Towards a recursive graph bipartitioning algorithm for well balanced domain decomposition.

Mini-Symposium on "Combinatorial Issues in Sparse Matrix Computation"

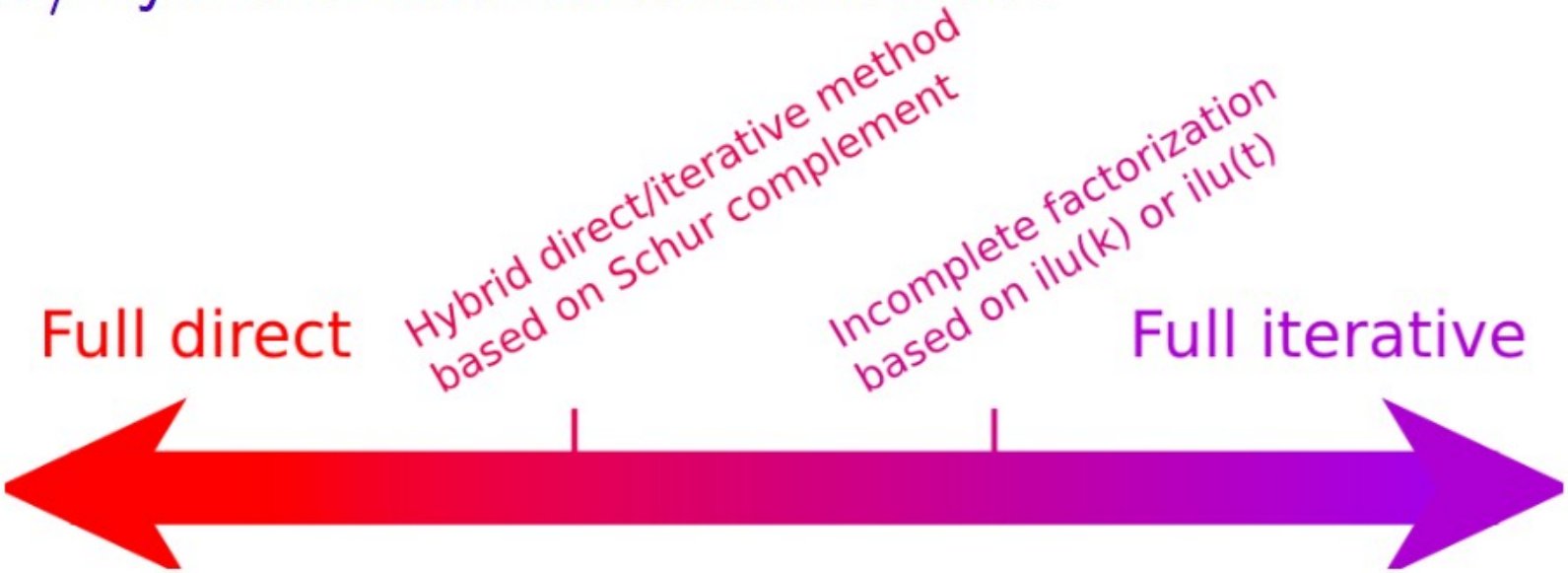
Astrid Casadei, Pierre Ramet and Jean Roman

ICIAM 2015, Beijing, China August 2015

Plan

- 1. Context: load balancing in hybrid solvers
- 2. Nested Dissection (ND)
- 3. Multilevel framework
- 4. Algorithm to refine a separator: Fiduccia-Mattheyses (FM)
- Algorithms to build a separator:
  - 5. Greedy graph growing (GG)
  - 6. Double greedy graph growing (DG)
  - 7. Halo-first greedy graph growing (HF)
- 8. Results
- 9. Conclusion / future work

Mixed/Hybrid direct-iterative methods



The "spectrum" of linear algebra solvers

- Robust/accurate for general problems
- BLAS-3 based implementation
- Memory/CPU prohibitive for large 3D problems
- Limited parallel scalability

- Problem dependent efficiency/controlled accuracy
- Only mat-vec required, fine grain computation
- Less memory consumption, possible trade-off with CPU
- Attractive "build-in" parallel features

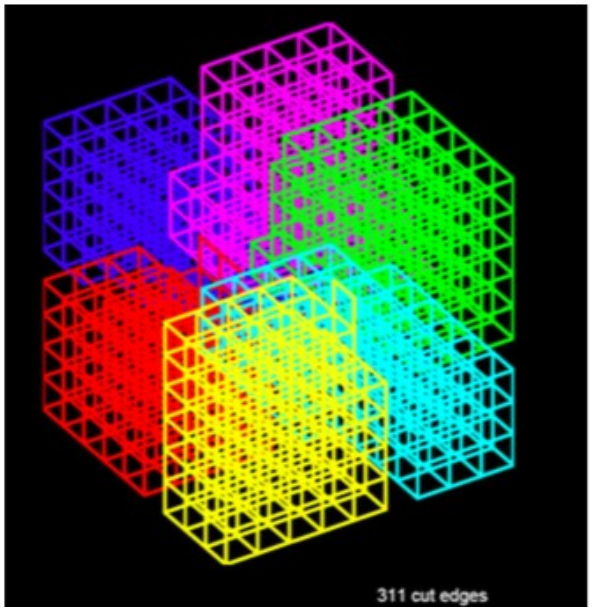
Hybrid Linear Solvers

Develop robust scalable parallel hybrid direct/iterative linear solvers

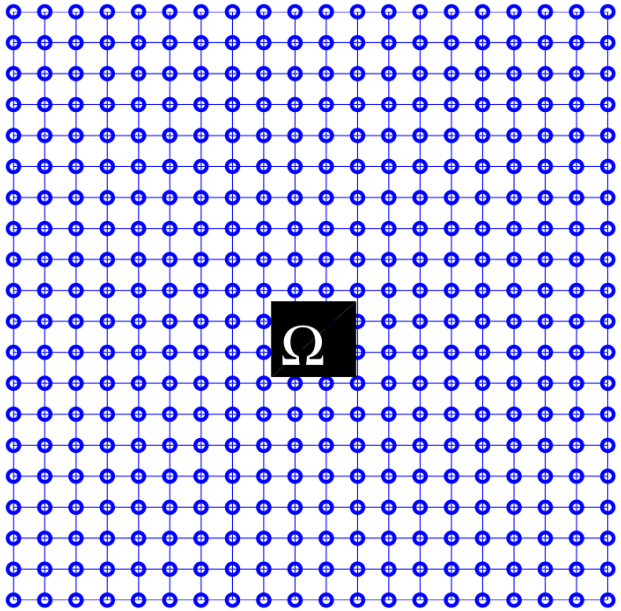
- Exploit the efficiency and robustness of the sparse direct solvers
- Develop robust parallel preconditioners for iterative solvers
- Take advantage of scalable implementation of iterative solvers

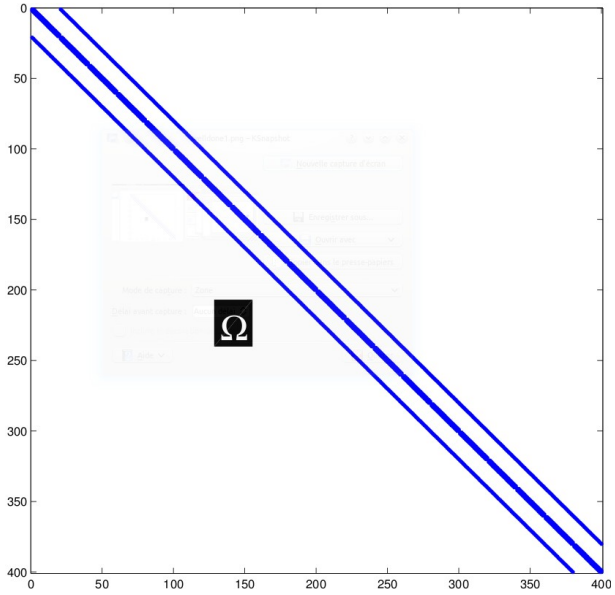
Domain Decomposition (DD)

- Natural approach for PDE's
- Extend to general sparse matrices
- Partition the problem into subdomains
- Use a direct solver on the subdomains
- Robust preconditioned iterative solver



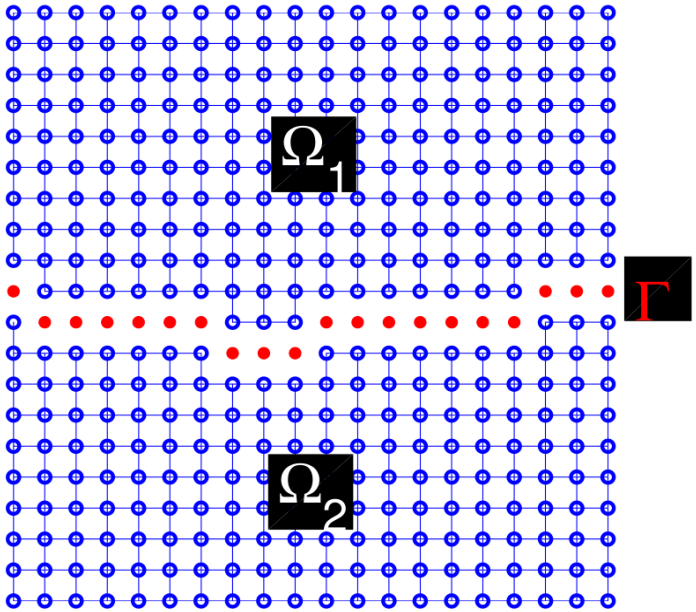
1 Context: load balancing in hybrid solvers

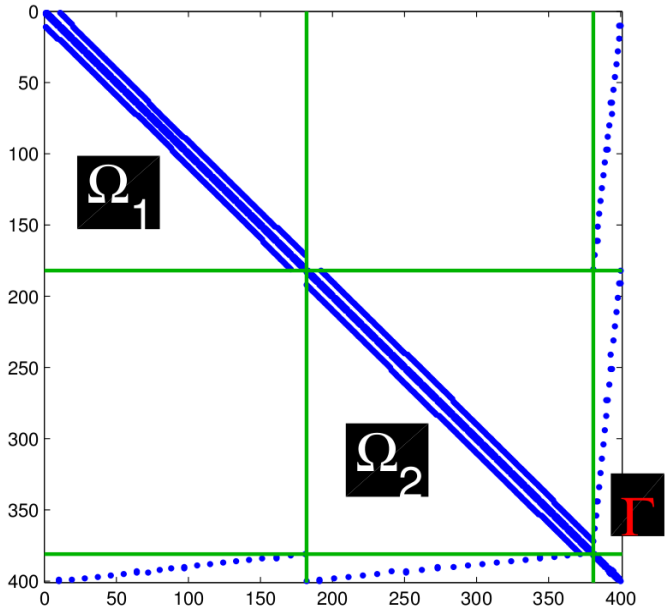




1. Building a domain decomposition
2. Factorizing each domain and computing local Schur complement (= local interface)
3. Iterating on Schur complement (= interface)

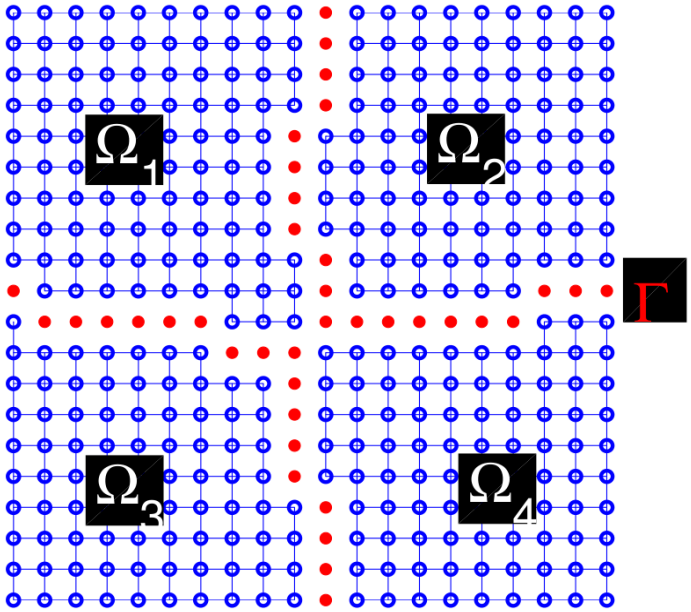
1 Context: load balancing in hybrid solvers

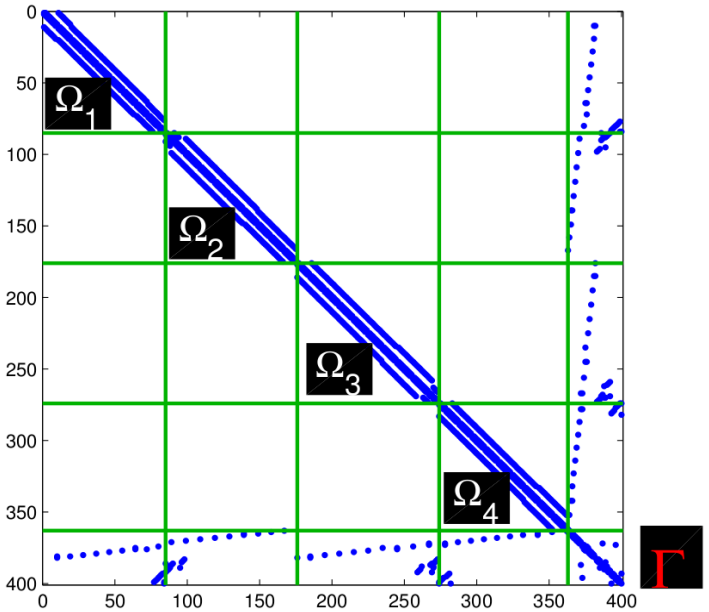




1. Building a domain decomposition
2. Factorizing each domain and computing local Schur complement (= local interface)
3. Iterating on Schur complement (= interface)

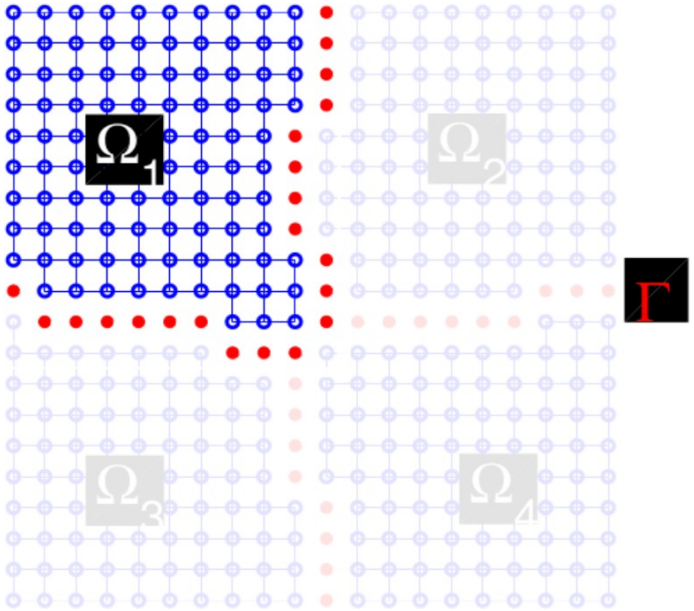
1 Context: load balancing in hybrid solvers

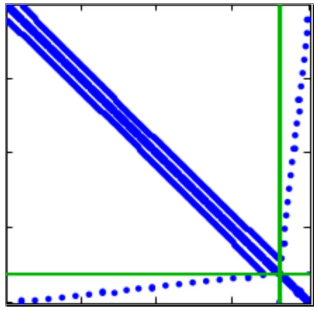




1. Building a domain decomposition
2. Factorizing each domain and computing local Schur complement (= local interface)
3. Iterating on Schur complement (= interface)

1 Context: load balancing in hybrid solvers

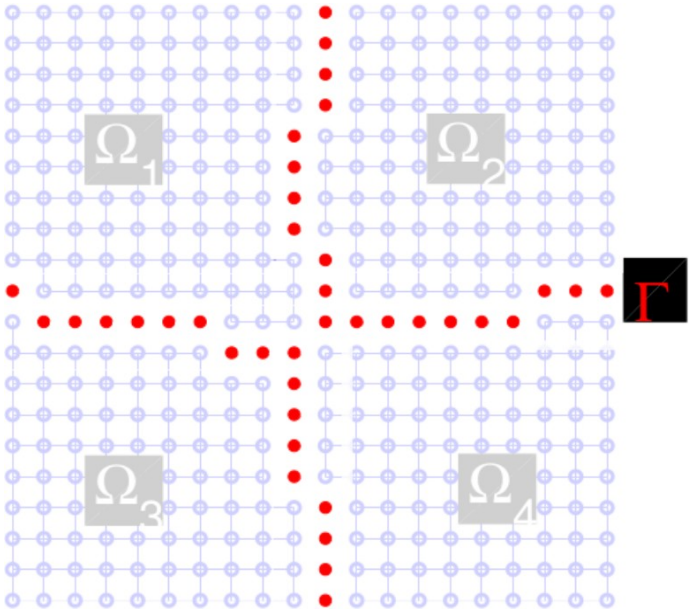


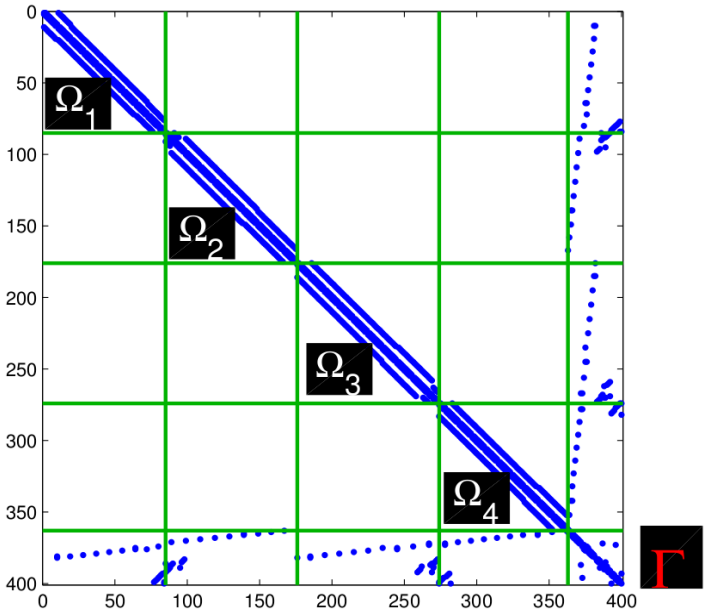


→ Both domain interior and domain interface must be balanced
- factorize subdomains
- build local Schur complement
- apply local part of precond.

1. Building a domain decomposition
2. Factorizing each domain and computing local Schur complement (= local interface)
3. Iterating on Schur complement (= interface)

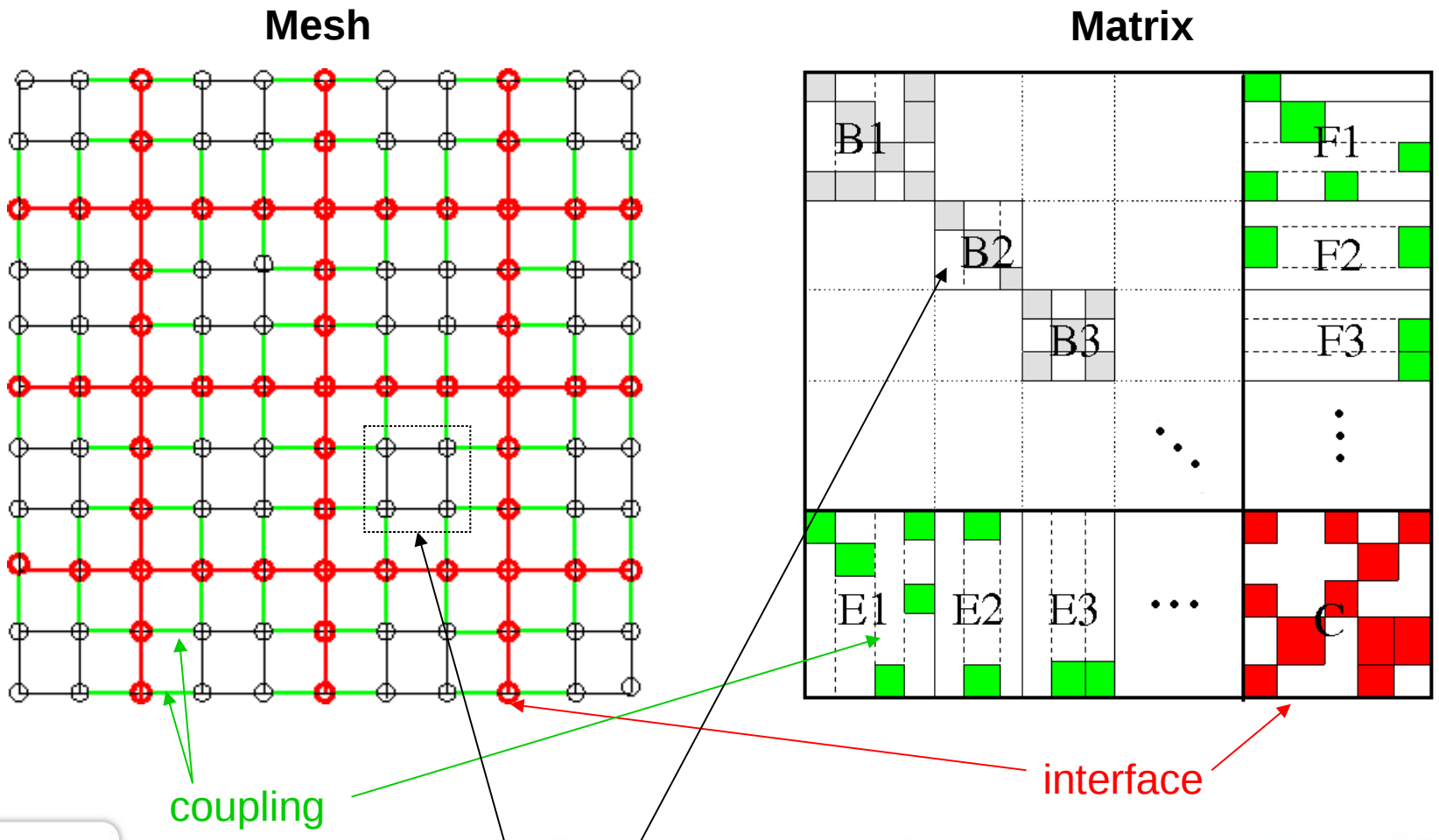
1 Context: load balancing in hybrid solvers





1. Building a domain decomposition
2. Factorizing each domain and computing local Schur complement (= local interface)
3. Iterating on Schur complement (= interface)

Hybrid method (MaPHYS, HIPS, PDSLIN, SHYLU, ...)

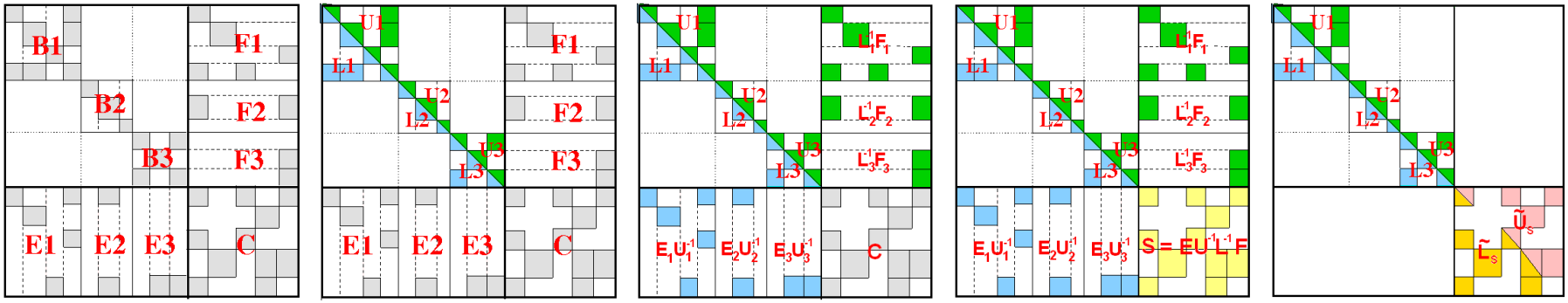


Hybrid method (MaPHYS, HIPS, PDSLIN, SHYLU, ...)

Step 1 :

Build an approximate factorization :

$$A \approx \tilde{L}\tilde{U} = \begin{pmatrix} L_B U_B & \\ & \tilde{L}_S \tilde{U}_S \end{pmatrix}$$

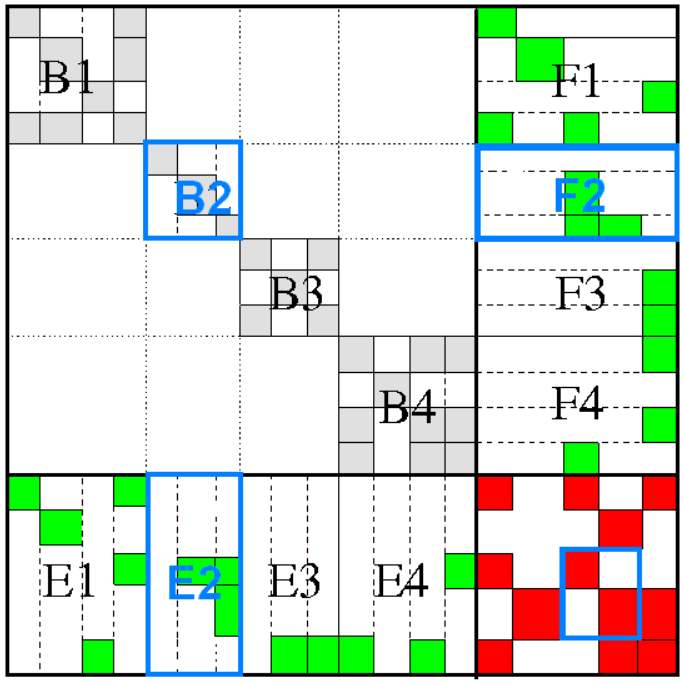


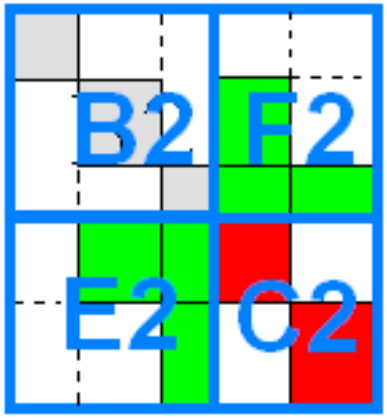
Initial Matrix | Factorize each domain | TRSM | Add contributions | Incomplete factorization of S

Step 2 :

Iterative method to solve Ax = b using $\tilde{L}$ & $\tilde{U}$ as preconditioner

Hybrid method (MaPHYS, HIPS, PDSLIN, SHYLU, ...)





Local matrix for subdomain 2

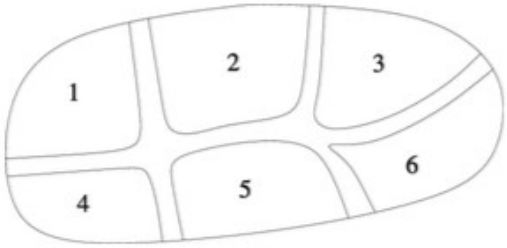
2 levels of parallelism :

- parallelism between subdomains
- parallel direct solver to factorize each subdomain

HIPS : hybrid direct-iterative solver

Based on a domain decomposition

$$\begin{pmatrix} A_B & F \\ E & A_C \end{pmatrix}$$



B : Interior nodes of subdomains (direct factorization).

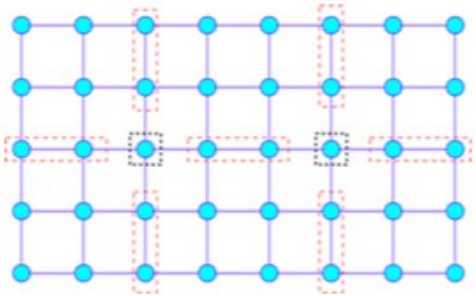
C : Interface nodes.

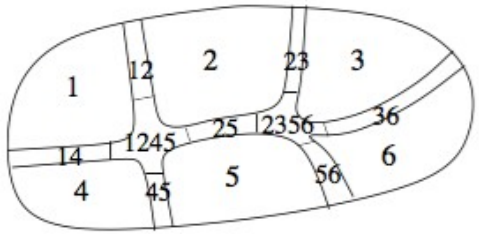
Special decomposition and ordering of the subset C :
Goal : Building a global Schur complement preconditioner (ILU) from the local domain matrices only.

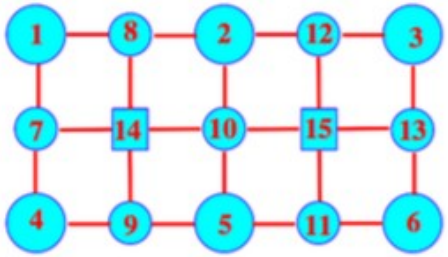
HIPS: domain interface based fill-in policy

Special decomposition and ordering of the subset C :

Hierachical interface decomposition into connectors :





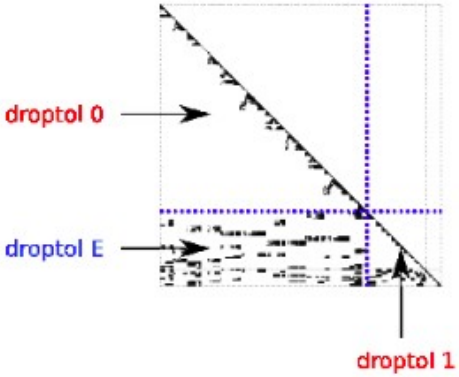


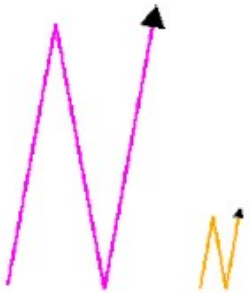
Rules :

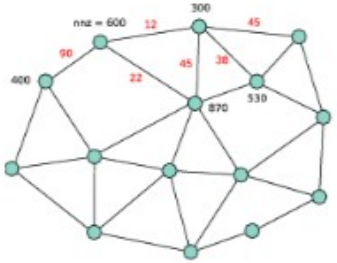
- No creation of edge (fill-in) outside the local domain matrices.
- Allow edges between connectors adjacent to the same subdomain.

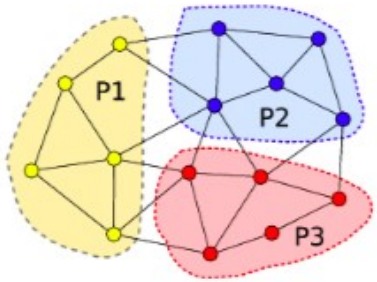
$\Rightarrow$ keep the parallelism (communication only between adjacent subdomains).

HIPS: preconditioners





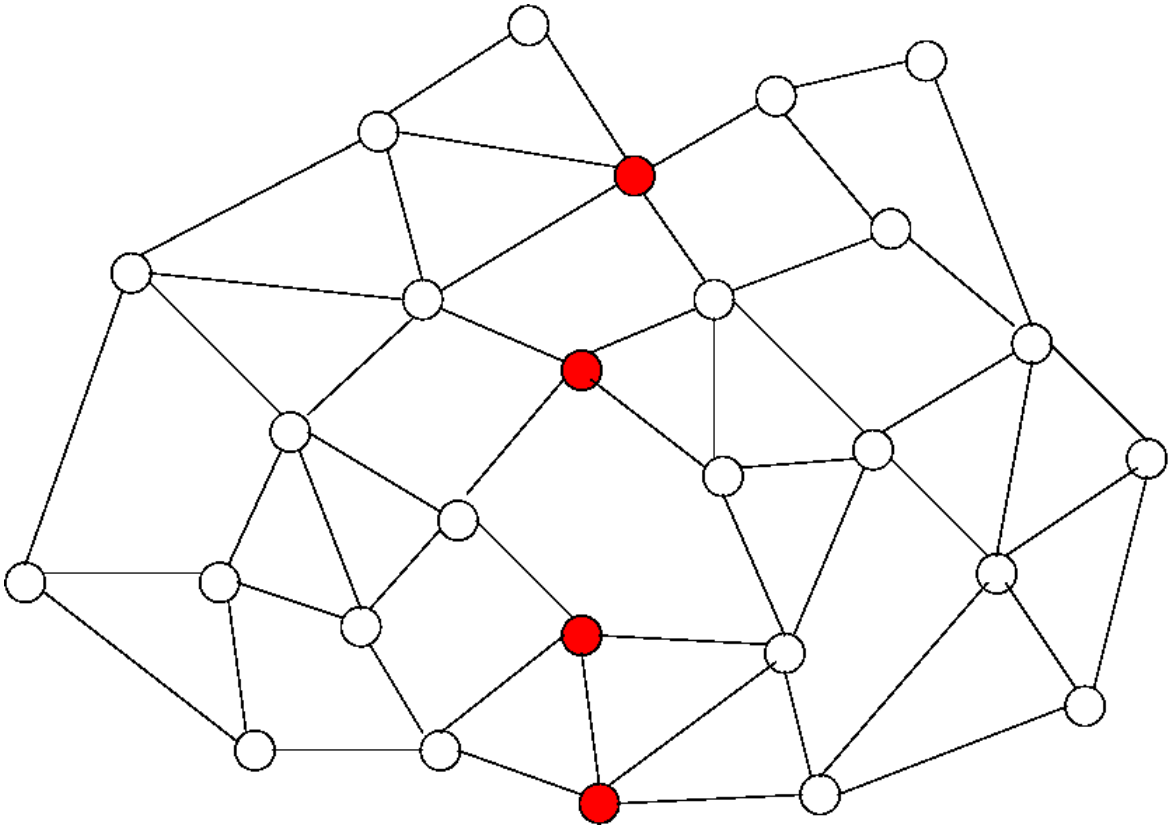




Main features

- Iterative or "hybrid" direct/iterative method are implemented.
- Mix direct supernodal (BLAS-3) and sparse ILUT factorization in a seamless manner.
- Memory/Load balancing : distribute the domains on the processors (domains $>$ processors).

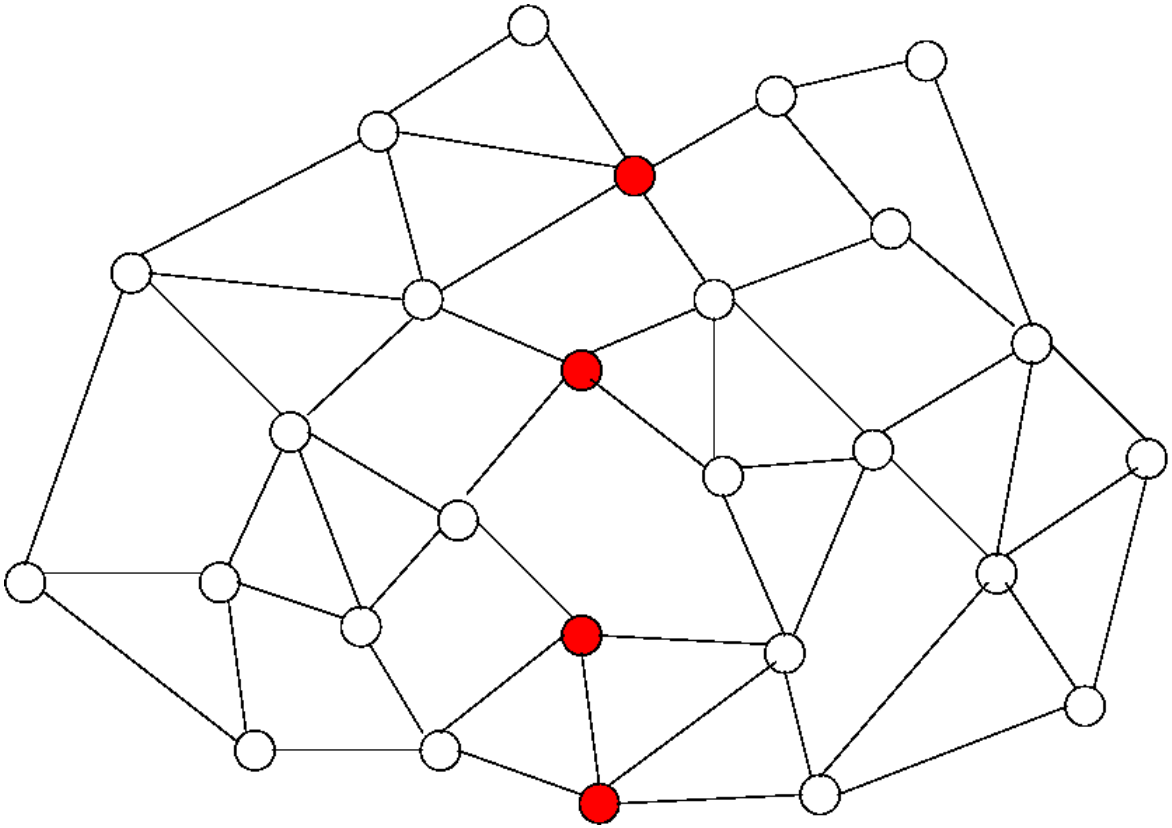
2 Nested dissection



When separating a subgraph, ensure that :

- Parts are balanced
- Separator size is minimized

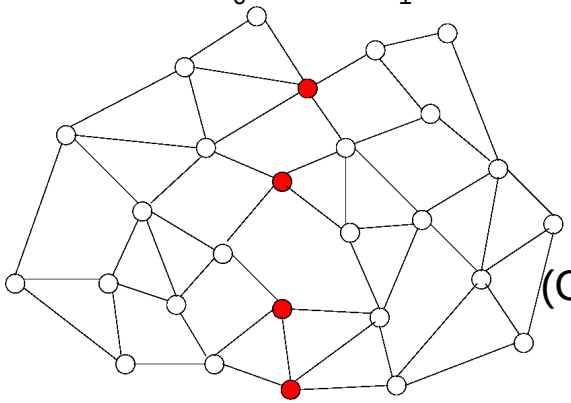
2 Nested dissection



When separating a subgraph, ensure that :

- Parts are balanced
- Separator size is minimized
- **NEW:** Halo is balanced

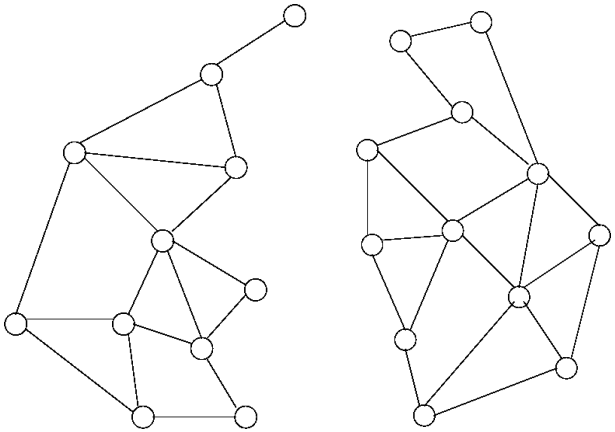
$Y = P_0 \cup S \cup P_1$



Classical nested dissection

Do ND on P_0 and P_1

(George)

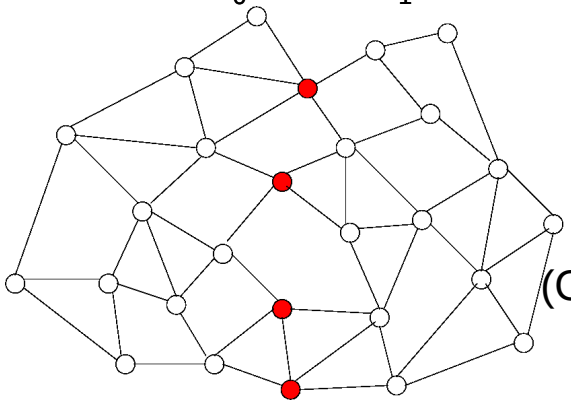


ND Keeping track of halo vertices

Do ND on $P_0 \cup S$ and $P_1 \cup S$

(Generalized ND, Lipton, Rose, Tarjan)

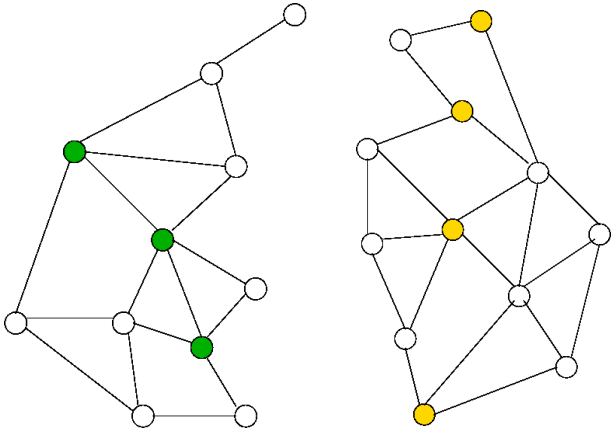
$Y = P_0 \cup S \cup P_1$



Classical nested dissection

Do ND on P_0 and P_1

(George)

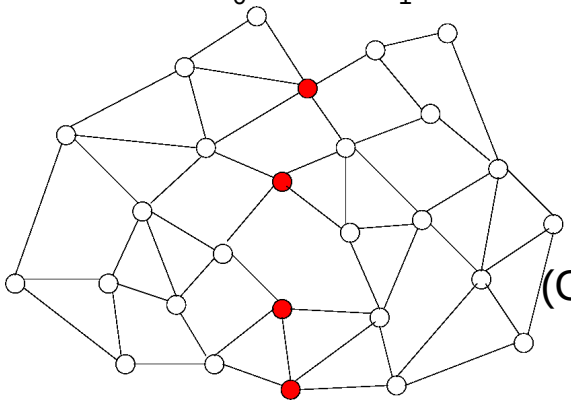


ND Keeping track of halo vertices

Do ND on $P_0 \cup S$ and $P_1 \cup S$

(Generalized ND, Lipton, Rose, Tarjan)

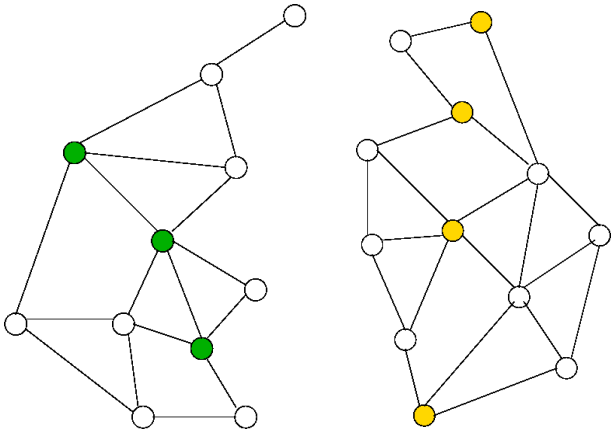
$Y = P_0 \cup S \cup P_1$

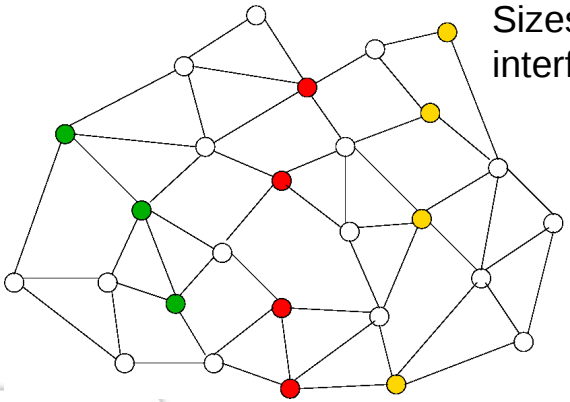


Classical nested dissection

Do ND on P_0 and P_1

(George)





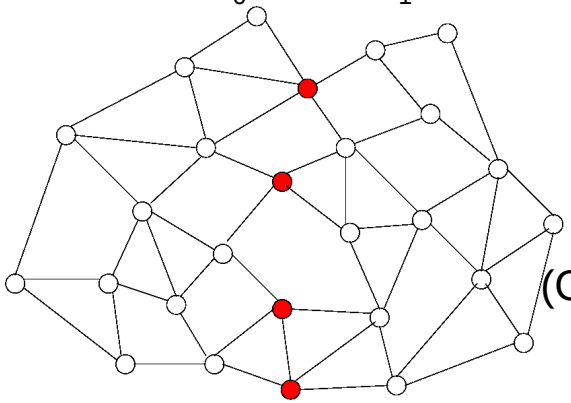
Sizes of local interfaces: 4, 5, 6, 8

ND Keeping track of halo vertices

Do ND on $P_0 \cup S$ and $P_1 \cup S$

(Generalized ND, Lipton, Rose, Tarjan)

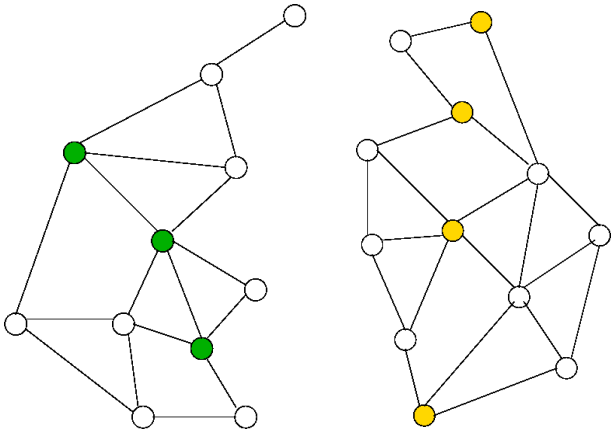
$Y = P_0 \cup S \cup P_1$

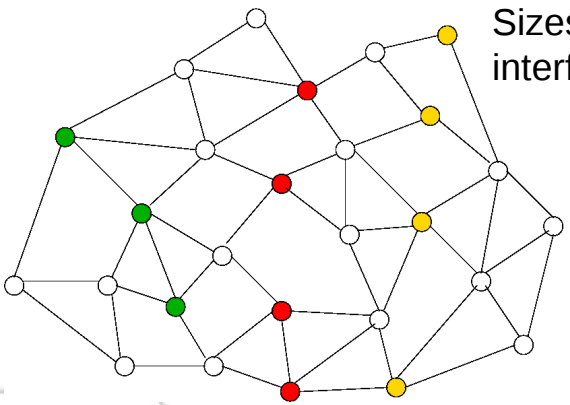


Classical nested dissection

Do ND on P_0 and P_1

(George)



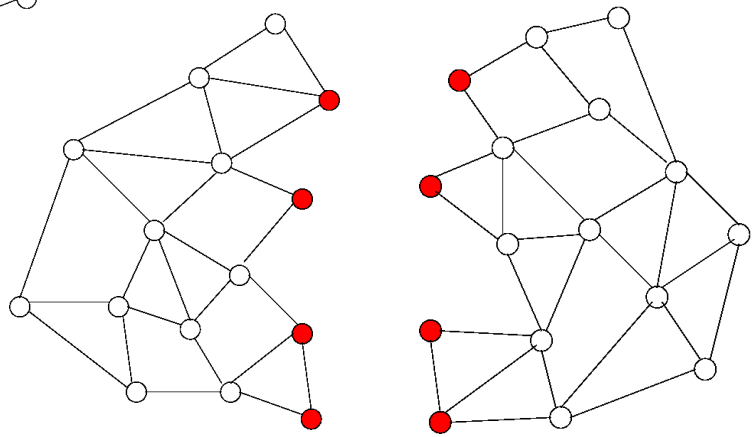


Sizes of local interfaces: 4, 5, 6, 8

ND Keeping track of halo vertices

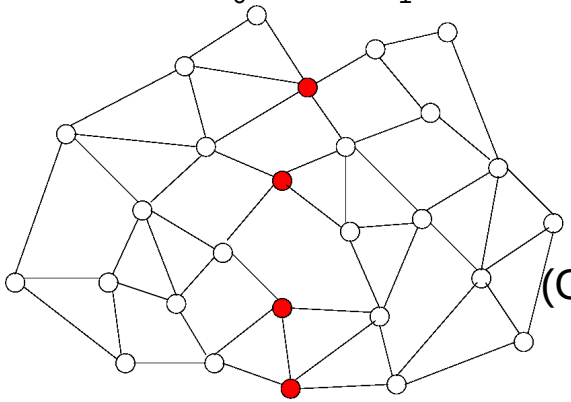
Do ND on $P_0 \cup S$ and $P_1 \cup S$

(Generalized ND, Lipton, Rose, Tarjan)



Halo: vertices of old separators adjacent to subgraph

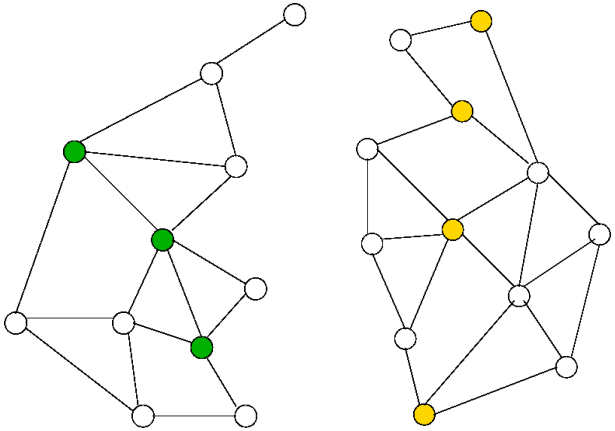
$Y = P_0 \cup S \cup P_1$

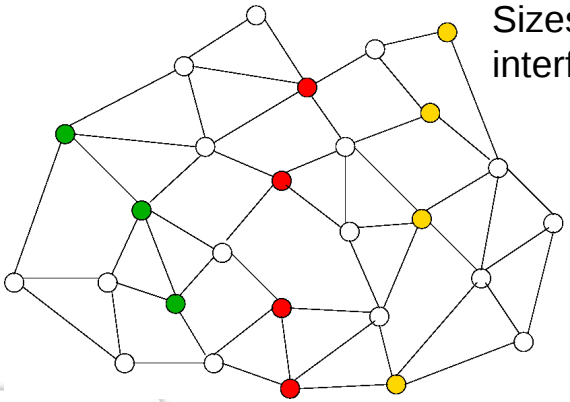


Classical nested dissection

Do ND on P_0 and P_1

(George)



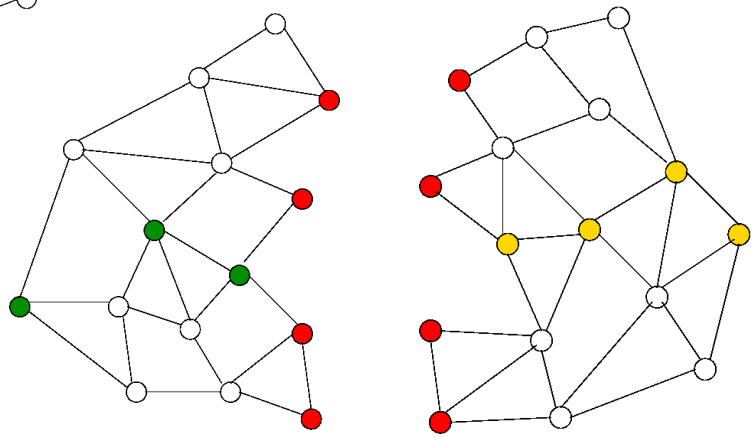


Sizes of local interfaces: 4, 5, 6, 8

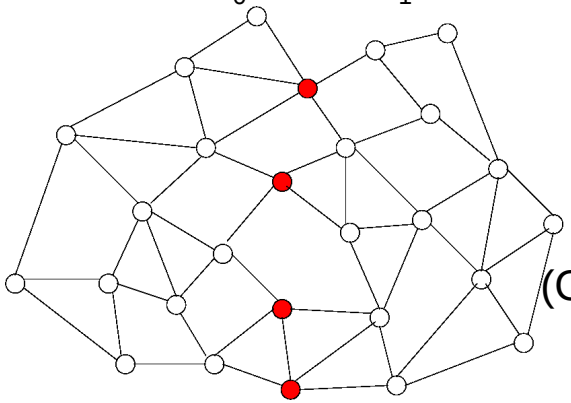
ND Keeping track of halo vertices

Do ND on $P_0 \cup S$ and $P_1 \cup S$

(Generalized ND, Lipton, Rose, Tarjan)



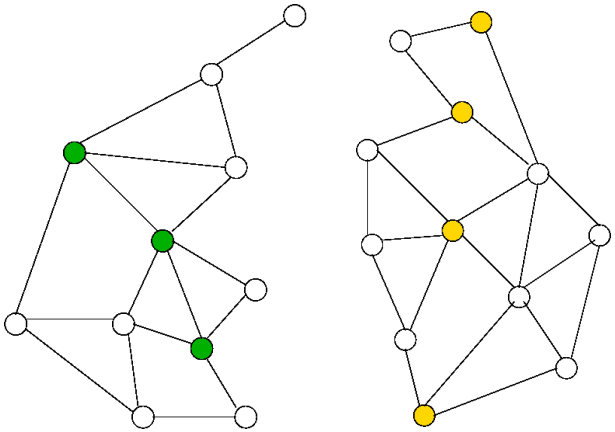
$Y = P_0 \cup S \cup P_1$

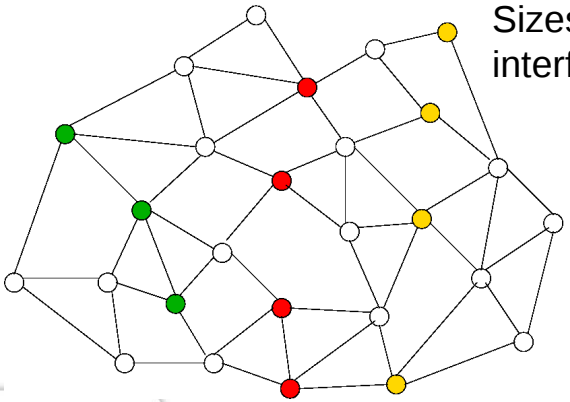


Classical nested dissection

Do ND on P_0 and P_1

(George)



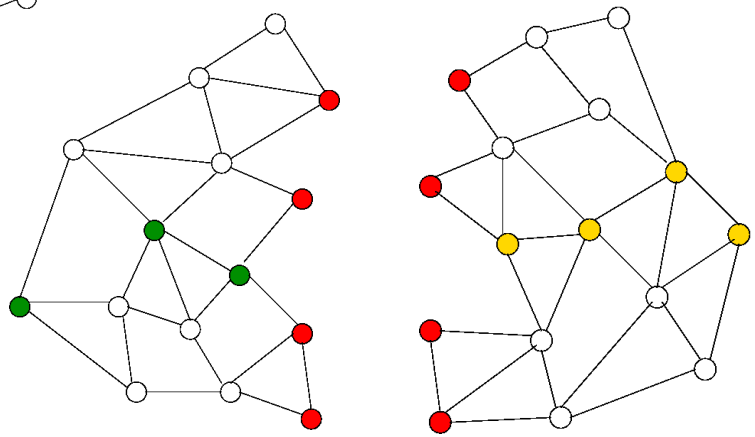


Sizes of local interfaces: 4, 5, 6, 8

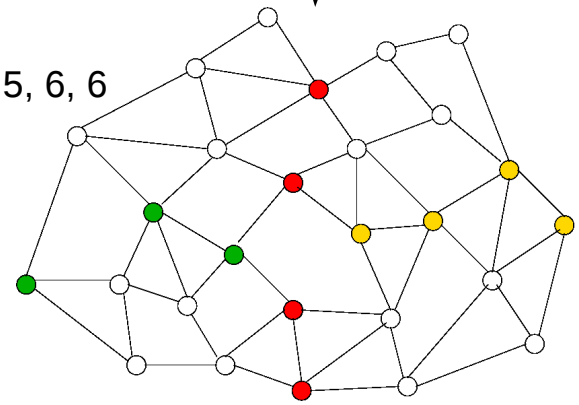
ND Keeping track of halo vertices

Do ND on $P_0 \cup S$ and $P_1 \cup S$

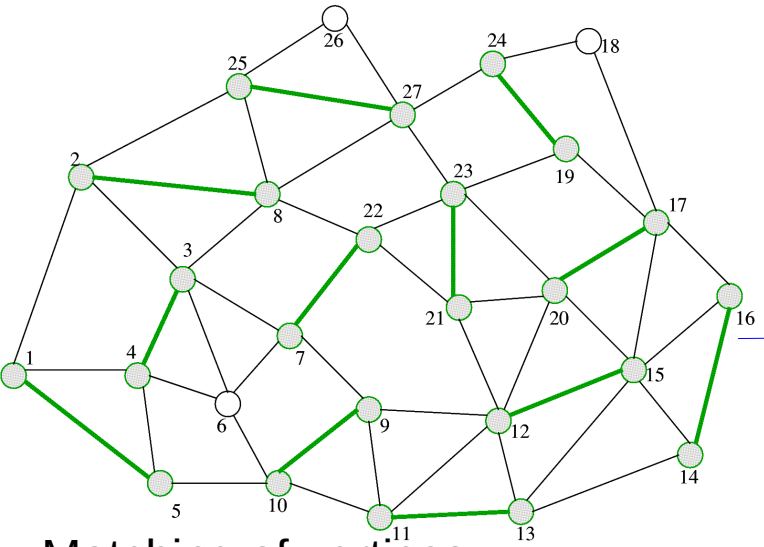
(Generalized ND, Lipton, Rose, Tarjan)



Sizes of local interfaces: 5, 5, 6, 6

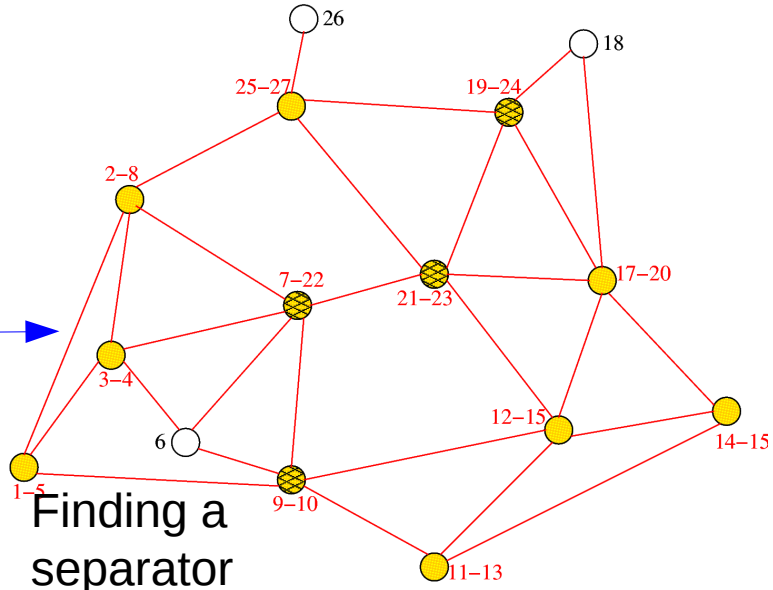


3 The multilevel framework



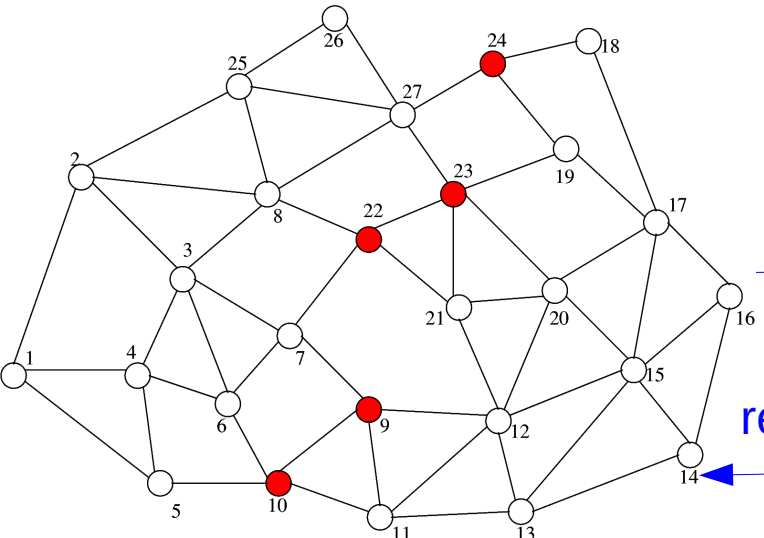
Matching of vertices

coarsening
(multiple times)

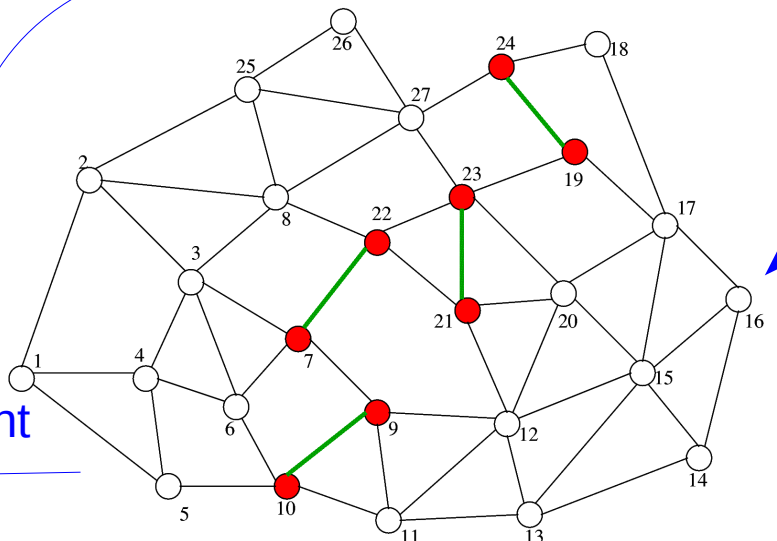


Finding a
separator

uncoarsening



refinement



3 The multilevel framework

$G_1 \leftarrow G;$
$i \leftarrow 1;$
while G_i *can be coarsened* **do**
  $G_{i+1} \leftarrow coarsenGraph(G_i);$
  $i++;$
$n \leftarrow i;$
$S_n \leftarrow findSeparator(G_n);$
for i *from* $n-1$ *to* 1 **do**
  $S_i \leftarrow uncoarsenSeparator(G_{i+1}, S_{i+1});$
  $S_i \leftarrow refineSeparator(G_i, S_i);$

Greedy graph growing

Fiduccia-Mattheyses algorithm

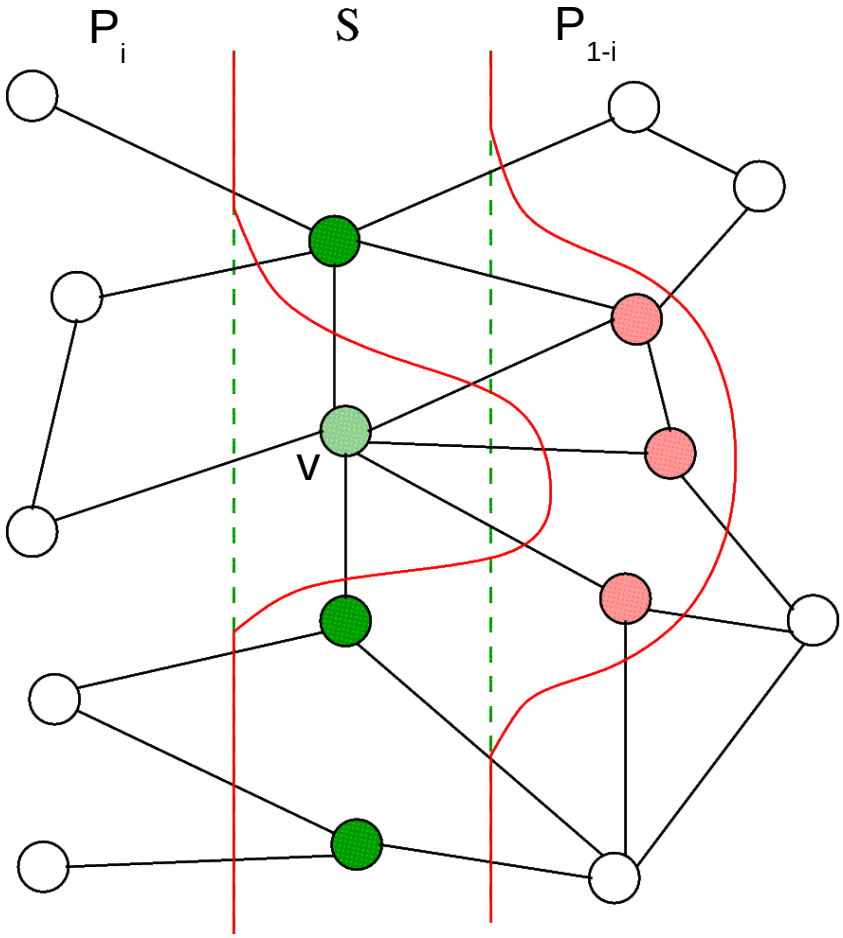
4 Fiduccia-Mattheyses algorithm

Original version

Main action of FM:
moving of vertex v from S to a part P_i

Choice based on:

- if $||P_0| - |P_1|| > \delta$, part balance
- separator minimization

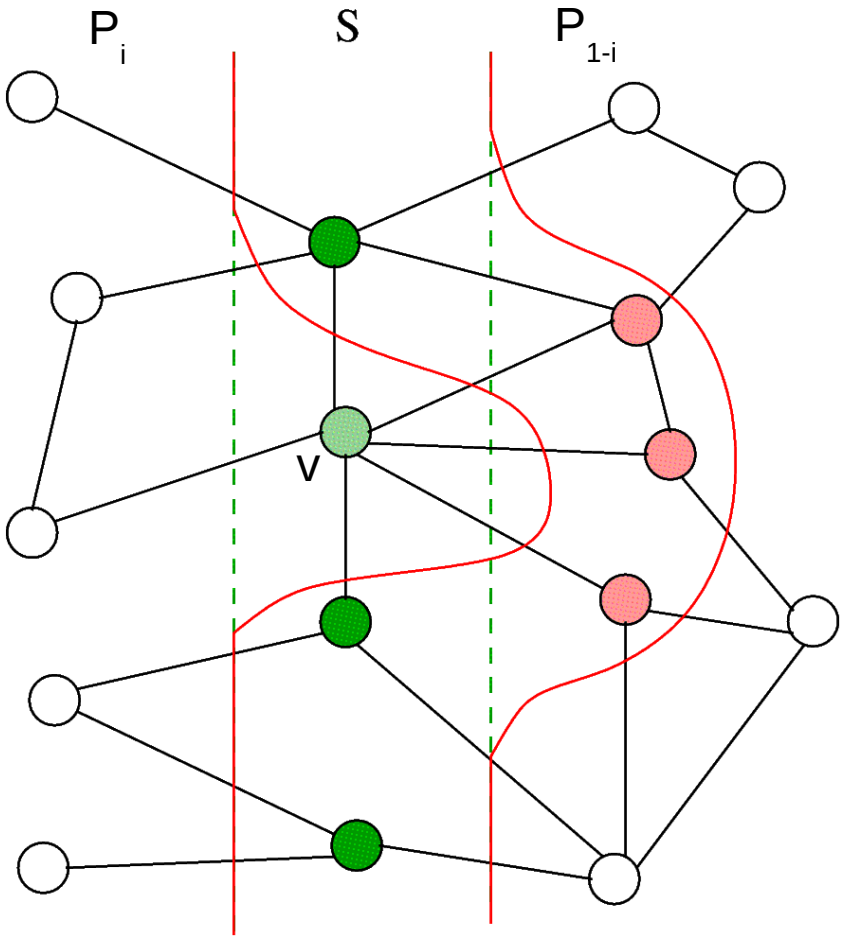


4 Fiduccia-Mattheyses algorithm

Modified version

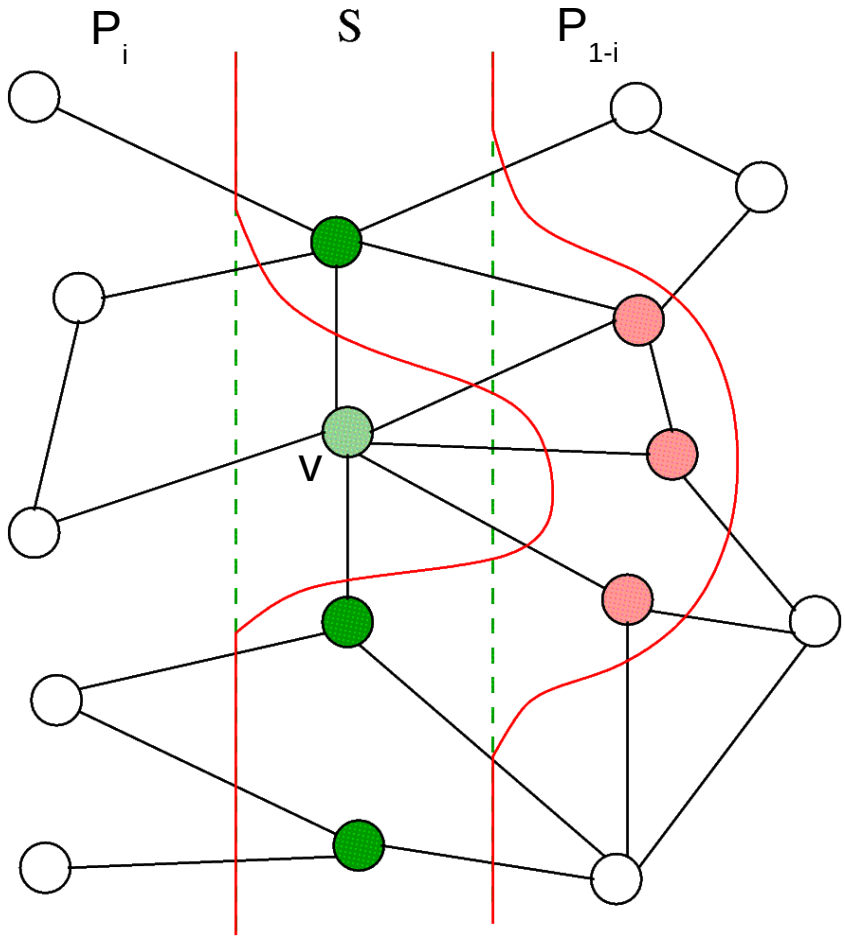
Main action of FM:
moving of vertex v from S to a part P_i

Choice based on:
- if $||P_0 \cap V_h| - |P_1 \cap V_h|| > \delta_{halo}$, halo balance
- if $||P_0| - |P_1|| > \delta$, part balance
- separator minimization



4 Fiduccia-Mattheyses algorithm

Modified version



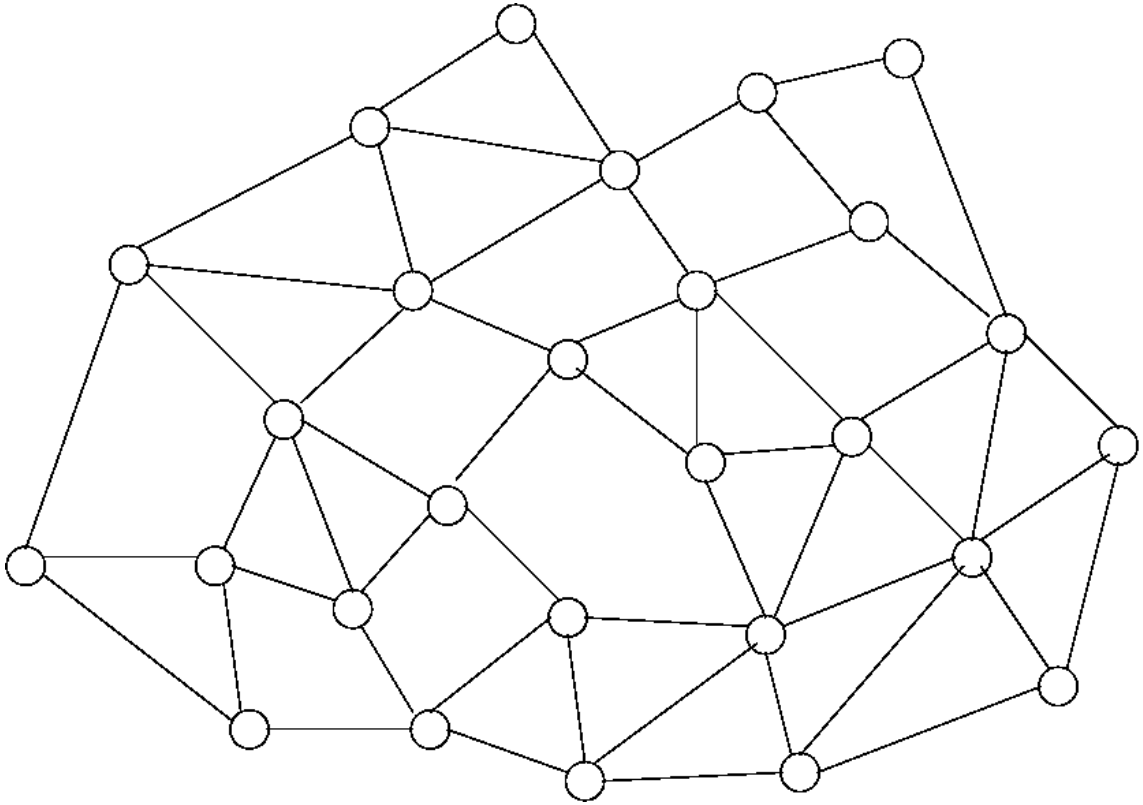
Main action of FM:
moving of vertex v from S to a part P_i

Choice based on:
- if $||P_0 \cap V_h| - |P_1 \cap V_h|| > \delta_{halo}$, halo balance
- if $||P_0| - |P_1|| > \delta$, part balance
- separator minimization

To select best separator, choice based on:
- reasonnable part balance
- reasonnable halo balance
- separator minimization
- halo balance
- part balance

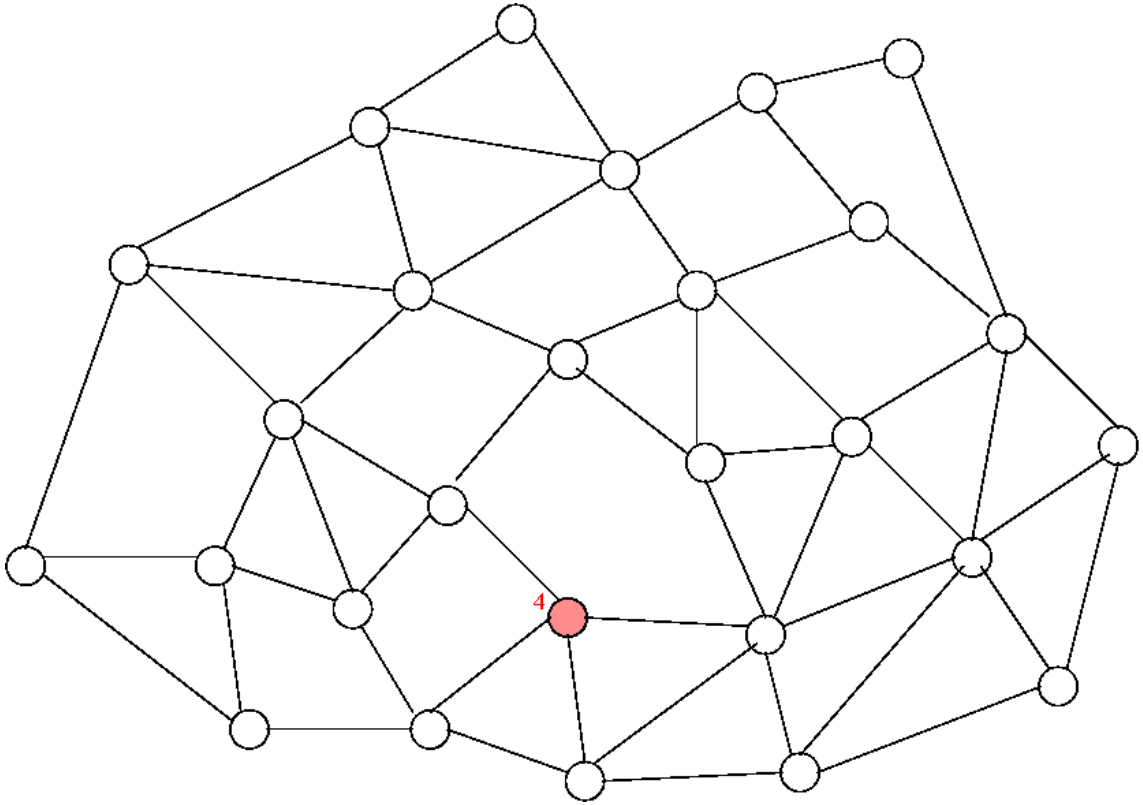
5 Greedy Graph Growing

- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



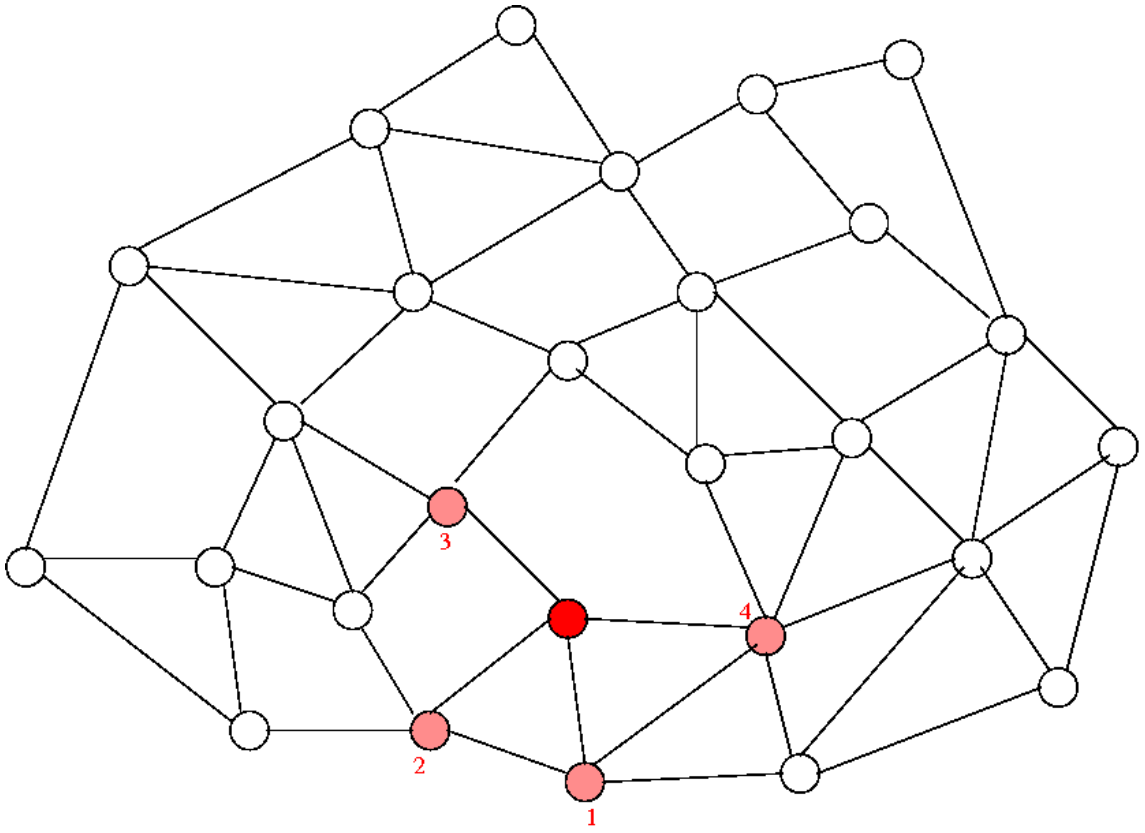
5 Greedy Graph Growing

- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

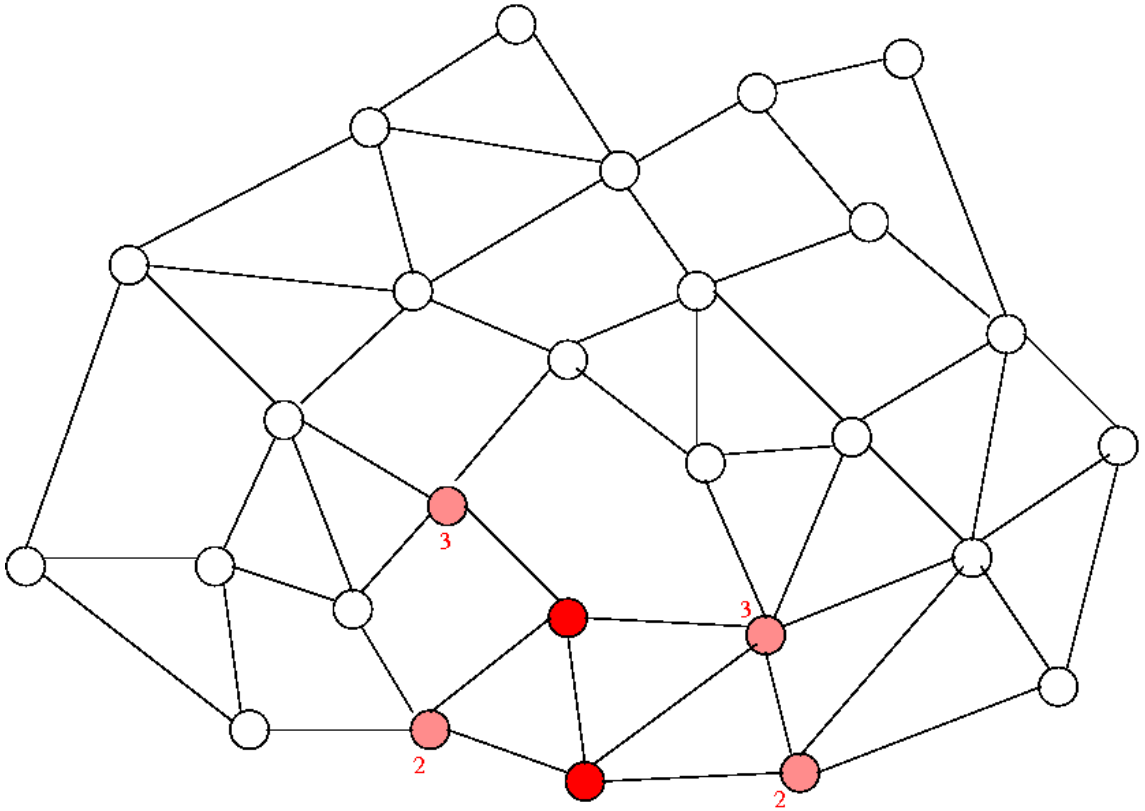
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

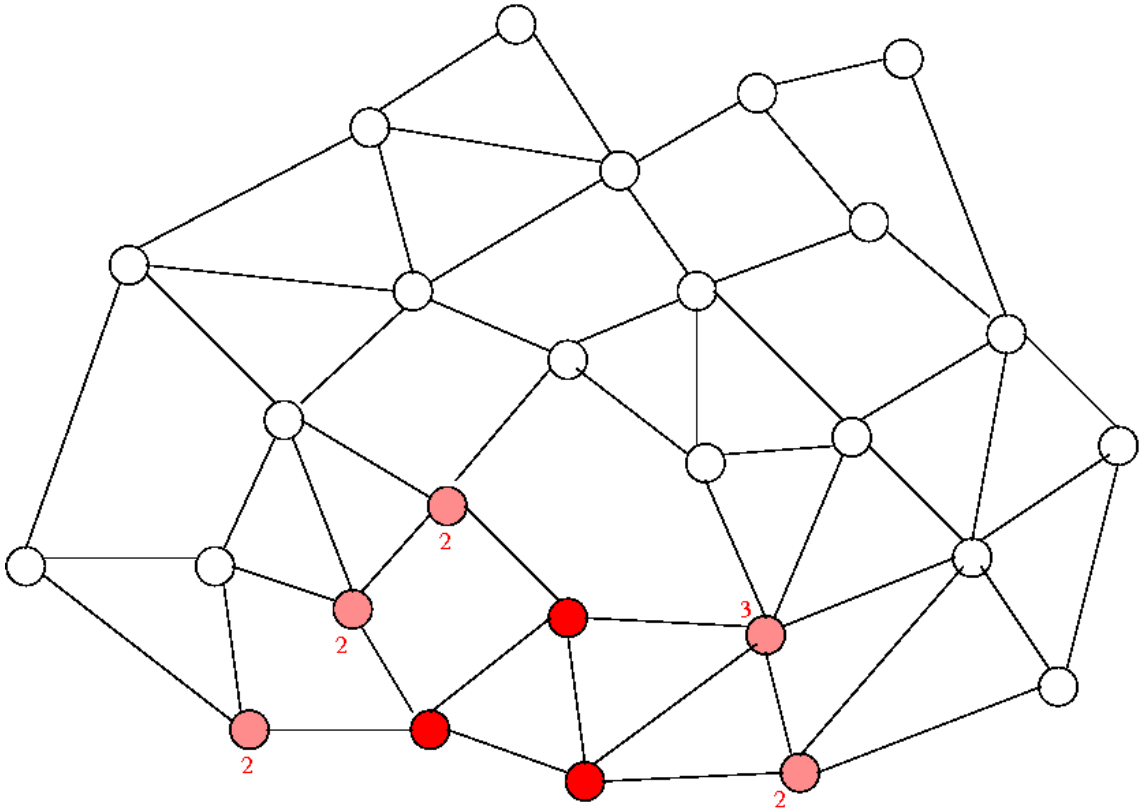
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

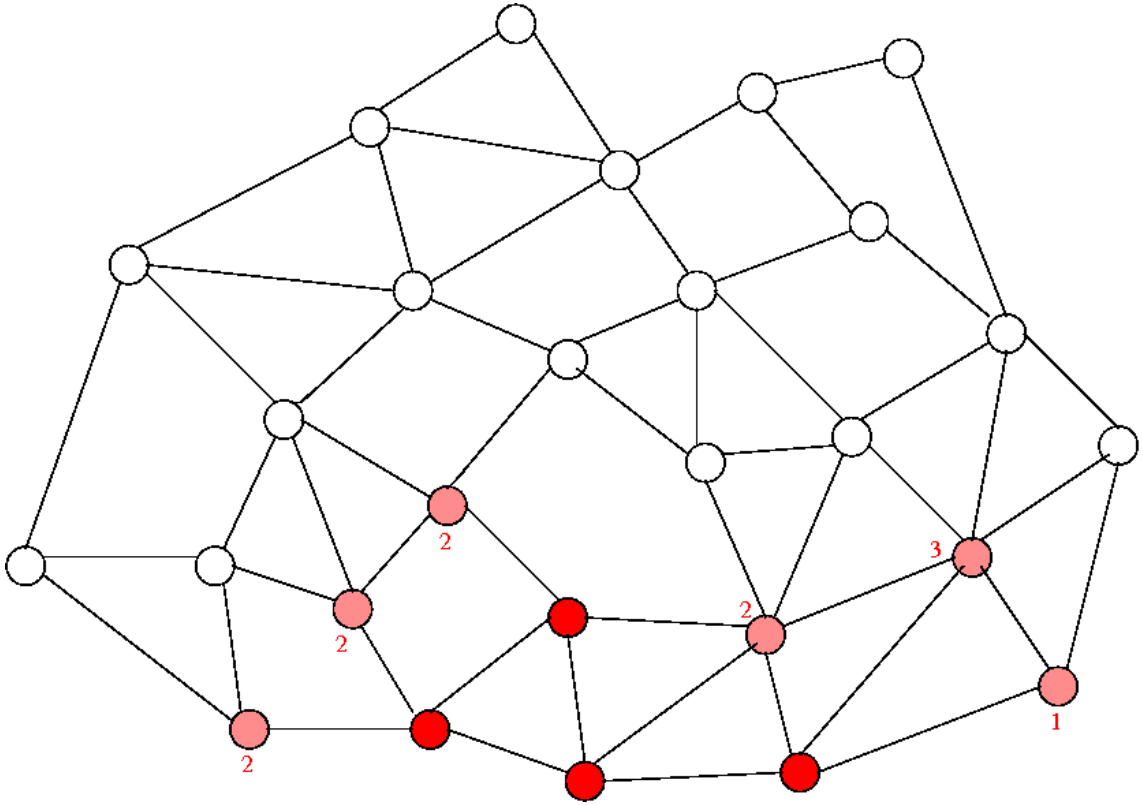
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

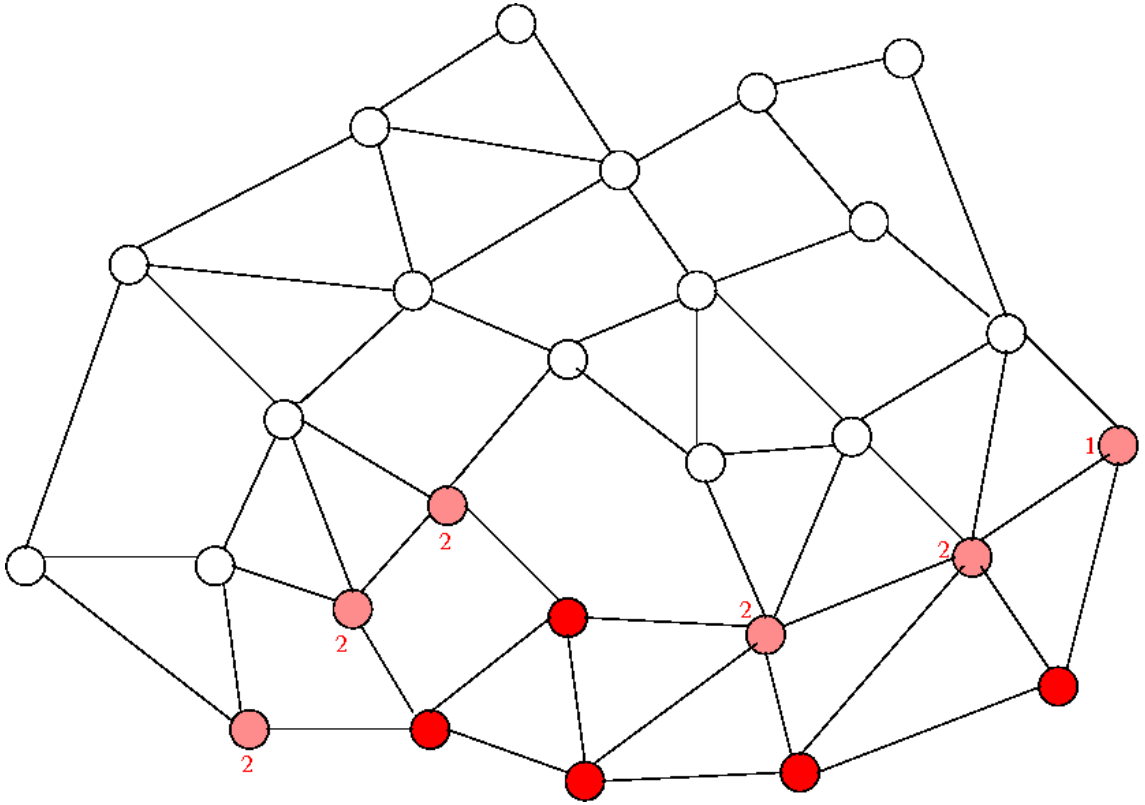
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

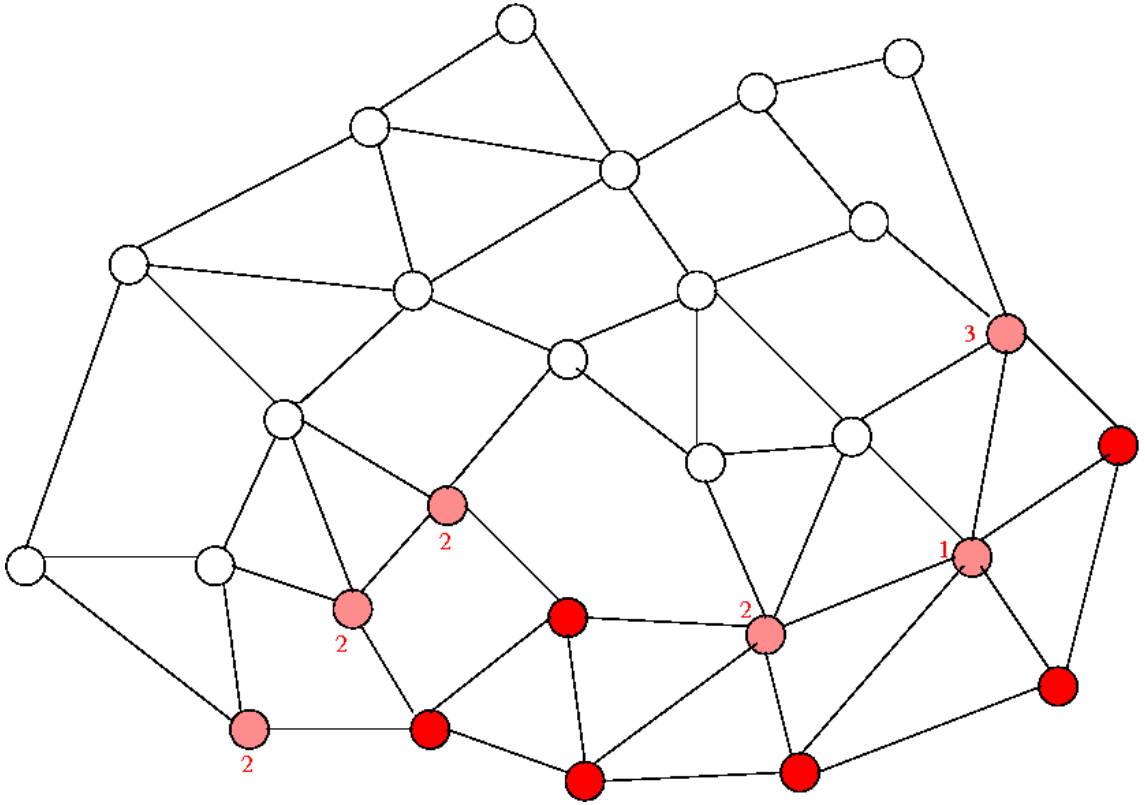
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

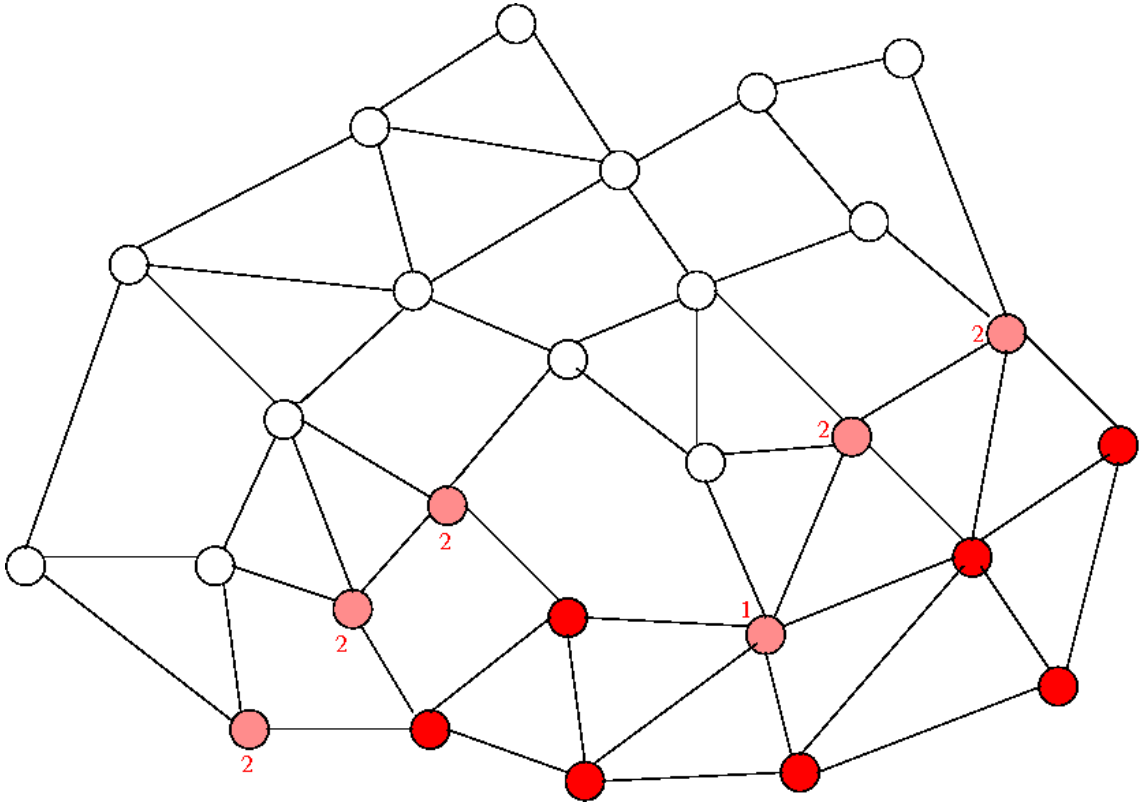
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

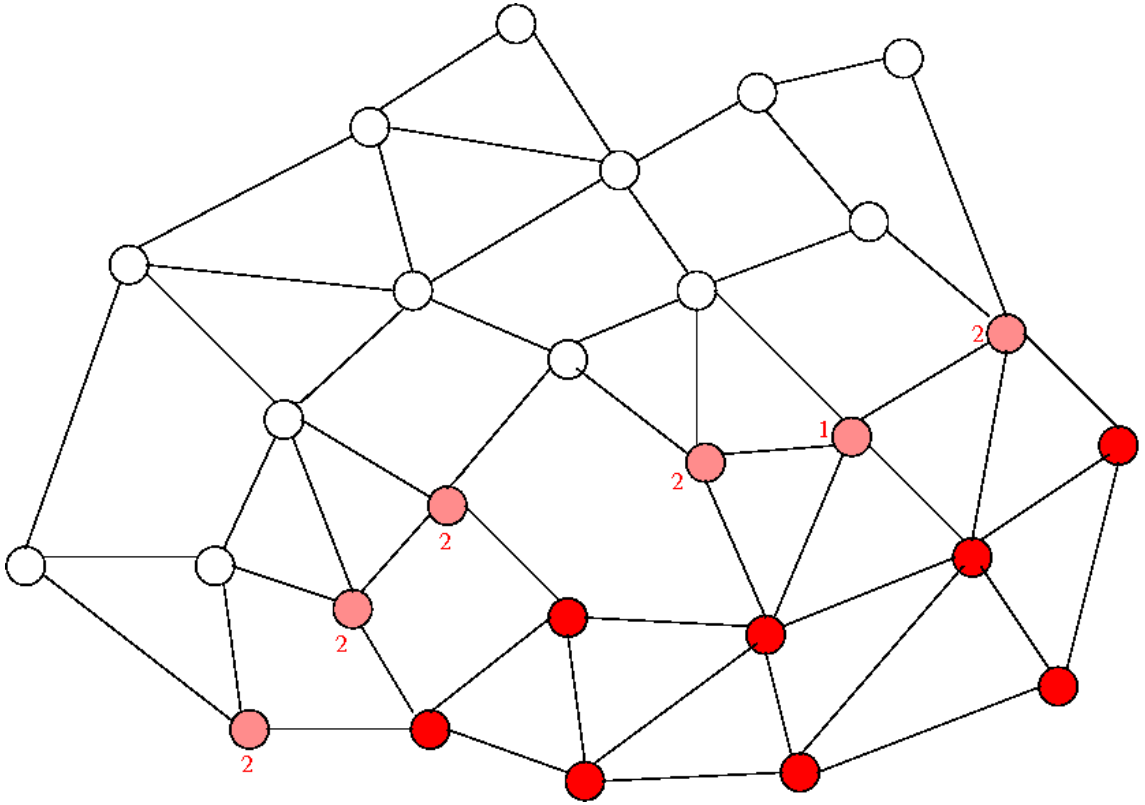
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

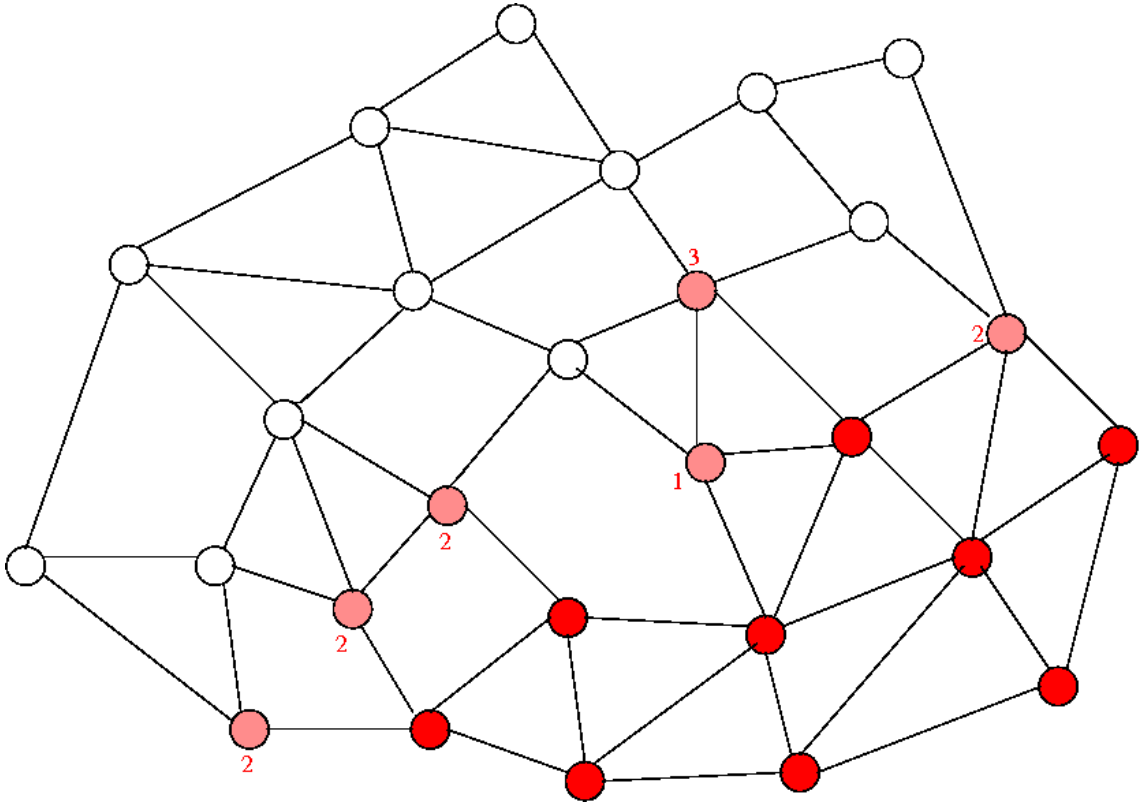
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

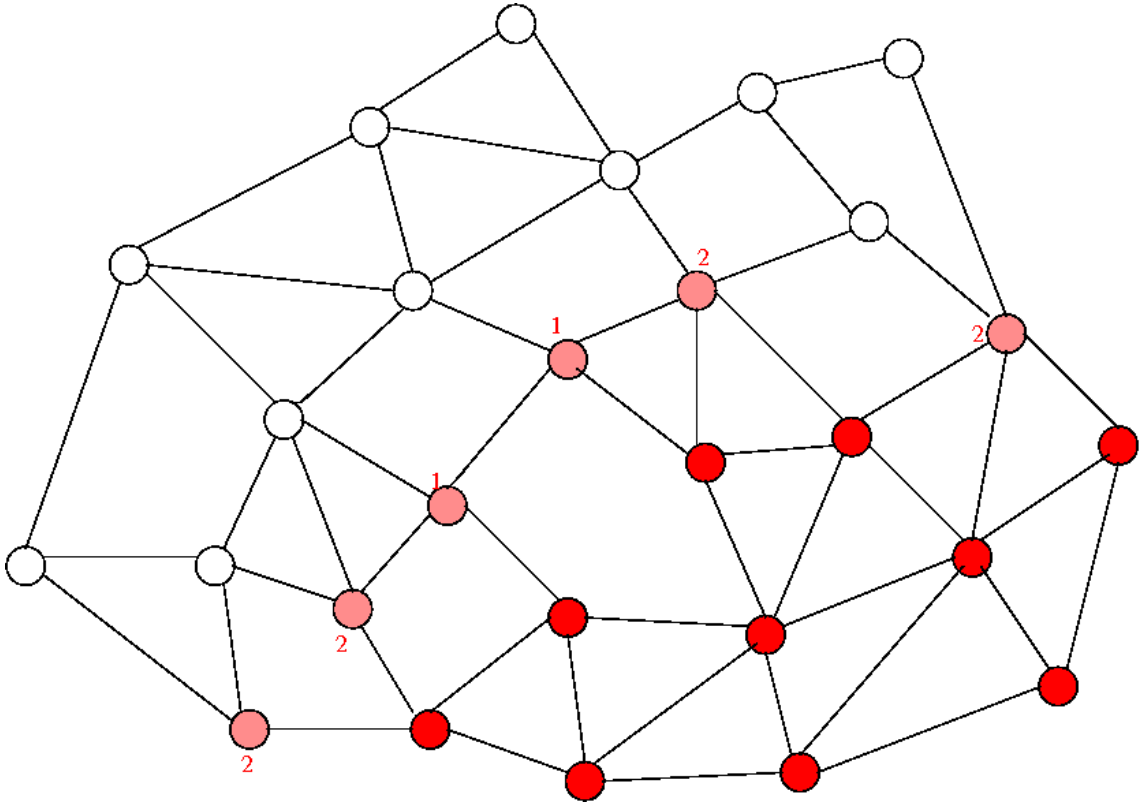
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

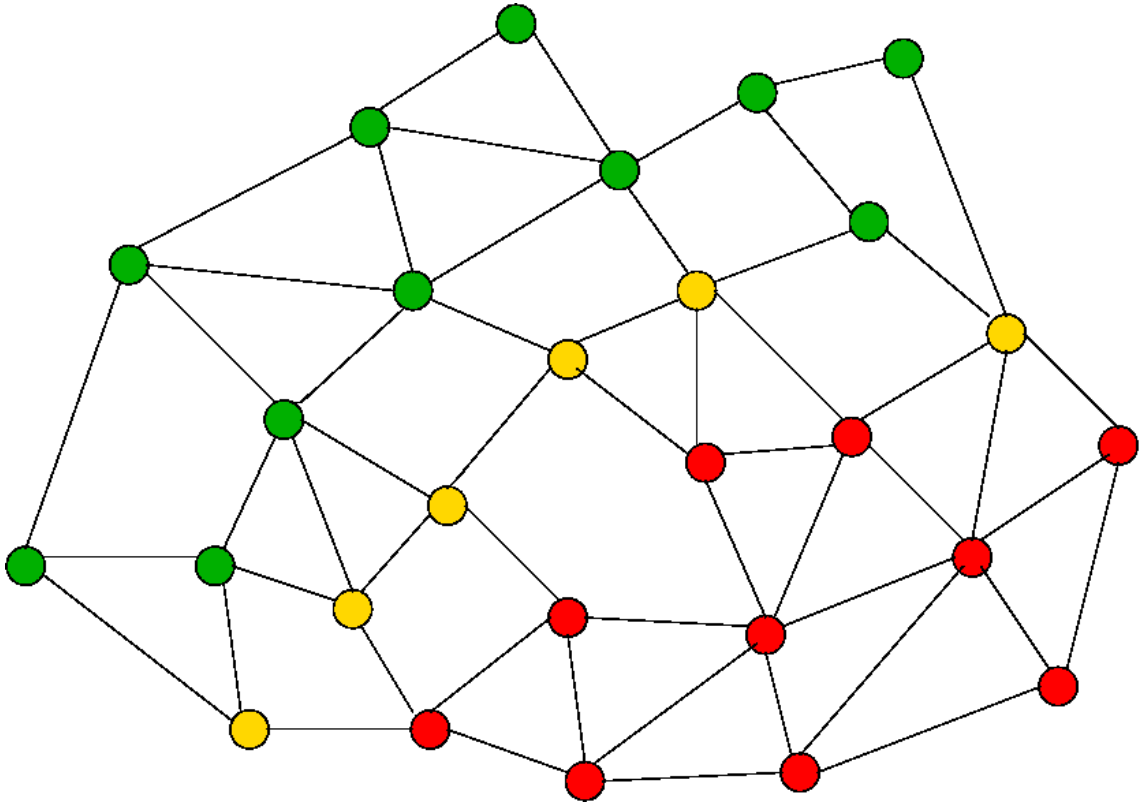
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



5 Greedy Graph Growing

Original version

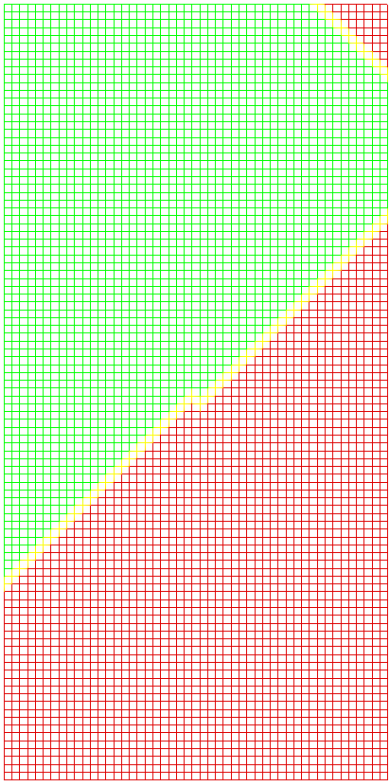
- **Initialization :** w ← randomVertexSeed(), $P_0 \leftarrow V \setminus \{w\}$, $P_1 \leftarrow \emptyset$, $S \leftarrow \{w\}$
- **At each step :** move a vertex v from S to P_1 and update separator (choose v which minimize S)
- **Stop** : whenever $|P_0| \approx |P_1|$



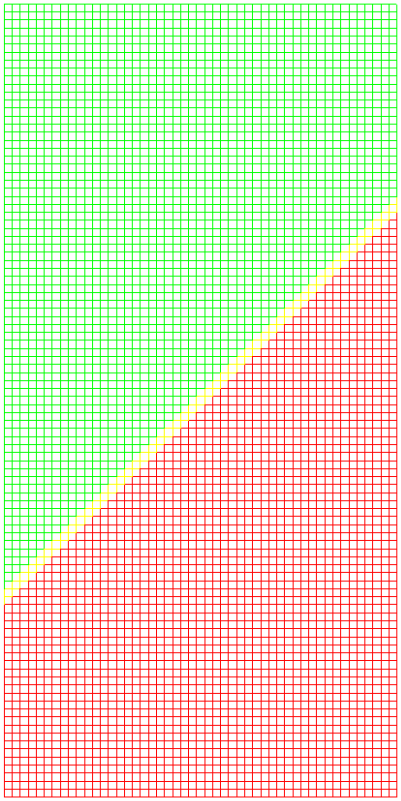
5 Greedy Graph Growing

Original version

- P_1 always connex, but not P_0
- Several passes made with different seeds, smallest separator eventually selected



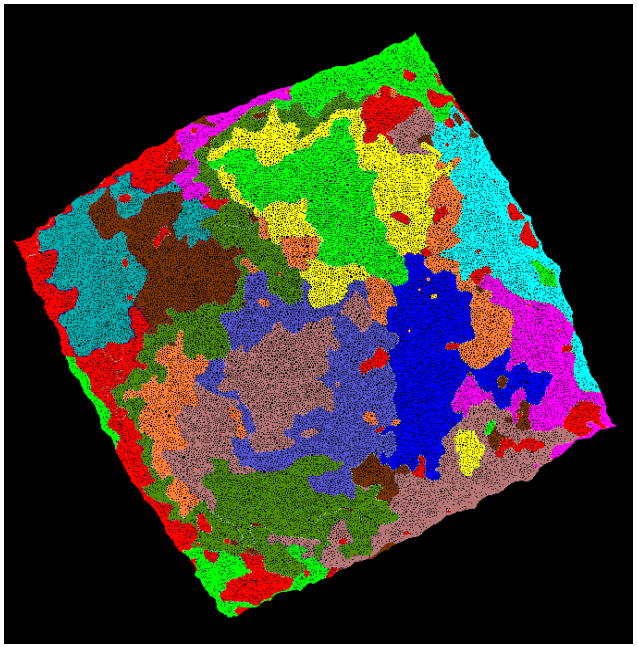
1 passe

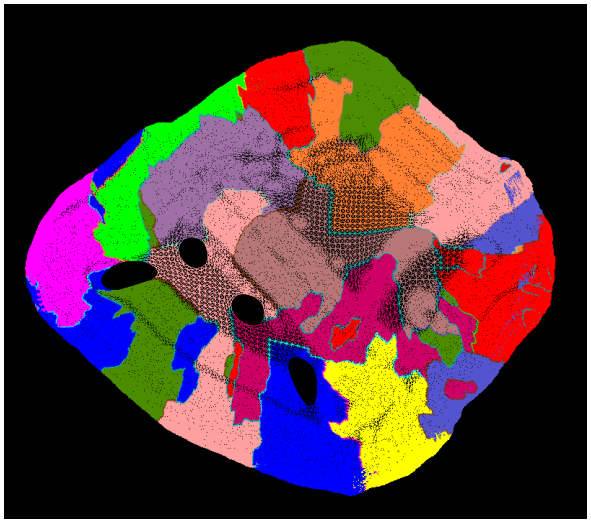


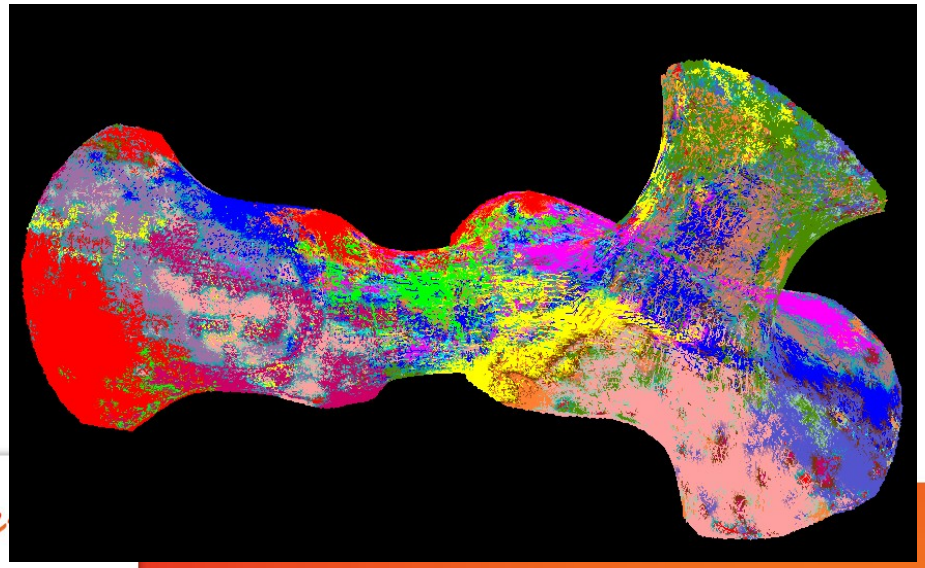
10 passes

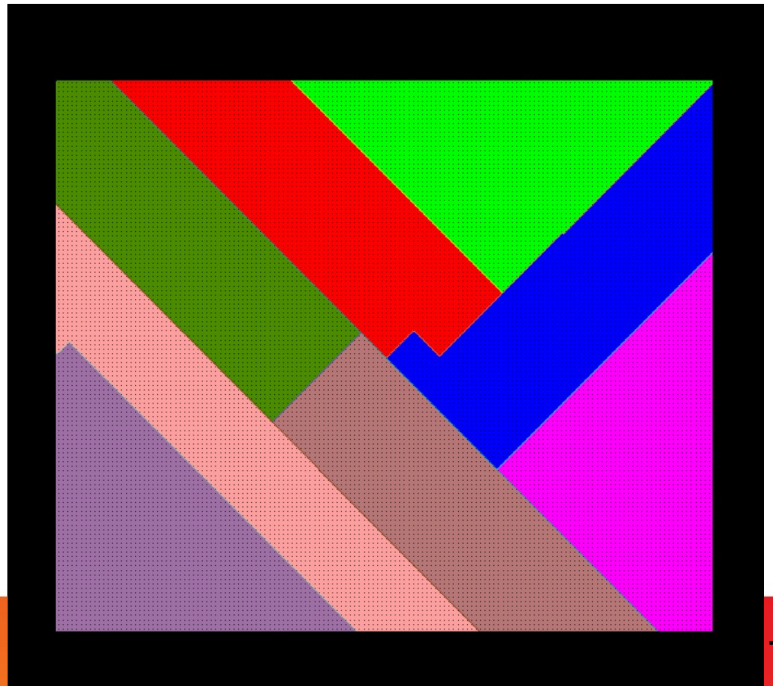
5 Greedy Graph Growing

Original version









5 Greedy Graph Growing

Modified version

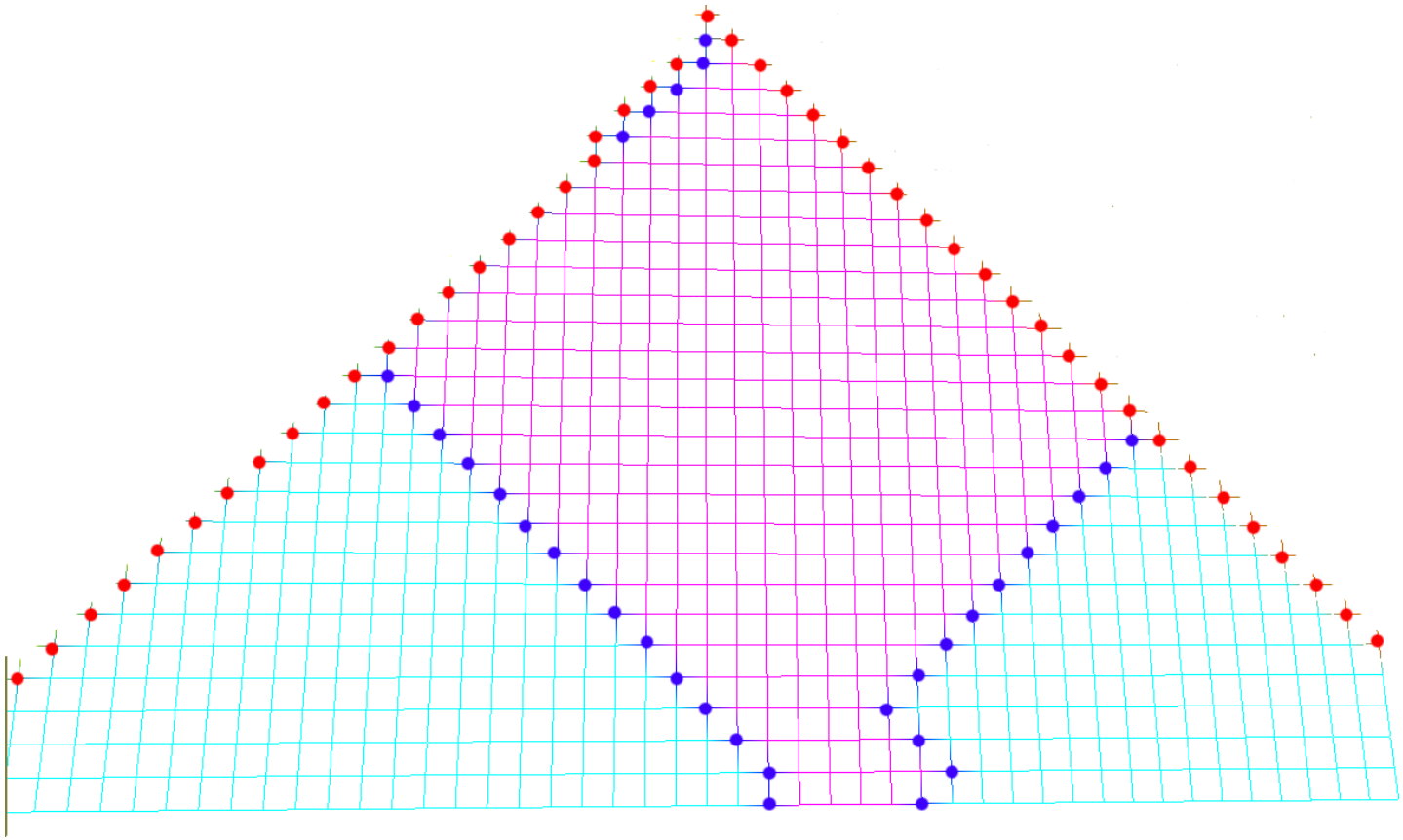
Idea: as P_1 grows, ensure $\frac{|P_0 \cap V_h|}{|P_0|} \simeq \frac{|P_1 \cap V_h|}{|P_1|}$

Changes:

- Choice of vertex v to move from S to P_1:
  - If $\frac{|P_1 \cap V_h|}{|P_1|} < \frac{|P_0 \cap V_h|}{|P_0|}$ and $S \cap V_h \neq \emptyset$ → choose among $S \cap V_h$
  - Minimize separator
- Choice of the seed among halo
- Choice of « best pass » based on separator minimization *and* halo balance

5 Greedy Graph Growing

Modified version



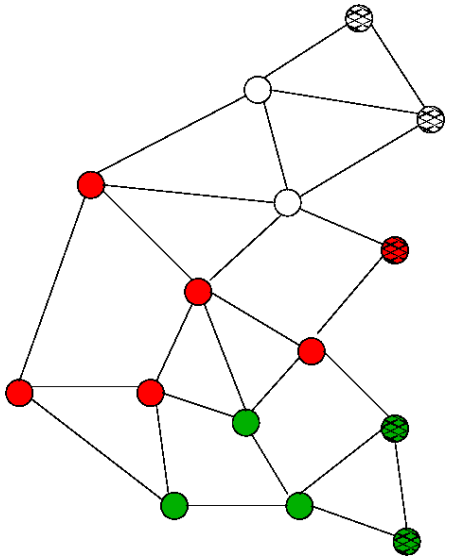
Problems:

- Allowing larger separators often results in non-connected P_0
- Always possible to enhance halo balance by adding a connected component to P_0

6 Double Greedy Graph Growing

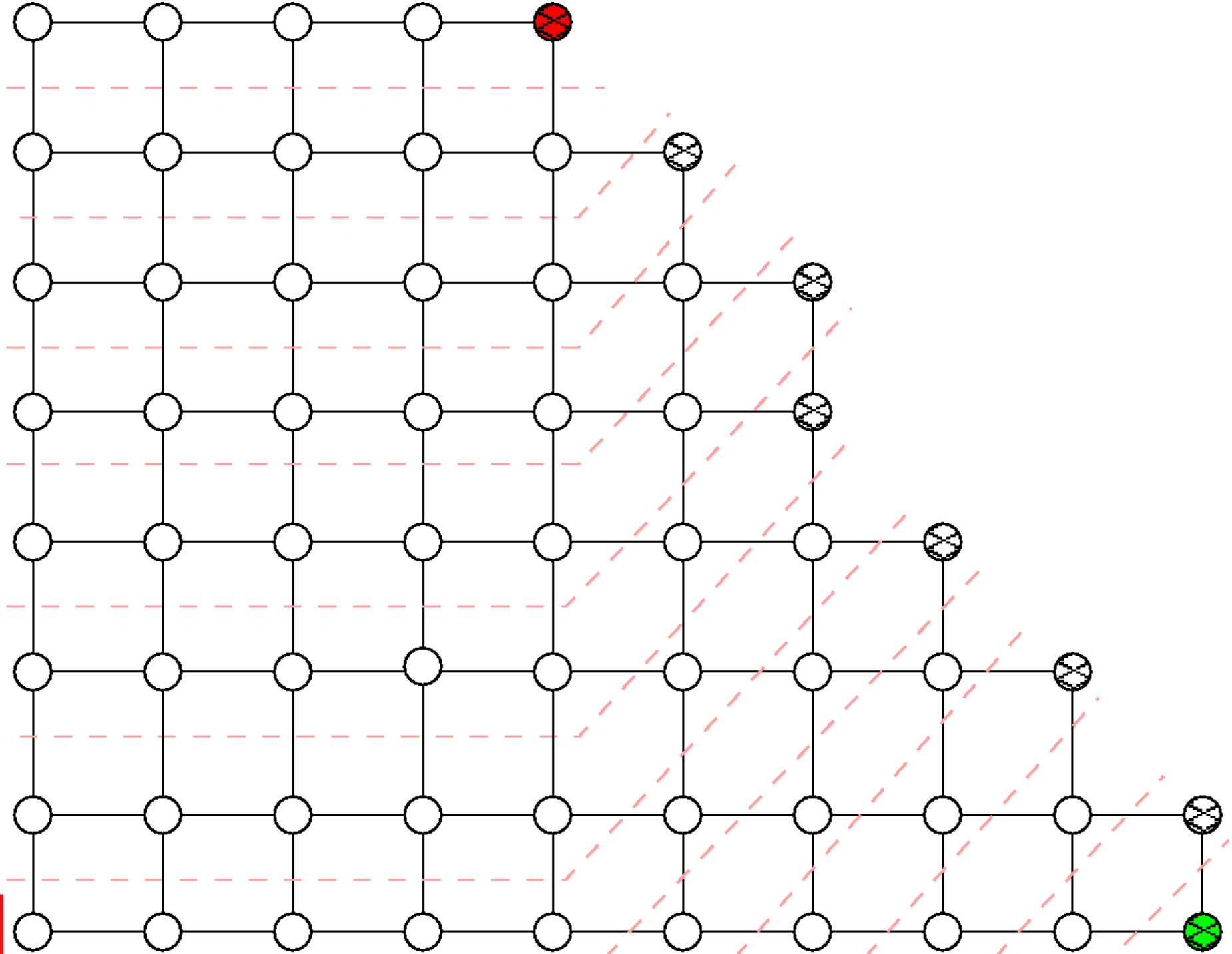
Idea: 2 random seeds, both parts are made grown simultaneously until all vertices are taken

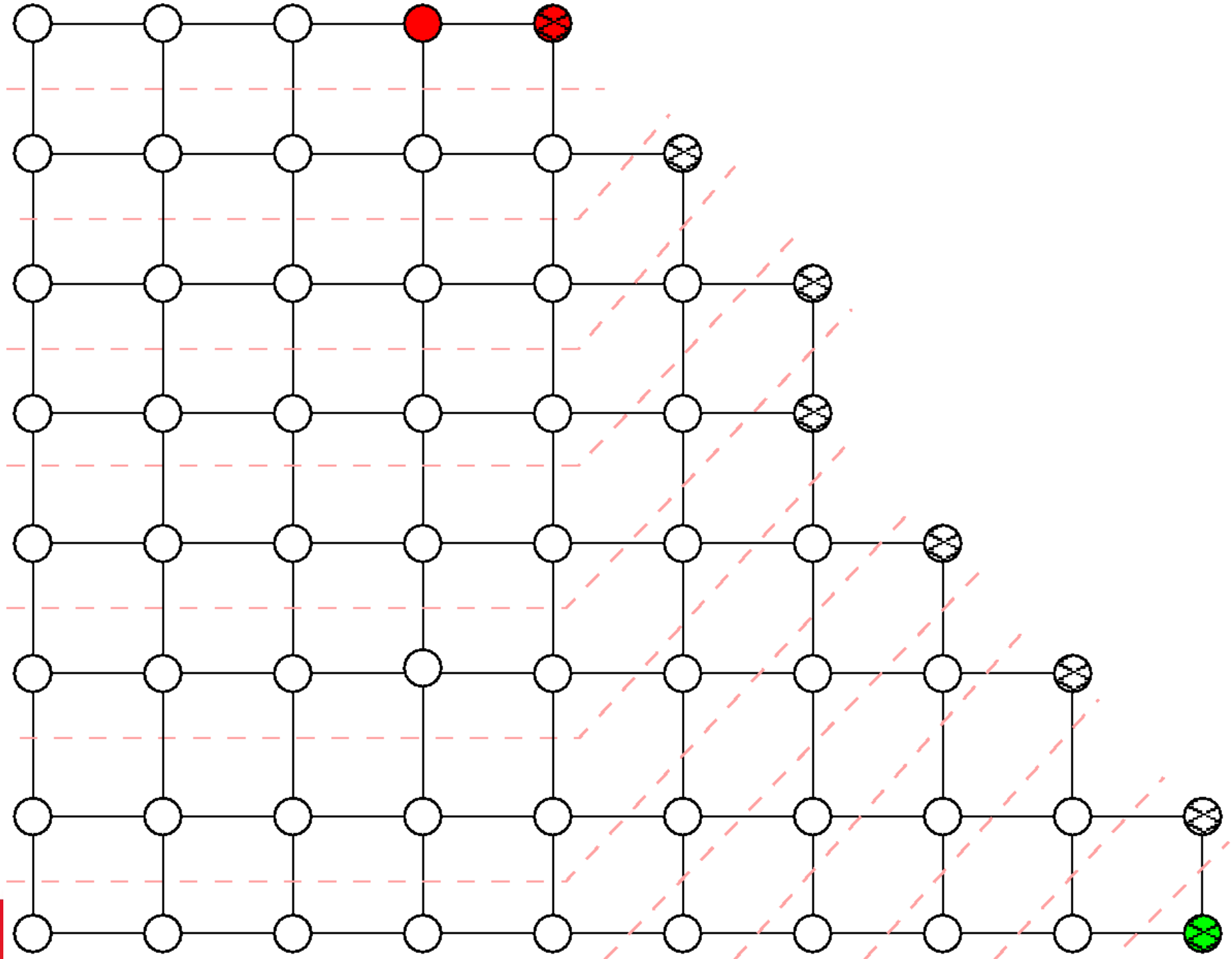
Problem: one part may block the progression of the other

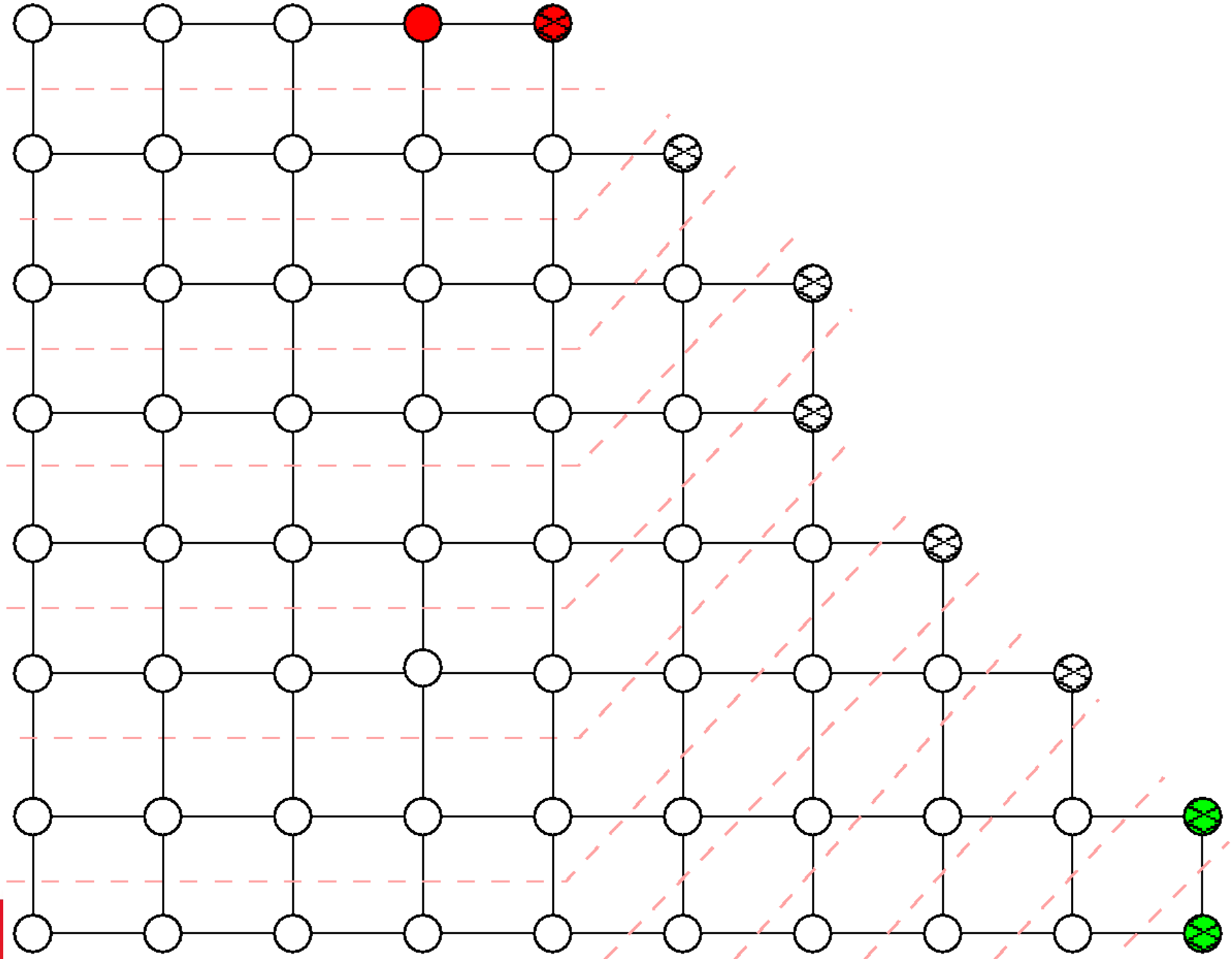


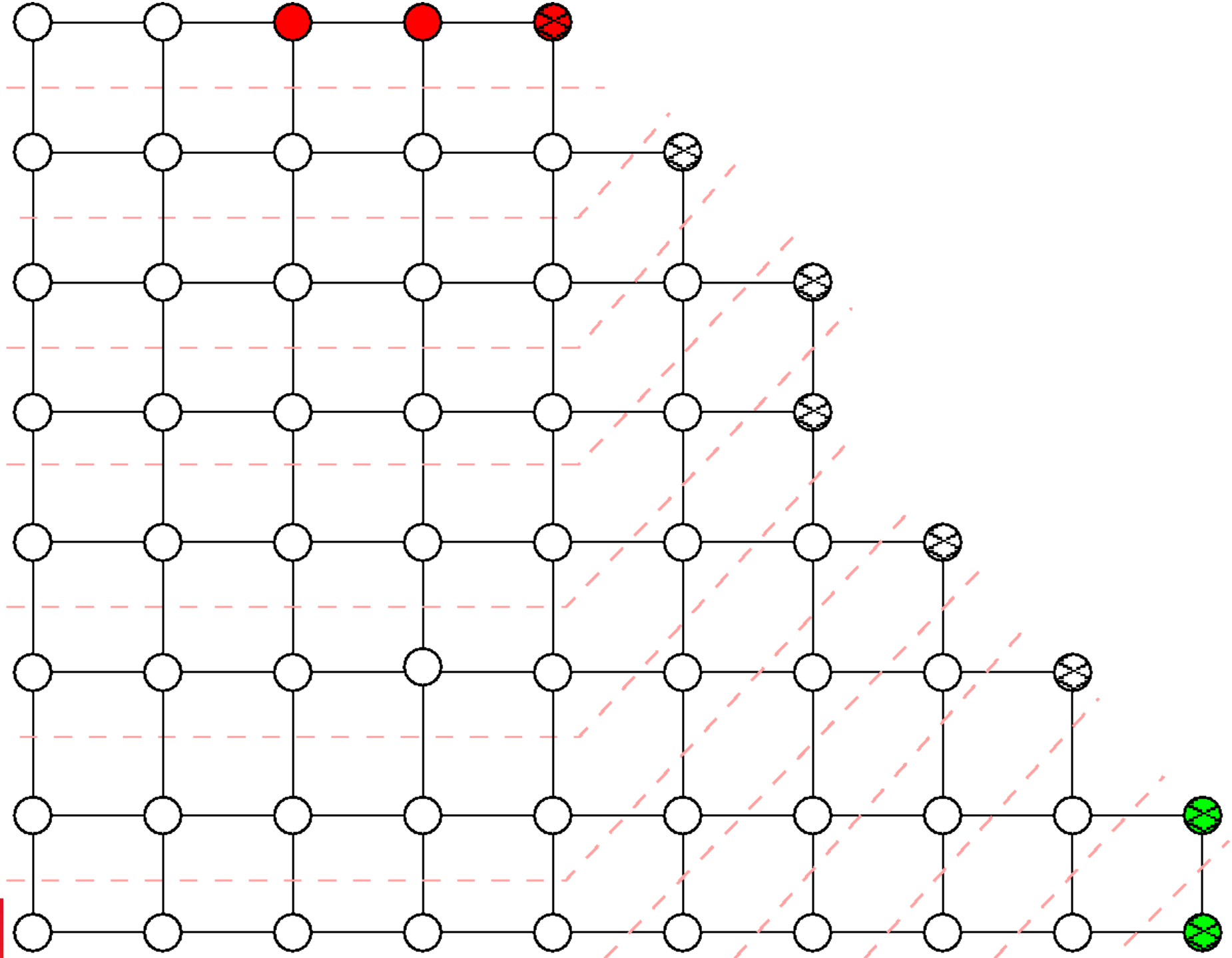
Solutions:

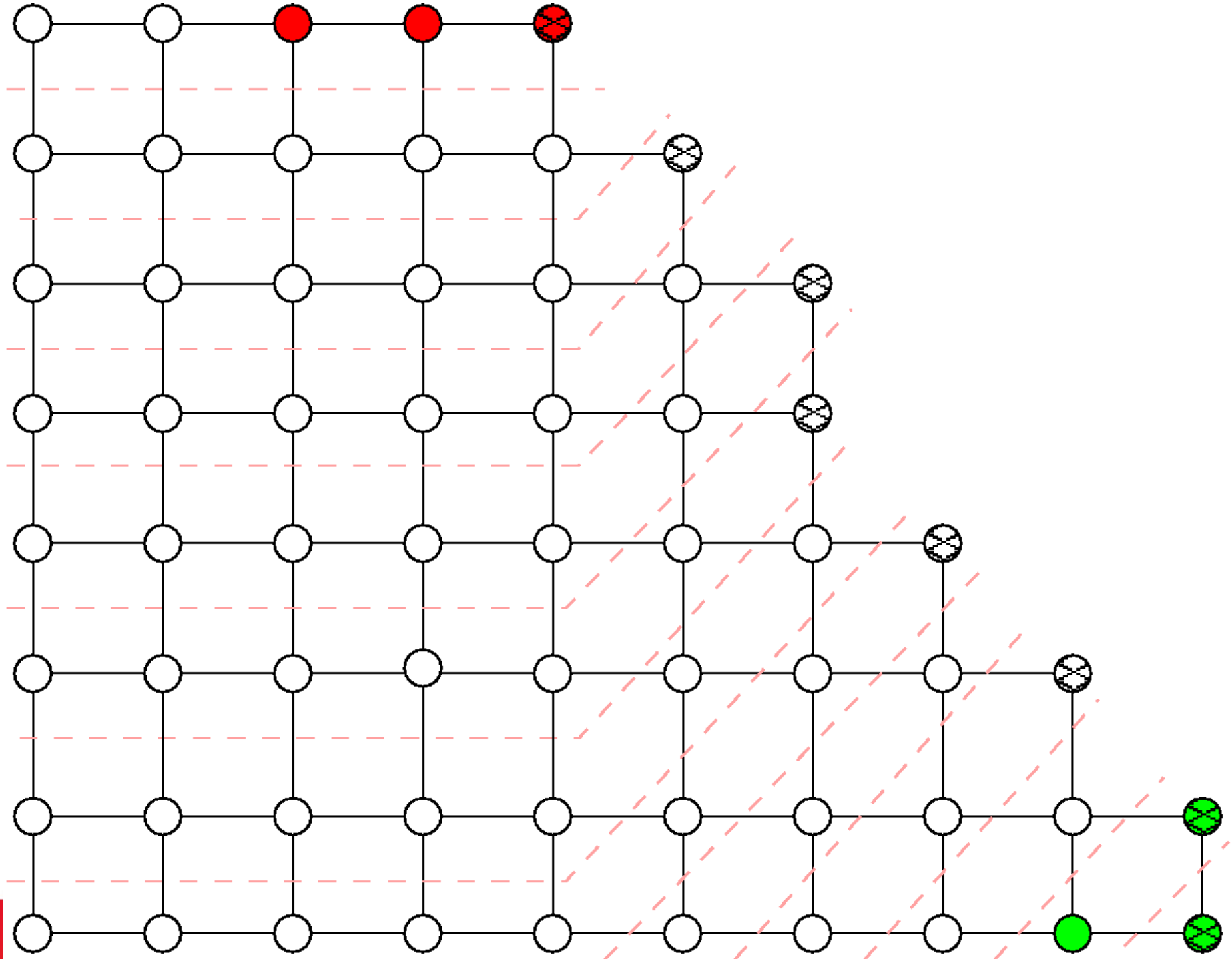
- Seeds s_0 and s_1 as far as possible in halo
- If several choices → choose suitable vertex the nearest from s_i / farthest from s_{1-i}
- If stucked with a lot of vertices left → make a new pass

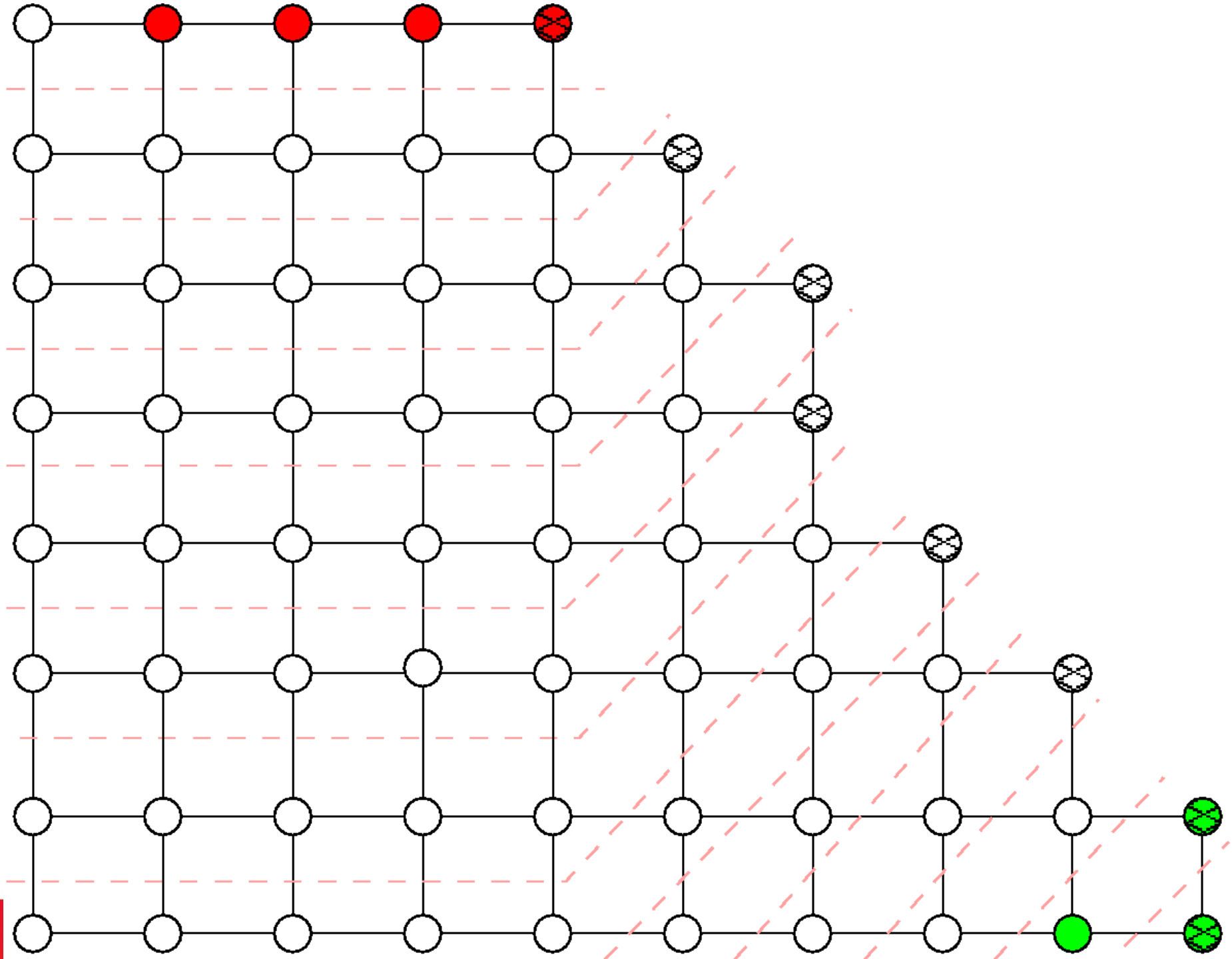


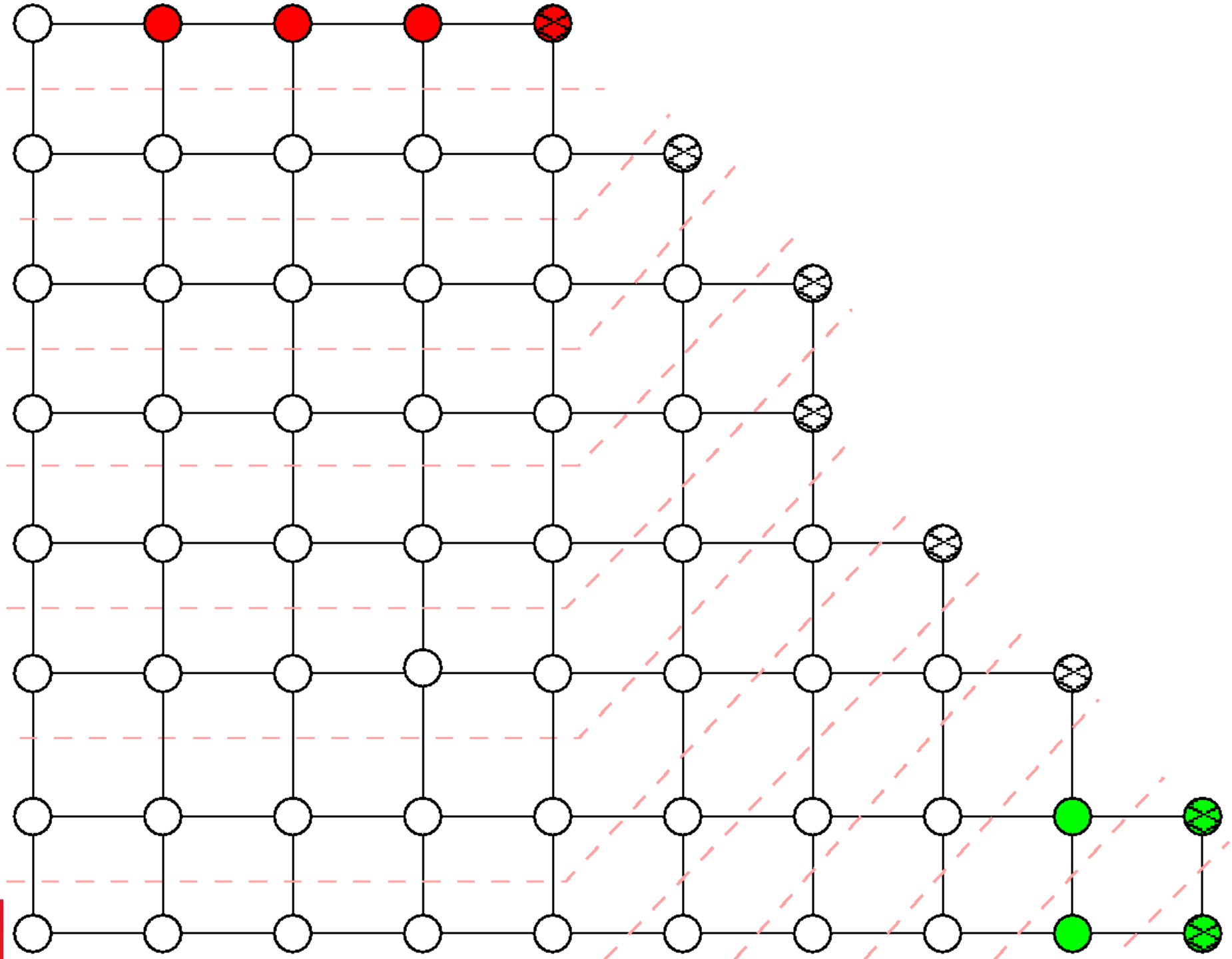


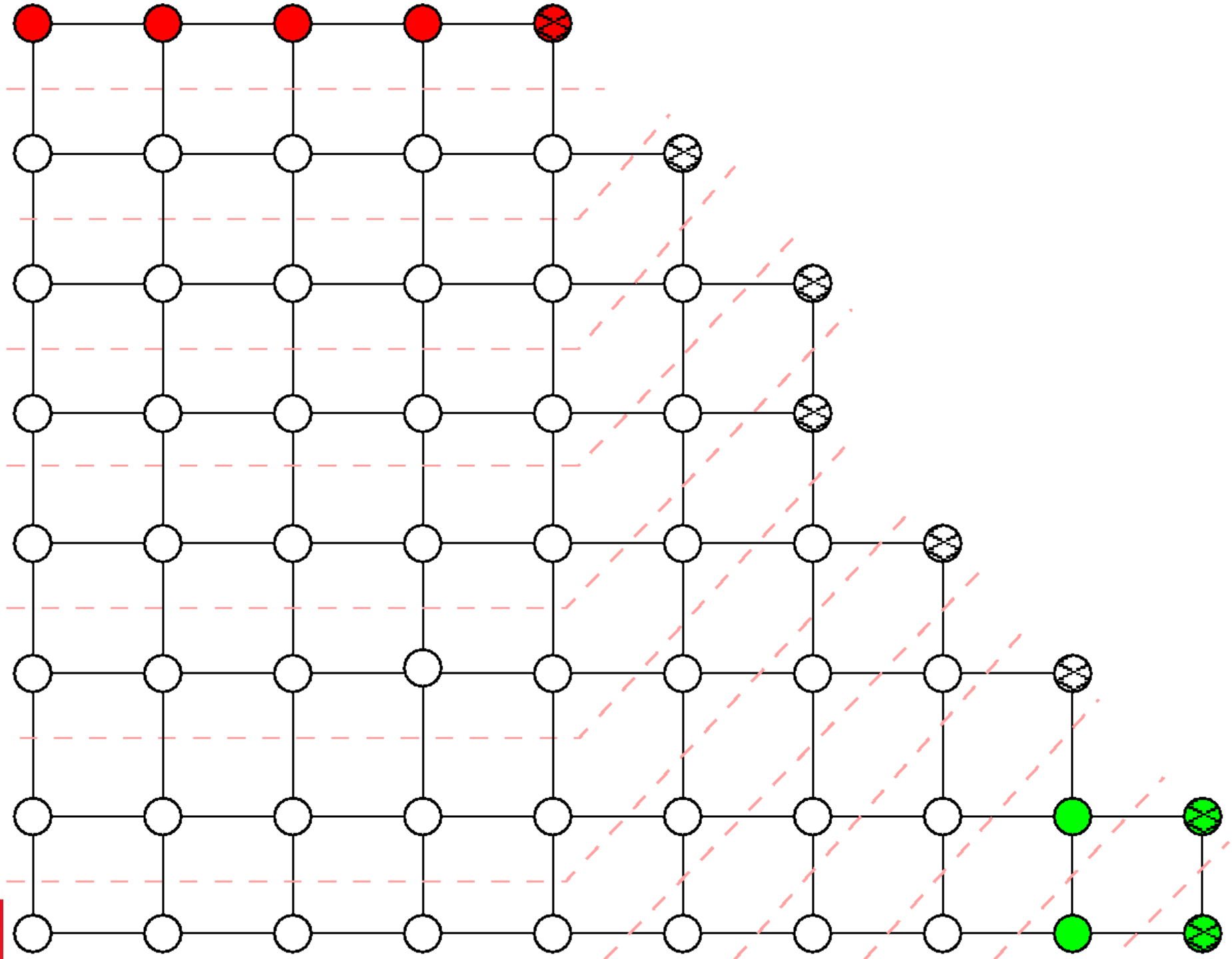


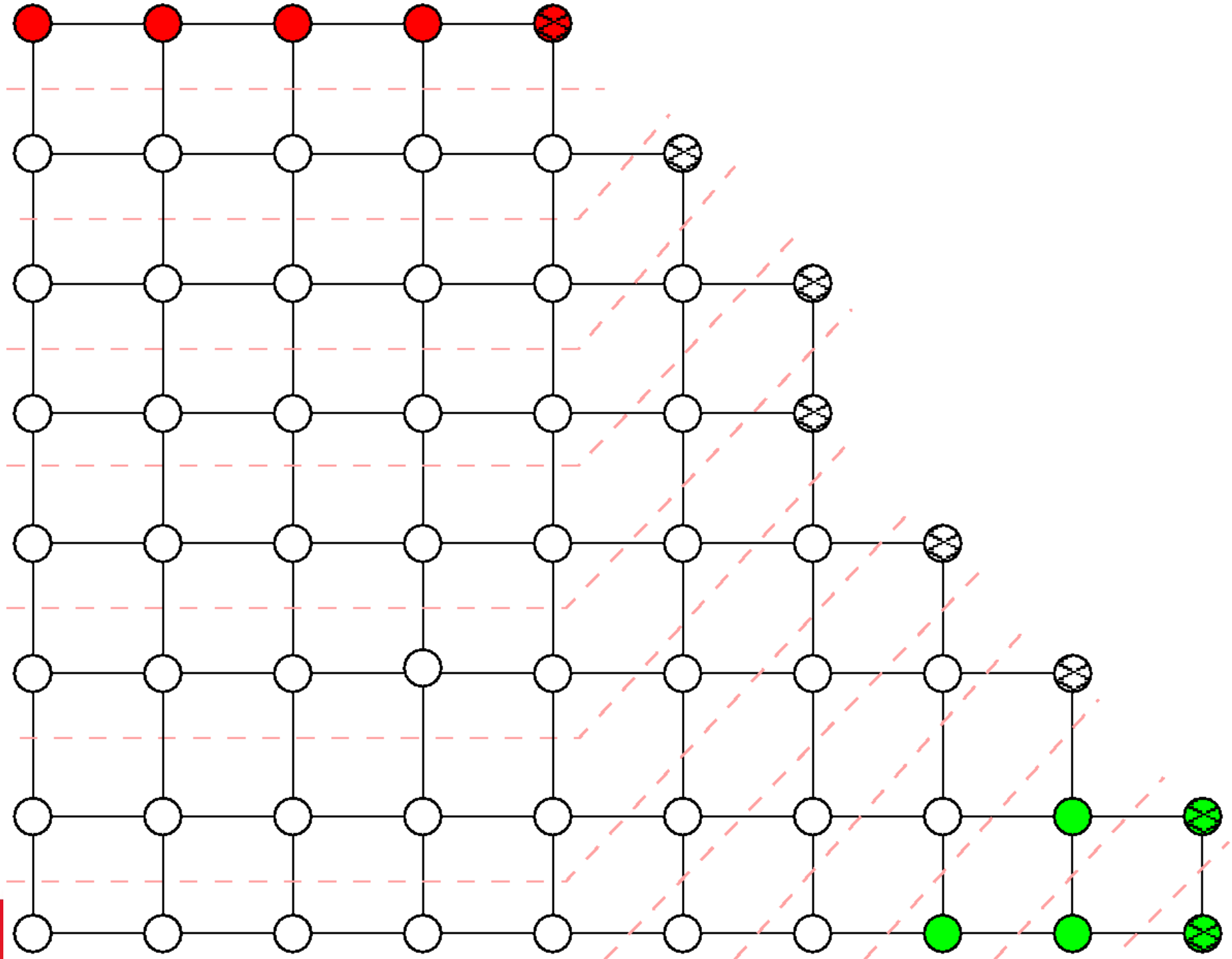


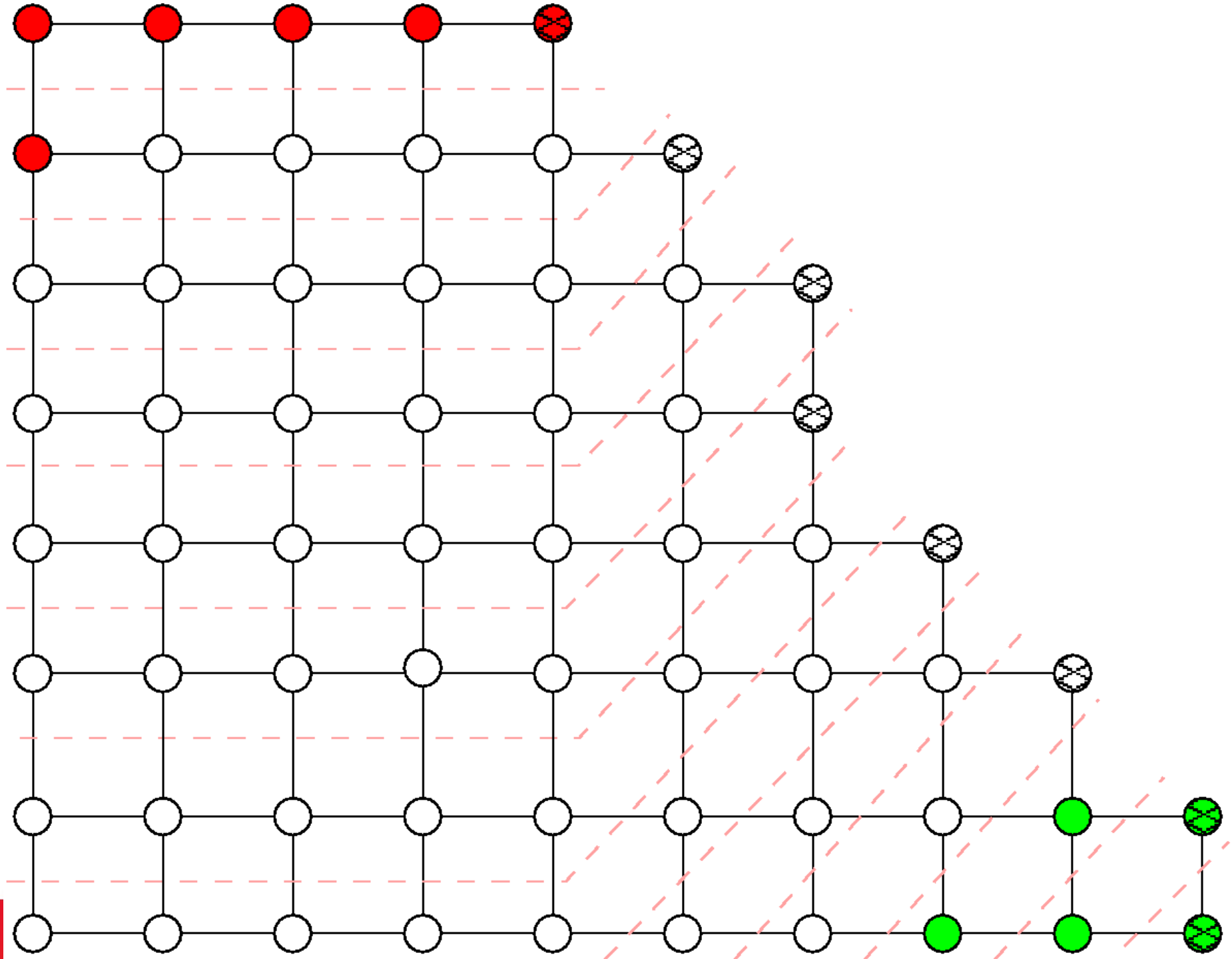


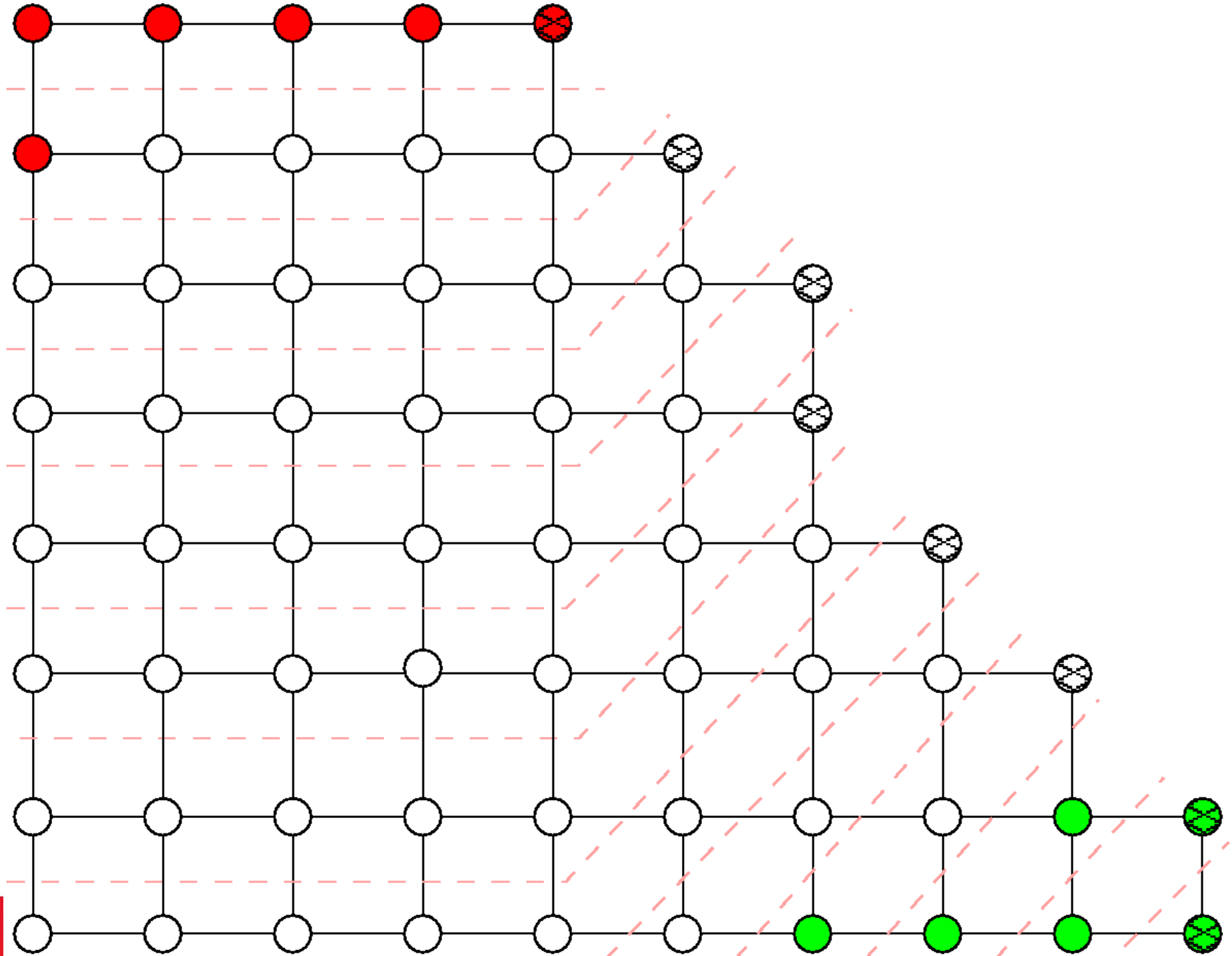


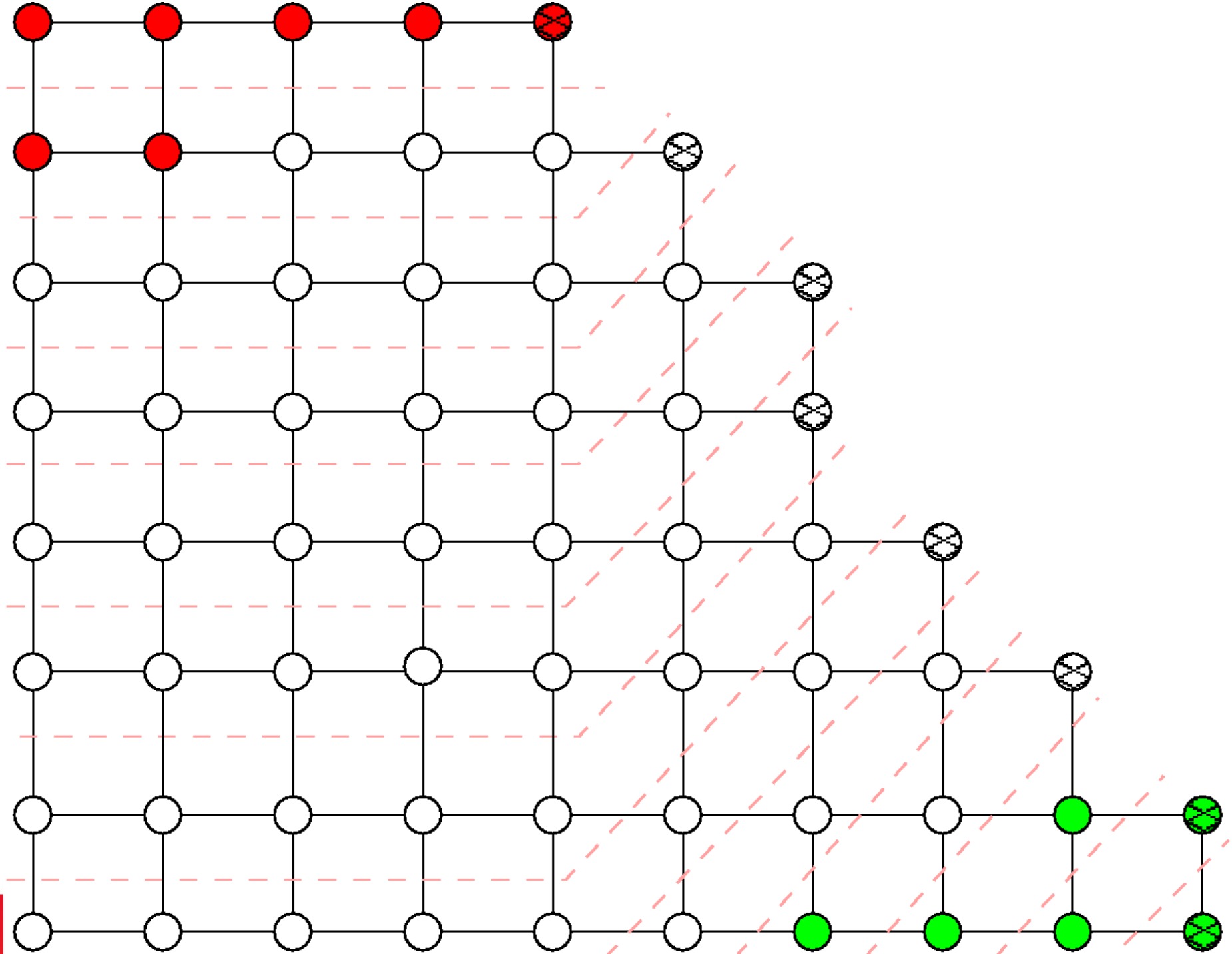


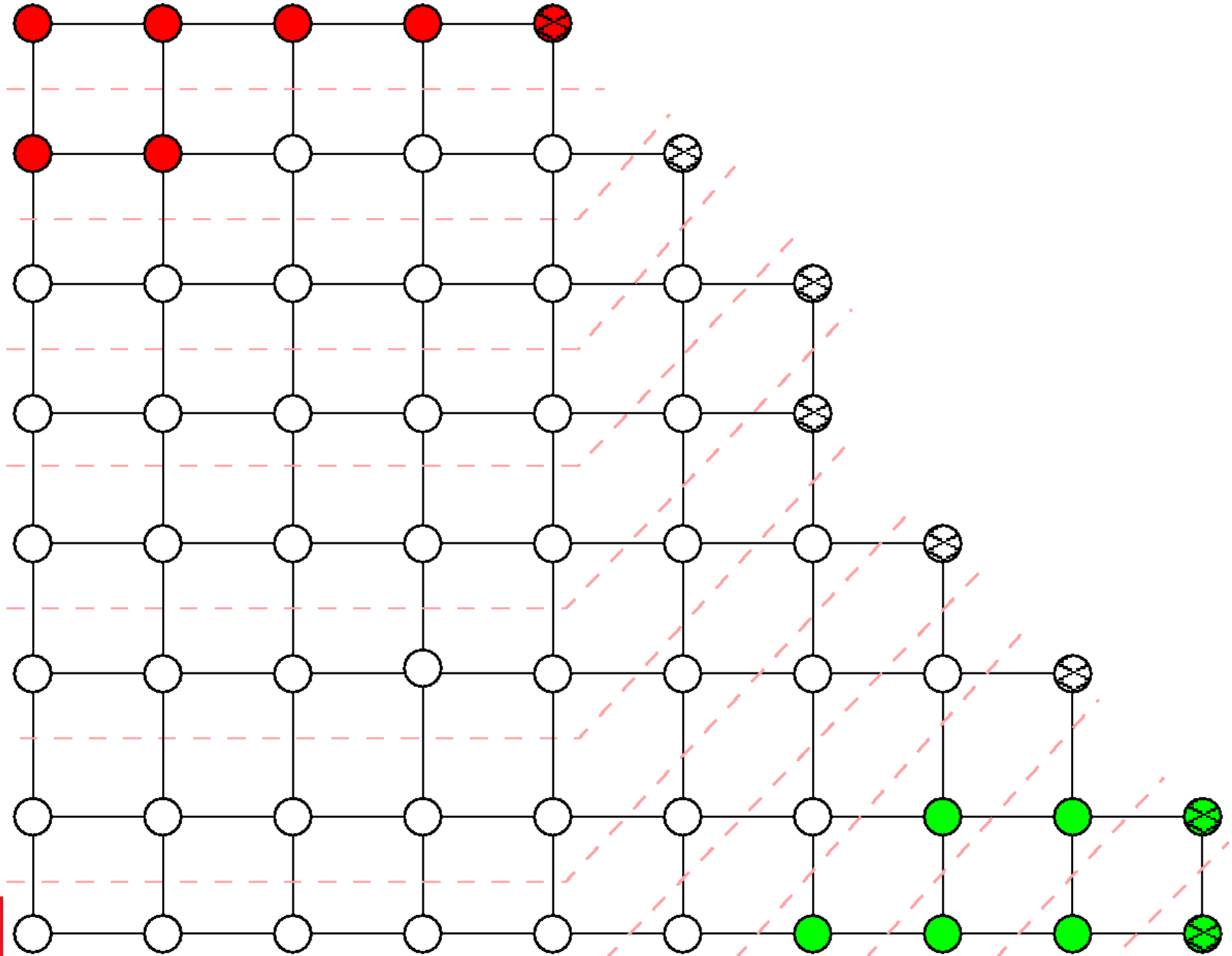


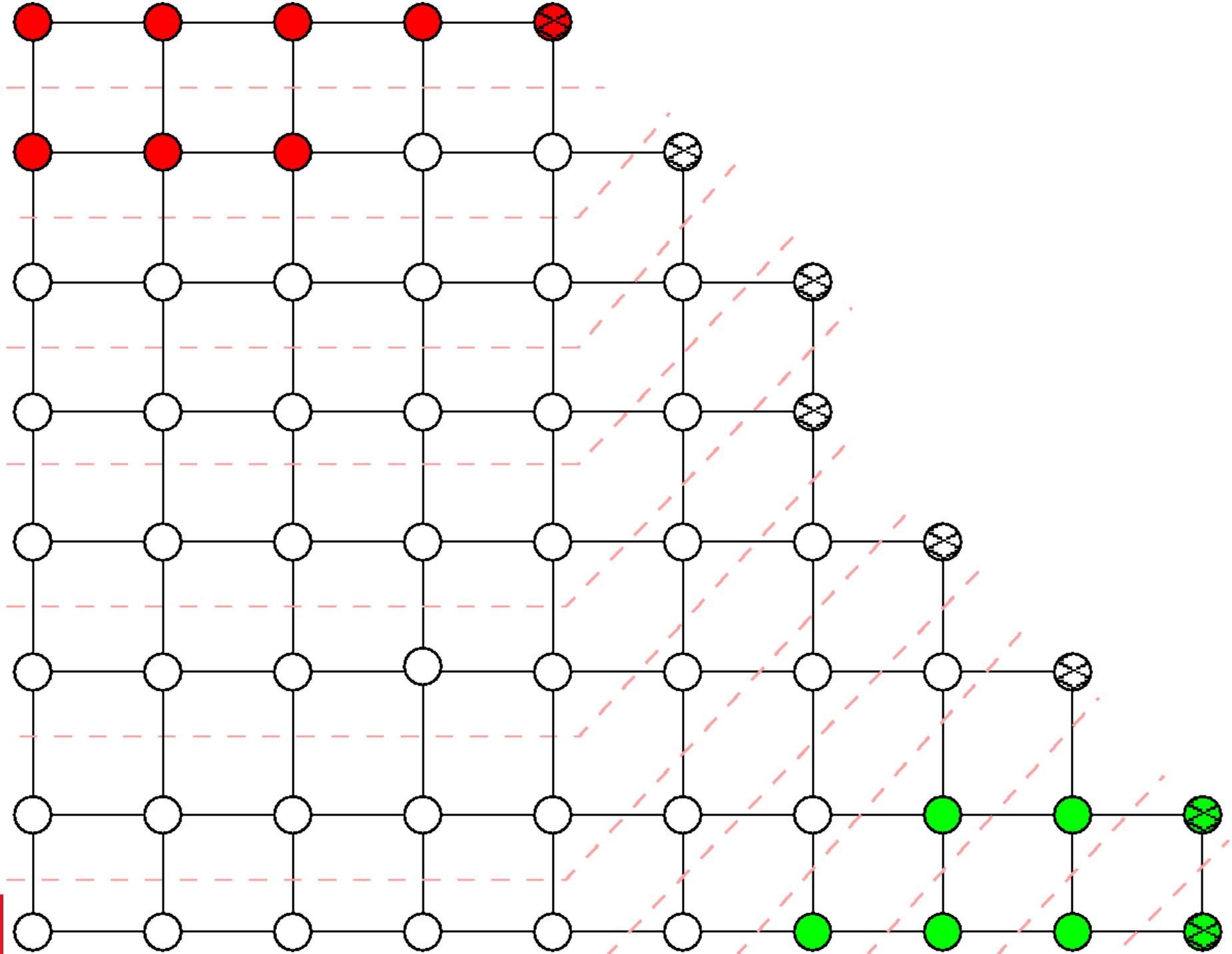


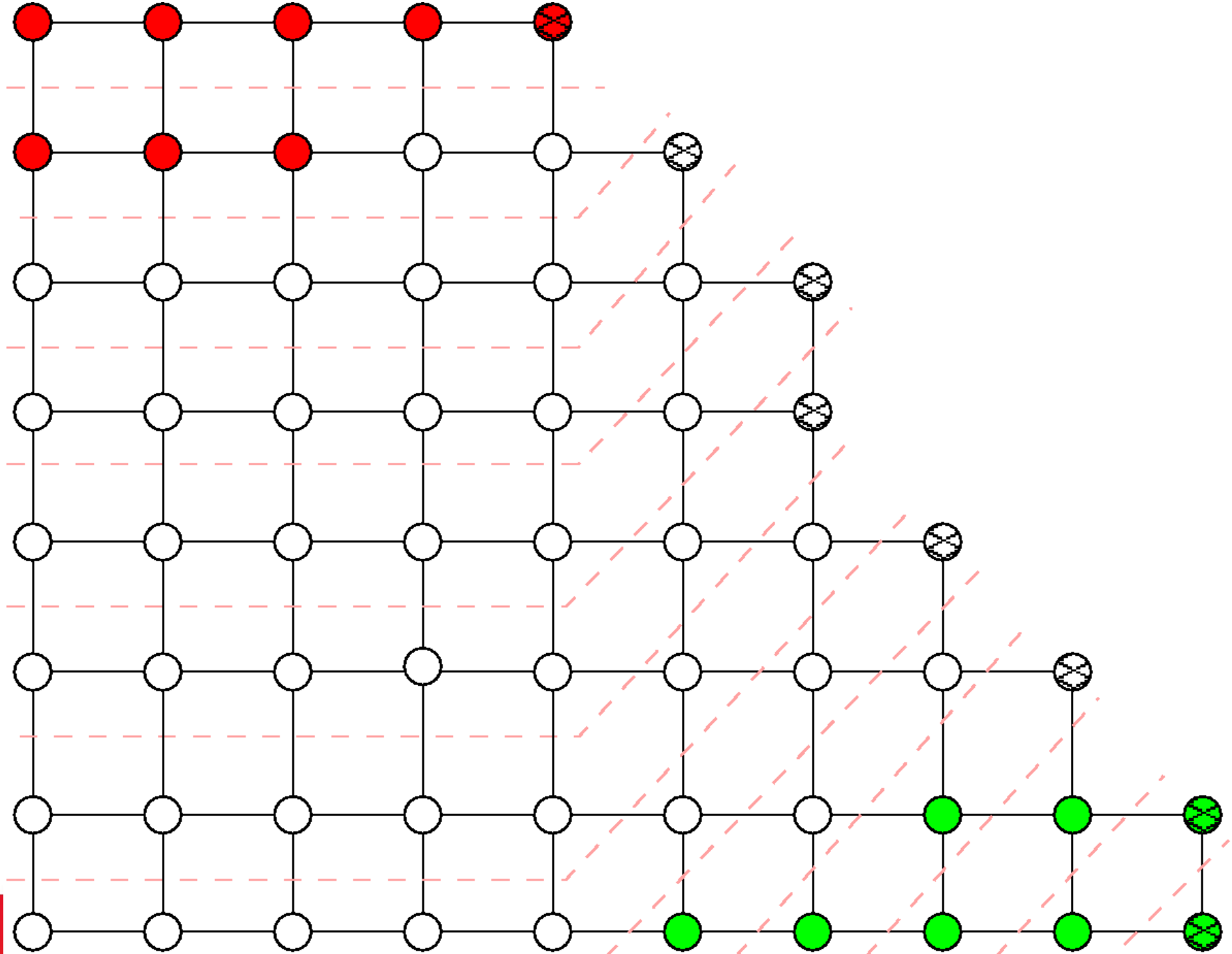


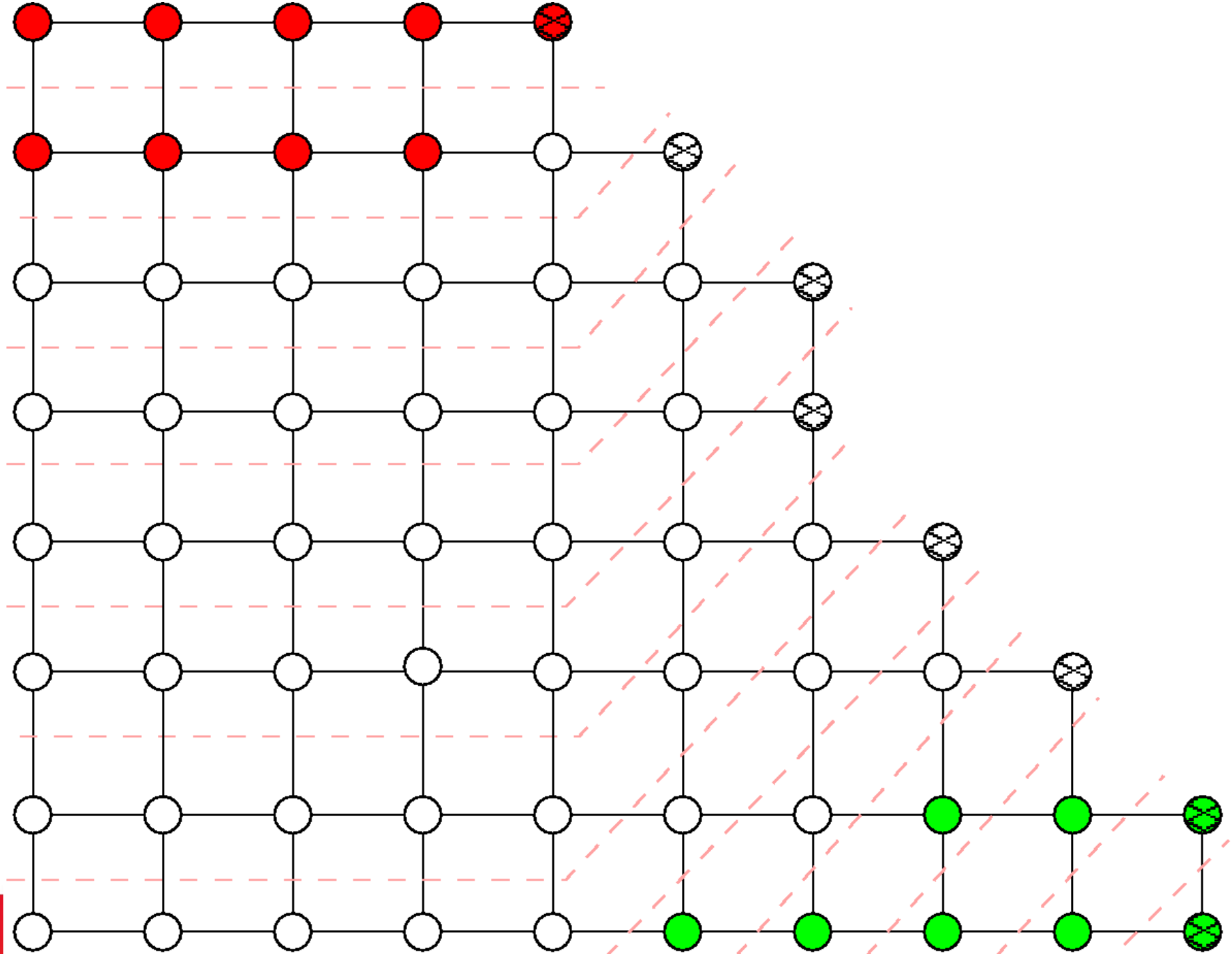


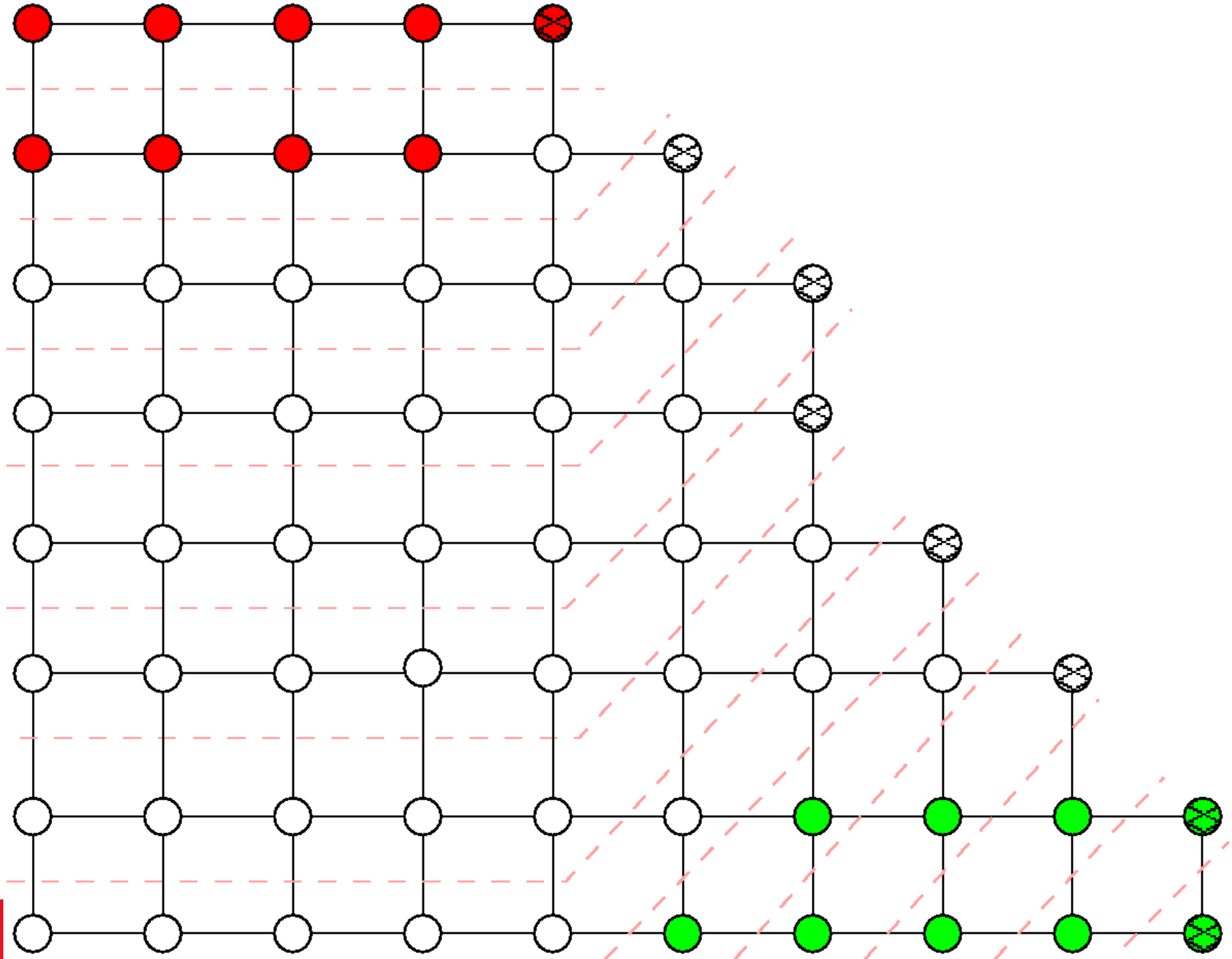


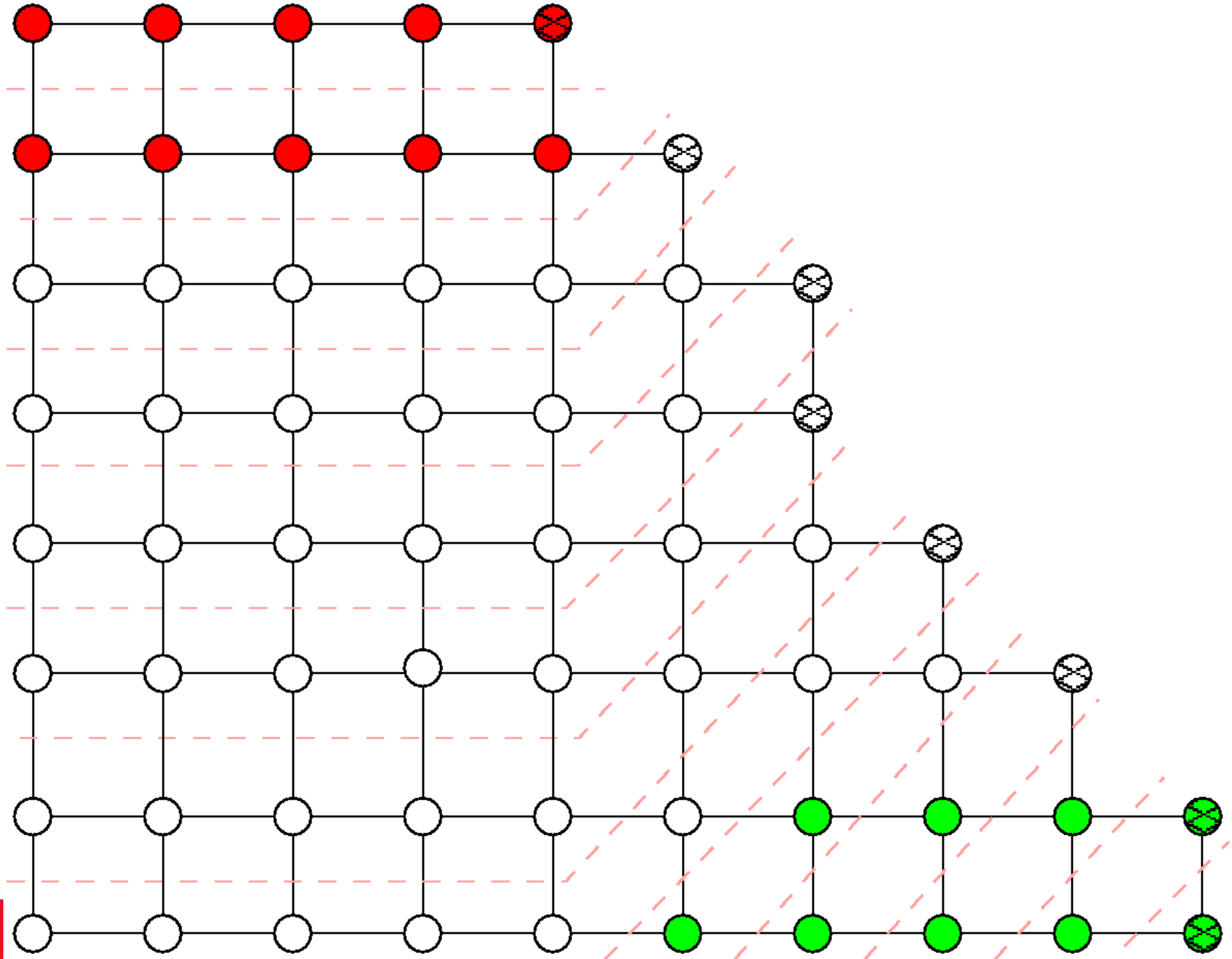


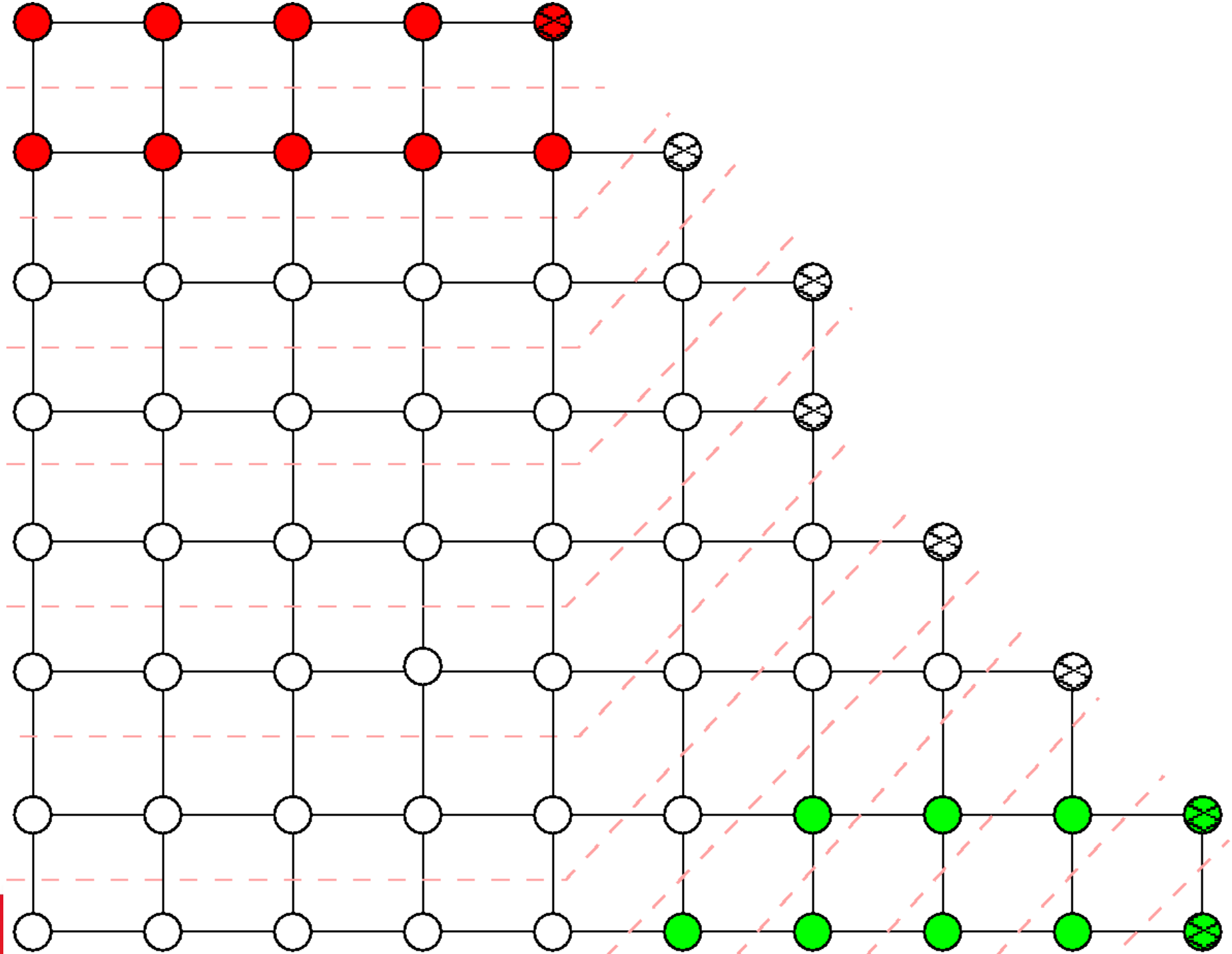


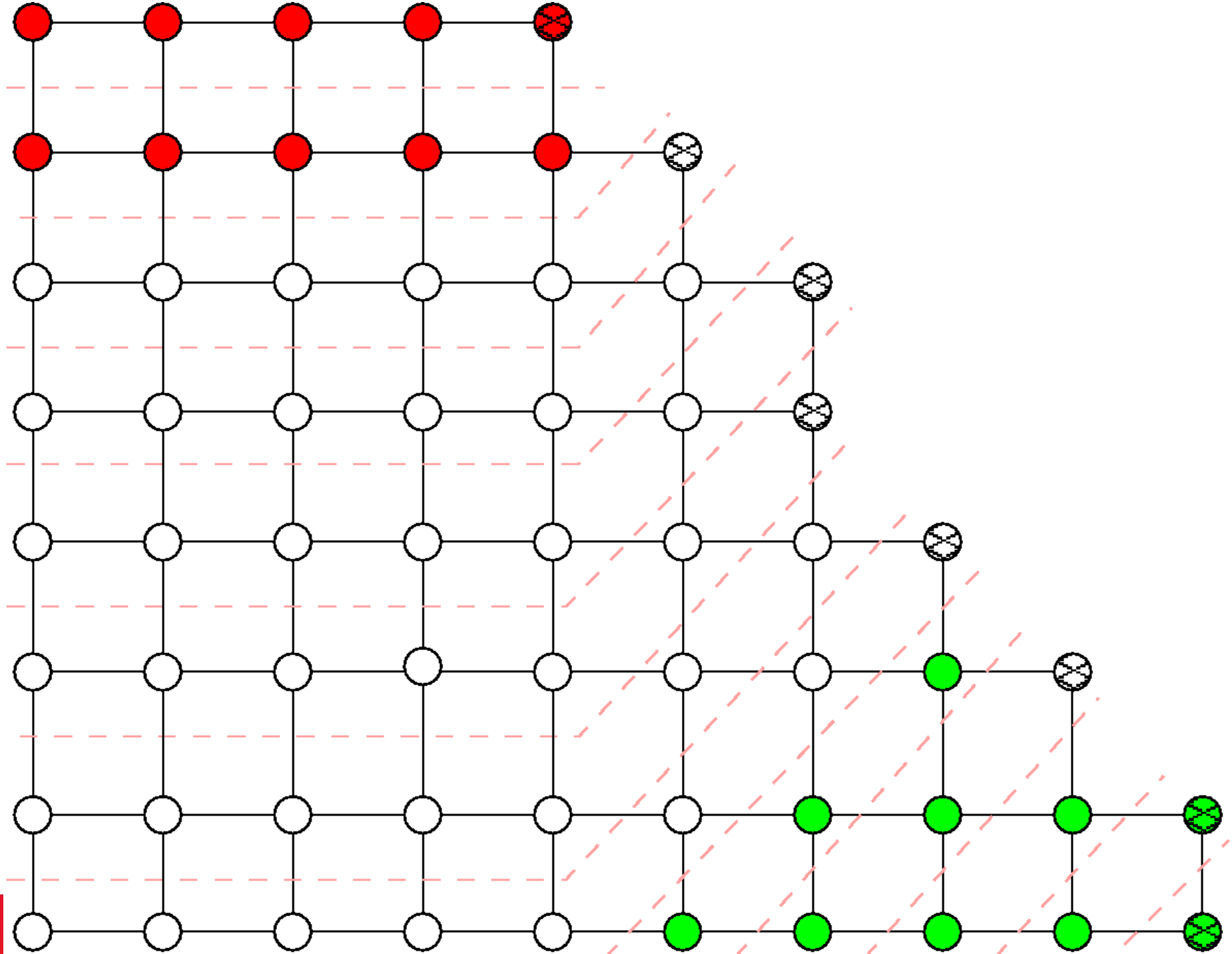


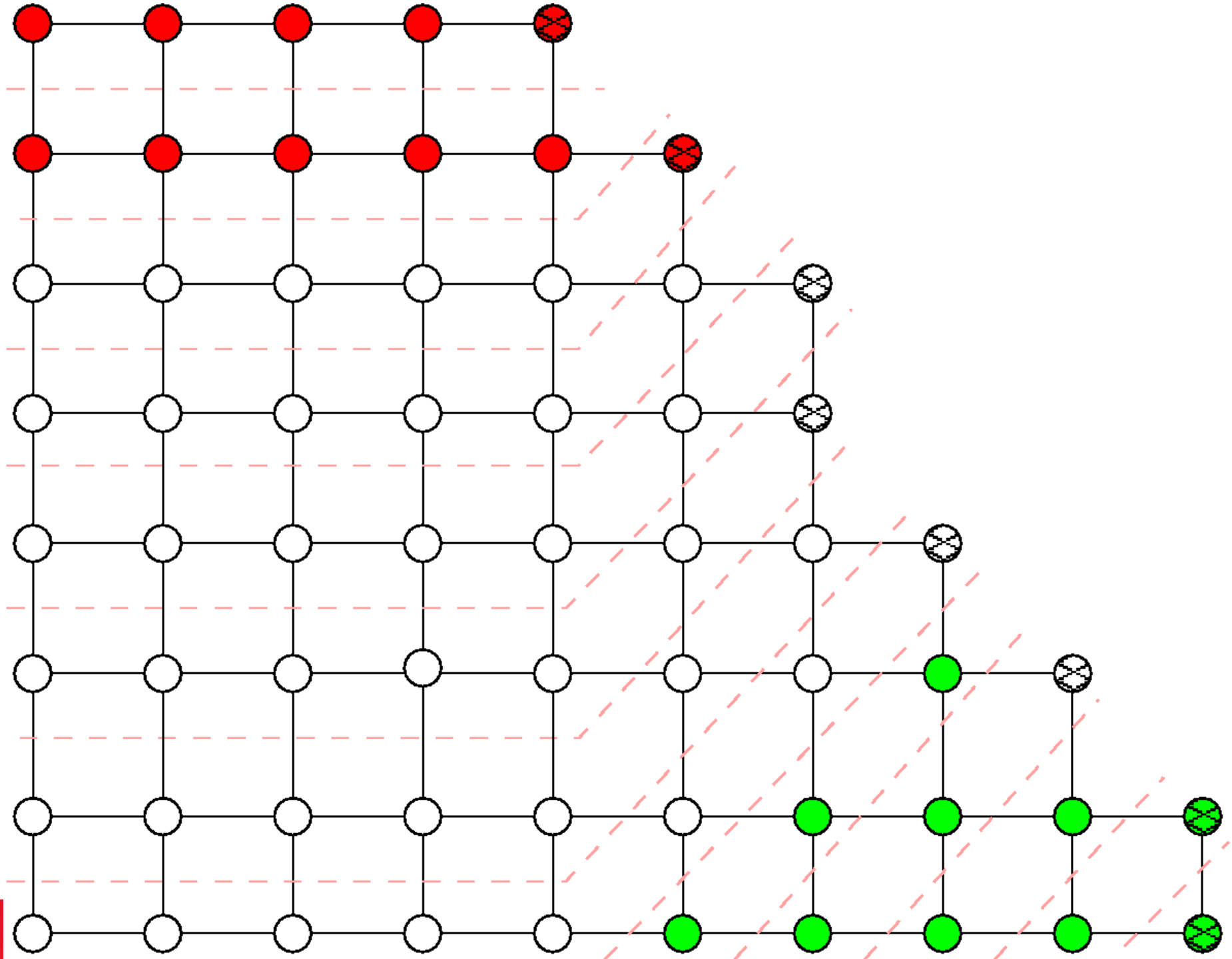


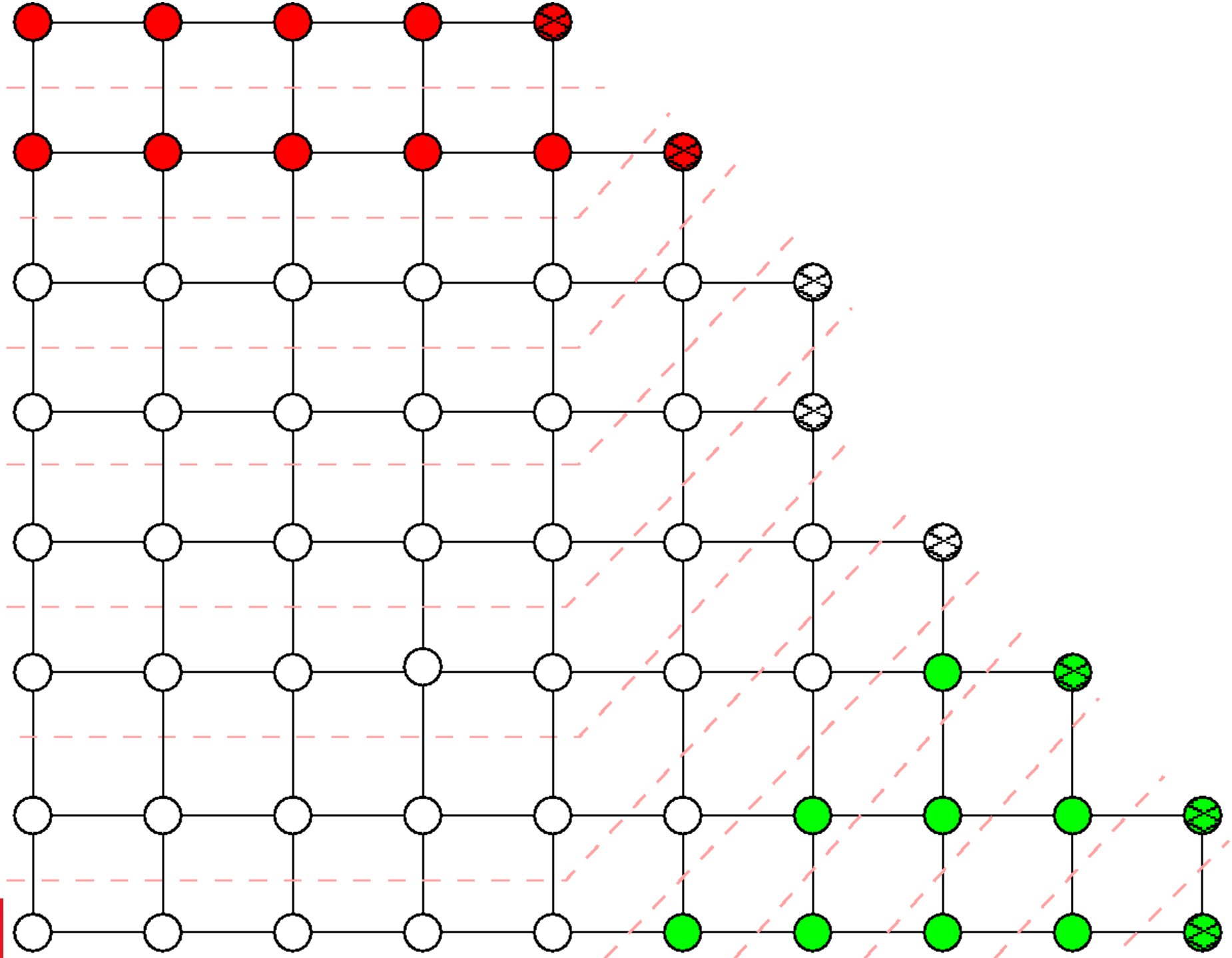


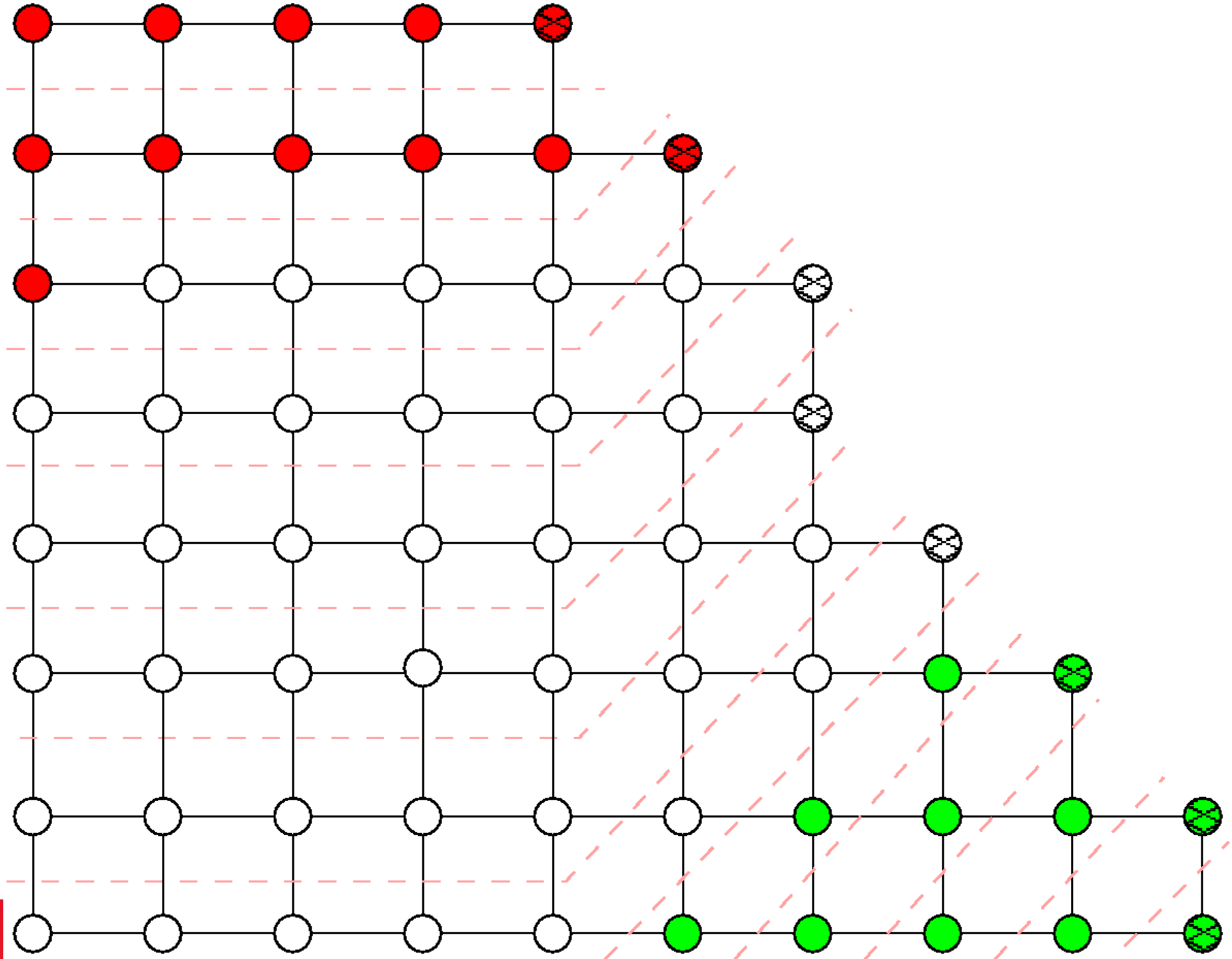


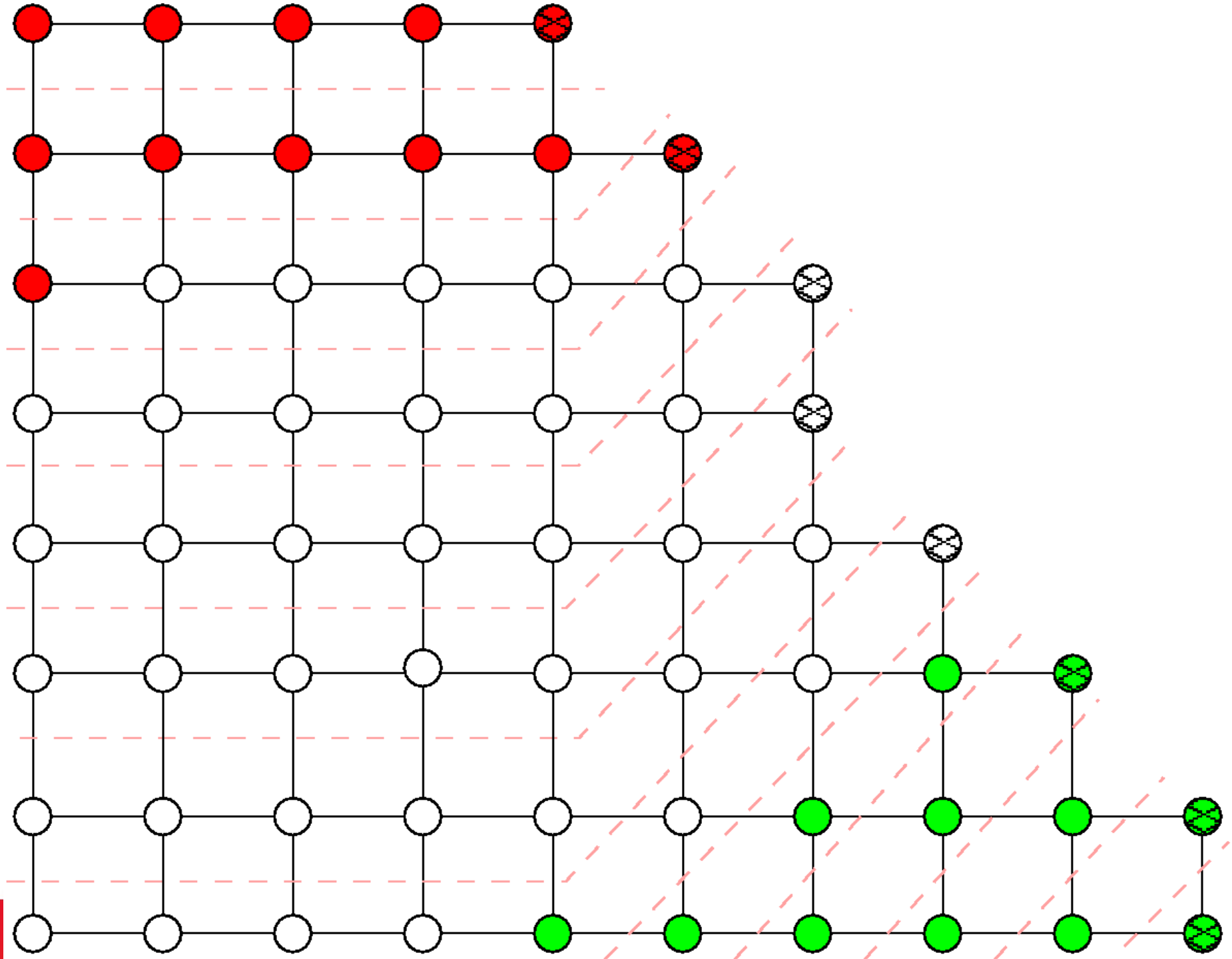


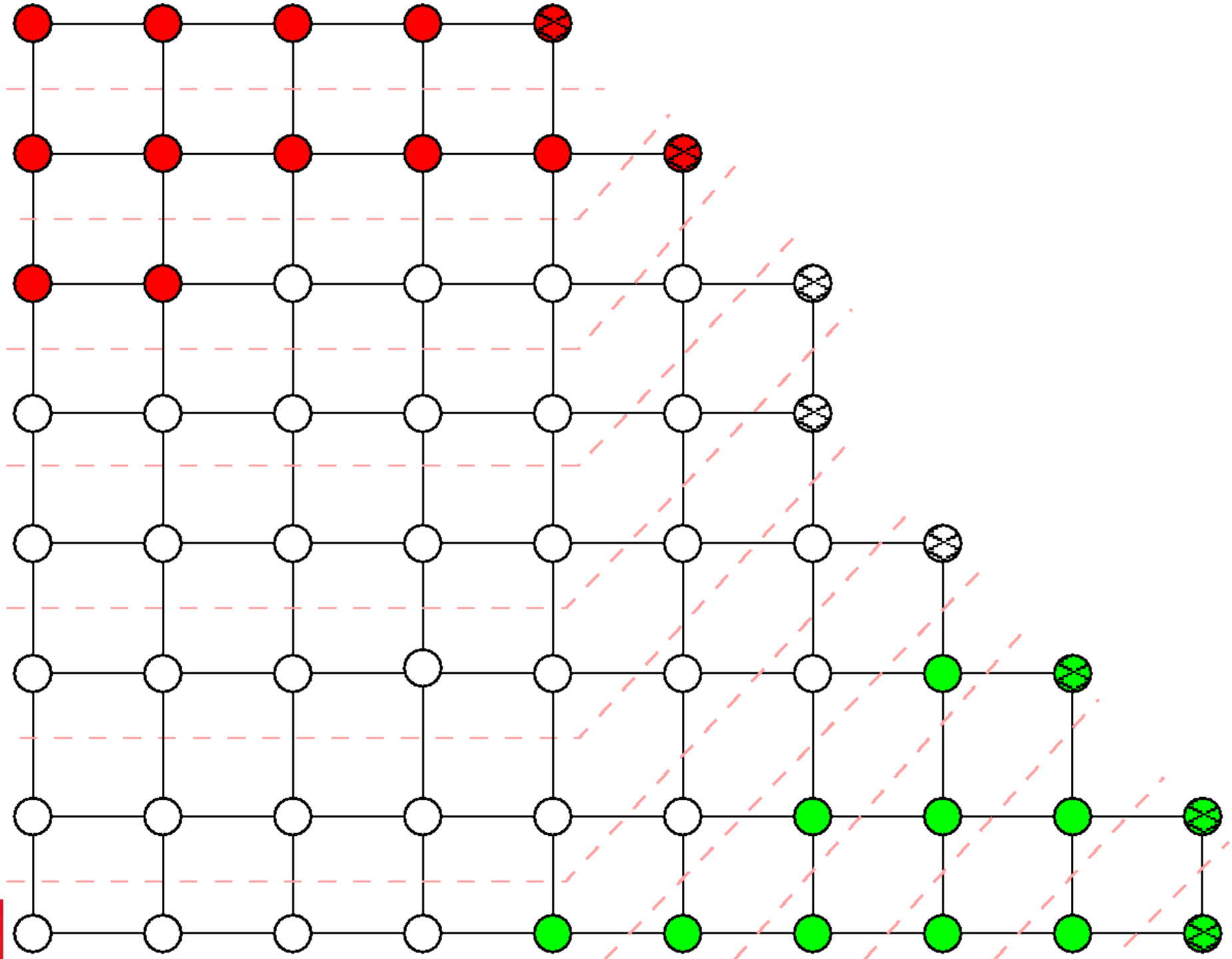


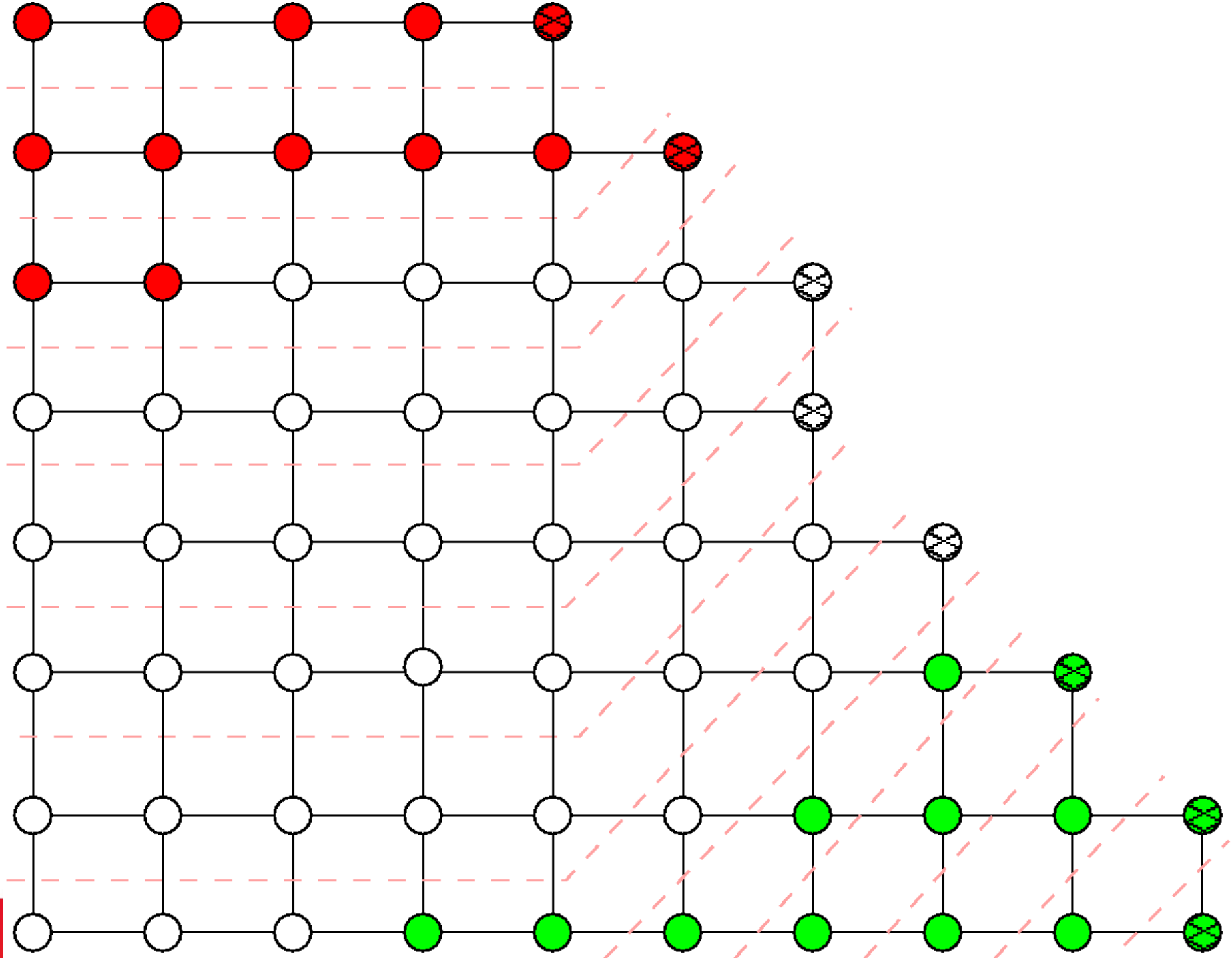


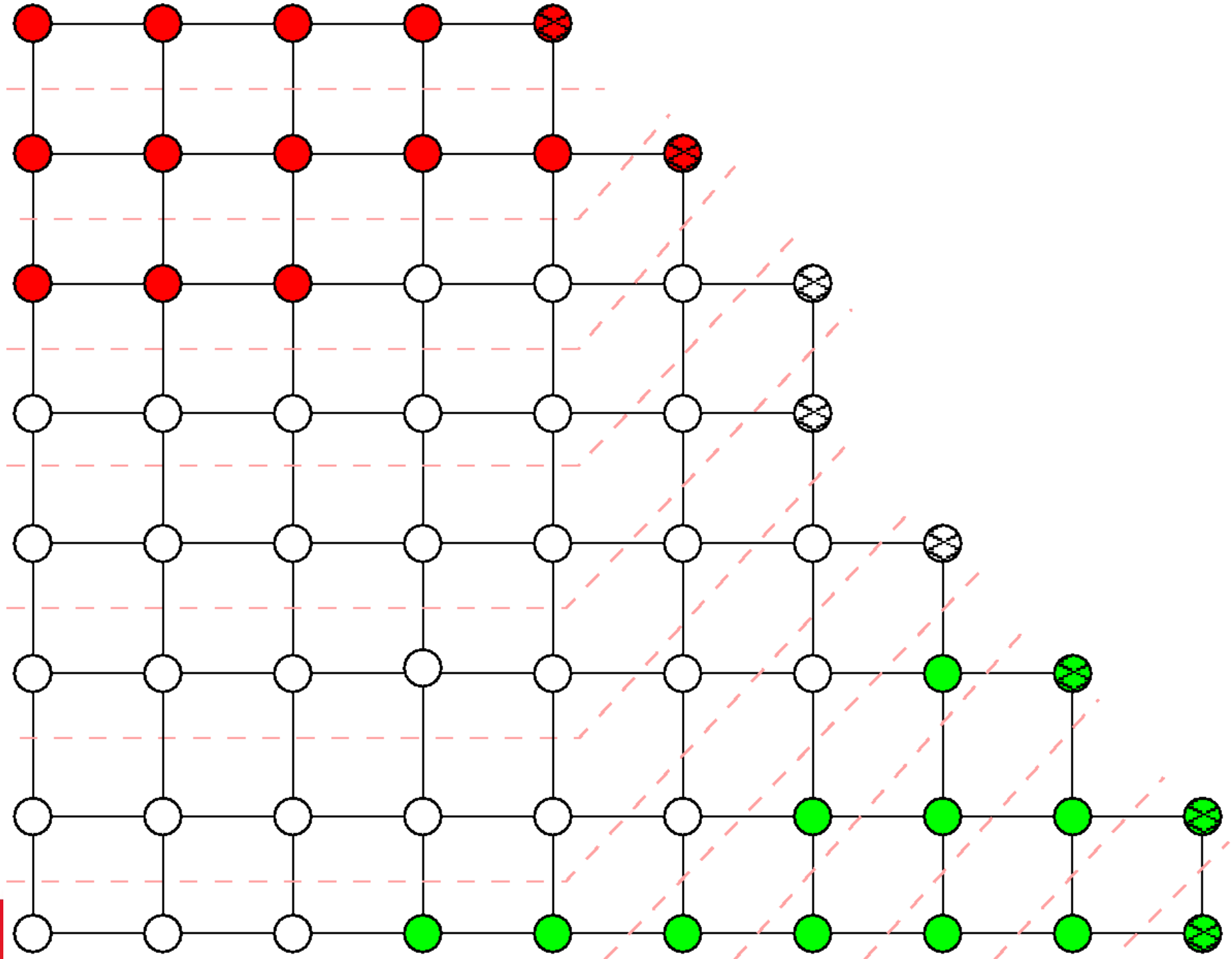


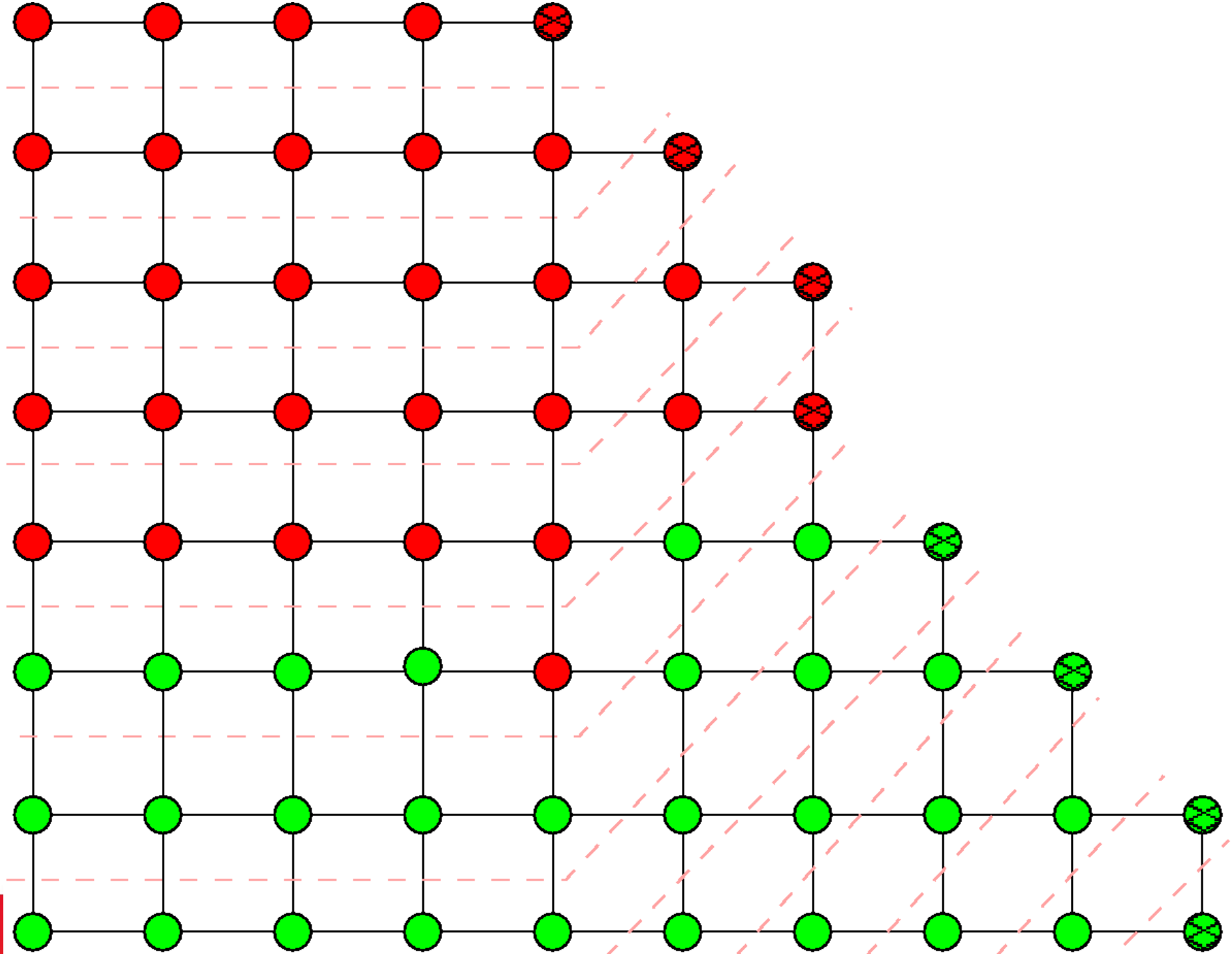


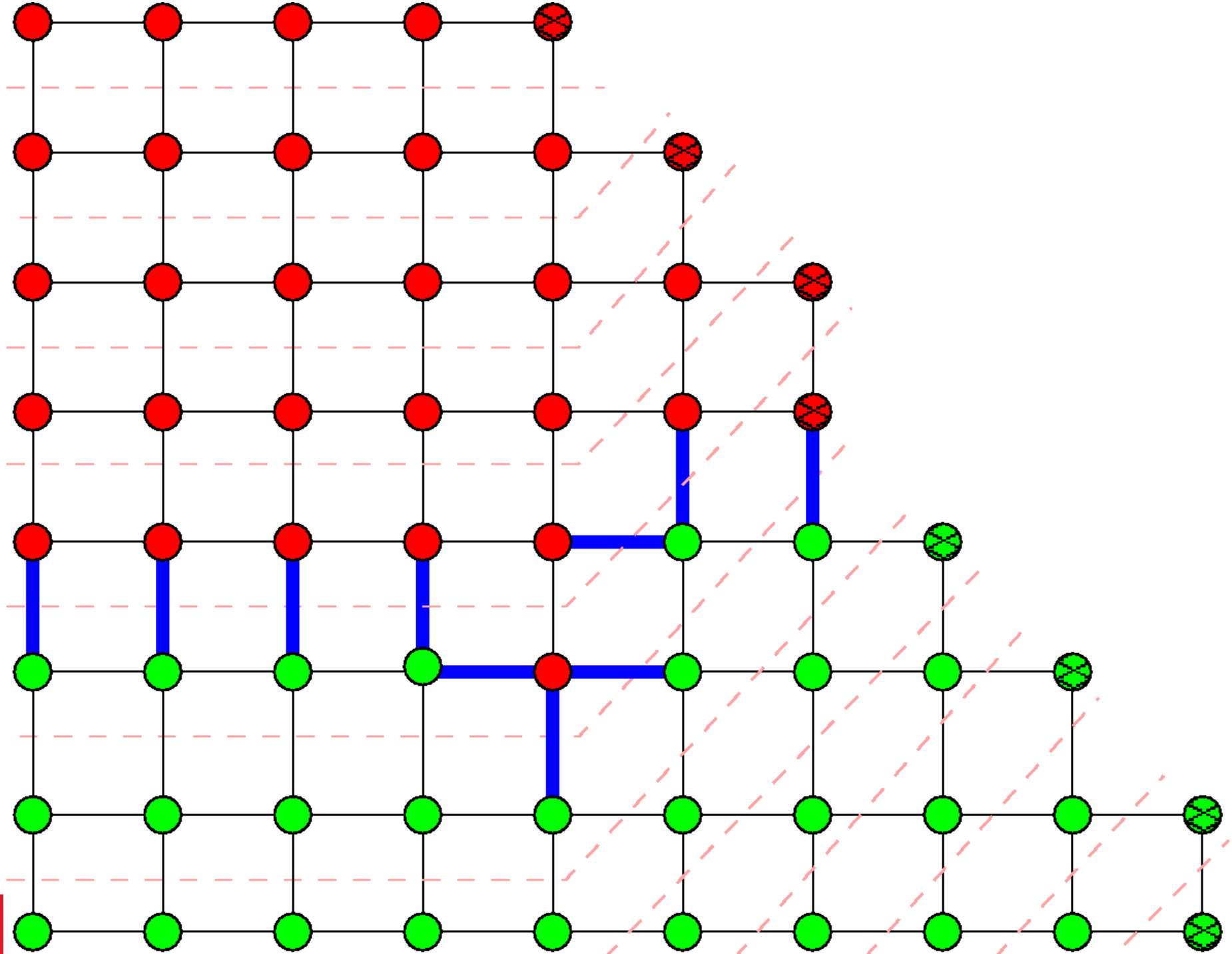


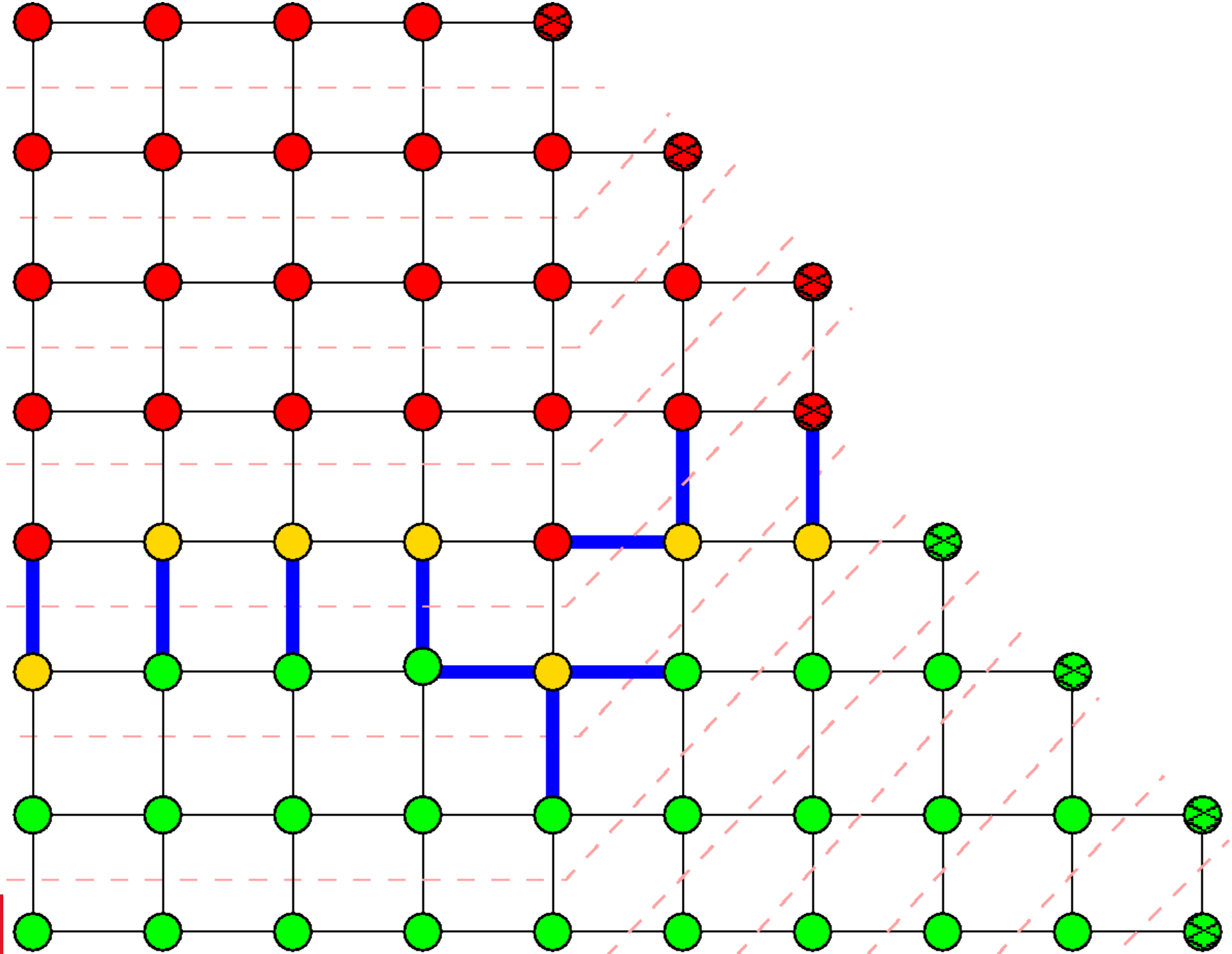


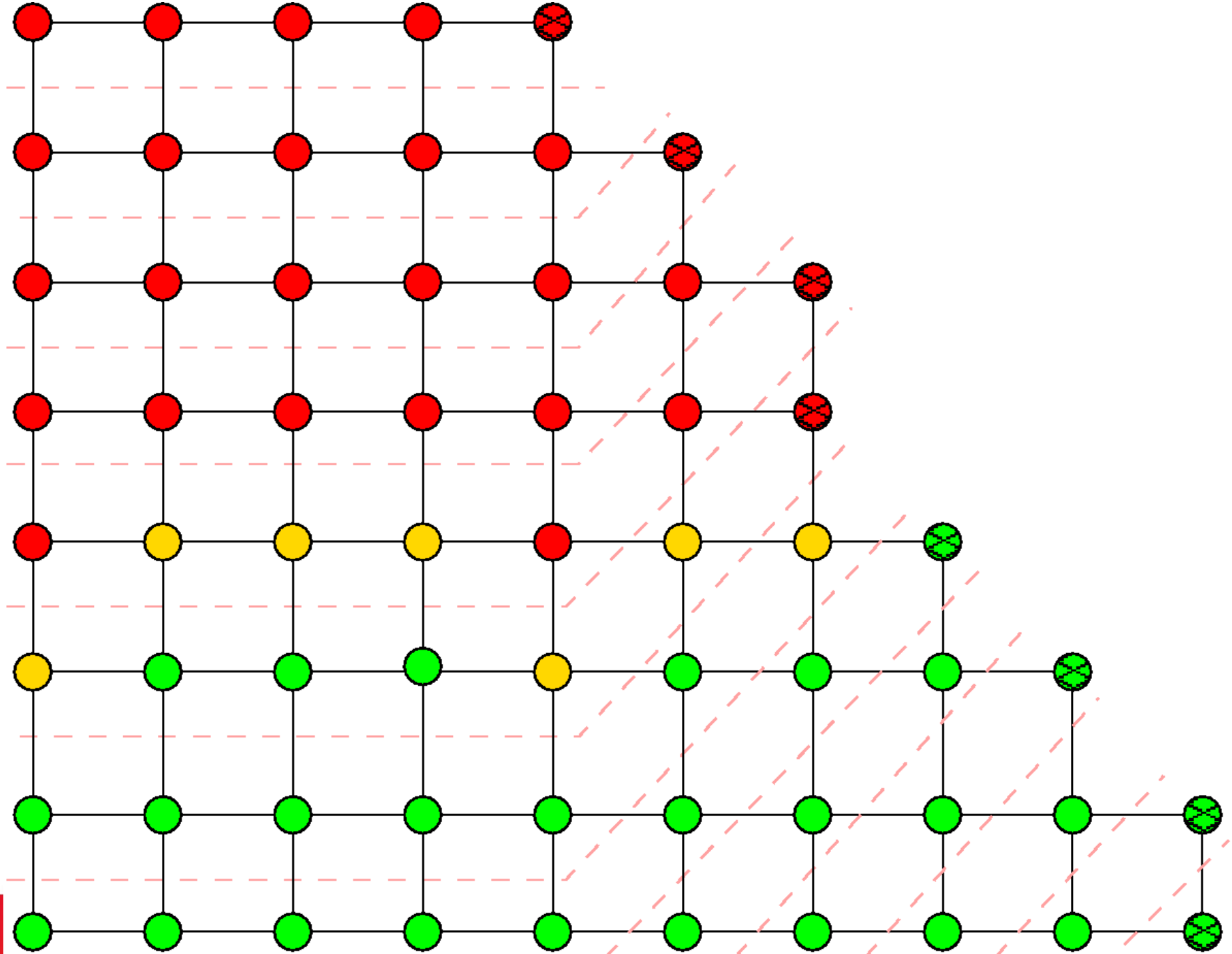




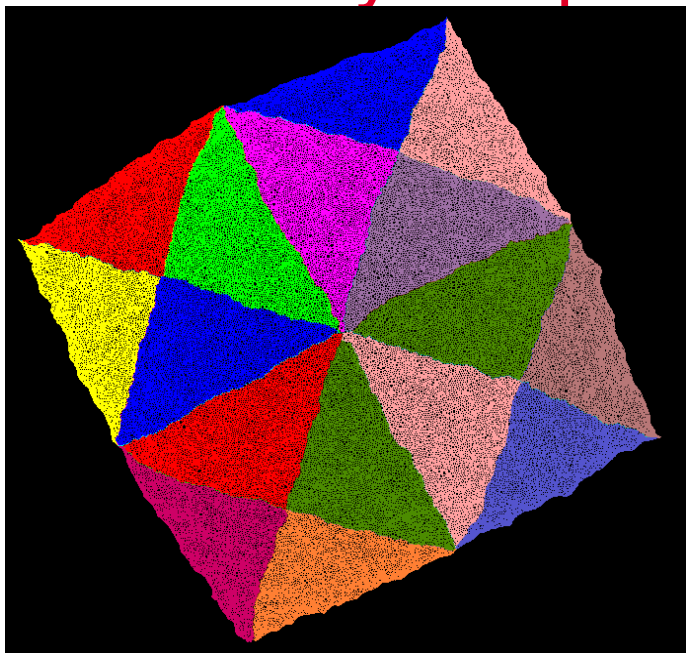


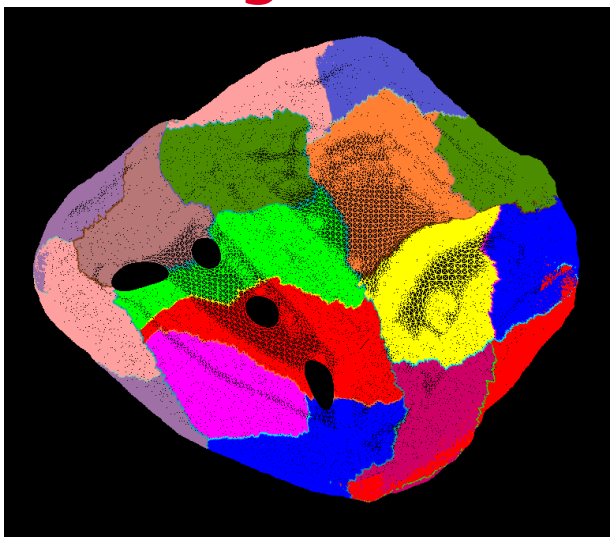


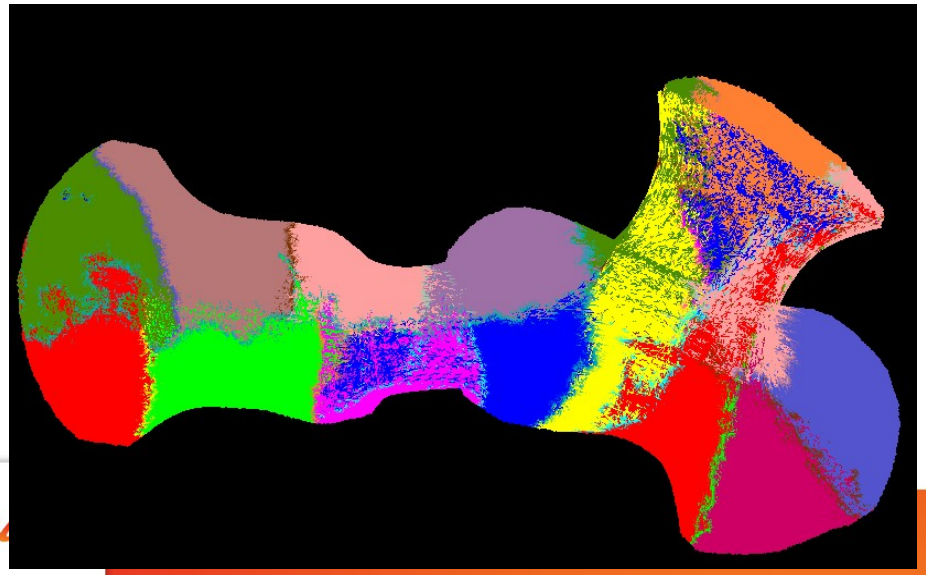


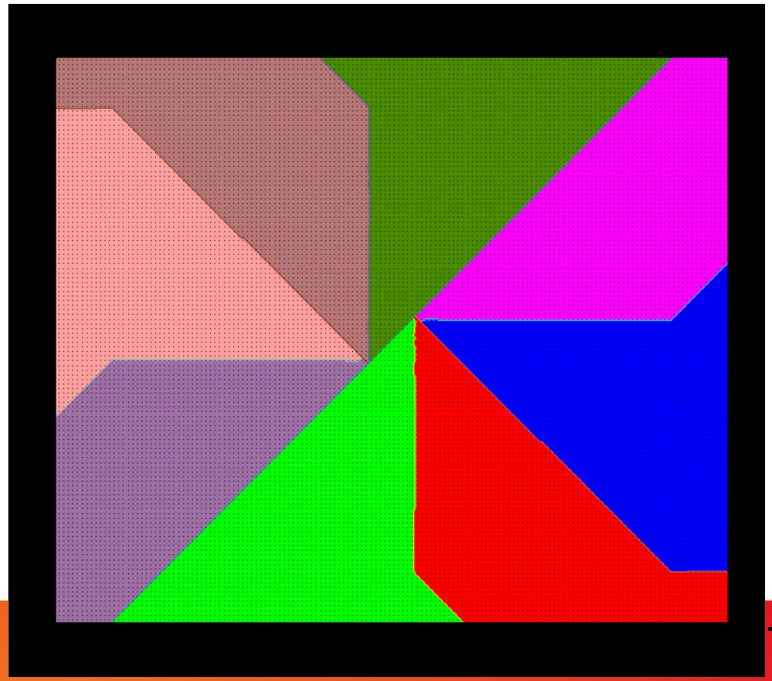


6 Double Greedy Graph Growing

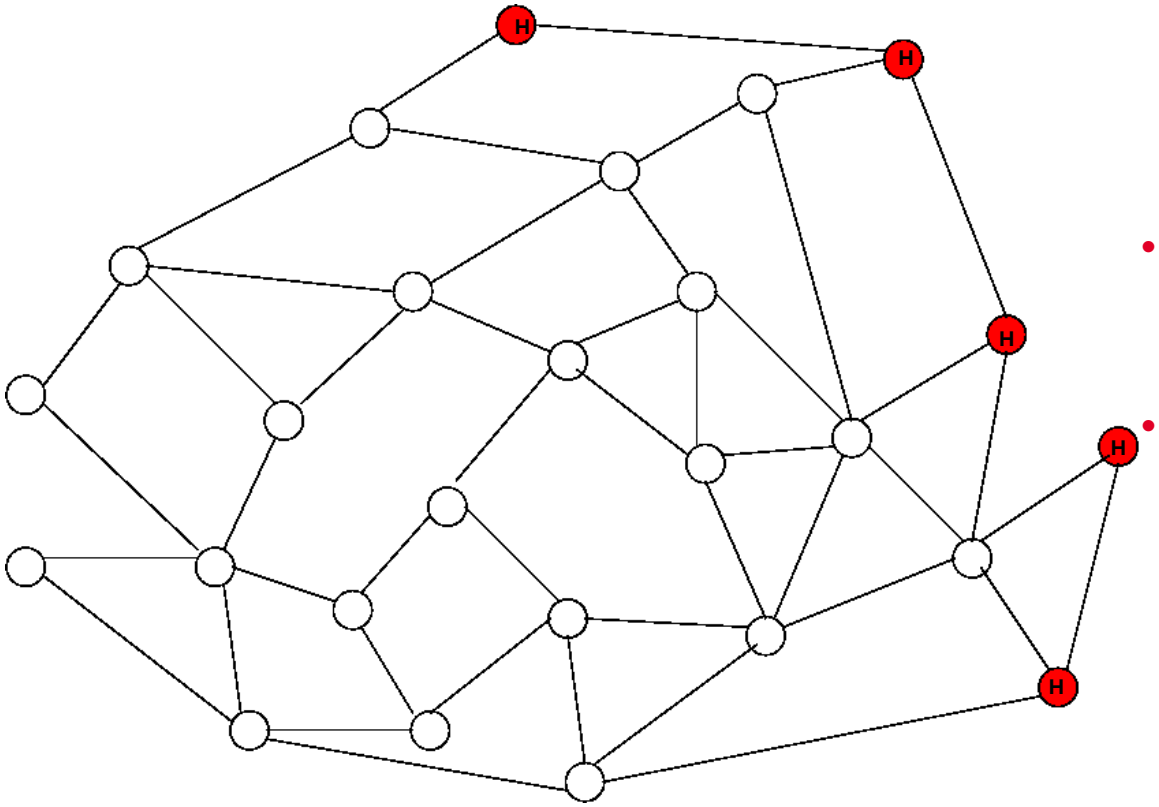








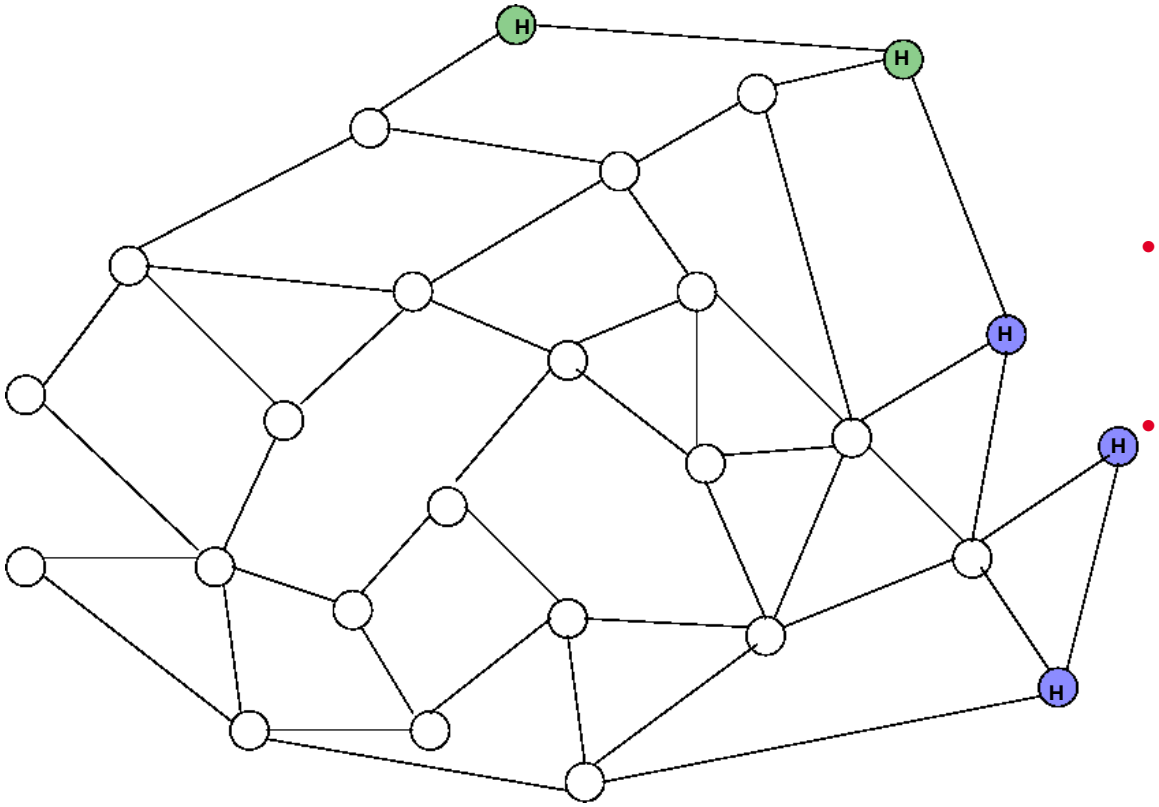
7 Halo-first Greedy Graph Growing



Idea:

- Split halo in H_0 and H_1 (using GG)
- Make grow P_0 and P_1 using H_0 and H_1 as set of seeds

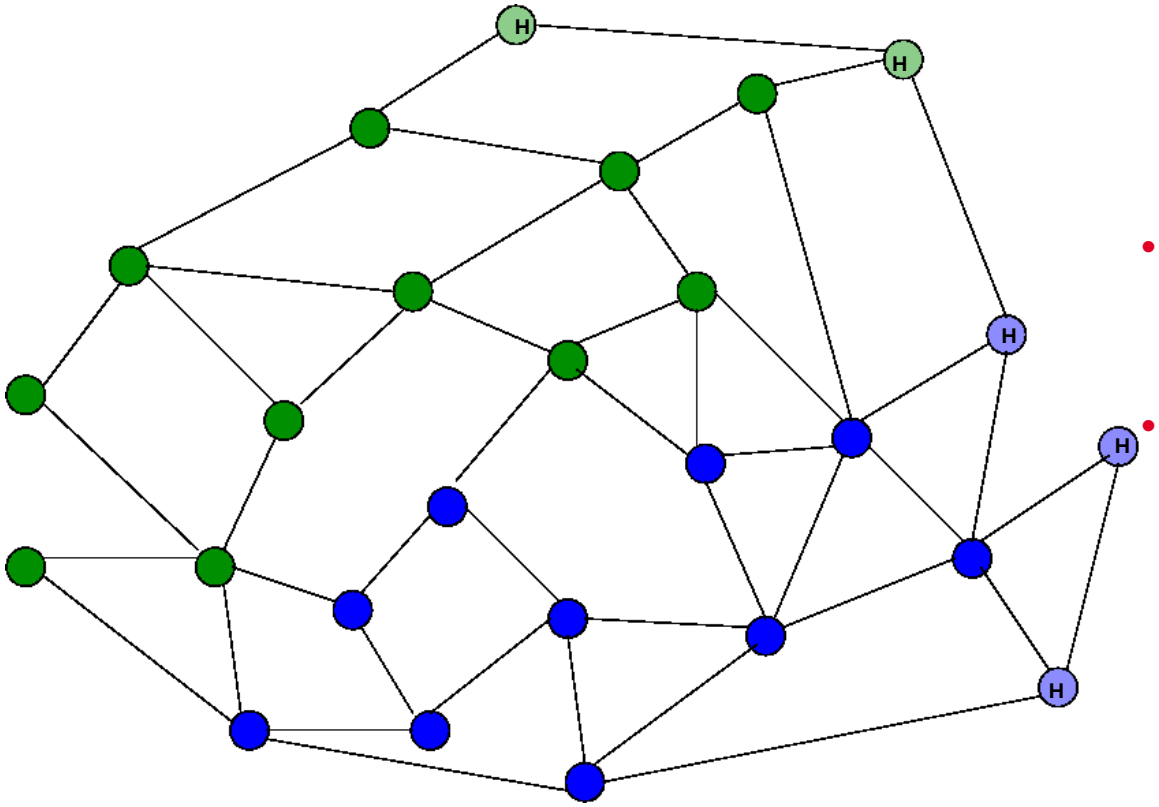
7 Halo-first Greedy Graph Growing



Idea:

- Split halo in H_0 and H_1 (using GG)
- Make grow P_0 and P_1 using H_0 and H_1 as set of seeds

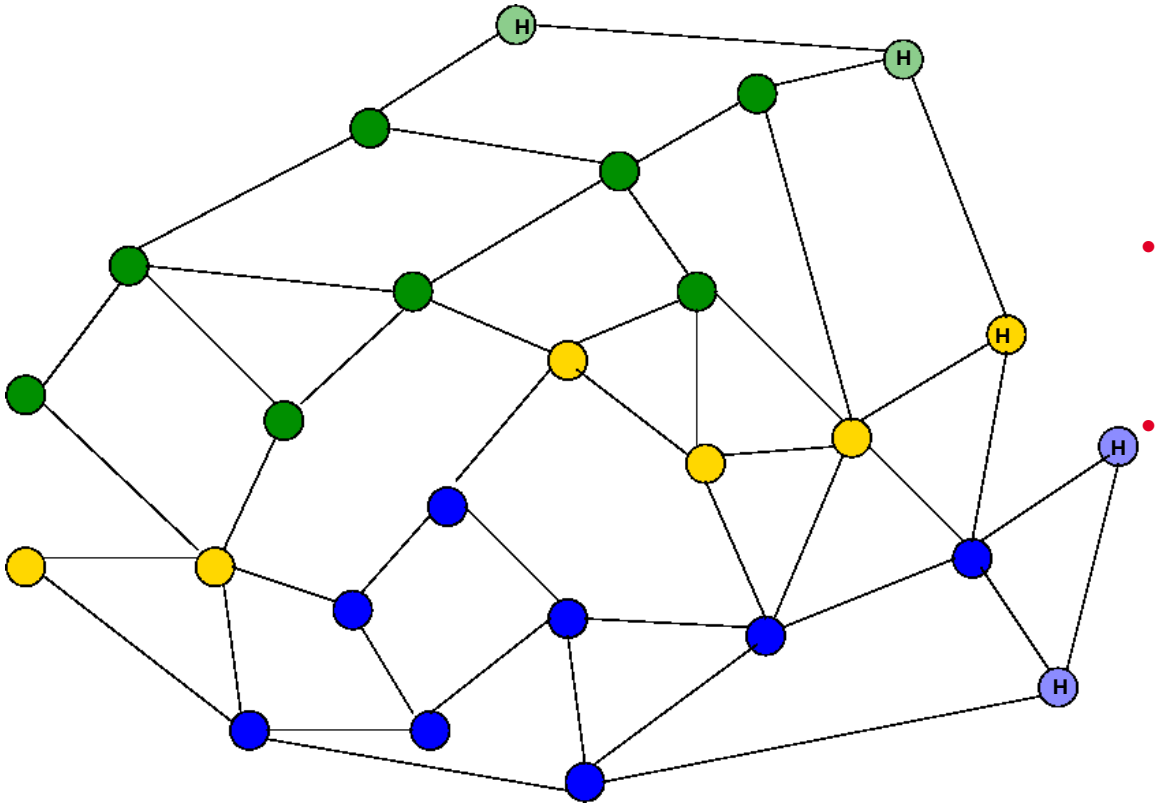
7 Halo-first Greedy Graph Growing



Idea:

- Split halo in H_0 and H_1 (using GG)
- Make grow P_0 and P_1 using H_0 and H_1 as set of seeds

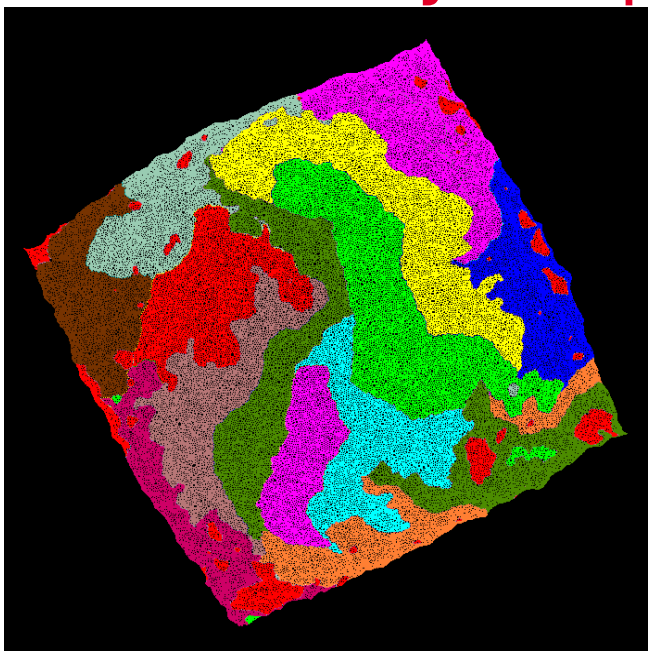
7 Halo-first Greedy Graph Growing

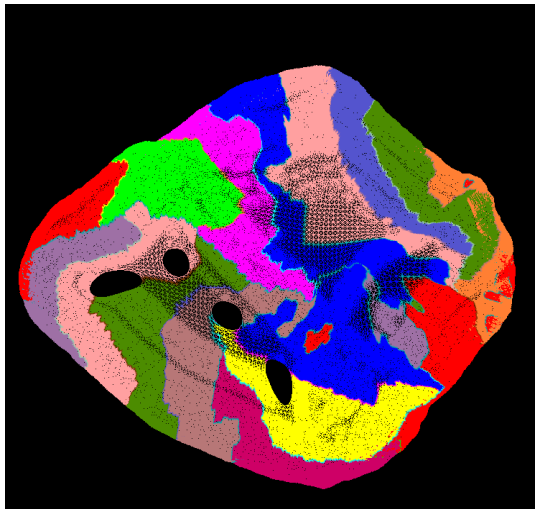


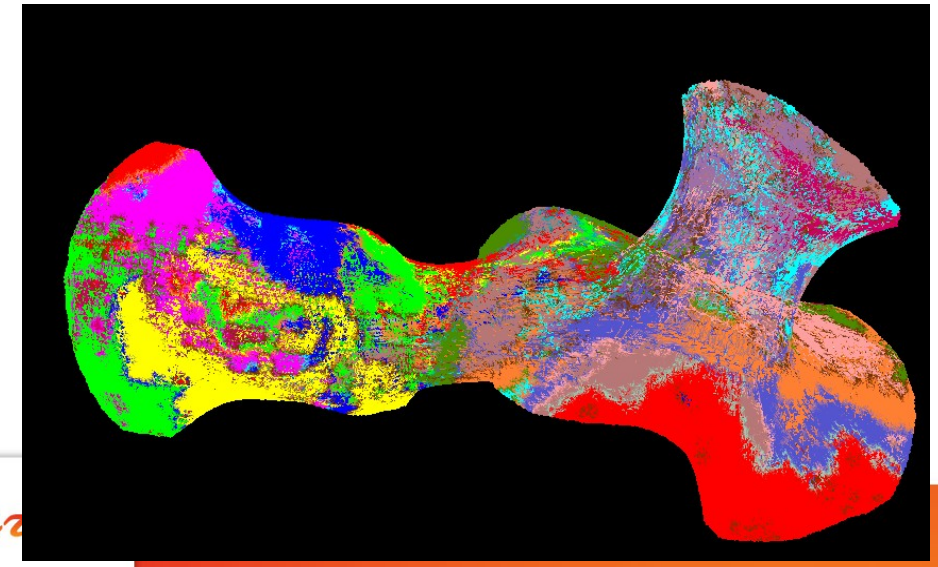
Idea:

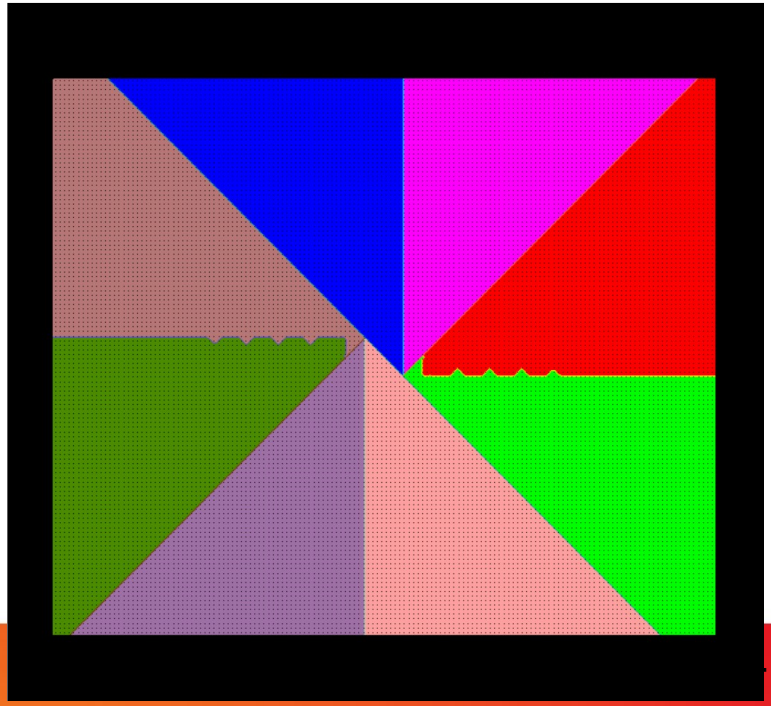
- Split halo in H_0 and H_1 (using GG)
- Make grow P_0 and P_1 using H_0 and H_1 as set of seeds

7 Halo-first Greedy Graph Growing

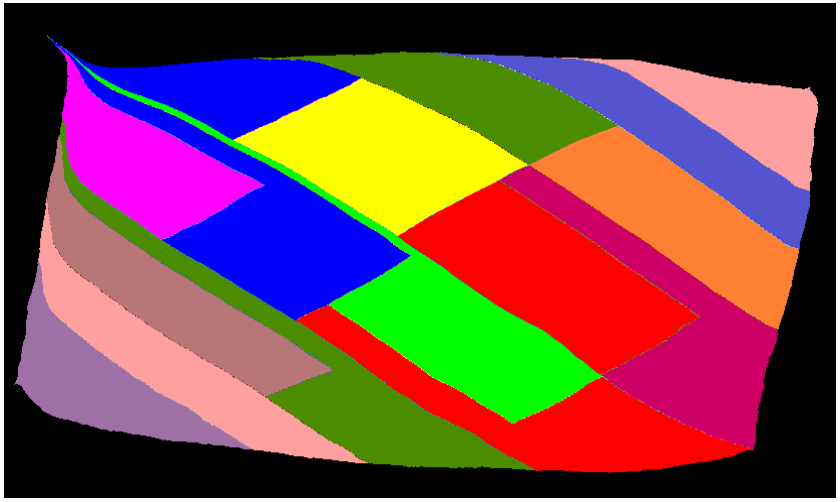




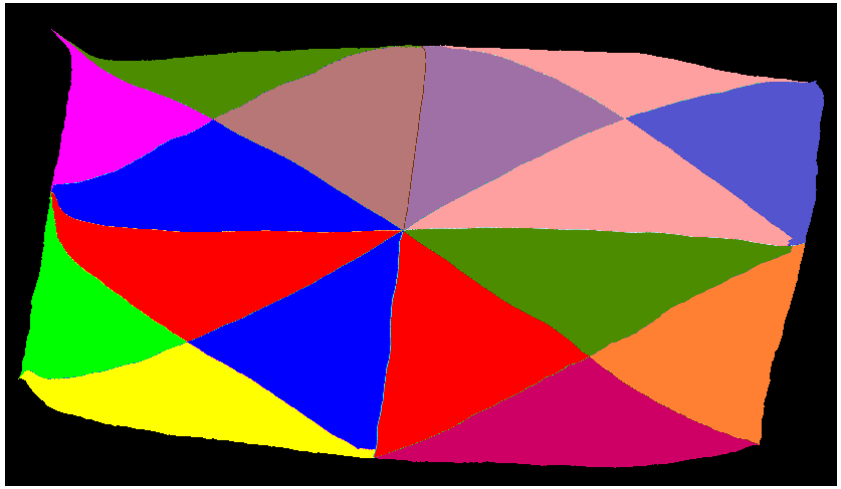




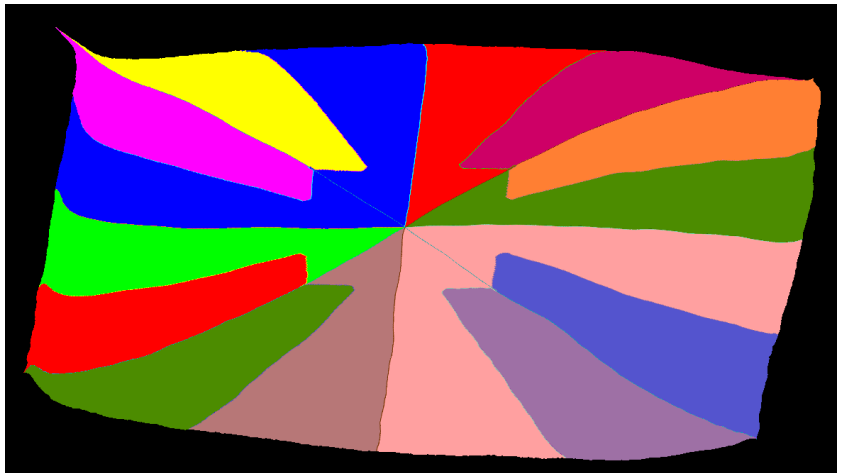
Matrix Ecology1, 1,000,000 unknowns
(without multilevel)



GG: Δ 1403, max 1758



DG: Δ 499, max 999



HF: Δ 254, max 1073

8 Remarks

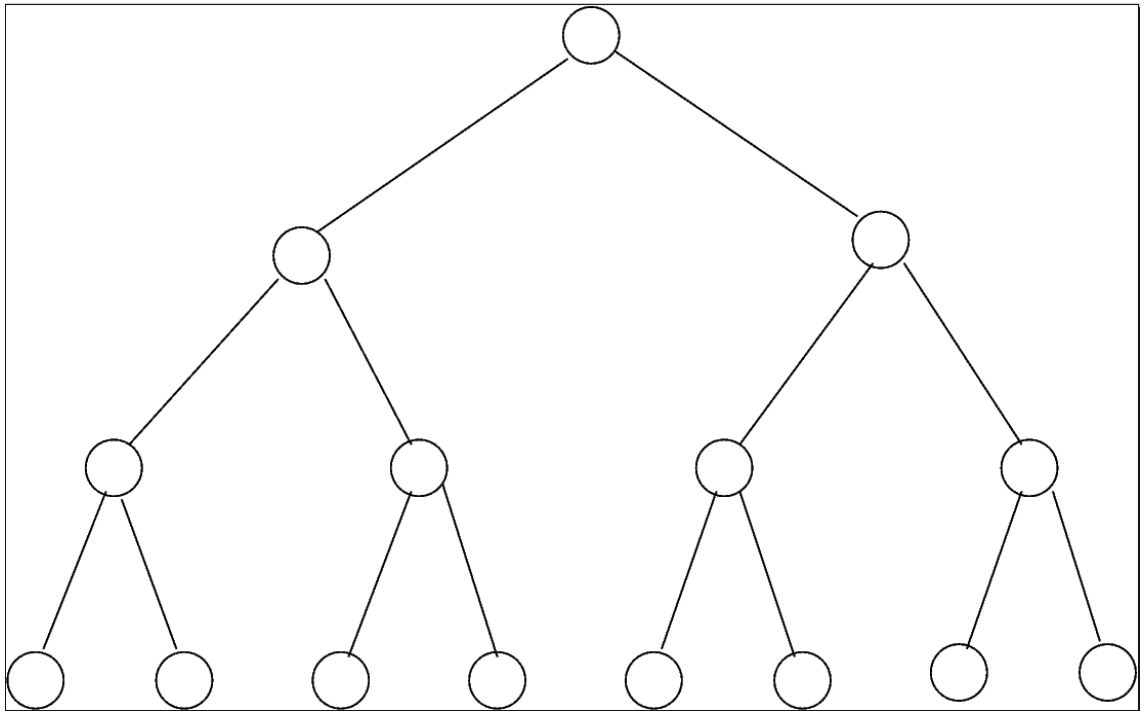
Imbalance

Level 0 0

Level 1 + δ %

Level 2 + δ %

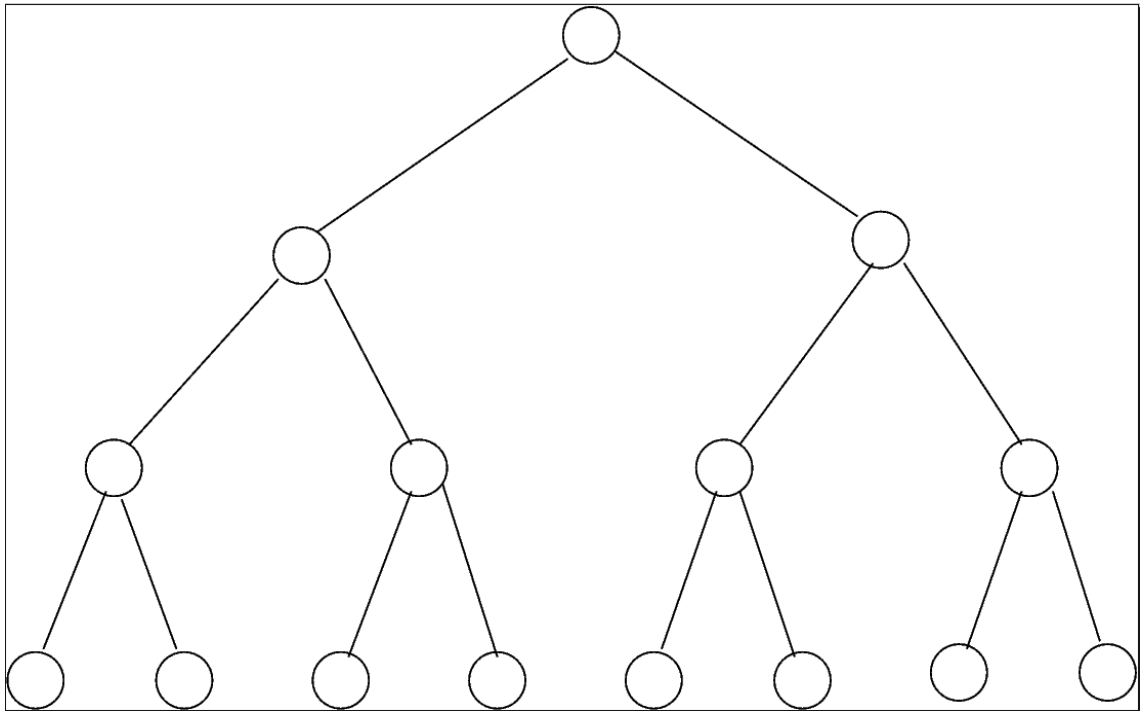
Level 3 + δ %



Total: δ x nb_levels %

8 Remarks

	Imbalance
Level 0	0
Level 1	+ δ / 4%
Level 2	+ δ / 2 %
Level 3	+ δ %
Total:	< 2δ %



In results,
« GG » denotes original Scotch
« GG* » denotes Scotch with this improvement

8 Results

- Interior imbalance: $\Delta = \max_{0 \leq i < ndom} |Dom_i| - \min_{0 \leq i < ndom} |Dom_i|$
- Interface imbalance: $\Delta_h = \max_{0 \leq i < ndom} |interface(Dom_i)| - \min_{0 \leq i < ndom} |interface(Dom_i)|$

Results on 16 domains with multilevel + Fiduccia-Mattheyses + (DG or HF):

	Δ_h DG	Δ_h HF
• Almond (n ≈ 7 x 10^6):	2544	2263
• Nice-7 (n ≈ 8 x 10^6):	5618	4967
• 10millions (n ≈ 10 x 10^6):	4771	4381

8 Results
16 domains, DG

	interface imbalance			interior imbalance		
id	*GG*	% *GG**	% *DG*	*GG*	% *GG**	% *DG*
1	297	7,4	**-19,2**	1311	**-69,1**	-23,3
2	1112	1,6	**-40,0**	4678	**-66,5**	-78,9
3	522	-11,3	**-39,7**	1635	**-13,9**	-13,0
4	244	-9,0	**-11,5**	1568	-23,7	**-31,1**
5	475	**-17,9**	-3,8	4605	**-85,6**	-62,3
6	869	-21,3	**-41,3**	4107	**-78,5**	-49,6
7	**905**	49,4	26,4	4942	**-69,1**	-63,7
8	261	0,0	**-46,7**	420	**0,0**	79,8
9	365	-3,8	**-44,9**	8128	**-82,8**	-65,2
10	1002	14,0	**-32,1**	4852	**-73,2**	-44,6
11	509	-3,7	**-44,4**	2831	**-84,1**	27,5
12	569	-25,0	**-59,1**	22416	**-79,5**	-67,5
13	532	-11,8	**-58,5**	12049	**-69,0**	-49,5
14	756	-23,1	**-46,3**	17458	-61,9	**-66,6**
15	6678	6,6	**-29,8**	35466	**-73,9**	-68,4
16	564	-11,9	**-62,9**	15092	**-65,5**	-58,0
17	2130	18,6	**-50,4**	32202	**-72,9**	-67,7
18	335	17,0	**-19,7**	28057	**-70,1**	-64,9
19	2604	-16,4	**-42,0**	25853	**-66,5**	-59,5
20	9990	-14,2	**-25,8**	73635	**-80,7**	-16,5
21	927	0,6	**-40,9**	2377	-58,6	**-59,5**
22	3468	5,0	**-78,5**	**3336**	4,3	18,2
23	1869	7,1	**-3,2**	14424	**-73,3**	-64,4
24	2460	-9,1	**-73,4**	8940	-44,1	**-60,6**
25	6837	-11,6	**-69,7**	26877	-63,0	**-68,9**
26	780	-15,9	**-53,8**	24987	-58,3	**-65,2**
27	6168	-27,5	**-68,4**	67721	-68,7	**-76,2**
28	4344	-15,6	**-41,4**	240729	-76,9	**-77,1**
29	11539	8,6	**-51,3**	244959	-73,9	**-77,2**
30	9936	-6,9	**-52,0**	286992	**-72,6**	-8,5

8 Results
16 domains, HF

	interface imbalance			interior imbalance		
id	GG	% $GG^\star$	% HF	GG	% $GG^\star$	% HF
1	297	7,4	**-29,3**	1311	-69,1	**-73,5**
2	1112	1,6	**-34,9**	4678	-66,5	**-74,2**
3	522	-11,3	**-63,4**	1635	**-13,9**	-7,2
4	244	**-9,0**	103,3	1568	-23,7	**-24,5**
5	475	-17,9	**-26,7**	4605	**-85,6**	-74,3
6	869	-21,3	**-48,2**	4107	**-78,5**	-65,6
7	905	49,4	**-24,2**	4942	-69,1	**-72,2**
8	261	0,0	**-69,0**	420	0,0	**-11,4**
9	365	-3,8	**-41,4**	8128	**-82,8**	-69,2
10	1002	14,0	**-66,9**	4852	**-73,2**	-47,5
11	509	-3,7	**-50,3**	2831	**-84,1**	-19,1
12	569	-25,0	**-70,1**	22416	**-79,5**	-72,9
13	532	-11,8	**-37,0**	12049	**-69,0**	-52,3
14	756	**-23,1**	-20,4	17458	-61,9	**-67,2**
15	6678	6,6	**-22,4**	35466	**-73,9**	-68,2
16	564	-11,9	**-54,4**	15092	**-65,5**	-59,2
17	2130	18,6	**-44,8**	32202	-72,9	**-73,0**
18	**335**	17,0	14,6	28057	**-70,1**	-68,3
19	2604	**-16,4**	-15,8	25853	**-66,5**	-49,7
20	9990	-14,2	**-55,4**	73635	**-80,7**	-79,6
21	927	0,6	**-49,4**	2377	-58,6	**-64,5**
22	3468	5,0	**-75,4**	**3336**	4,3	16,1
23	1869	7,1	**-24,2**	14424	**-73,3**	-72,0
24	2460	-9,1	**-55,9**	8940	-44,1	**-51,5**
25	6837	-11,6	**-65,5**	26877	-63,0	**-80,3**
26	780	-15,9	**-33,6**	24987	-58,3	**-66,0**
27	6168	-27,5	**-73,6**	67721	-68,7	**-74,5**
28	4344	-15,6	**-47,9**	240729	**-76,9**	-73,2
29	11539	8,6	**-57,0**	244959	**-73,9**	-70,9
30	9936	-6,9	**-55,9**	286992	**-72,6**	-71,4

8 Results on 16 domains

Multilevel + FM + DG

Average gain over GG:

- Δ_h: - 40%
- Δ: - 45%

DG beats both GG and GG* on:

- Δ_h: 28 / 30 cases
- Δ: 9 / 30 cases

Multilevel + FM + HF

Average gain over GG:

- Δ_h: - 38%
- Δ: - 56%

HF beats both GG and GG* on:

- Δ_h: 26 / 30 cases
- Δ: 12 / 30 cases

HF beats DG on:

- 16 / 30 cases
- 21 / 30 cases

→ HF slightly better on most cases

8 Results multilevel + DG/HF, 16 to 512 domains

Almond
n ≈ 7 x 10^6
nnz ≈ 100 x 10^6

	interface imbalance Δ_h				interior imbalance Δ			
dom	*GG*	% *GG**	% *DG*	% *HF*	*GG*	% *GG**	% *DG*	% *HF*
16	4344	-15,6	-41,4	**-47,9**	240729	-76,9	**-77,1**	-73,2
32	3179	**-34,8**	-31,5	-0,5	133886	-73,8	**-80,1**	-76,9
64	2258	-5,8	**-47,6**	-17,8	83819	**-80,5**	-79,2	-79,1
128	1822	-29,5	**-42,9**	-32,6	48087	-78,4	-77,6	**-80,9**
256	1071	-2,2	**-44,4**	2,5	27695	-81,6	-77,9	**-83,0**
512	910	0,8	-17,1	**-22,3**	16243	-83,8	-80,0	**-84,7**

Nice-7
n ≈ 8 x 10^6
nnz ≈ 660 x 10^6

	interface imbalance Δ_h				interior imbalance Δ			
dom	*GG*	% *GG**	% *DG*	% *HF*	*GG*	% *GG**	% *DG*	% *HF*
16	11539	8,6	-51,3	**-57,0**	244959	-73,9	**-77,2**	-70,9
32	10991	-26,3	**-45,7**	-38,5	188834	-80,3	-81,0	**-81,6**
64	8997	-27,5	40,7	**-30,8**	101838	-75,9	-6,6	**-80,2**
128	5694	-14,4	11,7	**-26,9**	66792	-83,9	-1,9	**-84,3**
256	4554	-3,2	**-33,1**	-27,6	34314	-77,4	7,3	**-82,4**
512	3762	-16,7	-30,8	**-33,1**	19734	-79,1	-6,3	**-84,2**

10millions
n ≈ 10 x 10^6
nnz ≈ 160 x 10^6

	interface imbalance Δ_h				interior imbalance Δ			
dom	*GG*	% *GG**	% *DG*	% *HF*	*GG*	% *GG**	% *DG*	% *HF*
16	9936	-6,9	-52,0	**-55,9**	286992	**-72,6**	-8,5	-71,4
32	6666	-0,5	43,7	**-56,6**	188900	-69,5	13,0	**-77,7**
64	7036	-13,3	**-47,4**	-31,1	125444	-76,9	-18,8	**-78,4**
128	4564	-11,2	4,0	**-48,7**	79754	**-82,9**	-52,9	-78,8
256	3114	-6,2	163,5	**-32,5**	42931	-79,5	-30,2	**-80,8**
512	2336	-19,5	22,2	**-54,2**	25800	**-83,5**	49,3	-83,4

Impact on performances for MaPHYS

Matrix Almond, divided in 8 domains

	Natif	DG	HF
Max-min Schur size	6546	**3221**	1779
Avg Schur size (n)	**10466**	10532	11254
Avg Schur size (nnz)	114M	**112M**	127M
Total interface size	**41604**	41869	44768
Avg time facto (s)	632	**596**	593
Avg time precond (s)	163	**117**	129
Avg time solve (s)	147	**85**	118

Impact on performances for MaPHYS

Matrix Almond, divided in 16 domains

	Natif	DG	HF
Max-min Schur size	5805	3455	**3307**
Avg Schur size (n)	**7523**	7709	7900
Avg Schur size (nnz)	**58M**	60M	63M
Total interface size	**59574**	61057	62608
Avg time facto (s)	232	**222**	223
Avg time precond (s)	139	**98**	102
Avg time solve (s)	128	**91**	98

Impact on performances for MaPHYS

Matrix Almond, divided in 32 domains

	Natif	DG	HF
Max-min Schur size	3118	2765	**2603**
Avg Schur size (n)	**5492**	5672	5749
Avg Schur size (nnz)	**31M**	32M	33M
Total interface size	**86551**	89452	90664
Avg time facto (s)	87	84	**84**
Avg time precond (s)	58	62	**54**
Avg time solve (s)	91	59	**50**

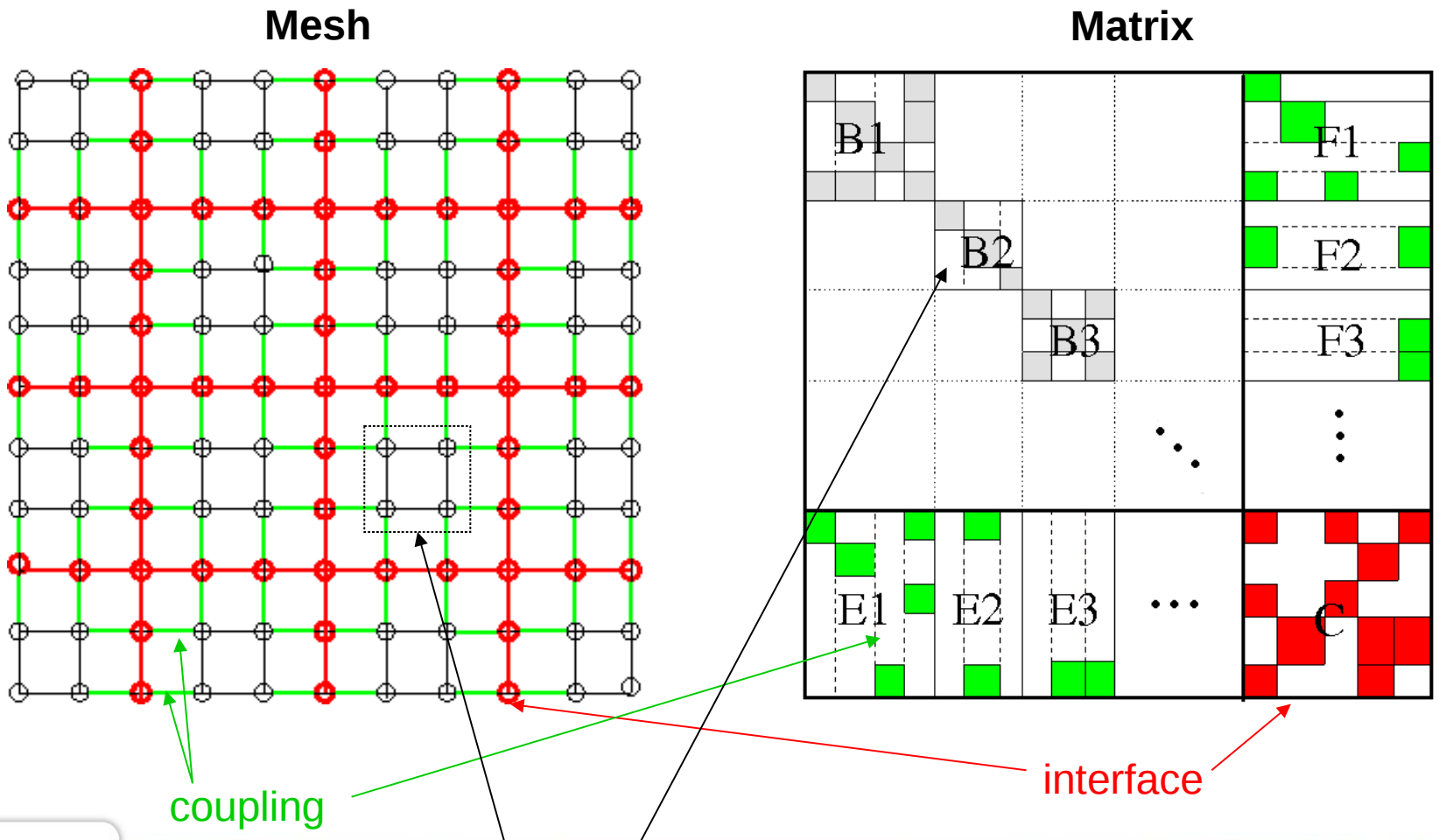
9 Future work

- In a parallel context, chose the best from the 3 variants (GG, DG, HF) (keep same complexity for the global reordering process, but time increases)
- An other important criteria to control : total size of the interface
- Comparison with k-way or other graph partitioning (Metis, ...)
- Release new algorithms in the master branch of Scotch package

Thanks for listening.
Questions ?

Memory optimization to build a Schur complement for hybrid solvers

Domain decomposition

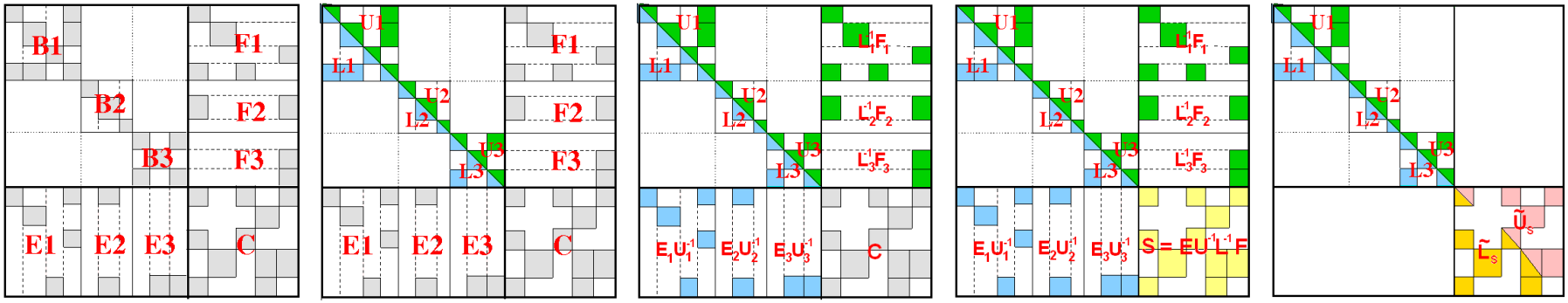


Domain decomposition

Step 1 :

Build an approximate factorization :

$$A \approx \tilde{L}\tilde{U} = \begin{pmatrix} L_B U_B & \\ & \tilde{L}_S \tilde{U}_S \end{pmatrix}$$

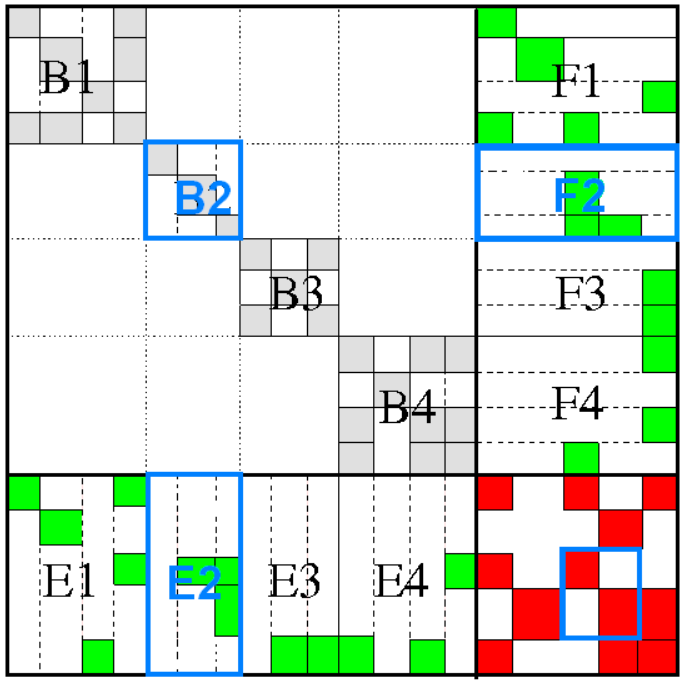


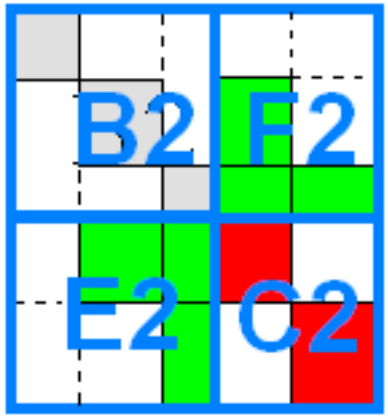
Initial Matrix | Factorize each domain | TRSM | Add contributions | Incomplete factorization of S

Step 2 :

Iterative method to solve Ax = b using $\tilde{L}$ & $\tilde{U}$ as preconditioner

Domain decomposition





Local matrix for subdomain 2

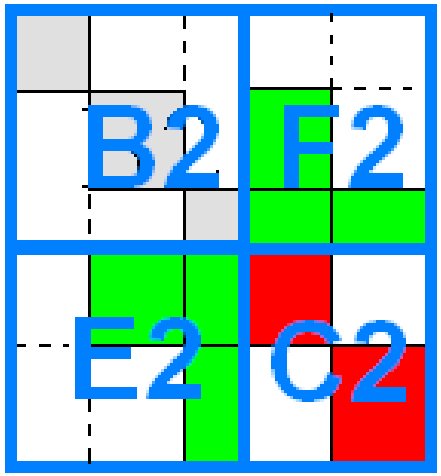
2 levels of parallelism :

- parallelism between subdomains
- parallel direct solver to factorize each subdomain

Memory optimization

Memory overcosts during the factorization of a subdomain :

- Aggregated blocs (Fan-In)
- Coupling blocs (E/F)

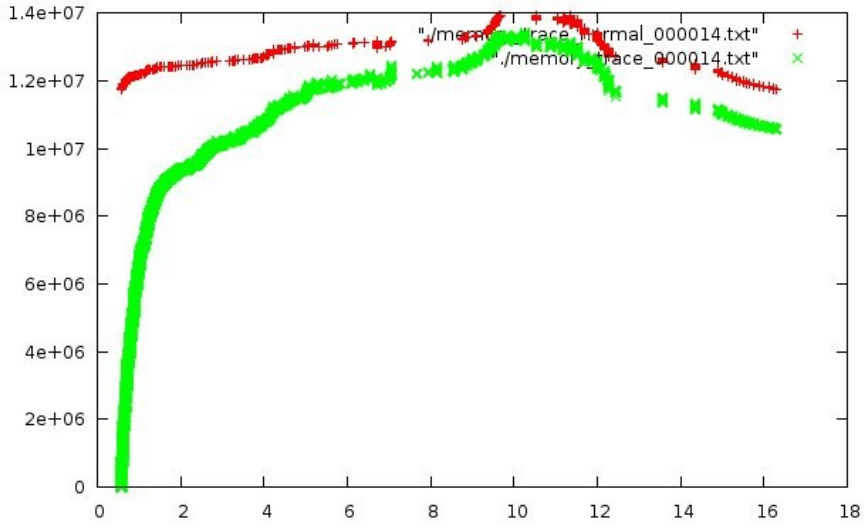


Idea 1 :
Dynamic allocations of column blocs (as late as possible)

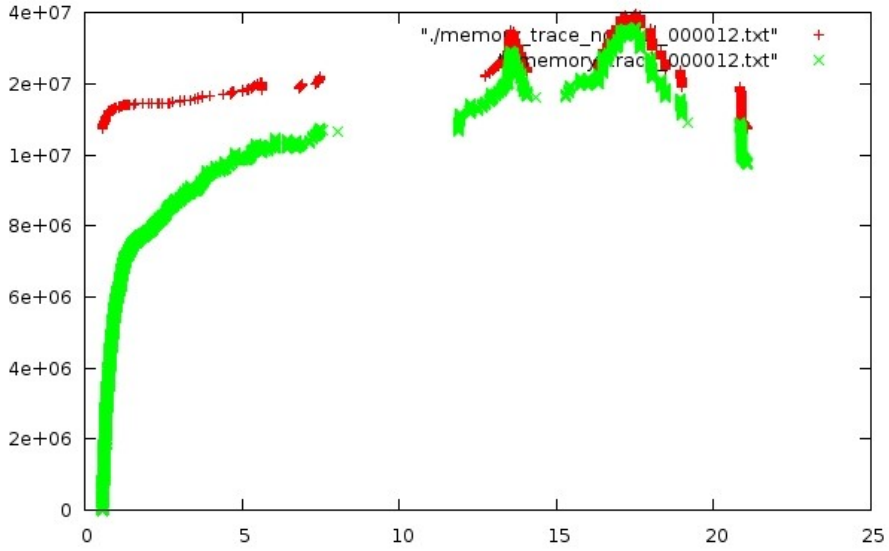
Idea 2 :
Free the coupling blocs as soon as possible

Memory Optimization

INLINE matrix, factorized using PaStiX with 16 processus and 4 threads per processus, size of the interface about 1500



Processus 14



Processus 12

Memory Optimization

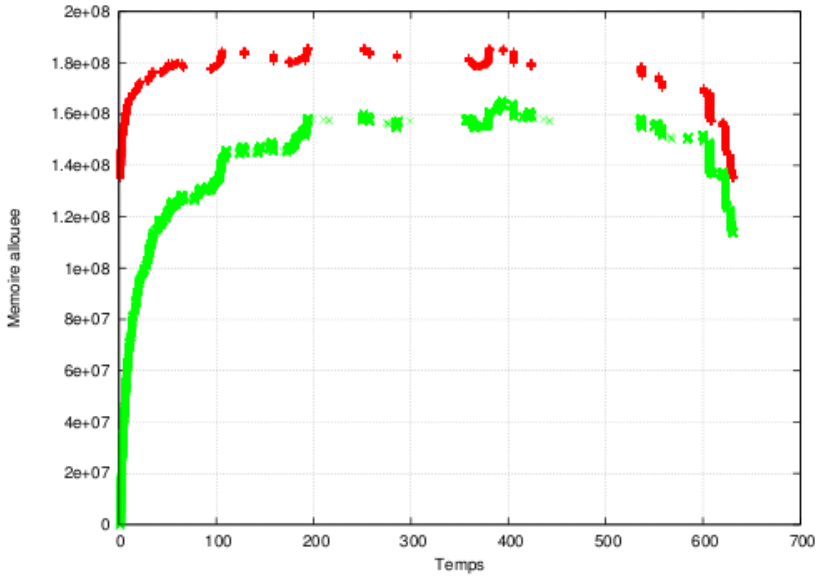
Idea 3 :

Use a « left looking version » to build the non-coupling part of columns blocs

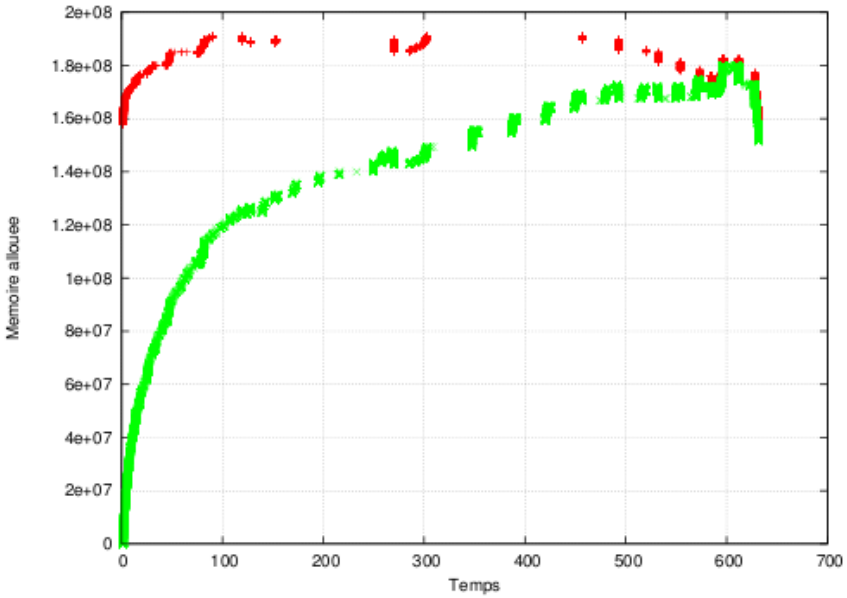
Memory Optimization

domain 1 for AUDI matrix, split in 2 subdomains,

factorized with 4 processus and 4 threads per processus



Processus 1



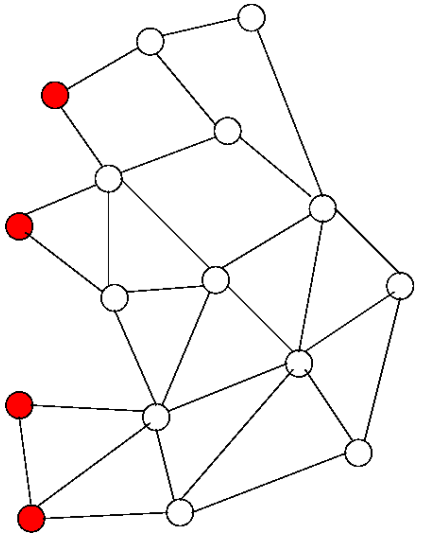
Processus 3

Some results

- left-right-looking version : from 1 to 4 MPI nodes, memory gain from 86 % to 10%
- in the case of hybrid solvers we expect small number of MPI node per subdomain
- Same gains when increasing the number of threads instead of MPI nodes

7 Halo-first Greedy Graph Growing

Problem : graph induced by halo may be disconnected ; we have to reconnect it before cutting halo



Algorithm :

Let begin with an empty induced graph
Let call H = {h_0, h_1, …, h_m } the halo vertices.
Let P be a partition of the halo vertices, initially P = {{h_0}, {h_1}, …, {h_m}}

While p ≠ {H} :
- find the shortest path between any two elements of P
- merge them
- add to the induced graph all vertices of the path

4 Fiduccia-Mattheyses algorithm

Original version

$(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
$passnum \leftarrow 0$;
repeat
  $(P_0, S, P_1) \leftarrow (P_0^*, S^*, P_1^*)$;
  $\Delta \leftarrow |P_0| - |P_1|$;
  $tabu \leftarrow \emptyset$;
  $movenum \leftarrow 0, enhanced \leftarrow false$;
  $pref \leftarrow mod(passnum, 2)$;
  while $movenum < movenbr$ **do**
    $(f, v, i) \leftarrow getSep(S \setminus tabu, \max(\Delta_{th}, |\Delta|), pref)$;
    if $\neg f$ **then** `/* No movable vertex */`
      break;
    `/* Move` v `from separator to part` i `*/`;
    $R \leftarrow \{w | (v, w) \in E \text{ and } w \in P_{\neg i}\}$;
    $S \leftarrow S \setminus \{v\} \cup R$;
    $P_i \leftarrow P_i \cup \{v\}$, $P_{\neg i} \leftarrow P_{\neg i} \setminus R$;
    $\Delta \leftarrow |P_0| - |P_1|$;
    $movenum{+}{+}$;
    $tabu \leftarrow tabu \cup \{v\}$;
    if (P_0, S, P_1) *is better than* (P_0^*, S^*, P_1^*) **then**
      $(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
      $movenum \leftarrow 0, enhanced \leftarrow true$;
  $passnum{+}{+}$;
until $\neg enhanced$ *or* $(passnum = passnbr)$;
return (P_0^*, S^*, P_1^*);

Choose a vertex v in S and a part i

Move v to i

If best partition ever, record it

4 Fiduccia-Mattheyses algorithm

Original version

$(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
$passnum \leftarrow 0$;
repeat
  $(P_0, S, P_1) \leftarrow (P_0^*, S^*, P_1^*)$;
  $\Delta \leftarrow |P_0| - |P_1|$;
  $tabu \leftarrow \emptyset$;
  $movenum \leftarrow 0$, $enhanced \leftarrow false$;
  $pref \leftarrow mod(passnum, 2)$;
  while $movenum < movenbr$ **do**
    $(f, v, i) \leftarrow getSep(S \setminus tabu, \max(\Delta_{th}, |\Delta|), pref)$;
    if $\neg f$ **then** `/* No movable vertex */`
      break;
    `/* Move` v `from separator to part` i `*/`;
    $R \leftarrow \{w | (v, w) \in E \text{ and } w \in P_{\neg i}\}$;
    $S \leftarrow S \setminus \{v\} \cup R$;
    $P_i \leftarrow P_i \cup \{v\}$, $P_{\neg i} \leftarrow P_{\neg i} \setminus R$;
    $\Delta \leftarrow |P_0| - |P_1|$;
    $movenum{+}{+}$;
    $tabu \leftarrow tabu \cup \{v\}$;
    if (P_0, S, P_1) *is better than* (P_0^*, S^*, P_1^*) **then**
      $(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
      $movenum \leftarrow 0$, $enhanced \leftarrow true$;
  $passnum{+}{+}$;
until $\neg enhanced$ *or* $(passnum = passnbr)$;
return (P_0^*, S^*, P_1^*);

- main loop: choose a vertex, move it, update best separator
- 1 pass = make moves until the last <movenbr> moves do not bring improvement

4 Fiduccia-Mattheyses algorithm

Original version

```
(P0*, S*, P1*) ← (P0, S, P1);
passnum ← 0;
repeat
    (P0, S, P1) ← (P0*, S*, P1*);
    Δ ← |P0| − |P1|;
    tabu ← ∅;
    movenum ← 0, enhanced ← false;
    pref ← mod(passnum, 2);
    while movenum < movenbr do
        (f, v, i) ← getSep(S \ tabu, max(Δth, |Δ|), pref);
        if ¬f then /* No movable vertex */
            break;
        /* Move v from separator to part i */;
        R ← {w|(v, w) ∈ E and w ∈ P¬i};
        S ← S \ {v} ∪ R;
        Pi ← Pi ∪ {v}, P¬i ← P¬i \ R;
        Δ ← |P0| − |P1|;
        movenum++;
        tabu ← tabu ∪ {v};
        if (P0, S, P1) is better than (P0*, S*, P1*) then
            (P0*, S*, P1*) ← (P0, S, P1);
            movenum ← 0, enhanced ← true;
    passnum++;
until ¬enhanced or (passnum = passnbr) ;
return (P0*, S*, P1*);
```

- main loop: choose a vertex, move it, update best separator
- 1 pass = make moves until the last <movenbr> moves do not bring improvement
- tabu search

4 Fiduccia-Mattheyses algorithm

Original version

$(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
$passnum \leftarrow 0$;
repeat
  $(P_0, S, P_1) \leftarrow (P_0^*, S^*, P_1^*)$;
  $\Delta \leftarrow |P_0| - |P_1|$;
  $tabu \leftarrow \emptyset$;
  $movenum \leftarrow 0$, $enhanced \leftarrow false$;
  $pref \leftarrow mod(passnum, 2)$;
  while $movenum < movenbr$ **do**
    $(f, v, i) \leftarrow getSep(S \setminus tabu, \max(\Delta_{th}, |\Delta|), pref)$;
    if $\neg f$ **then** `/* No movable vertex */`
      break;
    `/* Move` v `from separator to part` i `*/`;
    $R \leftarrow \{w | (v, w) \in E \text{ and } w \in P_{\neg i}\}$;
    $S \leftarrow S \setminus \{v\} \cup R$;
    $P_i \leftarrow P_i \cup \{v\}$, $P_{\neg i} \leftarrow P_{\neg i} \setminus R$;
    $\Delta \leftarrow |P_0| - |P_1|$;
    $movenum$++;
    $tabu \leftarrow tabu \cup \{v\}$;
    if (P_0, S, P_1) *is better than* (P_0^*, S^*, P_1^*) **then**
      $(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
      $movenum \leftarrow 0$, $enhanced \leftarrow true$;
  $passnum$++;
until $\neg enhanced$ *or* $(passnum = passnbr)$;
return (P_0^*, S^*, P_1^*);

- main loop: choose a vertex, move it, update best separator
- 1 pass = make moves until the last <movenbr> moves do not bring improvement
- tabu search
- passes keep going on while last pass bring improvement
- each pass begins with best separator ever found

4 Fiduccia-Mattheyses algorithm

Original version

$(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
$passnum \leftarrow 0$;
repeat
  $(P_0, S, P_1) \leftarrow (P_0^*, S^*, P_1^*)$;
  $\Delta \leftarrow |P_0| - |P_1|$;
  $tabu \leftarrow \emptyset$;
  $movenum \leftarrow 0, enhanced \leftarrow false$;
  $pref \leftarrow mod(passnum, 2)$;
  while $movenum < movenbr$ **do**
    $(f, v, i) \leftarrow getSep(S \setminus tabu, \max(\Delta_{th}, |\Delta|), pref)$;
    if $\neg f$ **then** `/* No movable vertex */`
      break;
    `/* Move` v `from separator to part` i `*/`;
    $R \leftarrow \{w | (v, w) \in E \text{ and } w \in P_{\neg i}\}$;
    $S \leftarrow S \setminus \{v\} \cup R$;
    $P_i \leftarrow P_i \cup \{v\}$, $P_{\neg i} \leftarrow P_{\neg i} \setminus R$;
    $\Delta \leftarrow |P_0| - |P_1|$;
    $movenum{+}{+}$;
    $tabu \leftarrow tabu \cup \{v\}$;
    if (P_0, S, P_1) *is better than* (P_0^*, S^*, P_1^*) **then**
      $(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
      $movenum \leftarrow 0, enhanced \leftarrow true$;
  $passnum{+}{+}$;
until $\neg enhanced$ *or* $(passnum = passnbr)$;
return (P_0^*, S^*, P_1^*);

- main loop: choose a vertex, move it, update best separator
- 1 pass = make moves until the last <movenbr> moves do not bring improvement
- tabu search
- passes keep going on while last pass bring improvement
- each pass begins with best separator ever found
- preferred part switched at each pass

4 Fiduccia-Mattheyses algorithm

Modified version

$(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
$passnum \leftarrow 0$;
repeat
  $(P_0, S, P_1) \leftarrow (P_0^*, S^*, P_1^*)$;
  $\Delta \leftarrow |P_0| - |P_1|$, $\overline{\Delta} \leftarrow |P_0|_h - |P_1|_h$;
  $tabu \leftarrow \emptyset$;
  $movenum \leftarrow 0, enhanced \leftarrow false$;
  $pref \leftarrow mod(passnum, 2)$;
  while $movenum < movenbr$ **do**
    $f \leftarrow false$;
    if $|\overline{\Delta}| > \overline{\Delta}_{th}$ **then**
      $(f, v, i) \leftarrow getHalo(S, tabu, -\overline{\Delta})$;
    if $\neg f$ **then**
      $(f, v, i) \leftarrow getSep(S, tabu, \max(\Delta_{th}, |\Delta|), pref)$;
    if $\neg f$ **then** `/* No movable vertex */`
      break;
    `/* Move` v `from separator to part` i `*/`;
    $R \leftarrow \{w | (v, w) \in E \text{ and } w \in P_{\neg i}\}$;
    $S \leftarrow S \setminus \{v\} \cup R$;
    $P_i \leftarrow P_i \cup \{v\}$, $P_{\neg i} \leftarrow P_{\neg i} \setminus R$;
    $\Delta \leftarrow |P_0| - |P_1|$, $\overline{\Delta} \leftarrow |P_0|_h - |P_1|_h$;
    $movenum$++;
    $tabu \leftarrow tabu \cup \{v\}$;
    if (P_0, S, P_1) *is better than* (P_0^*, S^*, P_1^*) **then**
      $(P_0^*, S^*, P_1^*) \leftarrow (P_0, S, P_1)$;
      $movenum \leftarrow 0, enhanced \leftarrow true$;
  $passnum$++;
until $\neg enhanced$ *or* $(passnum = passnbr)$;
return (P_0^*, S^*, P_1^*);

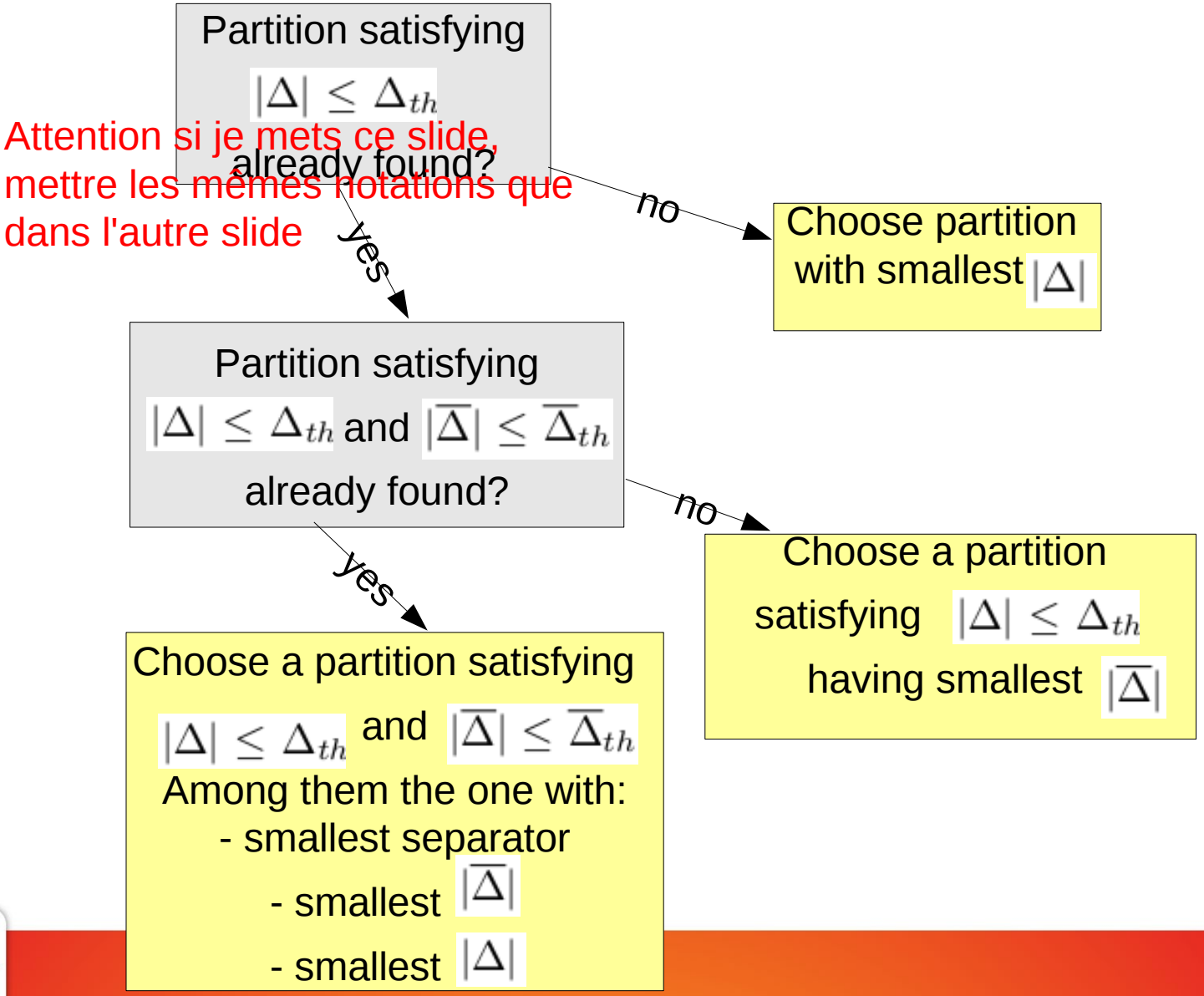
GetSep choice based on (as before):
- part balance (if large imbalance)
- separator minimization
- preferred part

GetHalo choice based on:
- halo imbalance

Next slide :-)

4 Fiduccia-Mattheyses algorithm

Modified version, choice of « the best separator »



Attention si je mets ce slide, mettre les mêmes notations que dans l'autre slide

8 Results
16 domains, DG

	interface imbalance			interior imbalance		
id	*GG*	% *GG**	% *DG*	*GG*	% *GG**	% *DG*
1	297	7,4	**-19,2**	1311	**-69,1**	-23,3
2	1112	1,6	**-40,0**	4678	**-66,5**	-78,9
3	522	-11,3	**-39,7**	1635	**-13,9**	-13,0
4	244	-9,0	**-11,5**	1568	-23,7	**-31,1**
5	475	**-17,9**	-3,8	4605	**-85,6**	-62,3
6	869	-21,3	**-41,3**	4107	**-78,5**	-49,6
7	**905**	49,4	26,4	4942	**-69,1**	-63,7
8	261	0,0	**-46,7**	420	**0,0**	79,8
9	365	-3,8	**-44,9**	8128	**-82,8**	-65,2
10	1002	14,0	**-32,1**	4852	**-73,2**	-44,6
11	509	-3,7	**-44,4**	2831	**-84,1**	27,5
12	569	-25,0	**-59,1**	22416	**-79,5**	-67,5
13	532	-11,8	**-58,5**	12049	**-69,0**	-49,5
14	756	-23,1	**-46,3**	17458	-61,9	**-66,6**
15	6678	6,6	**-29,8**	35466	**-73,9**	-68,4
16	564	-11,9	**-62,9**	15092	**-65,5**	-58,0
17	2130	18,6	**-50,4**	32202	**-72,9**	-67,7
18	335	17,0	**-19,7**	28057	**-70,1**	-64,9
19	2604	-16,4	**-42,0**	25853	**-66,5**	-59,5
20	9990	-14,2	**-25,8**	73635	**-80,7**	-16,5
21	927	0,6	**-40,9**	2377	-58,6	**-59,5**
22	3468	5,0	**-78,5**	**3336**	4,3	18,2
23	1869	7,1	**-3,2**	14424	**-73,3**	-64,4
24	2460	-9,1	**-73,4**	8940	-44,1	**-60,6**
25	6837	-11,6	**-69,7**	26877	-63,0	**-68,9**
26	780	-15,9	**-53,8**	24987	-58,3	**-65,2**
27	6168	-27,5	**-68,4**	67721	-68,7	**-76,2**
28	4344	-15,6	**-41,4**	240729	-76,9	**-77,1**
29	11539	8,6	**-51,3**	244959	-73,9	**-77,2**
30	9936	-6,9	**-52,0**	286992	**-72,6**	-8,5

8 Results
16 domains, HF

	interface imbalance			interior imbalance		
id	GG	% $GG^\star$	% HF	GG	% $GG^\star$	% HF
1	297	7,4	**-29,3**	1311	-69,1	**-73,5**
2	1112	1,6	**-34,9**	4678	-66,5	**-74,2**
3	522	-11,3	**-63,4**	1635	**-13,9**	-7,2
4	244	**-9,0**	103,3	1568	-23,7	**-24,5**
5	475	-17,9	**-26,7**	4605	**-85,6**	-74,3
6	869	-21,3	**-48,2**	4107	**-78,5**	-65,6
7	905	49,4	**-24,2**	4942	-69,1	**-72,2**
8	261	0,0	**-69,0**	420	0,0	**-11,4**
9	365	-3,8	**-41,4**	8128	**-82,8**	-69,2
10	1002	14,0	**-66,9**	4852	**-73,2**	-47,5
11	509	-3,7	**-50,3**	2831	**-84,1**	-19,1
12	569	-25,0	**-70,1**	22416	**-79,5**	-72,9
13	532	-11,8	**-37,0**	12049	**-69,0**	-52,3
14	756	**-23,1**	-20,4	17458	-61,9	**-67,2**
15	6678	6,6	**-22,4**	35466	**-73,9**	-68,2
16	564	-11,9	**-54,4**	15092	**-65,5**	-59,2
17	2130	18,6	**-44,8**	32202	-72,9	**-73,0**
18	**335**	17,0	14,6	28057	**-70,1**	-68,3
19	2604	**-16,4**	-15,8	25853	**-66,5**	-49,7
20	9990	-14,2	**-55,4**	73635	**-80,7**	-79,6
21	927	0,6	**-49,4**	2377	-58,6	**-64,5**
22	3468	5,0	**-75,4**	**3336**	4,3	16,1
23	1869	7,1	**-24,2**	14424	**-73,3**	-72,0
24	2460	-9,1	**-55,9**	8940	-44,1	**-51,5**
25	6837	-11,6	**-65,5**	26877	-63,0	**-80,3**
26	780	-15,9	**-33,6**	24987	-58,3	**-66,0**
27	6168	-27,5	**-73,6**	67721	-68,7	**-74,5**
28	4344	-15,6	**-47,9**	240729	**-76,9**	-73,2
29	11539	8,6	**-57,0**	244959	**-73,9**	-70,9
30	9936	-6,9	**-55,9**	286992	**-72,6**	-71,4

8 Results multilevel + DG/HF, 16 to 512 domains

Almond
n ≈ 7 x 10^6
nnz ≈ 100 x 10^6

	interface imbalance			interior imbalance		
dom	*GG*	% *DG*	% *HF*	*GG*	% *DG*	% *HF*
16	4344	-41,4	**-47,9**	240729	**-77,1**	-73,2
32	3179	**-31,5**	-0,5	133886	**-80,1**	-76,9
64	2258	**-47,6**	-17,8	83819	**-79,2**	-79,1
128	1822	**-42,9**	-32,6	48087	-77,6	**-80,9**
256	1071	**-44,4**	2,5	27695	-77,9	**-83,0**
512	910	-17,1	**-22,3**	16243	-80,0	**-84,7**

Nice-7
n ≈ 8 x 10^6
nnz ≈ 660 x 10^6

	interface imbalance			interior imbalance		
dom	*GG*	% *DG*	% *HF*	*GG*	% *DG*	% *HF*
16	11539	-51,3	**-57,0**	244959	**-77,2**	-70,9
32	10991	**-45,7**	-38,5	188834	-81,0	**-81,6**
64	8997	40,7	**-30,8**	101838	-6,6	**-80,2**
128	5694	11,7	**-26,9**	66792	-1,9	**-84,3**
256	4554	**-33,1**	-27,6	34314	7,3	**-82,4**
512	3762	-30,8	**-33,1**	19734	-6,3	**-84,2**

10millions
n ≈ 10 x 10^6
nnz ≈ 160 x 10^6

	interface imbalance			interior imbalance		
dom	*GG*	% *DG*	% *HF*	*GG*	% *DG*	% *HF*
16	9936	-52,0	**-55,9**	286992	-8,5	**-71,4**
32	6666	43,7	**-56,6**	188900	13,0	**-77,7**
64	7036	**-47,4**	-31,1	125444	-18,8	**-78,4**
128	4564	4,0	**-48,7**	79754	-52,9	**-78,8**
256	3114	163,5	**-32,5**	42931	-30,2	**-80,8**
512	2336	22,2	**-54,2**	25800	49,3	**-83,4**