Seminar summary of "Interval Coloring of Stencil Graphs"

Camilo Tello Fachin 12127084
TU Wien (Computational Science & Engineering)
Vienna, Austria
e12127084@student.tuwien.ac.at

1 ABSTRACT

The paper 'Coloring the Vertices of 9-pt and 27-pt Stencils with Intervals' by D. Durrman and E. Saule [1] is the subject of this summary which is done in the context of the seminar in Algorithms/Software Engineering Winter Semester 2023/24 at TU Vienna. In essence, the authors used graph theory to schedule parallel tasks which they then executed according to the found solution of their graph problem. They successfully derived and proved upper bounds for their algorithms, the most relevant of which is the bipartite decomposition for 2D stencils, which is at most twice as slow or less efficient as the optimal solution of the problem. The authors extended also to the 3D stencils where they arrived at an upper bound of their solution being up to 4 times slower or less efficient than the optimal solution. They successfully used their algorithm on the space-time-kernel density estimation to obtain faster solutions to this known data processing problem which benefits vastly from parallelization of subtasks which are not adjacent to each other in space and time, see [2] and [3]. In order to find the true solution qualities of their algorithms, they solved a large part of the problems with an Integer Program to obtain optimal solutions. For their best algorithm, Bipartite Decomposition + Post, 95% of the solved instances were at most 20% worse than the best solution. The + Post part being a greedy procedure recoloring each vertex if neighbors admit it.

2 INTRODUCTION

For a given parallel processing problem, one can model processing tasks that cannot be computed at the same time with 2 vertices connected by an edge. For a large problem, the simple solution is to solve the Vertex Coloring Problem of an undirected graph $G(V, E)$. Suppose the single tasks have different computational intensity, i.e. take varying amout of time for completion, the classical vertex coloring problem is rendered suboptimal since it will most certainly introduce idling times during parallel execution. The Interval Vertex Coloring Problem or short IVC Problem is a more suitable model for such an instance. The authors of the paper focused specifically on 2D and 3D stencils, which often occur in data processing such as the space-time-kernel density estimation or short STKDE. In 3D one can define a global voxel size, the number of datapoints within one voxel is approximatively proportional to the computational costs and is defined as a weight, a voxel is a weighted vertex connected to all other vertices that correspond to voxels which are adjacent to it, in 3D this corresponds to a 27-Pt Stencil whereas in 2D it corresponds to a 9-pt stencil. Meaning in 3D, if data is processed from one voxel, all the 26 adjacent voxels cannot be processed concurrently. The authors relied on the results of Saule et. al. [3] (author of main paper as well) and Hohl et. al. [2] where the focus was more on space-time-kernel density computation itself. Data that is used to compute STKDE is strictly composed of 2D and 3D stencil minors which is why the authors claim their results to be relevant. The authors exploit the structure of such graphs composed of 2D or 3D stencils to design algorihtms that find very good solutions to the IVC-Problems given. The defintion Interval Vertex Coloring Problem is somewhat misleading since it actually doesnt have anything to do with colors. The Problem broken down informally can be understood as: every vertex has assigned an interval, find interval assignments such that no two adjacent vertices have overlapping intervals. The largest interval end in the graph is then approximately the total runtime of the parallel program.

3 IVC PROBLEM DEFINITION

The formal definition of the IVC-Problem is as follows:

Interval Vertex Coloring Problem
Instance: An undirected graph $G(V, E, \mathrm{w})$ where $\mathrm{w} : \mathrm{V} \mapsto \mathbb{Z}^+$ and $\forall \mathrm{v} \in \mathrm{V}\, \mathrm{w}(\mathrm{v}) > 0$
Problem: Find a coloring start : $\mathrm{V} \mapsto \mathbb{Z}^+$ s.t. $\forall(a, b) \in \mathrm{E}$:
$[\text{start}(a), \text{start}(a) + w(a)) \cap [\text{start}(b), \text{start}(b) + w(b)) = \emptyset$.
and $\min_{\text{start}}$ maxcolor(start)
where maxcolor(start) = $\max_{\mathrm{v} \in \mathrm{V}} start(\mathrm{v}) + w(\mathrm{v})$

This is an optimization problem where the solution start is a minimizer of maxcolor. The number maxcolor is a number that is proportional to some elapsed time measurement and should coincide with the time when the parallel program is done. From here on, maxcolor* denotes a maxcolor that is indeed minimal.

The special case analysis deals with only 2 types of instances, namely $G(V, E)$ eihter as 9-pt or 27-pt stencils, which arise from 2D or 3D datasets respectively.

Definition 3.1. A graph $G(V, E)$ is a 9-pt stencil, composed of $X \times Y$ vertices on a 2D grid such that two vertices (i, j) and (i', j') are connected iff $|i - i'| \leq 1$ and $|j - j'| \leq 1$.

Definition 3.2. A graph $G(V, E)$ is a 27-pt stencil, composed of $X \times Y \times Z$ vertices on a 3D grid such that two vertices (i, j, k) and (i', j', k') are connected iff $|i - i'| \leq 1$ and $|j - j'| \leq 1$ and $|k - k'| \leq 1$.

These two types of graphs are the ones the authors dealt with primarily, because they have minors composed of cliques, bipartites, cycles and paths, which are their main ingredients to derive bounds.

4 SPECIAL CASE ANALYSIS

Cliques

Cliques can be colored trivially, where one just assigns start(v) greedily with ascending w(v). This can be done in O(n) and yields maxcolor* = $\sum_{v \in V} w(v)$

Bipartite Graphs

Any path and any even cycle is a bipartite graph. This is relevant since the 2D and 3D stencils both contain bipartite graphs. Here one can rather quickly identify another coloring of simpler structure than the problem may suggests. Since a bipartite graph has edges only between vertex sets A and B, the edges are lower bounds maxcolor* $\geq w(i) + w(j) \, \forall (i, j) \in E$. A simple algorithm can color bipartite graphs in $O(E)$

Odd Cycles

The analysis of odd cycles requires a few definitions introduced:

Definition 4.1. Let maxpair be the maximal sum of any two vertex weights s.t. maxpair = $\max_i w(i) + w(i+1)$. From now on, w(.) can also return multiple weights s.t. $w(i) + w(i+1) = w(i, i+1)$.

Definition 4.2. Let minchain3 be the minimal sum of any three vertex weights s.t. minchain3 = $\min_i w(i, i+1, i+2)$.

Theorem 4.3. *If $G(V, E)$ is an odd cycle, then* maxcolor* = max(maxpair, minchain3) *[1].*

The theorem is proven by the authors by establishing lower and upper bounds on maxcolor* that coincide.

Lemma 4.4. *If $G(V, E)$ is an odd cycle, there is an algorithm that yields* max(maxpair, minchain3) *as color (largest interval end). This implies that* maxcolor$*$ ≤ max(maxpair, minchain3).

The lemma is proven by construction of an algorithm that correctly yields such a coloring.

Since algorithm 1 yields a valid coloring, this is for sure an upper bound to the optimal coloring maxcolor*. Since maxpair is always a lower bound on the number of colors, maxpair would always pose a lower bound on maxcolor* if it were larger than minchain3. Therefore it is sufficient to show the following lemma.

Lemma 4.5. *If $G(V, E)$ is an odd cycle, maxcolor* ≥ minchain3, which implies maxcolor* ≥ max(maxpair, minchain3) if minchain3 ≥ maxpair.*

The authors set K = maxchain3 and impose a coloring of K-1 on a general odd cycle graph, the always invalid coloring of K-1 leads to a contradiction which is sufficient to prove theorem 4.3. □

5 HEURISTICS & ALGORITHMS

The authors used different heuristics to find colorings to the instances FluAnimal, Dengue and Pollen which are datasets containing the spatio-temporal occurences of the diseases. Like mentioned before, these datasets are used to compute the STKDE, which are computed in parallel after computing a interval coloring from the graph corresponding to the dataset. The authors deployed a few different variations of greedy methods, while their bipartite decomposition + post method performed best. The basis of it being the bipartite decompositon shown in Algorihtm 2.

Algorithm 1 Interval Coloring Algorithm for Odd Cycles [1]

```
1:  Input: G(V,E,w)
2:  Output: start array with color intervals for each vertex
3:  Initialize maxpair, minchain3, start array with length equal to
    number of vertices
4:  Calculate maxpair and minchain3 based on the graph's proper-
    ties
5:  start[0] ← 0                                    ▷ Color vertex 0
6:  start[1] ← w(0)                                 ▷ Color vertex 1
7:  start[2] ← max(maxpair, minchain3) - w(2)       ▷ Color vertex 2
8:  for each vertex $x$ in vertices starting from 3 do
9:      if $x$ is odd then
10:         start[$x$] ← 0
11:     else if $x$ is even then
12:         start[$x$] ← max(maxpair, minchain3) - w($x$)
13:     end if
14: end for
15: for each vertex $x$ in vertices do
16:     for each vertex $y$ adjacent to $x$ do
17:         Ensure:
18:         [strt[$x$], strt[$x$] + w($x$)) ∩ [strt[$y$], strt[$y$] + w($y$)) = ∅
19:     end for
20: end for
21: return start
```

Algorithm 2 Bipartite Decomposition Coloring for 2DS-IVC [1]

```
Require: $X, Y$, weights $w(x, y)$ for each vertex $(x, y)$
1:  Initialize color intervals $c[x][y] \leftarrow 0$ for all $x, y$
2:  $RC \leftarrow 0$                     ▷ Maximum color used by any row
3:  for $y = 0$ to $Y - 1$ do
4:      for $x = 0$ to $X - 1$ do
5:          $c[x][y] \leftarrow$ OptimalBipartiteColoring$(x, y)$
6:          $RC \leftarrow \max(RC, c[x][y] + w(x, y))$
7:      end for
8:  end for
9:  for $y = 0$ to $Y - 1$ do
10:     for $x = 0$ to $X - 1$ do
11:         if $y \mod 2 = 0$ then
12:             $start \leftarrow c[x][y]$                ▷ Even rows
13:         else
14:             $start \leftarrow RC + c[x][y]$           ▷ Odd rows
15:         end if
16:         ColorVertex$(x, y, start)$
17:     end for
18: end for
19: return MaxColor
```

Theorem 5.1. *The bipartite decomposition algorithm is a 2-Approximation algorihtm for the 2DS IVC Problem [1].*

The bipartite decomposition algorithm always finds a solution which is at most 2 times worse than the best solution. In the 3D case the authors came up with an algorithm with an analogous procedure as in the just shown 2D case with the following result:

Theorem 5.2. *The bipartite decomposition algorithm is a 4 approximation algorithm for the 3DS IVC Problem [1].*

The algorithm bipartite decomposition which itself is in $O(XY)$ is succeeded by a post optimization which can be implemented in $O(\mathrm{n}\log(\mathrm{n}))$. The post optimization takes a solution found by the bipartite decomposition algorithm and, in 2D, lists vertices as members of a K_4, after that all K_4's are sorted in non-increasing order. Lastly, the vertices are sorted within their K_4 by increasing order of the lowest value in their interval. This produces an ordering of vertices that can be recolored one at a time.

6 EXPERIMENTAL RESULTS

A portion of the instances were solved with a Integer Program to obtain optimal solutions. The bipartite decomposition + post dominates in the 2D instances, where more than 90% of the instances had solutions that were up to at most 1.2 times as bad as the best solution, see figure 1, whereas in the 3D case the algorithm smart greedy largest clique first seemed to dominate, see 2. The elapsed time to find solutions in 2D is quite intriguing as well, where the best algoritm is among the fastest, see figure 3.

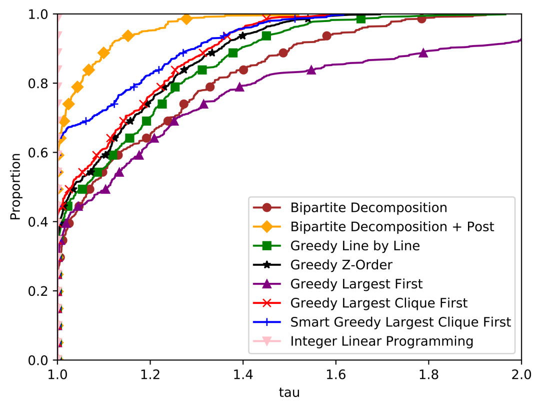

Figure 1: 2D Instances [1].

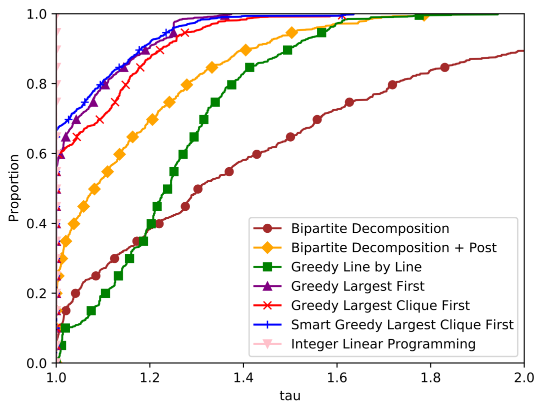

Figure 2: 3D Instances [1].

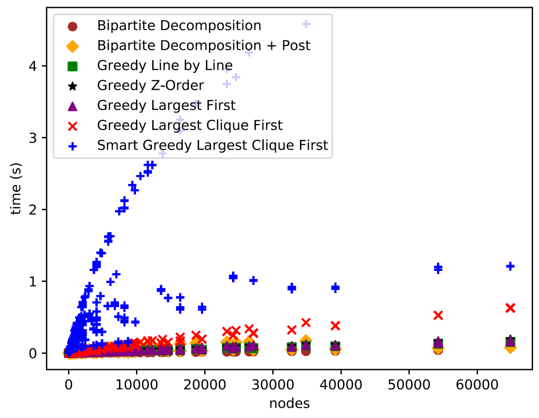

Figure 3: Execution Time (ET) for coloring solution in 2D instances [1].

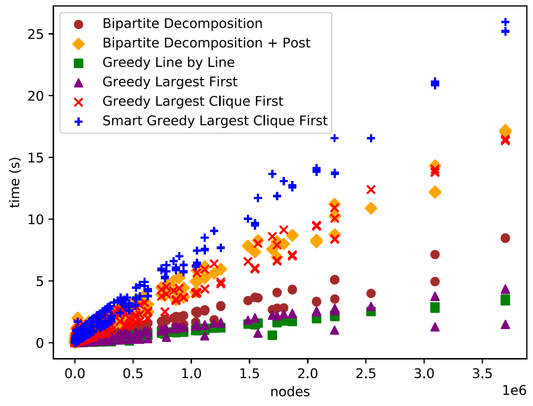

Figure 4: Execution Time (ET) for coloring solution in 3D instances [1].

7 CONCLUSION

The authors have additionally proven the NP-Completeness of the Interval Vertex Coloring Problem for 3D stencils while failing to deliver the proof for the 2D stencil case. While this critique seems harsh it is worth noting that the authors delivered algorithms that yield approximation solutions which were quite good, 90% of the solutions only 1.2 times as bad as optimal solution, and are proven to run in linear time. Nevertheless future work could contain the proof of the 2D case and extensions to graphs other than the 2D and 3D stencils.

REFERENCES

[1] Durrman, D., and Saule, E. Coloring the vertices of 9-pt and 27-pt stencils with intervals. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2022), pp. 963–973.
[2] Hohl, A., Delmelle, E., Tang, W., and Casas, I. Accelerating the discovery of space-time patterns of infectious diseases using parallel computing. *Spatial and Spatio-temporal Epidemiology 19* (2016), 10–20.
[3] Saule, E., Panchananam, D., Hohl, A., Tang, W., and Delmelle, E. M. Parallel space-time kernel density estimation. *2017 46th International Conference on Parallel Processing (ICPP)* (2017), 483–492.