Coloring the Vertices of 9-pt and 27-pt Stencils with Intervals

Dante Durrman[+], Erik Saule[*]
ddurrman@uncc.edu, esaule@uncc.edu

[+]Dept. Mathematics, [*]Dept. Computer Science
The University of North Carolina at Charlotte

IPDPS 2022

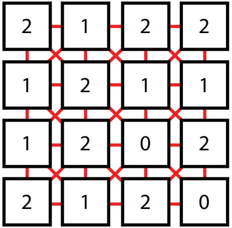
This material is based upon work supported by the National Science Foundation under Grant No CCF-1652442.

Outline

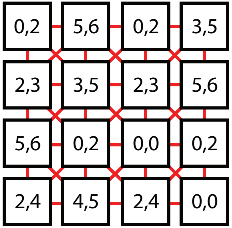
Outline

The Problem of Coloring Stencils with Intervals

2D Example (4x4)

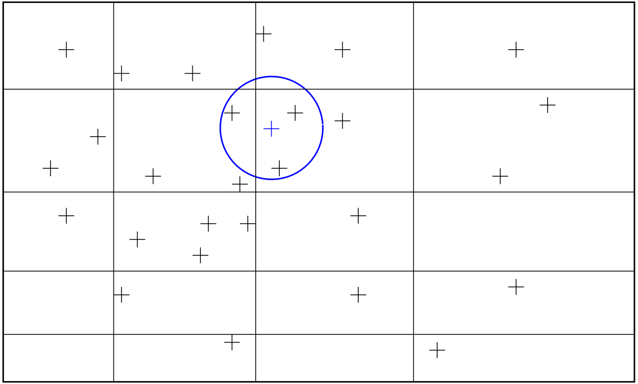


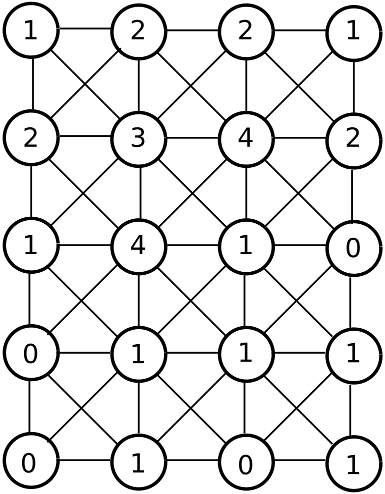
Example Solution



- 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

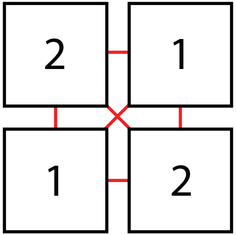




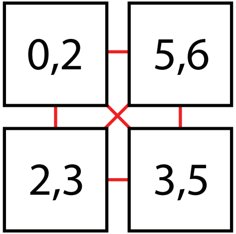
Outline

Cliques can be Colored in Linear Time

K4 Example



K4 Solution



Algorithm

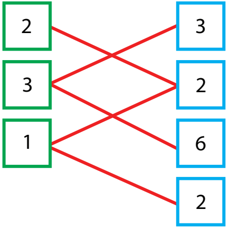
- No vertex can share any color with any other vertex in clique
- We must use at least $\sum_{v \in K} w(v)$ colors
- Greedily color the interval with the lowest available $start(v)$
- Complexity $\Theta(V)$

Implications

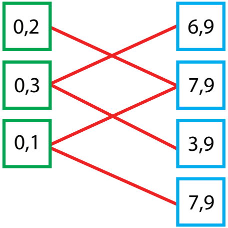
- Each square block of 4 vertices is a K4
  - Sum of weights of K4 is a lower bound of 2D 9-pt stencil
- Each square block of 8 vertices is a K8
  - Sum of weights of K8 is a lower bound of 3D 27-pt stencil

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 $\Theta(E)$

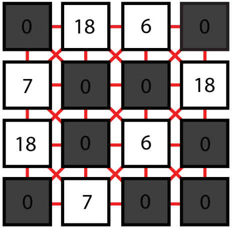
Implications

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

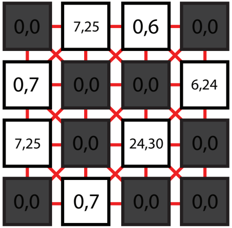
- Each edge in the graph
- 2D 5-pt stencils
- 3D 7-pt stencils
- Many cycles of even length

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

Outline

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

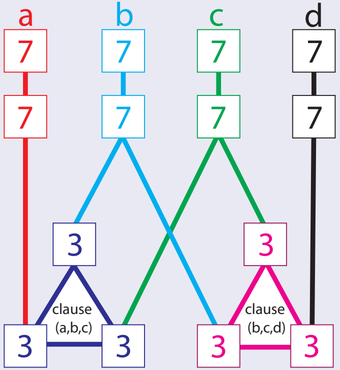
NAE-3SAT is known to be NP-Complete

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

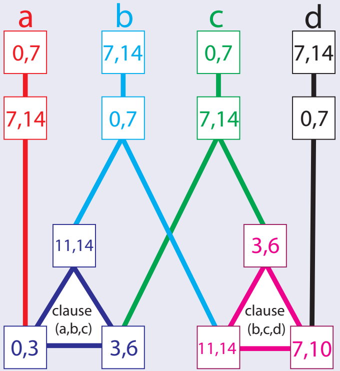
NAE-3SAT Instance

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

Constructed Graph Instance

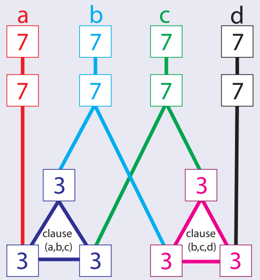


Solution in 14 Colors



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

From Graph ...



... To a 27-pt 3D stencil

Layer 1

0	0	0	0	0	0	0	0	0	0	7	7	0	0	0	0	0	0	0
0	0	7	7	7	7	7	7	7	7	0	0	7	0	0	0	7	7	0
0	7	0	0	0	0	0	0	0	0	0	0	7	0	0	**3**	0	0	7
0	7	0	0	7	7	7	7	7	7	7	0	0	7	**3**	**3**	0	0	7
0	7	0	7	0	0	0	0	0	0	0	7	0	0	0	0	7	0	7
0	7	0	7	0	0	7	7	7	7	0	0	7	0	0	7	0	0	7
0	7	0	7	0	7	0	0	0	0	7	0	0	7	7	0	0	0	7
0	7	0	7	0	7	0	7	0	0	0	7	0	0	0	0	0	7	0
0	0	0	0	0	0	0	0	0	0	0	0	7	7	7	7	7	0	0

Layer 2

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	7	0	7	0	7	0	7	0	0	0	0	0	0	0	0	0	0	0

Layer 3

0	0	0	0	0	0	0	0	0	0	7	7	0	0	0	0	0	0	0
0	0	0	0	7	7	7	7	7	7	0	0	7	0	0	0	7	7	0
0	0	0	7	0	0	0	0	0	0	0	0	7	0	0	**3**	0	0	7
0	0	0	7	0	0	7	7	7	7	7	0	0	7	**3**	**3**	0	0	7
0	0	0	7	0	7	0	0	0	0	0	7	0	0	0	0	7	0	7
0	0	0	7	0	7	0	0	7	7	0	0	7	0	0	7	0	0	7
0	0	0	7	0	7	0	7	0	0	7	0	0	7	7	0	0	0	7
0	7	0	7	0	7	0	7	0	0	0	7	0	0	0	0	0	7	0
0	0	0	0	0	0	0	0	0	0	0	0	7	7	7	7	7	0	0

Outline

Greedy

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

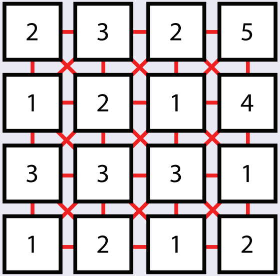
- Greedy Largest First
- Greedy Line by Line
- Greedy Z-Order

By Set

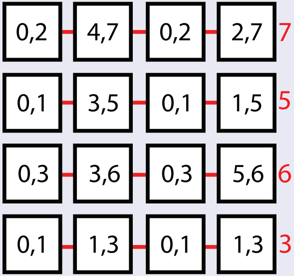
- Greedy Largest Clique First
  - Schedule vertices in the largest clique first; order within clique uses vertex id
- Smart Greedy Largest Clique First
  - Permute each clique and use the order with least *maxcolor*

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

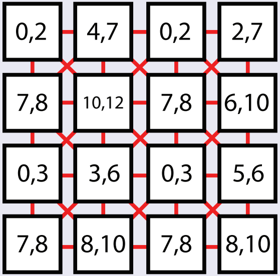
Example



Do Rows Independently



Shift Odd Rows by Max Color



Bipartite Decomposition with Post Optimization

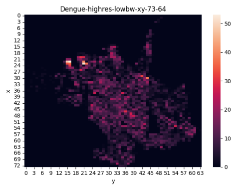
- Sort K4 or K8 (in 3D) by non-increasing order by the sum total of their weights
- Sort vertices within K4 by increasing order of lowest value in their scheduled interval
- Recolor each vertex one at a time using a greedy principle

Outline

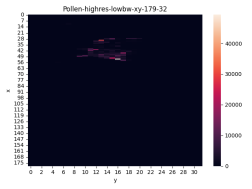
Experimental Settings

Instances from an Application

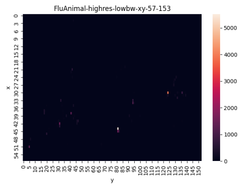
(lat,long) projection of (lat,long,time) instances



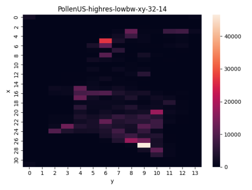
Dengue



Pollen



FluAnimal



PollenUS

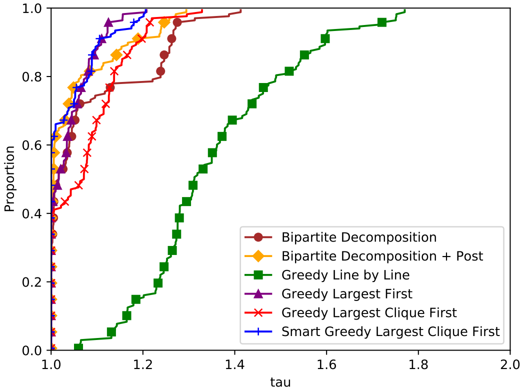
Software

- Algorithms written in Python
- ILP modeled in python-mip
  - Solved with Gurobi

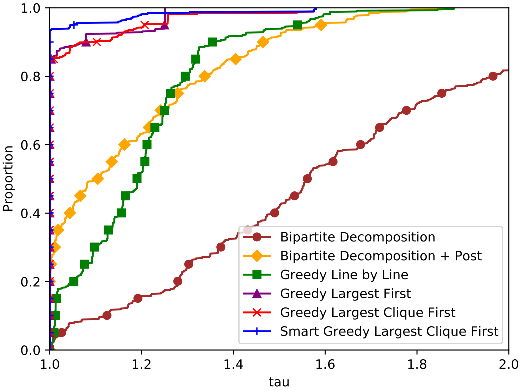
Machine

- Intel i9-9900K
- Windows 10
- CPython 3.9.4

Instance Structure Impacts Algorithm Performance Significantly

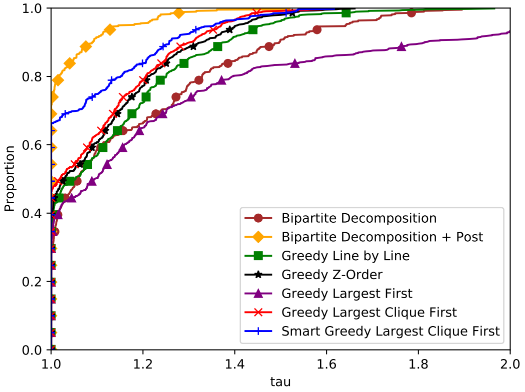


Pollen

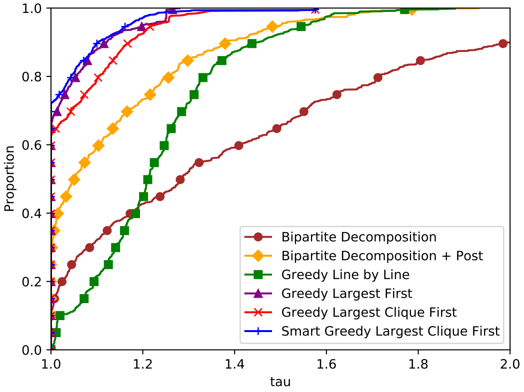


FluAnimal

Bipartite Decomposition is Particularly Good in 2D

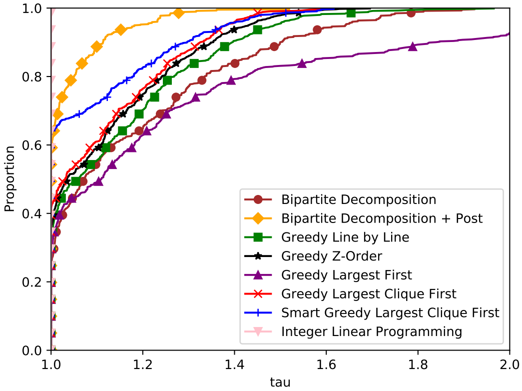


2D

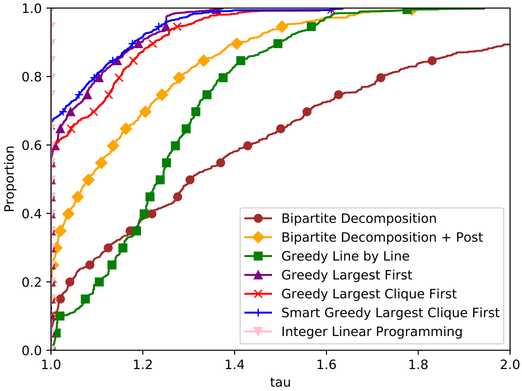


3D

Heuristics find Solutions Close to Optimal



2D

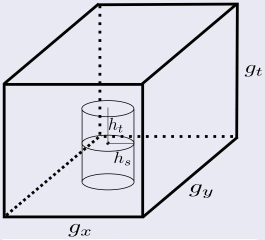


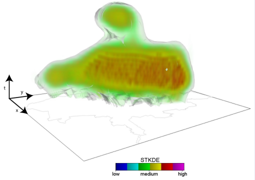
3D

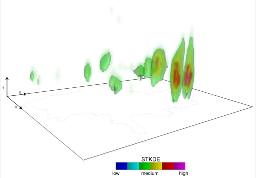
Outline

Problem of Space-Time Kernel Density Estimation

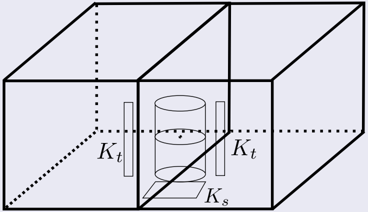
Space-Time Kernel Density Estimation





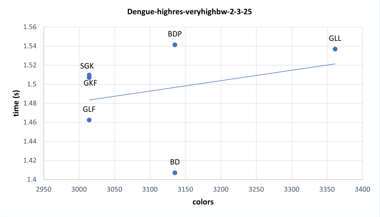


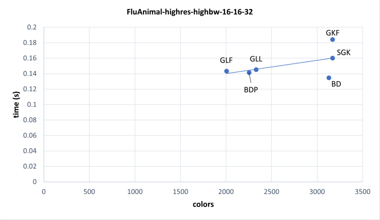
Parallelizing over Distant Boxes

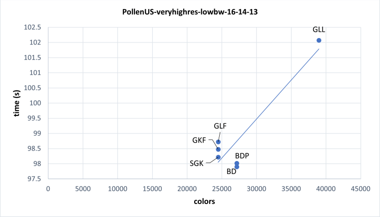


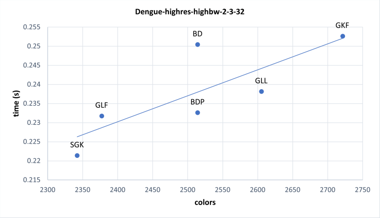
When parallelizing boxes, the points in a box can spill to neighboring boxes.
This creates the constraint that two neighboring boxes can't be processed at the same time.
Hence the stencil structure.

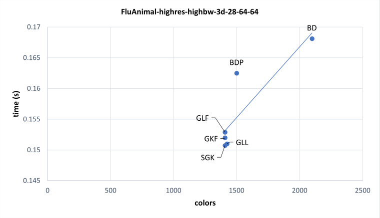
Number of Colors Correlate with Runtime

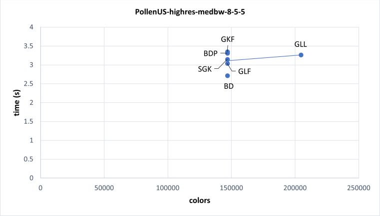












Outline

Conclusion

Work Accomplished

- Load balancing some spatial applications as interval coloring of stencil graphs
- Solved sub-problems
  - Clique
  - Bipartite graphs
  - Odd cycles
- Proved that interval coloring of 3D 27-pt stencil is NP-Complete
- Designed heuristics, including
  - 2-approximation for 2D 9-pt stencil
  - 4-approximation for 3D 27-pt stencil
- Evaluated heuristics
- Confirmed model validity on the STKDE application

Open Problems

- Can we find the odd cycles of highest number of colors in polynomial time?
- Is interval coloring 2D of 9-pt stencil NP-Complete?
- Are there better approximation algorithms than BDP for 3D 27-pt stencil?

Future Works

- Study cost of coloring vs benefit of coloring in STKDE application
- Use interval coloring in other applications