case: $w(v) = \delta(v)$

- Bandwidth or the cost of algorithms on the vertices themselves dominates the total execution time of the algorithm.
- Each vertex sends and receives $\delta(v)$ messages
- Most dataflow algorithms have each vertex do $O(\delta(v))$ computations
- Largest Degree First Order closely resembles that of Largest Processing Time First

Deriving Better Partial Orders for Distributed Graph Algorithms

`Uniform` (aka draw in $[0; 1)$)

- Existing method of random number generation in dataflow algorithms

`Linear` (aka draw in $[0; \delta(v))$)

- v is guaranteed to be after all vertices u, such that $\delta(u) = \delta(v) - 1$ with probability $\frac{1}{\delta(v)}$
- Good approximation of Largest Degree First with vertices of dramatic difference in degrees
- Poor approximation when $\Delta(G)$ is large and G has many vertices of large degrees

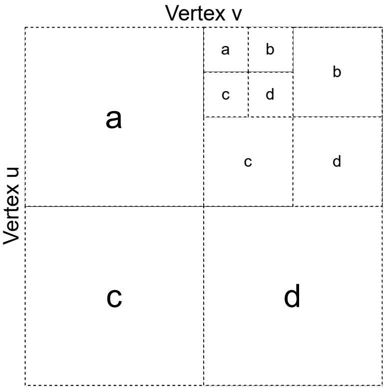
`Exponential` (aka draw in $[0; 2^{\delta(v)})$)

- v is guaranteed to be after all vertices u, such that $\delta(u) = \delta(v) - 1$ with probability $> \frac{1}{2}$
- Better approximation of Largest Degree First

- Communication and Computational cost is the same for each algorithm
- Sampling uniformly in those intervals despite naming conventions

Construction of RMAT Graphs

- 2^n nodes
- Recursively split square matrix into 4 quadrants: a, b, c, d
- Each quadrant has an associated probability that a given edge will fall into that quadrant: $a + b + c + d = 1$
- Edges are generated one at a time and placed in a quadrant recursively following those probabilities until the edge is placed in a 1×1 submatrix.
- $ef * 2^n$ edges



RMAT Graphs Study

Methodology

- Sampled RMAT parameter space with constant *ef*
- Computed critical path length for each algorithm
- Calculated 95% confidence intervals
- Computed pairwise ratios of critical paths
- Conducted Z-Test to validate statistical significance

Results

- `Exponential` path < `Linear` path < `Uniform` path
- `Exponential` was never worse than `Uniform`
- At best, `Exponential` was 50% better
- On average, `Exponential` was about 10% better

a	b	c	d	Uniform CI	Exponential CI	Linear CI	U/E	U/L	L/E
0.30	0.28	0.28	0.14	[2944; 2947]	[2623; 2625]	[2850; 2853]	1.123	1.033	1.087
0.40	0.24	0.24	0.12	[4950; 4953]	[4680; 4683]	[4859; 4862]	1.058	1.019	1.038
0.50	0.20	0.20	0.10	[6968; 6971]	[6643; 6647]	[6845; 6848]	1.049	1.018	1.030
0.60	0.16	0.16	0.08	[8353; 8357]	[7949; 7952]	[8175; 8178]	1.051	1.022	1.028
0.70	0.12	0.12	0.06	[9211; 9214]	[8738; 8740]	[8987; 8990]	1.054	1.025	1.029
0.30	0.28	0.17	0.25	[2111; 2113]	[2064; 2066]	[2103; 2105]	1.023	1.004	1.019
0.30	0.28	0.25	0.17	[2633; 2635]	[2437; 2439]	[2583; 2585]	1.080	1.019	1.060
0.30	0.28	0.34	0.08	[3782; 3785]	[3112; 3115]	[3510; 3513]	**1.215**	1.077	1.128
0.30	0.35	0.14	0.21	[2625; 2627]	[2047; 2049]	[2402; 2404]	**1.282**	1.093	**1.173**
0.30	0.35	0.21	0.14	[3064; 3067]	[2464; 2467]	[2825; 2827]	**1.243**	1.085	1.146
0.30	0.35	0.28	0.07	[3959; 3962]	[3221; 3225]	[3644; 3647]	**1.229**	1.086	1.131
0.30	0.42	0.11	0.17	[3632; 3635]	[2181; 2183]	[2880; 2883]	**1.665**	**1.261**	**1.321**
0.30	0.42	0.17	0.11	[3821; 3824]	[2433; 2436]	[3061; 3064]	**1.570**	**1.248**	**1.258**
0.30	0.42	0.22	0.06	[4343; 4346]	[3110; 3113]	[3685; 3688]	**1.396**	**1.178**	**1.185**
0.30	0.49	0.08	0.13	[4852; 4855]	[2245; 2249]	[3437; 3441]	**2.160**	**1.411**	**1.530**
0.30	0.49	0.13	0.08	[4775; 4778]	[2505; 2508]	[3325; 3330]	**1.906**	**1.435**	**1.328**
0.30	0.49	0.17	0.04	[5052; 5056]	[3151; 3155]	[3883; 3887]	**1.603**	**1.301**	**1.232**
0.40	0.24	0.14	0.22	[3246; 3249]	[3185; 3188]	[3242; 3245]	1.019	1.001	1.018
0.40	0.24	0.22	0.14	[4571; 4574]	[4377; 4380]	[4509; 4512]	1.044	1.014	1.030
0.40	0.24	0.29	0.07	[5946; 5950]	[5500; 5504]	[5759; 5763]	1.081	1.032	1.047
0.40	0.30	0.12	0.18	[4026; 4029]	[3457; 3460]	[3811; 3814]	**1.165**	1.056	1.102
0.40	0.30	0.18	0.12	[5003; 5006]	[4551; 4555]	[4821; 4825]	1.099	1.038	1.059
0.40	0.30	0.24	0.06	[6141; 6144]	[5652; 5656]	[5932; 5935]	1.086	1.035	1.049
0.40	0.36	0.10	0.14	[4903; 4906]	[3767; 3771]	[4333; 4336]	**1.301**	1.132	1.150
0.40	0.36	0.14	0.10	[5506; 5509]	[4637; 4641]	[5071; 5075]	**1.187**	1.086	1.094
0.40	0.36	0.19	0.05	[6383; 6387]	[5684; 5688]	[6045; 6049]	1.123	1.056	1.063
0.40	0.42	0.07	0.11	[5667; 5670]	[3901; 3906]	[4639; 4643]	**1.452**	**1.221**	**1.189**
0.40	0.42	0.11	0.07	[6196; 6199]	[4927; 4932]	[5478; 5483]	**1.257**	1.131	1.112
0.40	0.42	0.14	0.04	[6670; 6674]	[5681; 5685]	[6132; 6136]	**1.174**	1.088	1.079
0.50	0.20	0.12	0.18	[5009; 5012]	[4881; 4884]	[4981; 4984]	1.026	1.006	1.020
0.50	0.20	0.18	0.12	[6489; 6492]	[6213; 6216]	[6399; 6402]	1.044	1.014	1.030
0.50	0.20	0.24	0.06	[7813; 7816]	[7365; 7368]	[7616; 7620]	1.061	1.026	1.034
0.50	0.25	0.10	0.15	[5687; 5690]	[5230; 5234]	[5515; 5519]	1.087	1.031	1.054
0.50	0.25	0.15	0.10	[6920; 6924]	[6500; 6504]	[6748; 6751]	1.065	1.026	1.038
0.50	0.25	0.20	0.05	[7984; 7987]	[7503; 7507]	[7766; 7770]	1.064	1.028	1.035
0.50	0.30	0.08	0.12	[6294; 6297]	[5524; 5528]	[5943; 5946]	1.139	1.059	1.076
0.50	0.30	0.12	0.08	[7263; 7267]	[6644; 6648]	[6969; 6972]	1.093	1.042	1.049
0.50	0.30	0.16	0.04	[8073; 8076]	[7495; 7499]	[7786; 7789]	1.077	1.037	1.039
0.50	0.35	0.06	0.09	[6812; 6815]	[5778; 5781]	[6278; 6281]	**1.179**	1.085	1.087
0.50	0.35	0.09	0.06	[7537; 7540]	[6729; 6732]	[7108; 7111]	1.120	1.060	1.056
0.50	0.35	0.12	0.03	[8135; 8138]	[7420; 7423]	[7754; 7757]	1.096	1.049	1.045
0.60	0.16	0.10	0.14	[6491; 6495]	[6204; 6208]	[6406; 6409]	1.046	1.013	1.032
0.60	0.16	0.14	0.10	[7774; 7777]	[7418; 7422]	[7631; 7635]	1.048	1.019	1.029
0.60	0.16	0.19	0.05	[9077; 9080]	[8613; 8616]	[8843; 8846]	1.054	1.026	1.027
0.60	0.20	0.08	0.12	[6942; 6945]	[6427; 6431]	[6741; 6745]	1.080	1.030	1.049
0.60	0.20	0.12	0.08	[8257; 8260]	[7803; 7806]	[8044; 8048]	1.058	1.026	1.031
0.60	0.20	0.16	0.04	[9253; 9256]	[8754; 8757]	[8997; 9000]	1.057	1.028	1.028
0.60	0.24	0.06	0.10	[7261; 7264]	[6516; 6519]	[6926; 6929]	1.114	1.048	1.063
0.60	0.24	0.10	0.06	[8597; 8600]	[8041; 8044]	[8308; 8311]	1.069	1.035	1.033
0.60	0.24	0.13	0.03	[9283; 9286]	[8731; 8734]	[8987; 8990]	1.063	1.033	1.029
0.60	0.28	0.05	0.07	[7829; 7832]	[6958; 6962]	[7380; 7383]	1.125	1.061	1.061
0.60	0.28	0.07	0.05	[8537; 8540]	[7838; 7841]	[8157; 8160]	1.089	1.047	1.041
0.60	0.28	0.10	0.02	[9249; 9252]	[8631; 8634]	[8900; 8903]	1.072	1.039	1.031
0.70	0.12	0.07	0.11	[7229; 7232]	[6852; 6856]	[7101; 7105]	1.055	1.018	1.036
0.70	0.12	0.11	0.07	[8854; 8857]	[8424; 8427]	[8659; 8662]	1.051	1.023	1.028
0.70	0.12	0.14	0.04	[9813; 9816]	[9197; 9200]	[9514; 9517]	1.067	1.031	1.034
0.70	0.15	0.06	0.09	[7797; 7800]	[7252; 7255]	[7573; 7576]	1.075	1.030	1.044
0.70	0.15	0.09	0.06	[9093; 9096]	[8589; 8592]	[8850; 8853]	1.059	1.027	1.030
0.70	0.15	0.12	0.03	[10032; 10035]	[9347; 9350]	[9694; 9696]	1.073	1.035	1.037
0.70	0.18	0.05	0.07	[8267; 8270]	[7587; 7591]	[7941; 7945]	1.090	1.041	1.047
0.70	0.18	0.07	0.05	[9183; 9186]	[8599; 8602]	[8876; 8879]	1.068	1.035	1.032
0.70	0.18	0.10	0.02	[10074; 10076]	[9370; 9372]	[9700; 9702]	1.075	1.039	1.035
0.70	0.21	0.04	0.05	[8667; 8669]	[7899; 7902]	[8263; 8266]	1.097	1.049	1.046
0.70	0.21	0.05	0.04	[9168; 9171]	[8503; 8506]	[8798; 8800]	1.078	1.042	1.035
0.70	0.21	0.07	0.02	[9844; 9846]	[9198; 9200]	[9470; 9472]	1.070	1.039	1.030

Understanding the Results of the RMAT Graphs Study

Why is `Exponential` better on RMAT Graphs

- RMAT Graphs have the same properties of a social network
- Social networks are "Onion-like" - dense core, but outer layers become less dense
- `Exponential` is similar to Largest First

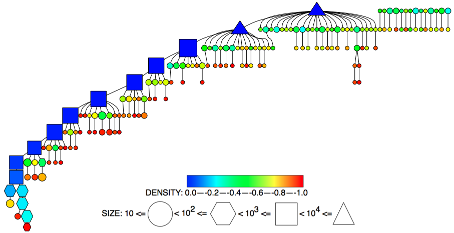


Figure 3: Hierarchy of Dense Subgraphs by Sariyuce et al. (2015)

Real World Application

- Conducted similar experiment on real world graphs from SNAP
- All graphs have small world properties, except roads of Pennsylvania
- `Exponential` was better except on ca-HepPh and roadNet-PA

Name	Vertices	Edges	Max Degree	Clustering Coefficient	Diameter	Uniform CI	Exponential CI	Linear CI	U/E	U/L	L/E
CA-HepPh	89,209	118,521	491	0.6115	13	[1030; 1036]	[1040; 1045]	[1032; 1037]	0.991	0.999	0.992
Email-Enron	36,692	183,831	1,383	0.4970	11	[43437; 43720]	[38836; 38982]	[40688; 41002]	1.120	1.067	1.050
p2p-Gnutella04	10,879	39,994	103	0.0062	9	[911; 925]	[568; 575]	[728; 740]	**1.606**	**1.251**	**1.284**
roadNet-PA	1,090,920	1,541,898	9	0.0465	786	[49; 49]	[49; 50]	[48; 49]	0.990	1.010	0.980
soc-Epinions1	75,888	405,740	3,044	0.1378	14	[94793; 95270]	[88297; 88593]	[89488; 90034]	1.074	1.059	1.015
soc-pokec-relationships	1,632,804	22,301,964	14,854	0.1094	11	[118924; 119528]	[96958; 97239]	[100836; 101775]	**1.228**	**1.177**	1.043
web-Google	916,428	4,322,051	6,332	0.5143	21	[80466; 81618]	[18166; 18192]	[20577; 21084]	**4.458**	**3.891**	1.146
WikiTalk	2,394,385	4,659,565	100,029	0.0526	9	[1352414; 1357165]	[1101248; 1103043]	[1145942; 1151894]	**1.229**	**1.179**	1.042

Results of the Real World Application

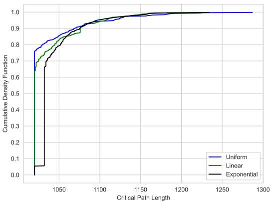


Figure 4: ca-HepPh

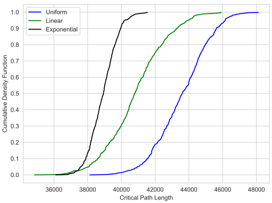


Figure 5: email-Enron

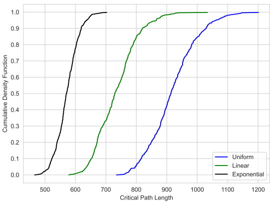


Figure 6: p2p-Gnutella04

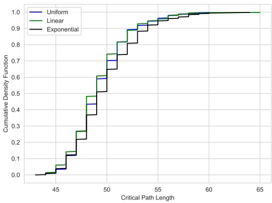


Figure 7: roadNet-PA

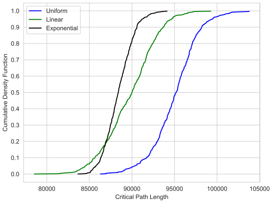


Figure 8: soc-Epinions1

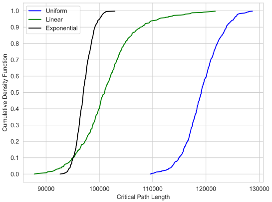


Figure 9: soc-pokec

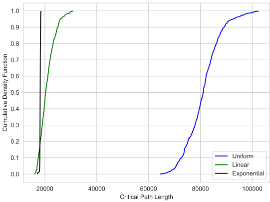


Figure 10: web-Google

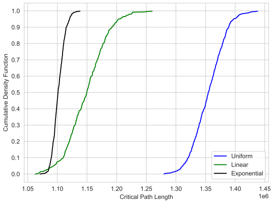


Figure 11: wiki-Talk

Outline

Tackled an understudied coloring problem

Structured graphs

- 27-pt 3D stencil is NP Complete
- Polynomiality of simple structures
- Approximation algorithms for 2D and 3D stencils
- Validated in simulation
- Validated in a real application

Distributed coloring with interval

- Recast graph dataflow algorithm optimization as interval coloring
- Suggested new algorithms for distributed interval coloring
- Statistically proved soundness on RMAT graphs
- Validated on some real world graphs

Future Works

- Complexity of coloring 2D 9pt stencil with intervals?
- Can we do better than 4-approximation for 3D 27-pt stencils?
- Are there other particular graphs it would make sense to consider?
- Can we prove that largest degree first lead to shorter path for some categories of graphs?
- Can we find more applications where coloring with intervals is a good model?

Thank you!

Papers

Dante Durrman and Erik Saule. Optimizing the critical path of distributed dataflow graph algorithms. In Proceedings of IPDPS Workshops (IPDPSW); PDCO, 2023.

Dante Durrman and Erik Saule. Coloring the vertices of 9-pt and 27-pt stencils with intervals. In Proc. of IPDPS, May 2022.

Contact

esaule@uncc.edu

Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No CCF-1652442.