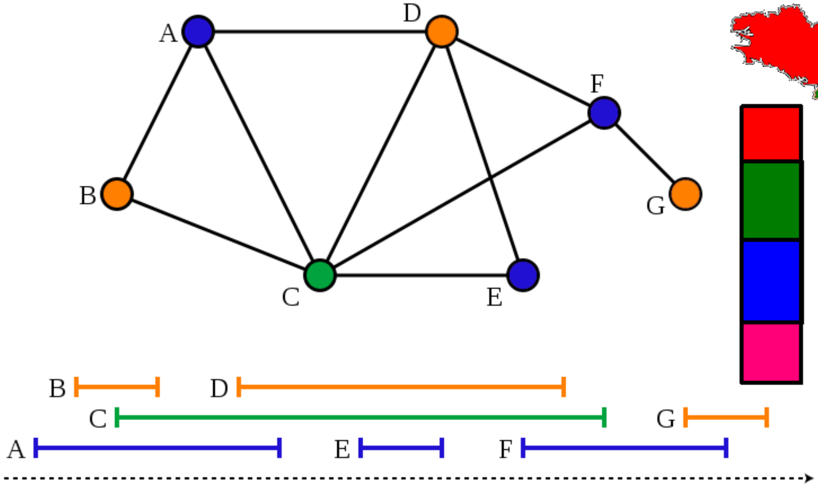
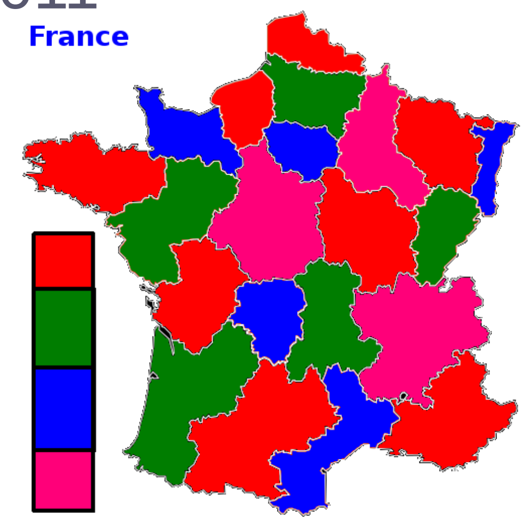
PARALLEL FIRST FIT COLORING

CSE 6490A Winter 2011

Loutfouz Zaman





			2	6		7		1
6	8			7			9	
1	9				4	5		
8	2		1				4	
		4	6		2	9		
	5				3		2	8
		9	3				7	4
	4			5			3	6
7		3		1	8			

Overview

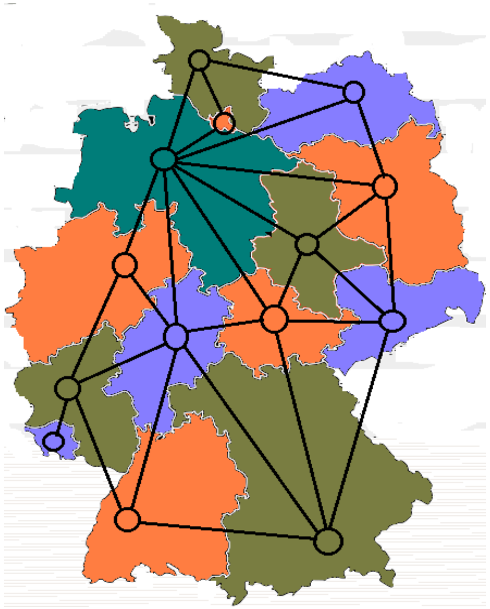
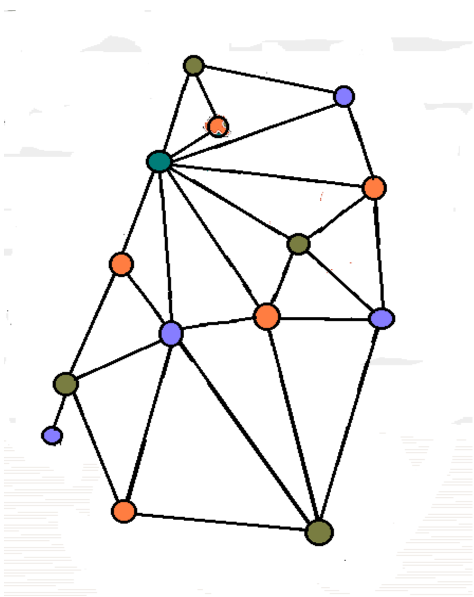
- What is graph coloring?
- Applications
- Related work
- Sequential First-Fit
- Parallel First-Fit
- Demo

Graph coloring

- Assignment of *"colors"* to certain objects in a graph subject to certain constraints
  - Vertex coloring
  - Edge coloring
  - Face coloring (planar)

Vertex coloring

- Coloring vertices of graph such that no two adjacent vertices share same color
- Edge and Face coloring can be transformed into Vertex version
- Edge coloring is vertex coloring of its line graph

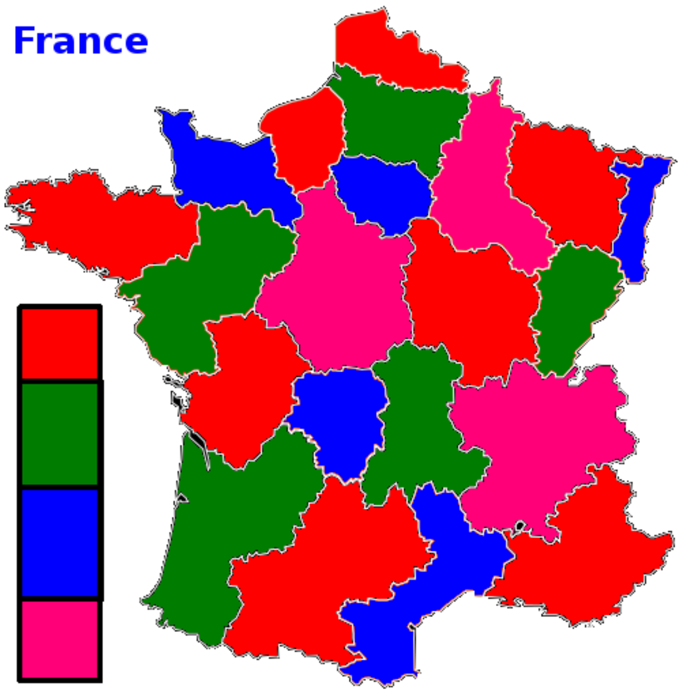



Chromatic Number

- χ - least number of colors needed to color a graph
  - Chromatic number of a complete graph:

$$\chi(K_n) = n$$

- χ(G) = 1 if and only if G is totally disconnected
- χ(G) ≤ 4, for any planar graph
  - The "four-color theorem"
    - More later

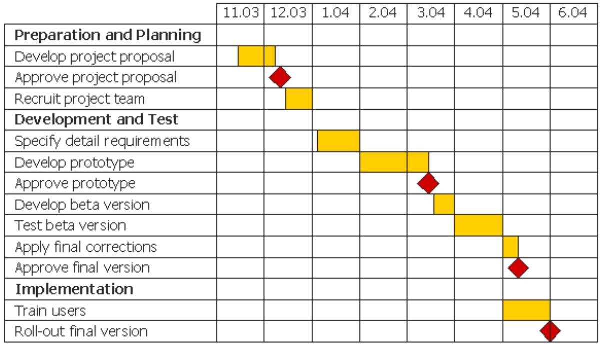


Applications of Graph Coloring

- Scheduling
- Register Allocation
- Sudoku

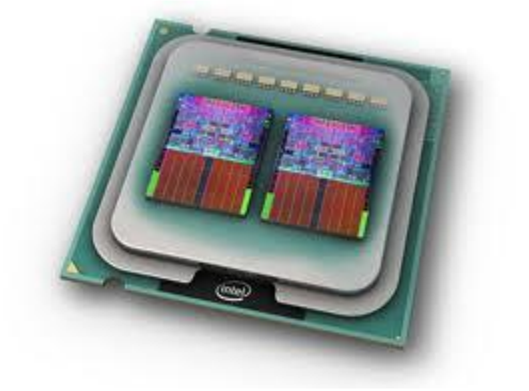
Scheduling (1)

- ***Job scheduling***
  - Schedule of interfering jobs
  - Conflict graph
    - Vertices for jobs
    - Edges, if jobs can't be executed at the same time
    - Colors – time slots



- ***Aircraft scheduling***
  - *k* aircrafts, *n* flights (k<n)
  - 2 flights overlap, same aircraft can't be used
  - Conflict graph
    - Vertices – flights
    - Edges, if flights overlap
    - Colors - aircrafts

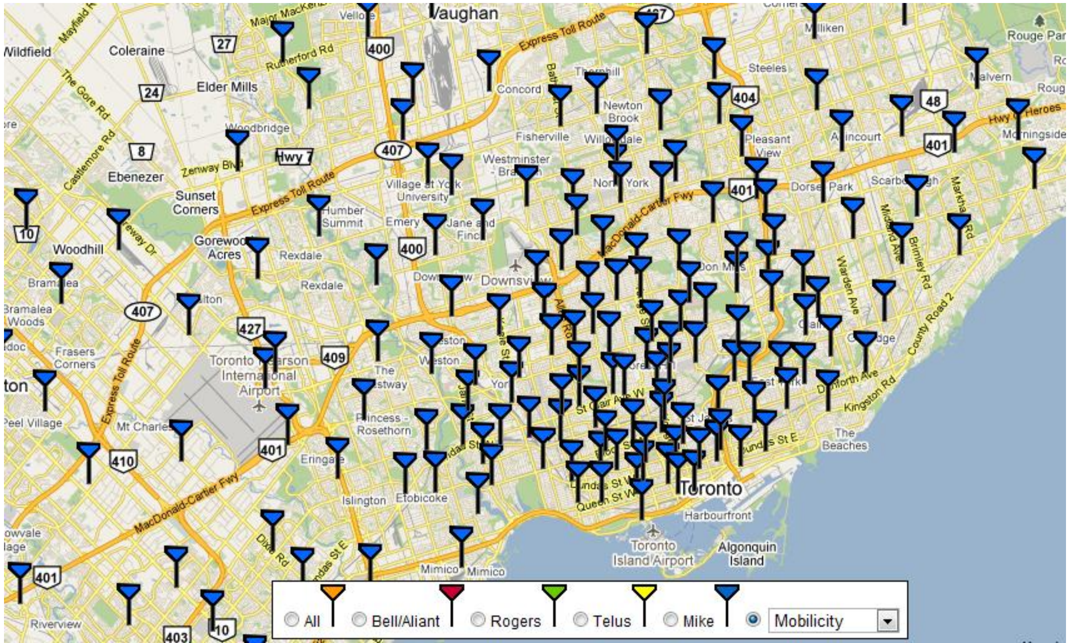
Scheduling (2)



- ***Bi-processor tasks***
  - K processors, n tasks
  - Each task has to be executed on pre-assigned processors simultaneously
  - Processor can’t execute 2 jobs at same time
    - E.g. schedule file transfers between processors
    - E.g. mutual diagnostic testing of processors
  - Graph
    - Vertices – processors
    - Edge – task between two processors
    - Edge coloring – edge appears at most once at a vertex

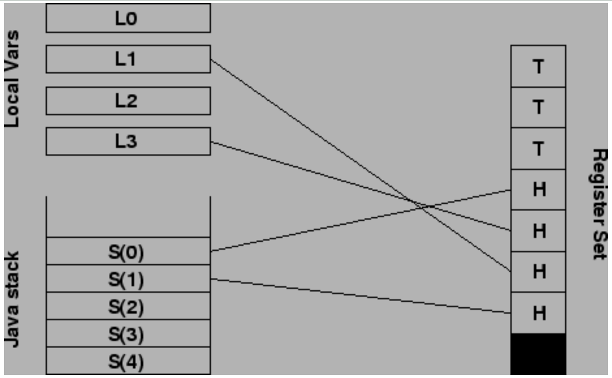
Scheduling (3)

- ***Frequency assignment***
  - Radio stations at locations marked (x,y)
  - Frequency assigned to each station
    - Interference, must receive different frequencies those that are close
    - E.g. frequency assignment of base stations in cellular phone networks
  - Solved using a 3-approximation algorithm for coloring unit disk graphs



Scheduling (4)

- ***Multi-coloring***
  - Earlier example: jobs to have more than one time slots.
- ***Pre-coloring extension problem***
  - unassigned vertices using the minimum number of colors
- ***List coloring problem***
  - only in certain time slots or machines available
  - Colors are takes from a list of available colors
- ***minimum sum coloring***
  - sum of the colors assigned to the vertices is minimal
  - E.g. minimize sum of job completion times ->minimize average completion time



Register Allocation

- Compiler optimization
- Frequently used values are kept in fast processor registers
  - build interference graph (*G*) of program
  - if variables interfere, can't be assigned to same register
  - Given k register, find *k*-coloring of *G*
  - Uncolored variables are "spilt" into memory
- Recent findings
  - Heuristic approach better allocation than optimal (counter-intuitive) (Koes and Goldstein 2006)

Sudoku (1)

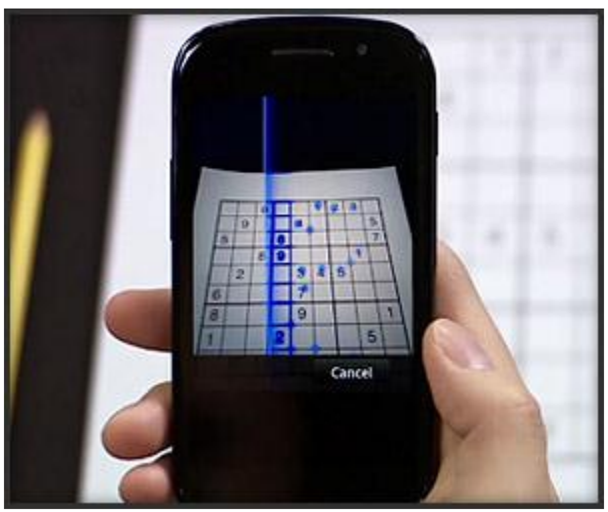
- Fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 sub-grids that compose the grid contains all of the digits from 1 to 9

			2	6		7		1
6	8			7			9	
1	9				4	5		
8	2		1				4	
		4	6		2	9		
	5				3		2	8
		9	3				7	4
	4			5			3	6
7		3		1	8			

4	3	5	2	6	9	7	8	1
6	8	2	5	7	1	4	9	3
1	9	7	8	3	4	5	6	2
8	2	6	1	9	5	3	4	7
3	7	4	6	8	2	9	1	5
9	5	1	7	4	3	6	2	8
5	1	9	3	2	6	8	7	4
2	4	8	9	5	7	1	3	6
7	6	3	4	1	8	2	5	9

Sudoku (2)

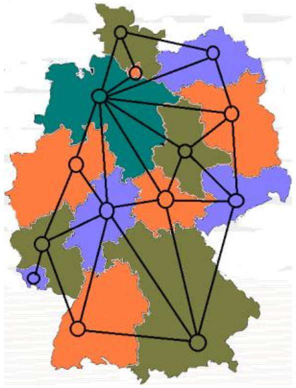
- Can be viewed graph coloring, here is how:
  - Each one of 81 squares is vertex in graph
  - Edge connects every pair of vertices whose squares are buddies
  - Each vertex connects to 20 other vertices (81x20/2 = 810 edges)
  - Same as to find 9-coloring
  - Also ***pre-coloring extension problem***



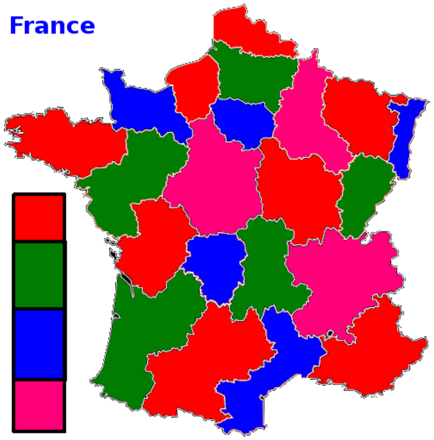
			2	6		7		1
6	8			7			9	
1	9				4	5		
8	2		1				4	
		4	6		2	9		
	5				3		2	8
		9	3				7	4
	4			5			3	6
7		3		1	8			

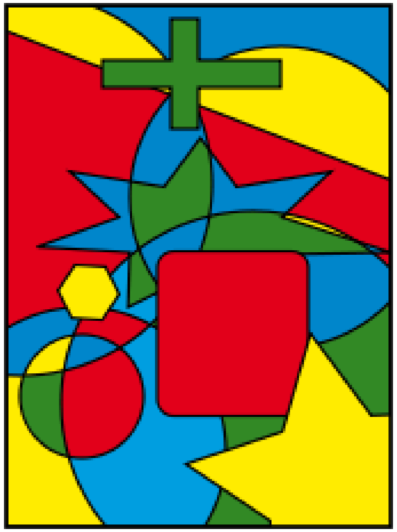
4	3	5	2	6	9	7	8	1
6	8	2	5	7	1	4	9	3
1	9	7	8	3	4	5	6	2
8	2	6	1	9	5	3	4	7
3	7	4	6	8	2	9	1	5
9	5	1	7	4	3	6	2	8
5	1	9	3	2	6	8	7	4
2	4	8	9	5	7	1	3	6
7	6	3	4	1	8	2	5	9

Related Work (1)



- Four-Color Theorem
  - Dates back to 1852 to Francis Guthrie
  - Any given plane separated into regions may be colored using no more than 4 colors
    - Used for political boundaries, states, etc
    - Shares common segment (not a point)
  - Many failed proofs, until finally proved using a computer (Appel and Haken 1977)
    - Started in 1972
    - took 1200 hours of computer time
    - Finished 4 years later ☺





Related Work (2)

- Studied as algorithmic problem since early 1970s
- Minimal vertex coloring algorithm using brute-force search
  - Christodes 1971
  - Wilf 1984
- Finding minimum coloring: **NP**-hard
  - You can't do it efficiently for large graphs
- Approximations
  - guarantee performance at expense of quality
    - quality = # colors used
  - E.g. Brelaz 1979
    - Good but not minimal solution
    - Minimal for certain type of graphs

Related Work (3)

- State of the art
  - Pushing tradeoff limits between performance and used number of colors
- Schneider and Wattenhofer 2010
  - Algorithm for ***distributed*** symmetry breaking

Colors	Time
$\Delta + 1$	$O(\log \Delta + \sqrt{\log n})$
$O(\Delta + \log n)$	$O(\log \log n)$
$O(\Delta + log^{1+1/log^* n} n)$	$O(log^* n)$
$O(\Delta log^{(c)} n + log^{1+1/c} \; n)$	$O(1)$

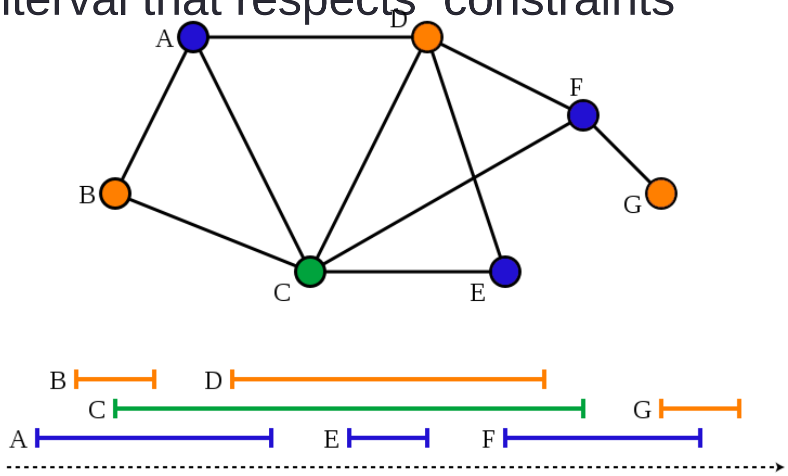
Related Work (4)

- Online coloring
  - Approximation
  - Heuristic algorithms used to produce proper graph coloring which is not necessarily minimal
  - Immediately colors vertices of *G* taken from list without looking ahead or changing colors already assigned
  - Any online algorithm lower bounds (Halldórsson and Szegedy 1994):

Deterministic	Randomized
$\geq 0\left(\frac{2n}{log^{n^2}}\right)$	$\geq 0\left(\frac{n}{16log^{n^2}}\right)$

Related Work (5)

- First-Fit (FF) simplest of all online coloring algorithms
- Assigns smallest possible integer as color to current vertex of *G* (Gyárfás and Lehel 1988)
- Appears extensively with ***interval graphs***
  - Interval graph captures intersection relation for some set of intervals on real line
    - E.g. resource requests arrive dynamically in unpredictable order
    - FF allocates lowest color to current interval that respects constraints imposed by colored intervals



Related Work (6)

- How bad is FF compared to optimal coloring?
  - $\chi_{FF}(G)$ – maximum number of colors used for colorings of G produced by FF for all orderings of vertices of *G*
  - $\chi(G)$ – chromatic number of *G*
  - *Performance ratio of FF:* $R_{FF} = \chi_{FF}(G)/\chi(G)$
  - Recent findings: $5 \leq \chi_{FF}(G) \leq 8$
  - Wan *et al.* (2010) used FF for First-Fit scheduling as an approximation algorithm for minimum latency beaconing schedule

Sequential FF (1)

- Umland (1998) demonstrates a 2-step sequential FF algorithm:
  - **(1)** $\boldsymbol{Build(L_i, v_j)}$: Determine a list L_i of all possible colors for v_i, i.e. exclude colors already used by vertices $v_j, j < i$ adjacent to v_i
    - L_i -- a boolean array (possibility list of v_i) with property:
      - $L_i[k] = false \leftrightarrow \exists v_j$ such that $j < i, (v_i, v_j) \in E$ and $f(v_j) = k$
  - **(2)** $\boldsymbol{Color(L_i, v_i)}$: Determine the smallest of all possible colors for v_i, i.e. look for the smallest entry in L_i where $L_i[k] = true$ and assign color k to v_i

Sequential FF (2)

Algorithm 1 Build(L_i, v_j)

Require: must be executed before $Color(L_i, v_j), \forall j < i$ and requires L_i initialized

Ensure: $L_i[k] = false \Leftrightarrow \exists v_j$ such that $j < i, (v_i, v_j) \in E, f(v_j) = k$

```
for all n in v_i.neighbours do
    if n.index > v_i.index then
        L_{n.index}[v_i.color] ← false
    end if
end for
```

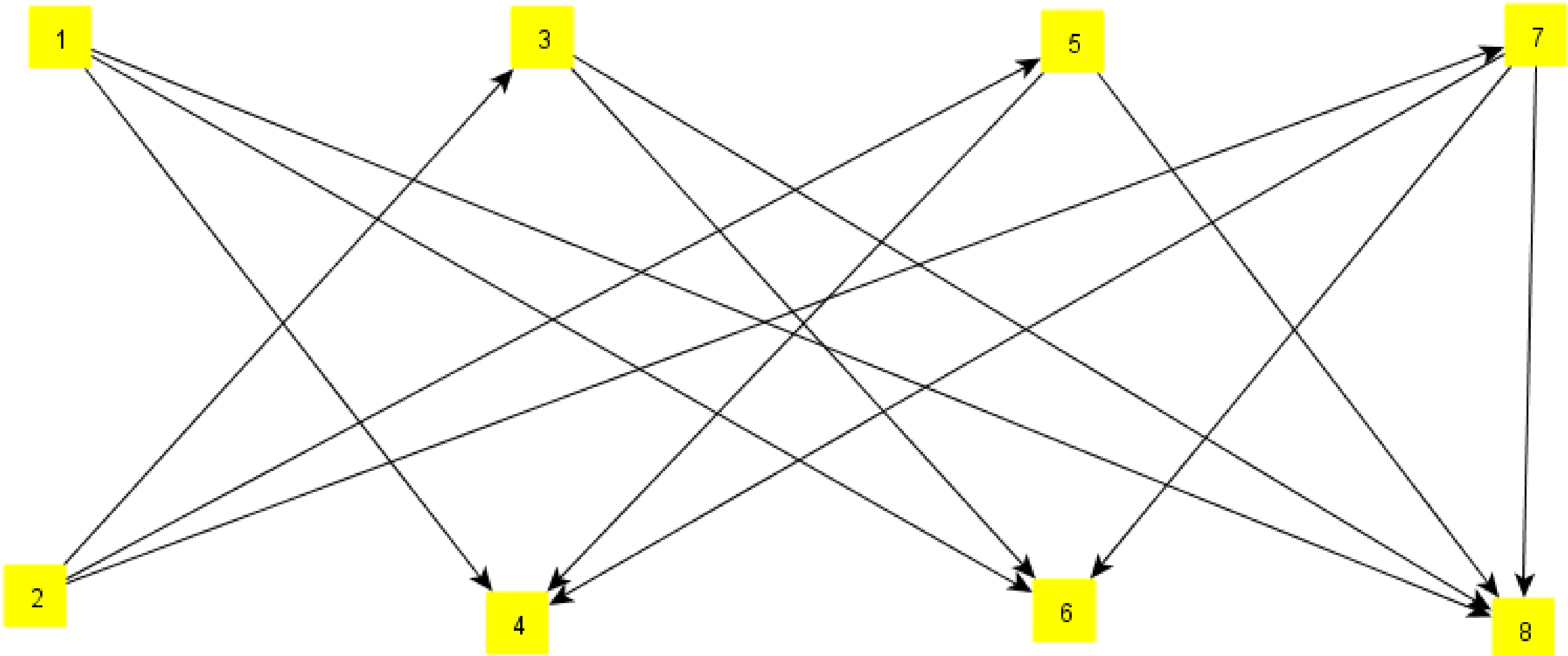
Algorithm 2 Color(L_i, v_i)

Require: must be executed before $Build(L_j, v_i), \forall j > i$

Ensure: v_i has the first color unused by neighbours

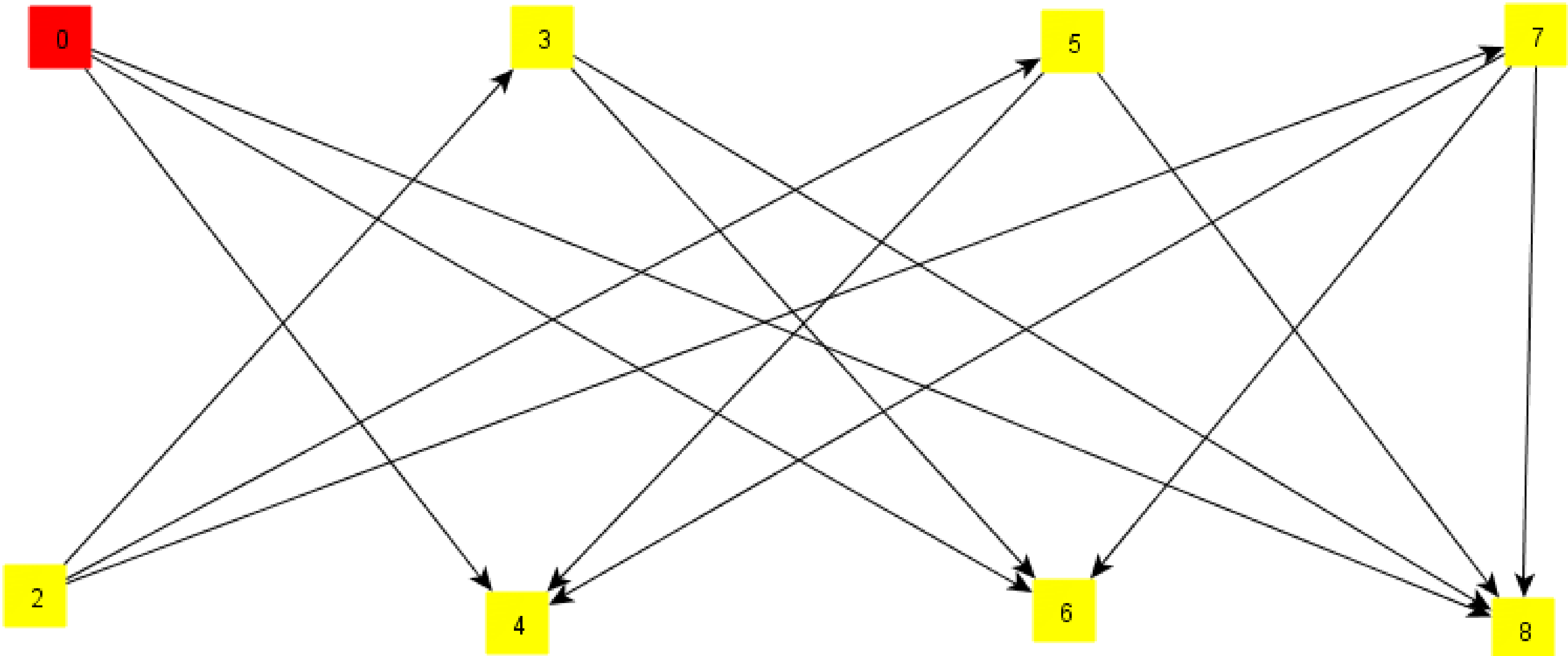
```
for k = 0 to L_i.length do
    if L_i[k] = true then
        v_i.color ← k
        break
    end if
end for
```

Sequential FF E.g. Step 0



Sequential FF E.g. Step 1

$L_1 = \{t, t, t, t\}$,k=0



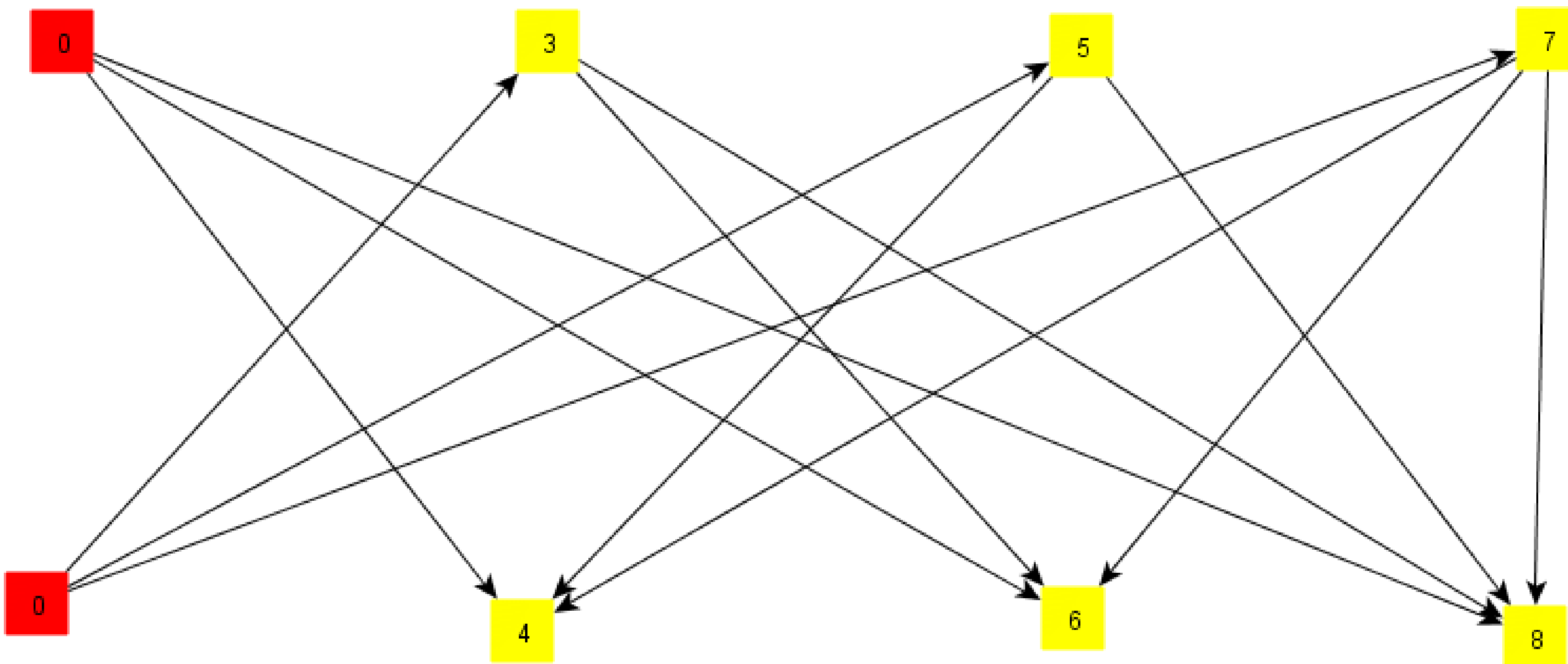
$L_4 = \{f, t, t, t\}$

$L_6 = \{f, t, t, t\}$

$L_6 = \{f, t, t, t\}$

Sequential FF E.g. Step 2

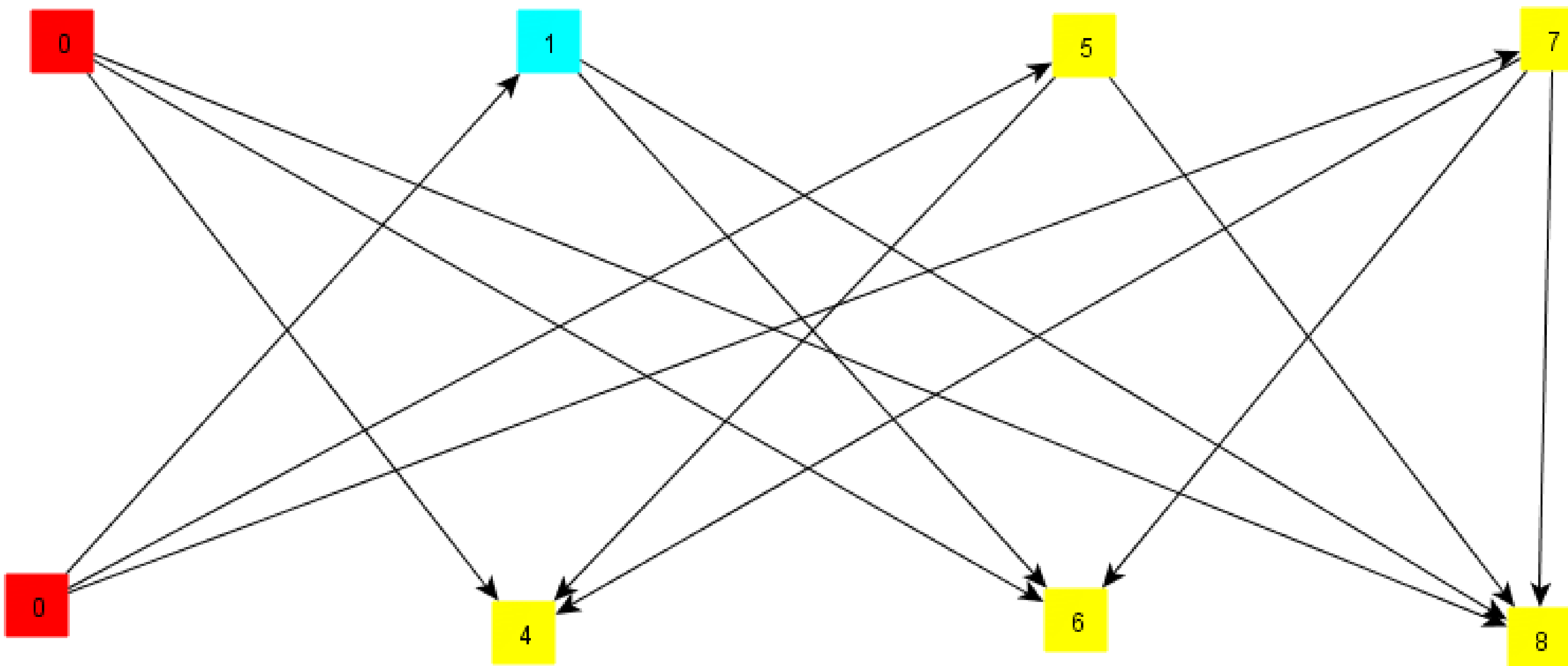
$L_1 = \{t, t, t, t\}$,k=0 $L_3 = \{f, t, t, t\}$ $L_5 = \{f, t, t, t\}$ $L_7 = \{f, t, t, t\}$



$L_2 = \{t, t, t, t\}$,k=0 $L_4 = \{f, t, t, t\}$ $L_6 = \{f, t, t, t\}$ $L_6 = \{f, t, t, t\}$

Sequential FF E.g. Step 3

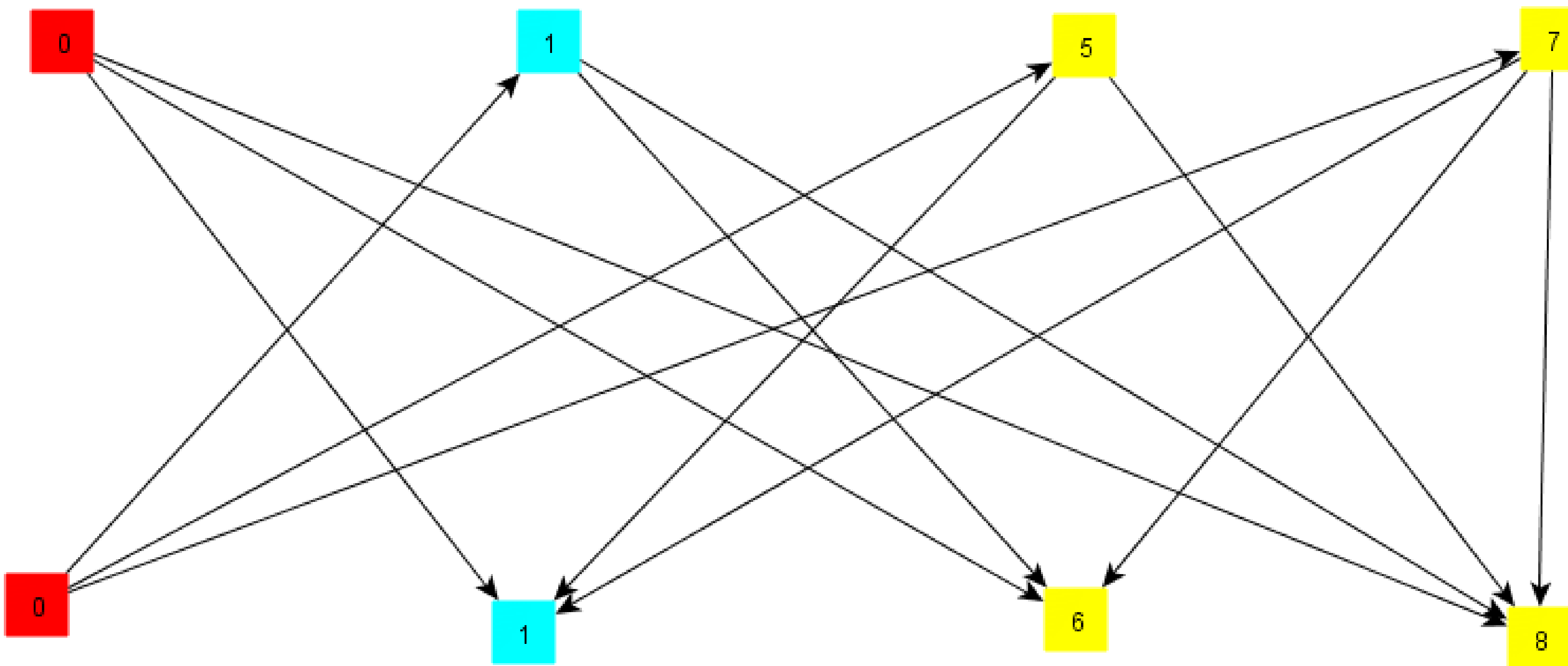
$L_1 = \{t, t, t, t\}$,k=0 $L_3 = \{f, t, t, t\}, k = 1$ $L_5 = \{f, t, t, t\}$ $L_7 = \{f, t, t, t\}$



$L_2 = \{t, t, t, t\}$,k=0 $L_4 = \{f, t, t, t\}$ $L_6 = \{f, f, t, t\}$ $L_6 = \{f, f, t, t\}$

Sequential FF E.g. Step 4

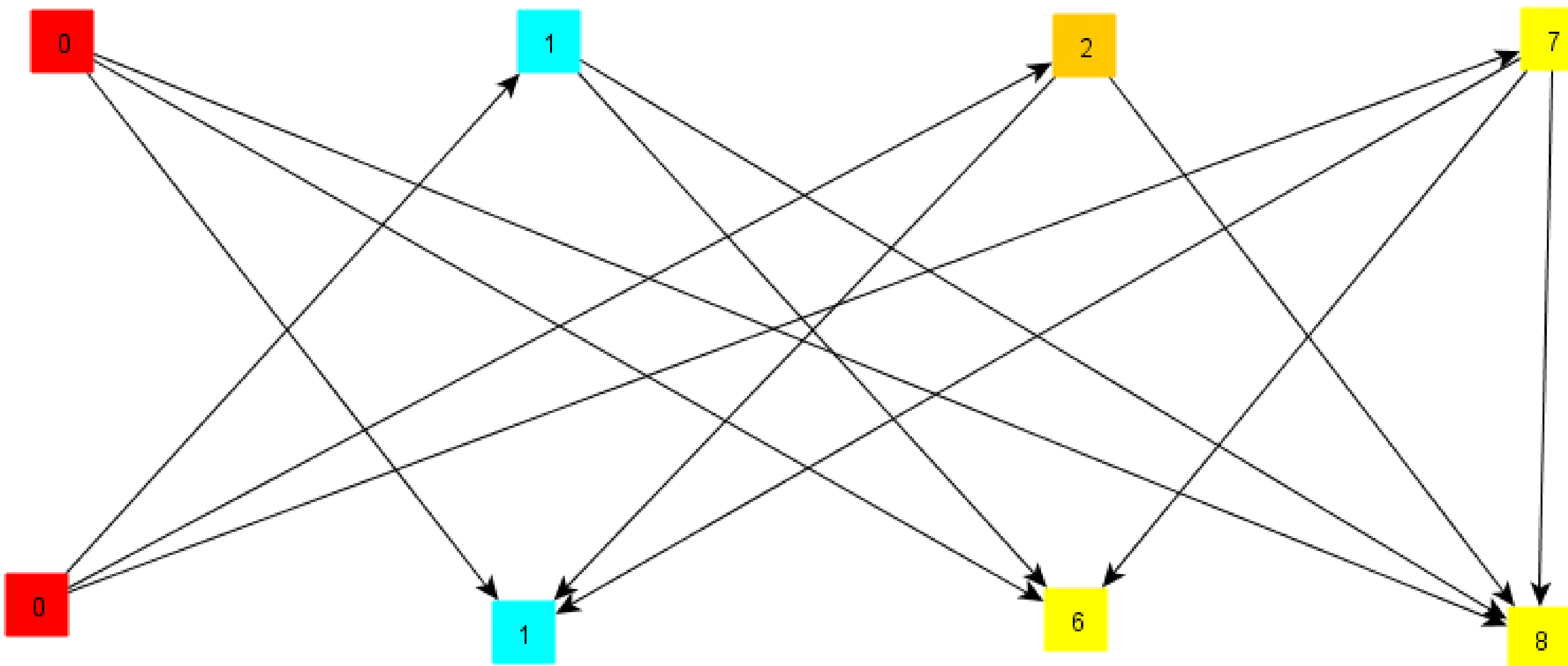
$L_1 = \{t, t, t, t\}$,k=0 $L_3 = \{f, t, t, t\}, k$ =1 $L_5 = \{f, f, t, t\}$ $L_7 = \{f, f, t, t\}$



$L_2 = \{t, t, t, t\}$,k=0 $L_4 = \{f, t, t, t\}$, k = 1 $L_6 = \{f, f, t, t\}$ $L_6 = \{f, f, t, t\}$

Sequential FF E.g. Step 5

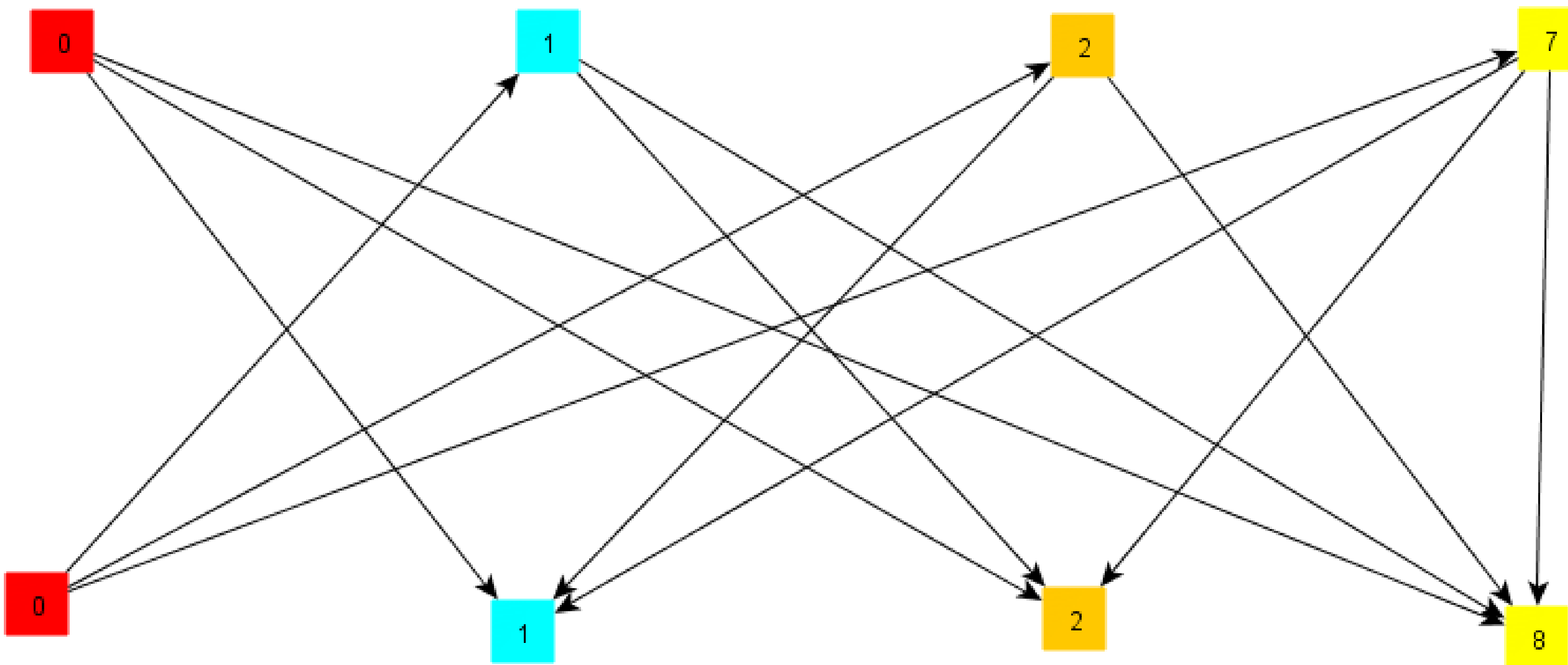
$L_1 = \{t, t, t, t\}$,k=0 $L_3 = \{f, t, t, t\}, k$ =1 $L_5 = \{f, f, t, t\}$,k=2 $L_7 = \{f, f, t, t\}$



$L_2 = \{t, t, t, t\}$,k=0 $L_4 = \{f, t, t, t\}, \mathrm{k} = 1$ $L_6 = \{f, f, t, t\}$ $L_6 = \{f, f, f, t\}$

Sequential FF E.g. Step 6

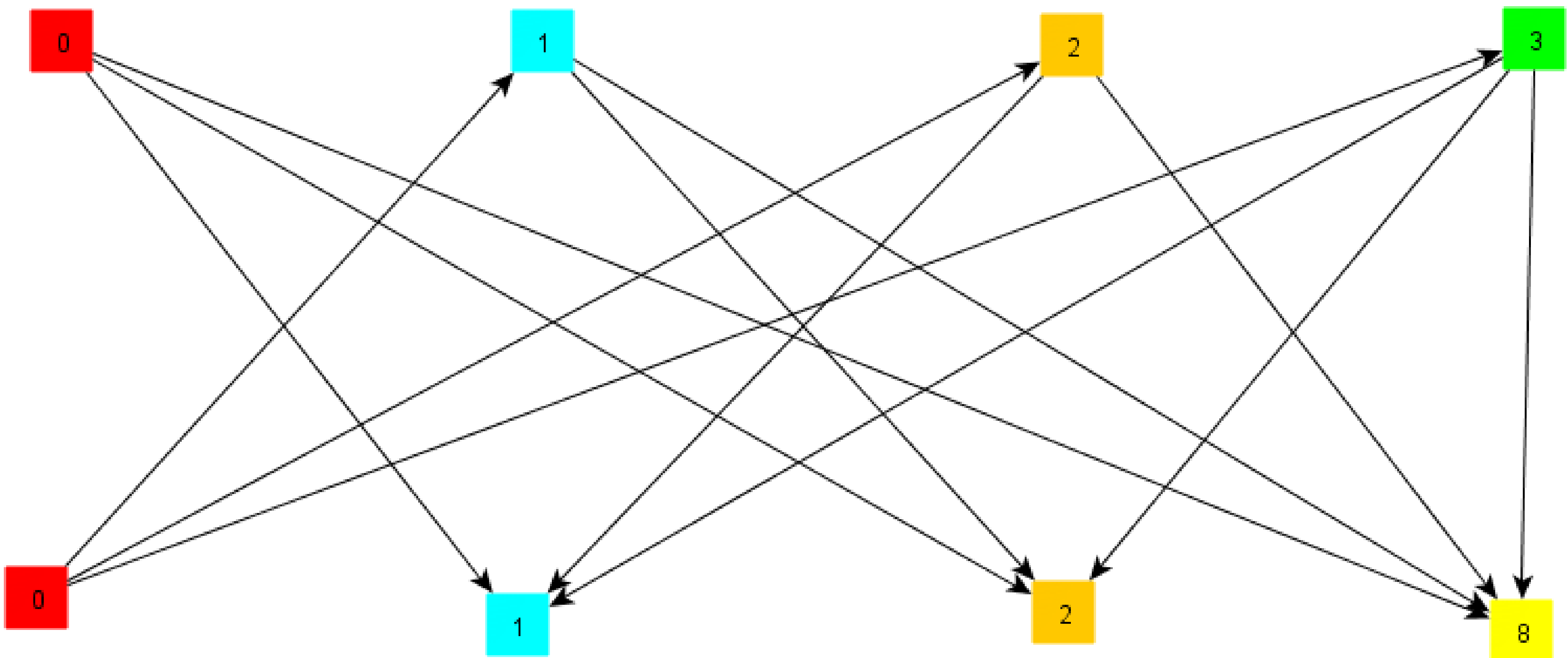
$L_1 = \{t, t, t, t\}$,k=0 $L_3 = \{f, t, t, t\}, k$ =1 $L_5 = \{f, f, t, t\}$,k=2 $L_7 = \{f, f, f, t\}$



$L_2 = \{t, t, t, t\}$,k=0 $L_4 = \{f, t, t, t\}, \mathrm{k} = 1$ $L_6 = \{f, f, t, t\}$,k=2 $L_6 = \{f, f, f, t\}$

Sequential FF E.g. Step 7

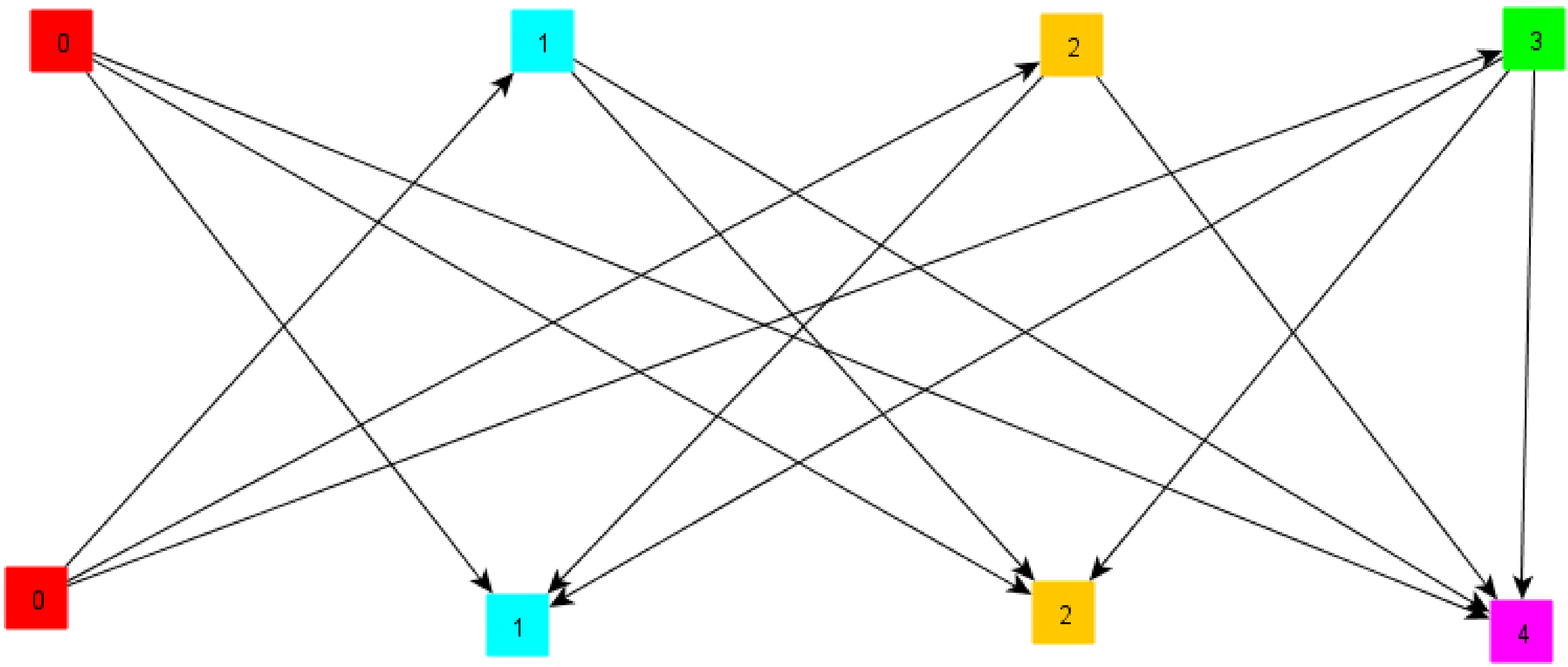
$L_1 = \{t, t, t, t\}$,k=0 $L_3 = \{f, t, t, t\}, k$ =1 $L_5 = \{f, f, t, t\}$,k=2 $L_7 = \{f, f, f, t\}, k = 3$



$L_2 = \{t, t, t, t\}$,k=0 $L_4 = \{f, t, t, t\}, \mathrm{k} = 1$ $L_6 = \{f, f, t, t\}$,k=2 $L_6 = \{f, f, f, f\}$

Sequential FF E.g. Step 8

$L_1 = \{t, t, t, t\}$,k=0 $L_3 = \{f, t, t, t\}, k$ =1 $L_5 = \{f, f, t, t\}$,k=2 $L_7 = \{f, f, f, t\}, k = 3$



$L_2 = \{t, t, t, t\}$,k=0 $L_4 = \{f, t, t, t\}, \mathrm{k} = 1$ $L_6 = \{f, f, t, t\}$,k=2 $L_6 = \{f, f, f, f\}$,k=4

Parallel FF (1)

Step	**Processor$_1$**	**Processor$_2$**	**Processor$_3$**	**Processor$_4$**	**Processor$_5$**
1.	Color(L_1, v_1)				
2.	Build(L_2, v_1)				
3.	Build(L_3, v_1)	Color(L_2, v_2)			
4.	Build(L_4, v_1)	Build(L_3, v_2)			
5.	Build(L_5, v_1)	Build(L_4, v_2)	Color(L_3, v_3)		
6.		Build(L_5, v_2)	Build(L_4, v_3)		
7.			Build(L_5, v_3)	Color(L_4, v_4)	
8.				Build(L_5, v_4)	
9.					Color(L_5, v_5)

Figure 2: Parallel first fit with 5 vertices and 5 processors.

Parallel FF (2)

- Problem
  - Requires same number of cores as there are vertices in *G*
- Generalized algorithm
  - Processors $P_1,\ldots,P_n (1 \leq \mathrm{N} \leq \mathrm{n})$, *n* – vertices
  - Every processor colors whole subgraph with *n/N* instead of single vertex unlike
  - Possibility lists prepared on previous processors
  - $Build(L_i, V_j)$ excludes colors of *all* vertices

$V_j = \{v_{1+(j-1)n/N}, \ldots, v_{jn/N}\}$ in jth subgraph from L_i which will be

  - later by another processor

Generalized Parallel FF (1)

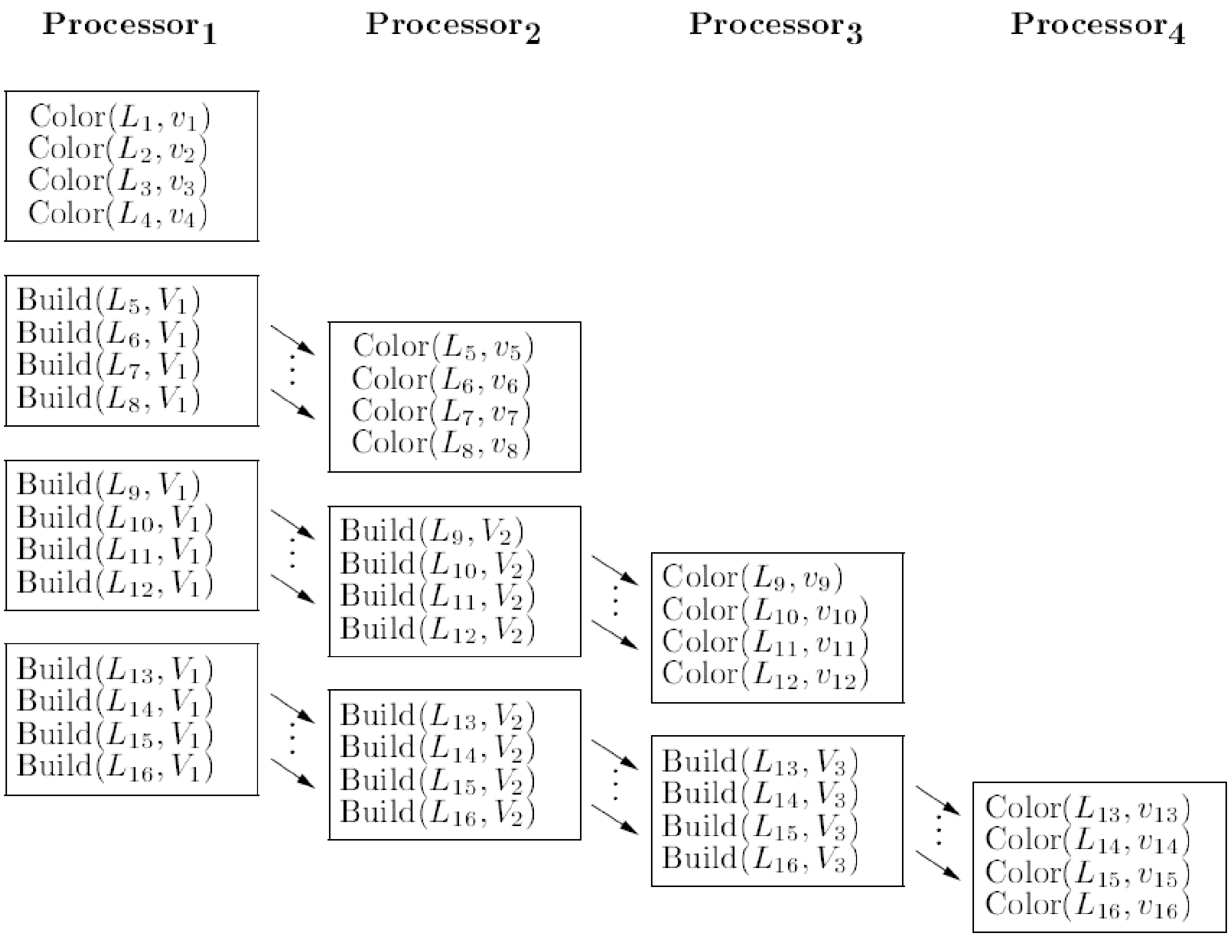
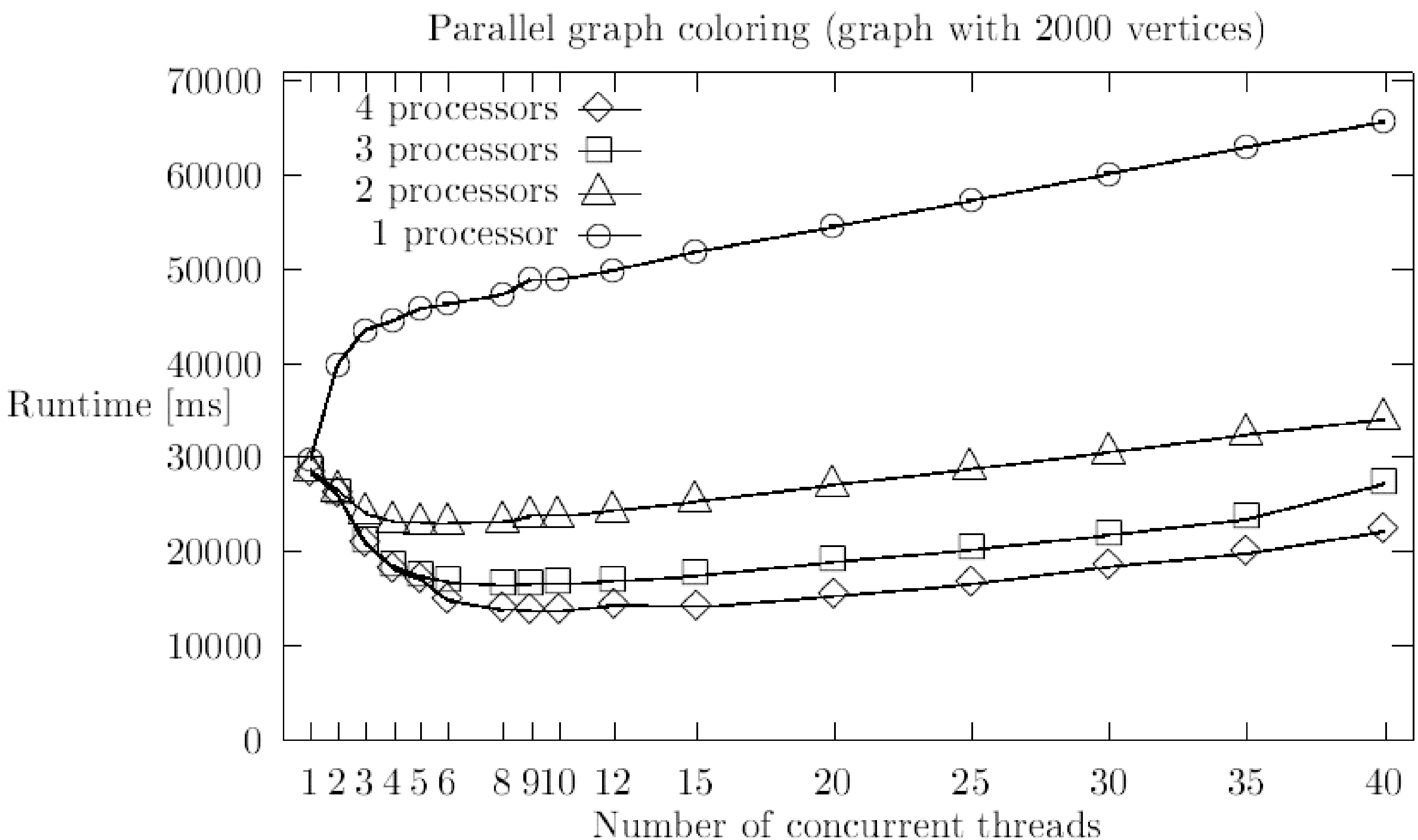


Figure 3: Generalized parallel first fit (16 vertices, 4 processors).

Generalized FF (2)

- Rougly 50% of resources not used
  - Speedup is not expected to exceed half the number of cores
  - Still good for this type of algorithm
- Implementation
  - Share graph among cores
  - Flow of control over L_i (illustrated by arrows) can be implemented by passing tokens from thread to thread
    - CSP
    - No need to transfer entire list

Umland's Results



Plan(1)

- Umland ran algorithm on SPARC 40 MHz and 128 MB RAM in 1998
  - 2000 vertices, 999001 edges
  - 1001 colors
  - 30 seconds
- Implement generalized parallel algorithm
- Determine graph of comparable size for modern hardware
- Run with different number of threads and observe speedup or improvements in total time
- DEMO TIME!!

Questions?

- [1] Daniel Brélaz. New methods to color the vertices of a graph. Commun. ACM, 22:251-256, April 1979.
- [2] N. Christodes. An algorithm for the chromatic number of a graph. 14(1):38-39, 1971.
- [3] Tom Davis. The mathematics of sudoku. http://geometer.org/mathcircles/sudoku.pdf, 2008. [Online; accessed January 30, 2011].
- [4] Keith Devlin. Last doubts removed about the proof of the four color theorem. http://www.maa.org/devlin/devlin_01_05.html, January 2005. [Online; accessed January 30, 2011].
- [5] A. Gyárfás and J. Lehel. On-line and rst t colorings of graphs. Journal of Graph Theory, 12(6):217-227, 1988.
- [6] Magnús M. Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring. Theoretical Computer Science, 130:163-174, 1994.
- [7] David Koes and Seth Copen Goldstein. An analysis of graph coloring register allocation. Technical Report CMU-CS-06-111, Carnegie Mellon University, March 2006.
- [8] Dániel Marx. Graph coloring problems and their applications in scheduling. Periodica Polytechnica Ser.El. Eng., 48(1-2):5-10, 2004.
- [9] N. Narayanaswamy and R. Babu. A note on rst-t coloring of interval graphs. Order, 25:4-53, 2008.10.1007/s11083-008-9076-6.
- [10] Sriram V. Pemmaraju, Rajiv Raman, and Kasturi Varadarajan. Buer minimization using max-coloring. In Proceedings of the 15th annual ACM-SIAM symposium on Discrete algorithms, SODA '04, pages 562-571, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.
- [11] Johannes Schneider and Roger Wattenhofer. A new technique for distributed symmetry breaking. In Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing , PODC '10, pages 257266, New York, NY, USA, 2010. ACM.
- [12] David A. Smith. The First-Fit Algorithm Uses Many Colors on Some Interval Graphs. PhD thesis, Arizona State University, United States, 2010.
- [13] Thomas Umland. Parallel graph coloring using JAVA. In Architectures, Languages and Patterns for Parallel and Distributed Applications, pages 211-218. IOS Press, 1998.
- [14] Peng-Jun Wan, Zhu Wang, Hongwei Du, Scott C.-H. Huang, and Zhiyuan Wan. First-t scheduling for beaconing in multihop wireless networks. In Proceedings of the 29th conference on Information communications, INFOCOM'10, pages 2205-2212, Piscataway, NJ, USA, 2010. IEEE Press.
- [15] Eric W. Weisstein. Four-color theorem. http://mathworld.wolfram.com/Four-ColorTheorem.html. [Online; accessed January 30, 2011].
- [16] Eric W. Weisstein. Interval graph. http://mathworld.wolfram.com/IntervalGraph.html. [Online; accessed January 30, 2011].
- [17] Herbert S. Wilf. Backtrack: An o(1) expected time algorithm for the graph coloring problem. Information Processing Letters, 18(3):119121, 1984.
- [18] Hamid Zarrabi-Zadeh. Online coloring co-interval graphs. Scientia Iranica, 12(6):17, 2009.