Growing a Graph Matching from a Handful of Seeds

Ehsan KAZEMI[1], S. Hamed HASSANI[2], and Matthias GROSSGLAUSER[1]

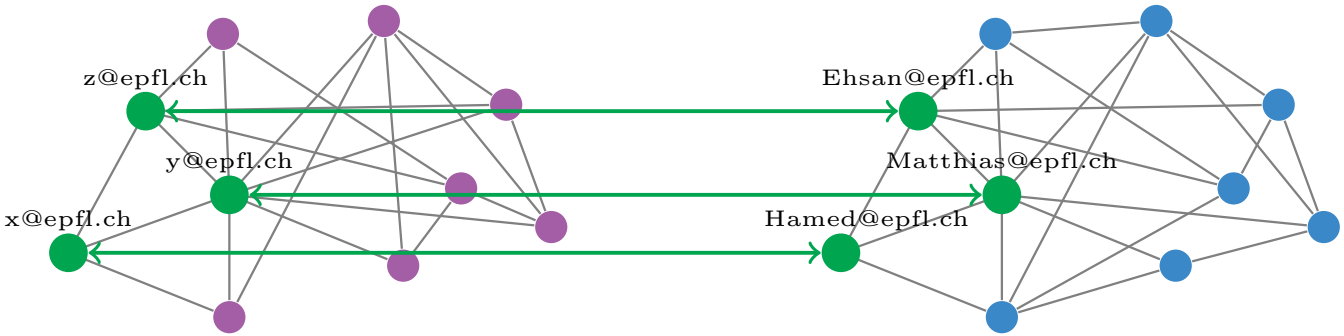
[1]School of Computer and Communication Sciences, EPFL

[2]Department of Computer Science, ETHZ

September 1, 2015

Motivation

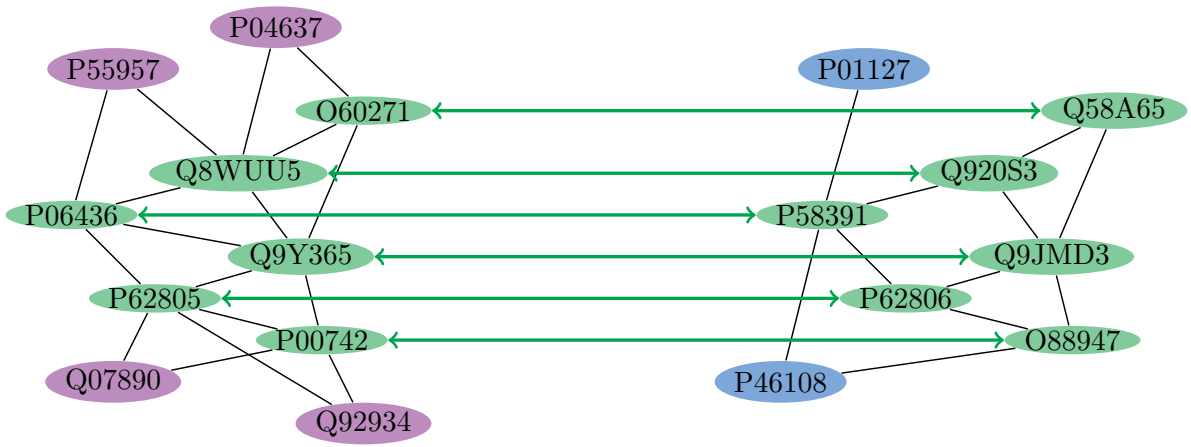
Example 1: network de-anonymization



Anonymized e-mail network

Linkedin connections

Example 2: protein-protein interaction network alignment



Human network

Mouse network

Motivation

Graph matching (also known as **network reconciliation** or **network alignment**) is studied in many fields:

- Network analysis: matching networks in similar domains for friend suggestion and personalized advertisements
- Bioinformatics: protein-protein interaction networks alignment
- Document and Image processing: OCR and handwritten recognition
- Biometric identification: face authentication and recognition
- Image database: matching graph segments of two scenes

Matching graph segments of scenes [Lazebnik et al., 2006]

What is Graph Matching?

Goal: find the unknown matching (bijection) between nodes in the intersection of the two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ where the presence of edges between the same nodes in the two graphs are correlated

Questions:

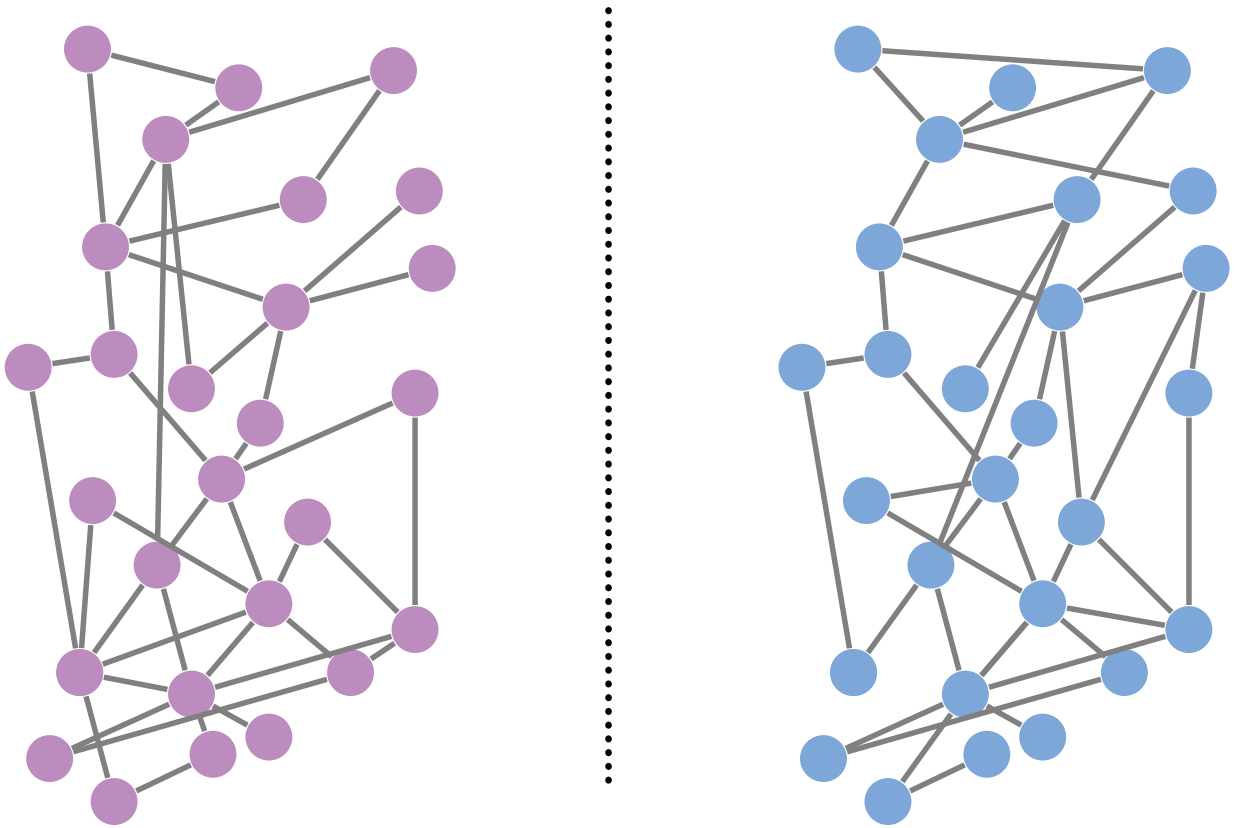
- When is it possible to align?
- **How to align? graph matching algorithms**
  - Is it possible to use only the graph structures to establish the true matching between the nodes?

Algorithm, Model and Performance Guarantee

- Algorithm: percolation graph matching [Yartseva and Grossglauser, 2013; Chiasserini et al., 2014; Korula and Lattanzi, 2014]
- Model: a random bigraph generator [Pedarsani and Grossglauser, 2011; Kazemi et al., 2015]
- Performance guarantee: theory of bootstrap percolation over random graphs [Janson et al., 2010]

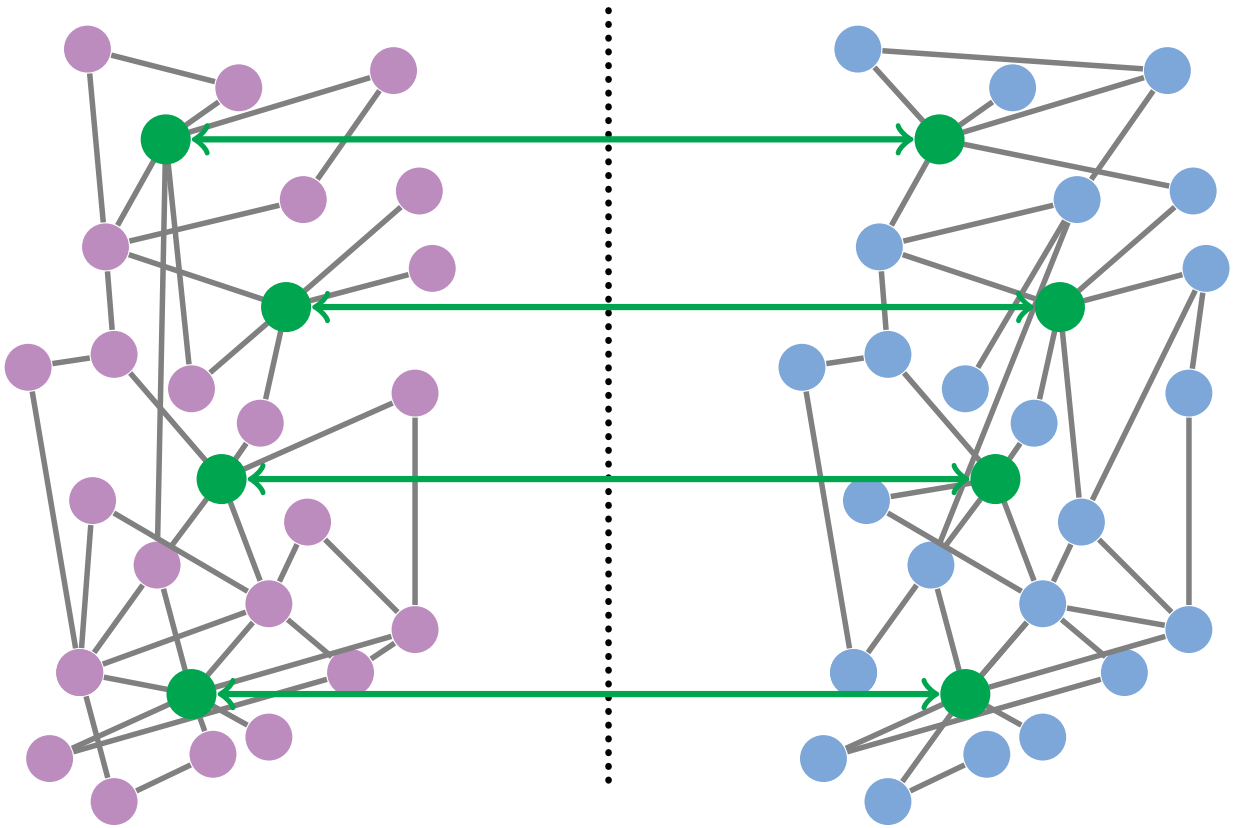
Percolation Graph Matching

- An initial candidate set of **seed pairs**
- Every non-matched pair with r **neighbouring seed-pairs** get matched and becomes a new seed



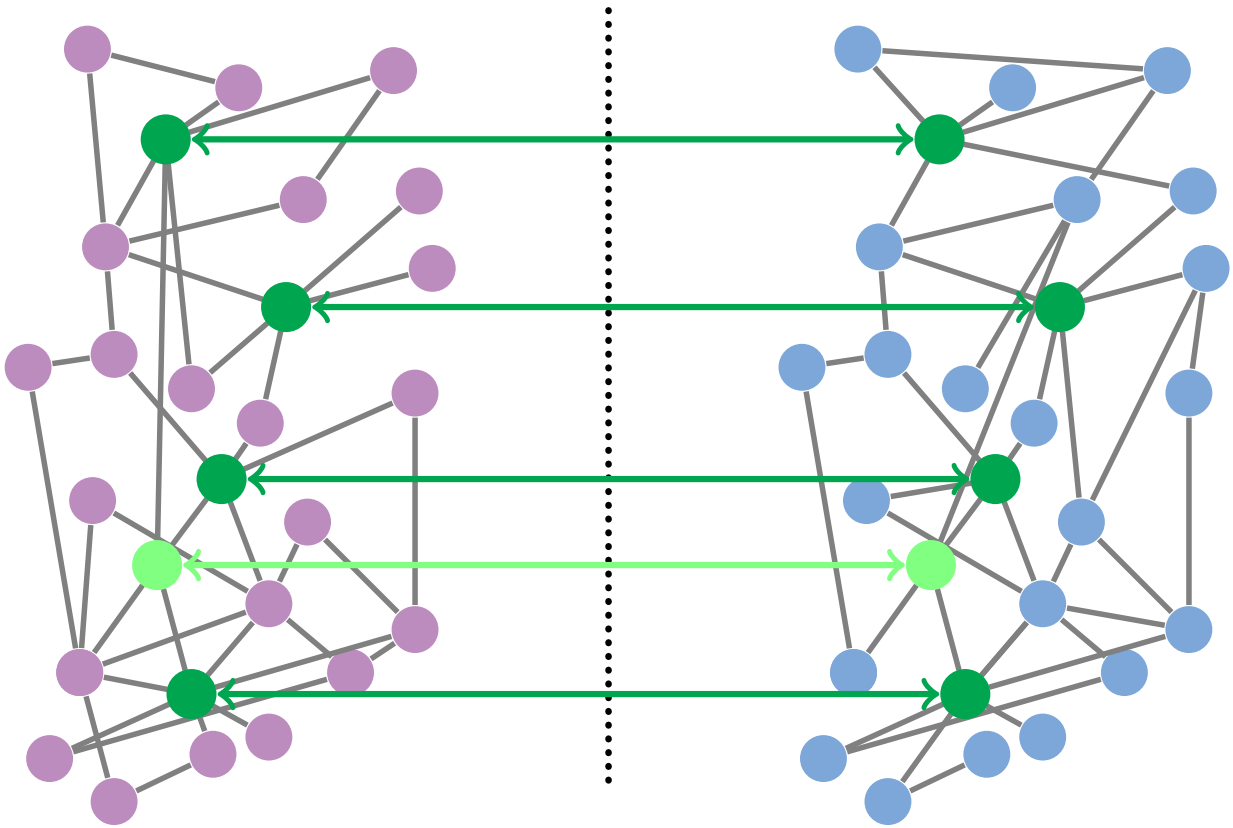
Percolation Graph Matching

- An initial candidate set of **seed pairs**
- Every non-matched pair with r **neighbouring seed-pairs** get matched and becomes a new seed



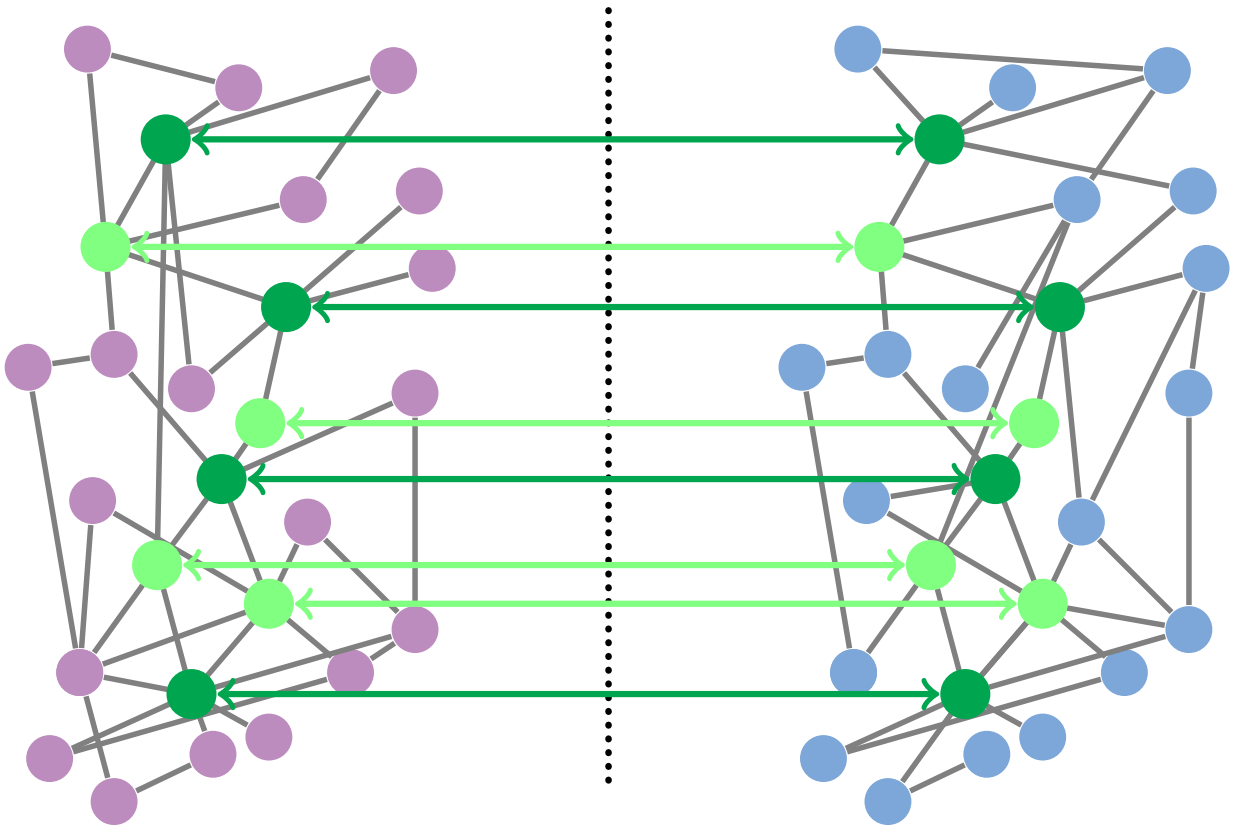
Percolation Graph Matching

- An initial candidate set of **seed pairs**
- Every non-matched pair with r **neighbouring seed-pairs** get matched and becomes a new seed



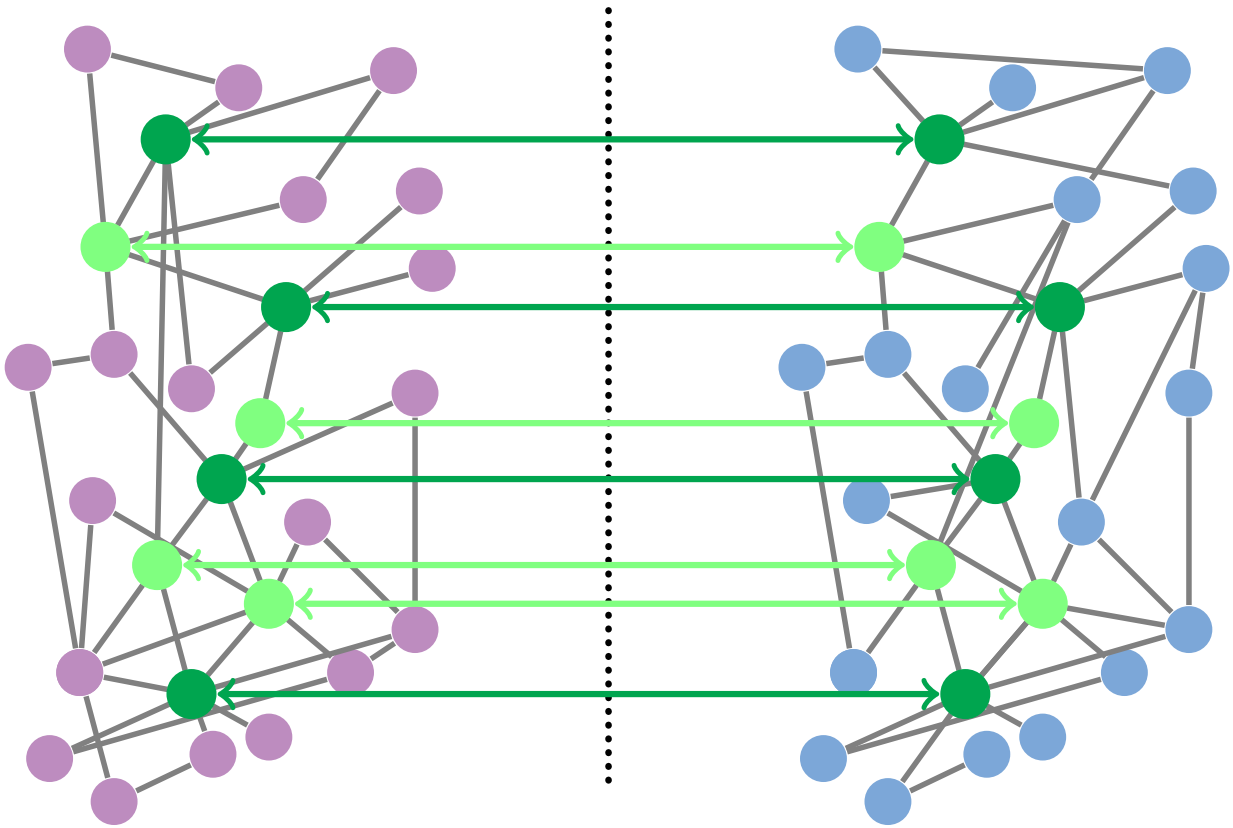
Percolation Graph Matching

- An initial candidate set of **seed pairs**
- Every non-matched pair with r **neighbouring seed-pairs** get matched and becomes a new seed



Percolation Graph Matching

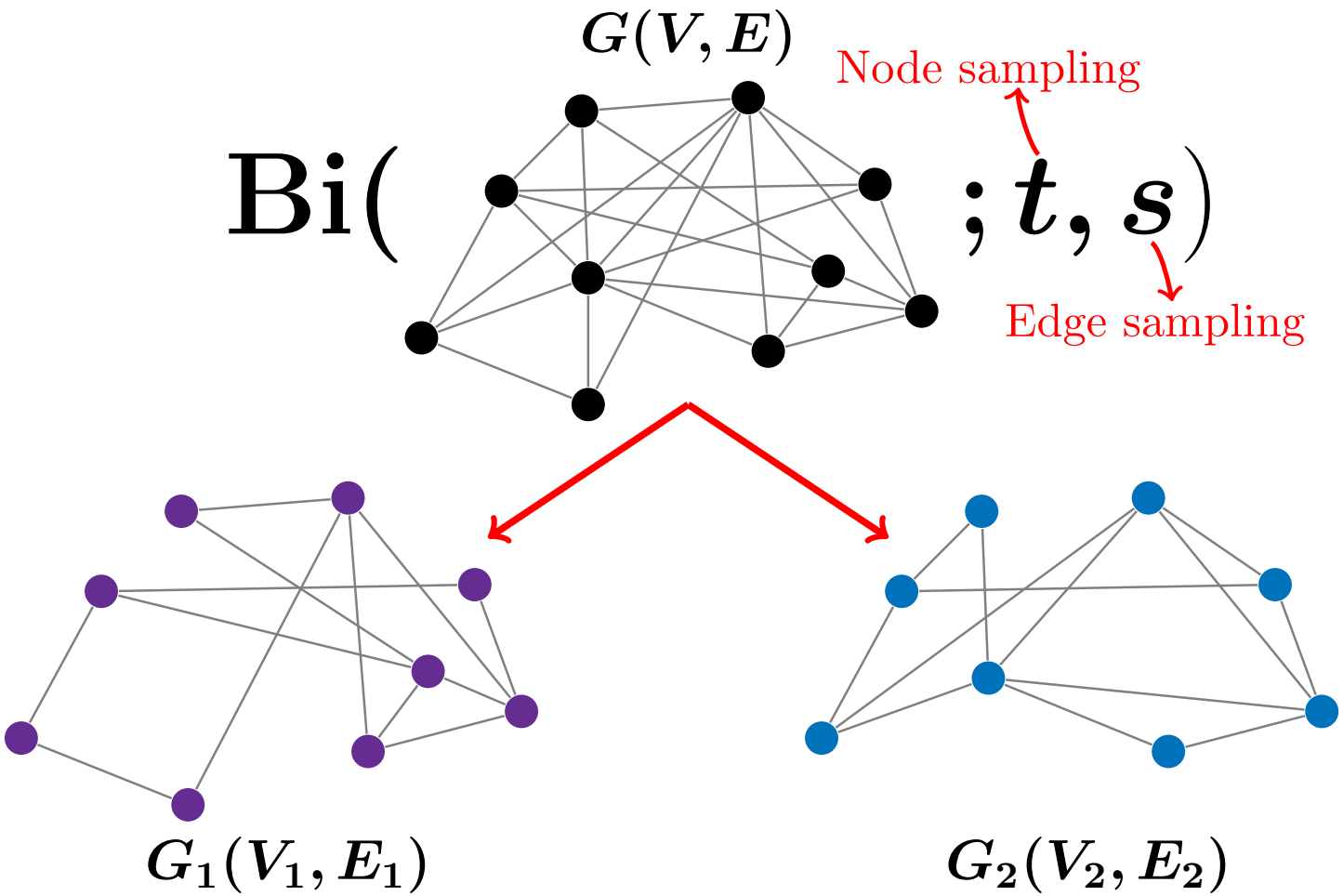
- An initial candidate set of **seed pairs**
- Every non-matched pair with r **neighbouring seed-pairs** get matched and becomes a new seed



Size of the final matching vs. number of initial seeds

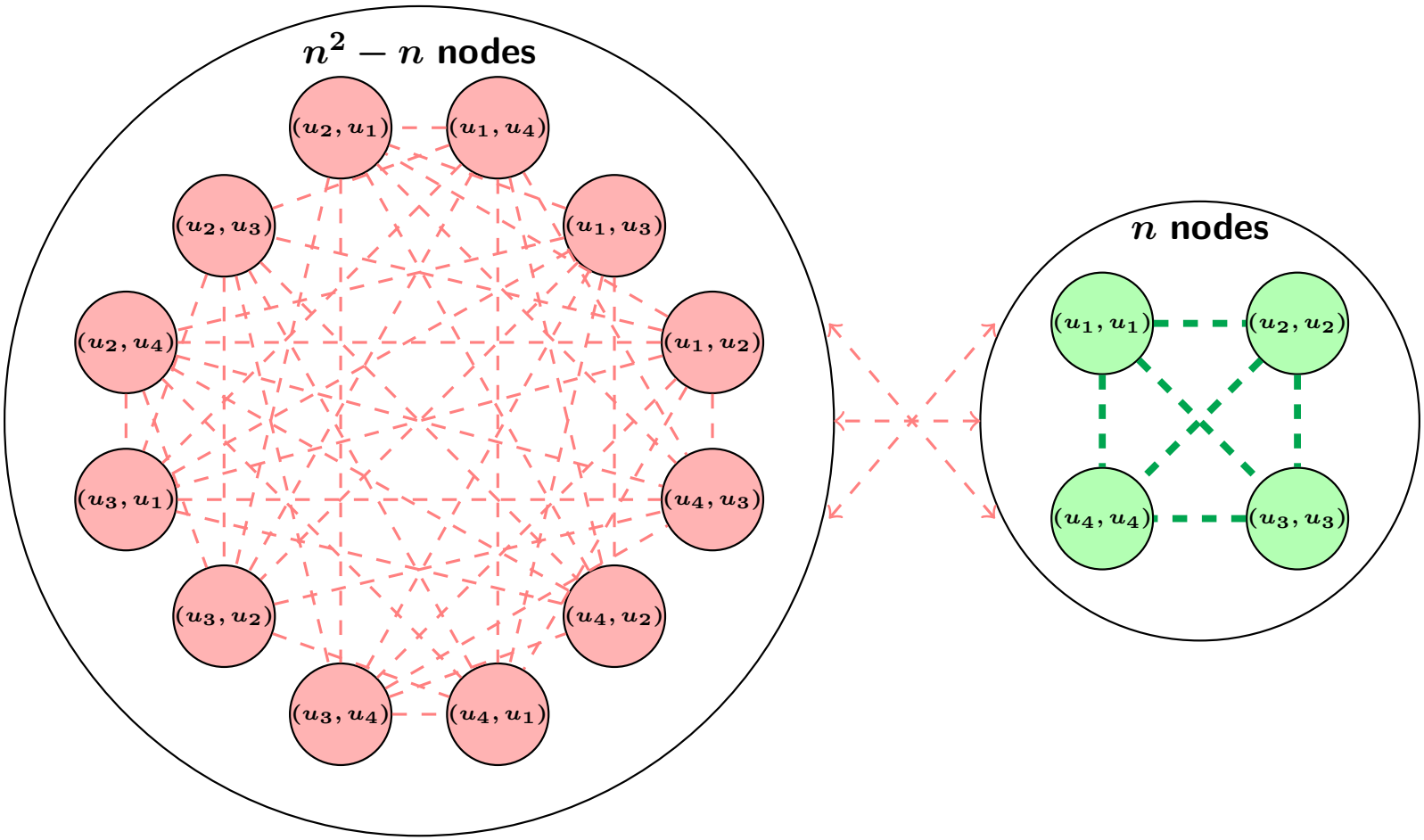
$\mathrm{Bi}(G; t, s)$: A Random Bigraph Model

- $\mathrm{Bi}(G; t, s)$ is a random bigraph model to generate two correlated graphs



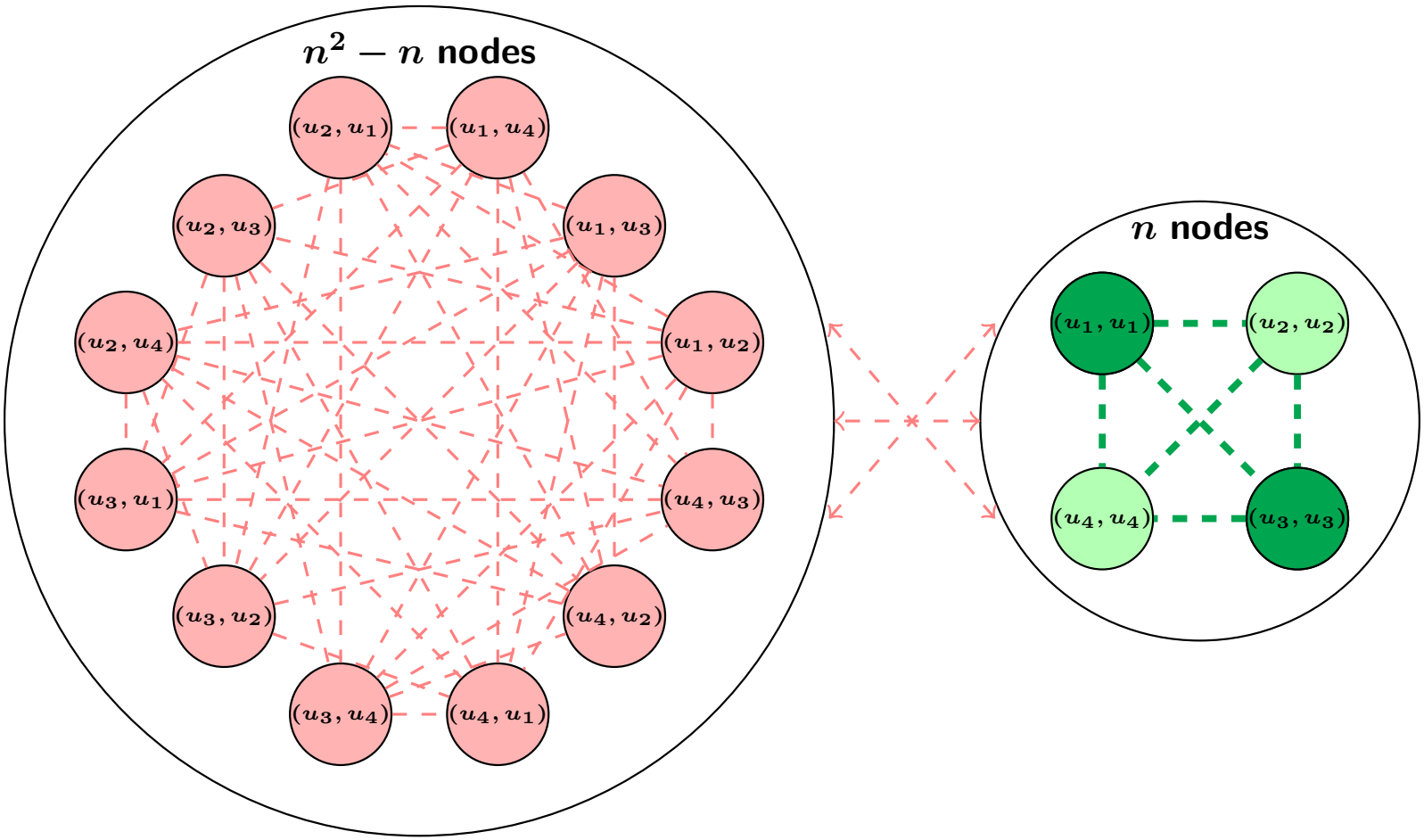
Bootstrap Percolation

- Marks are spread over the tensor product of the two graphs:
  - Green nodes are correct pairs
  - Red nodes are wrong pairs
  - Green nodes are more connected



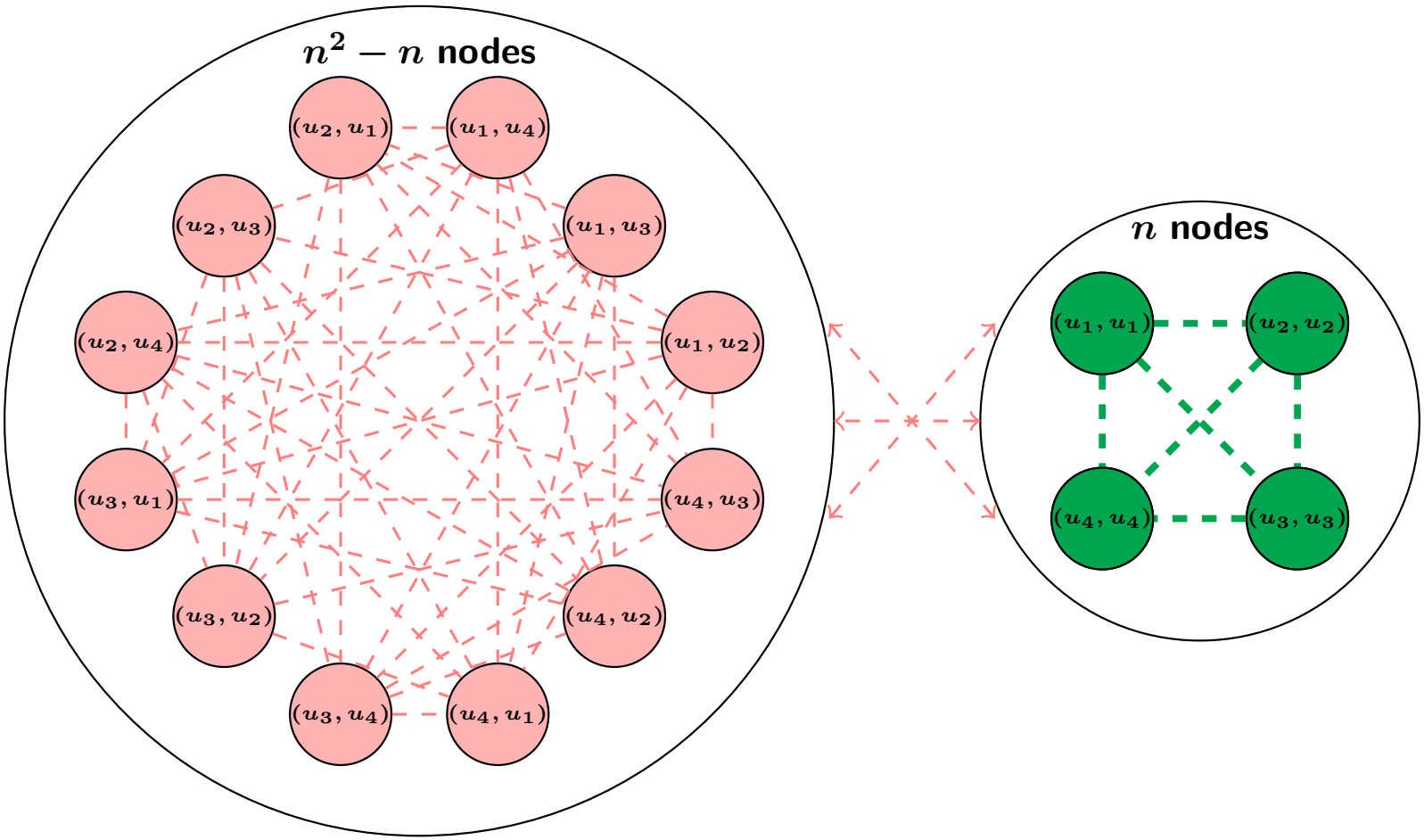
Bootstrap Percolation

- Marks are spread over the tensor product of the two graphs:
  - Green nodes are correct pairs
  - Red nodes are wrong pairs
  - Green nodes are more connected



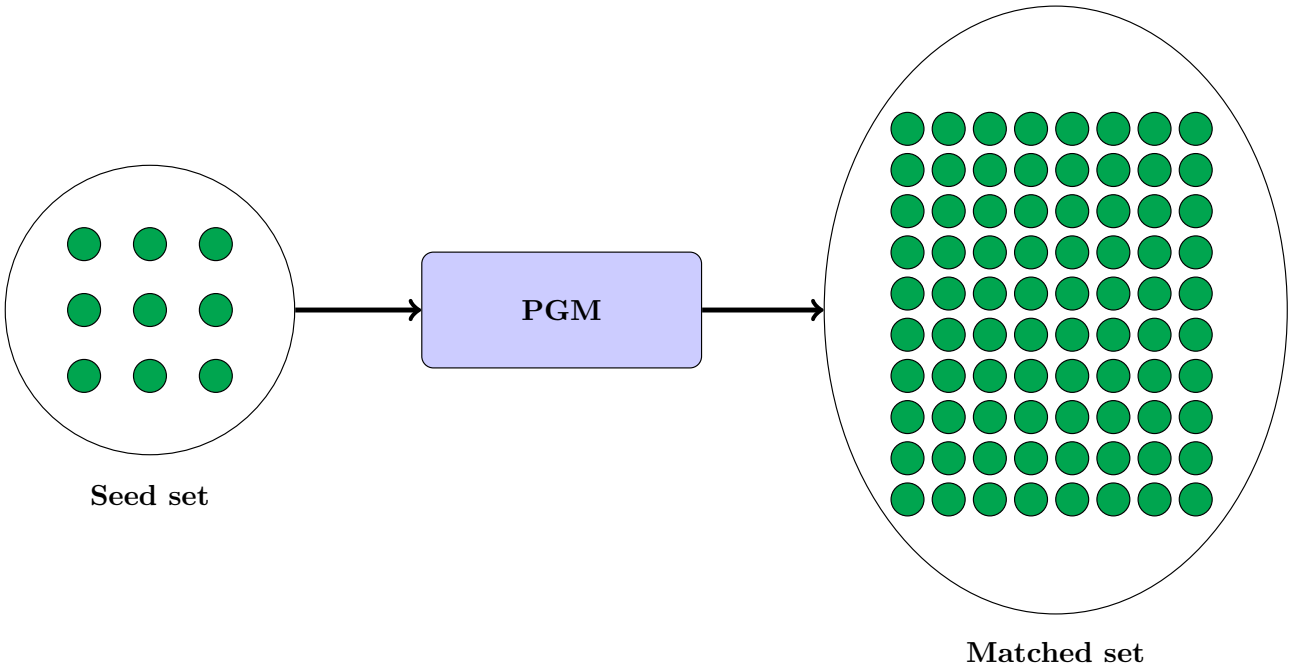
Bootstrap Percolation

- Marks are spread over the tensor product of the two graphs:
  - Green nodes are correct pairs
  - Red nodes are wrong pairs
  - Green nodes are more connected

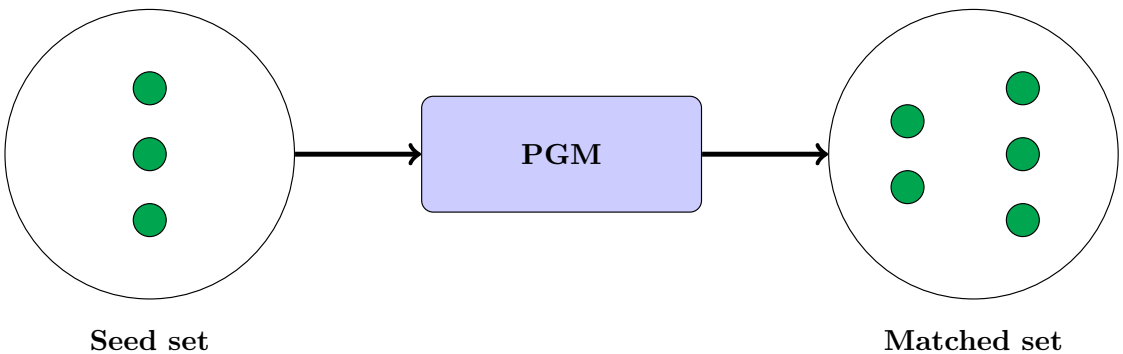


Bootstrap Percolation: Phase Transition

- Supercritical regime: **percolates** to whole network

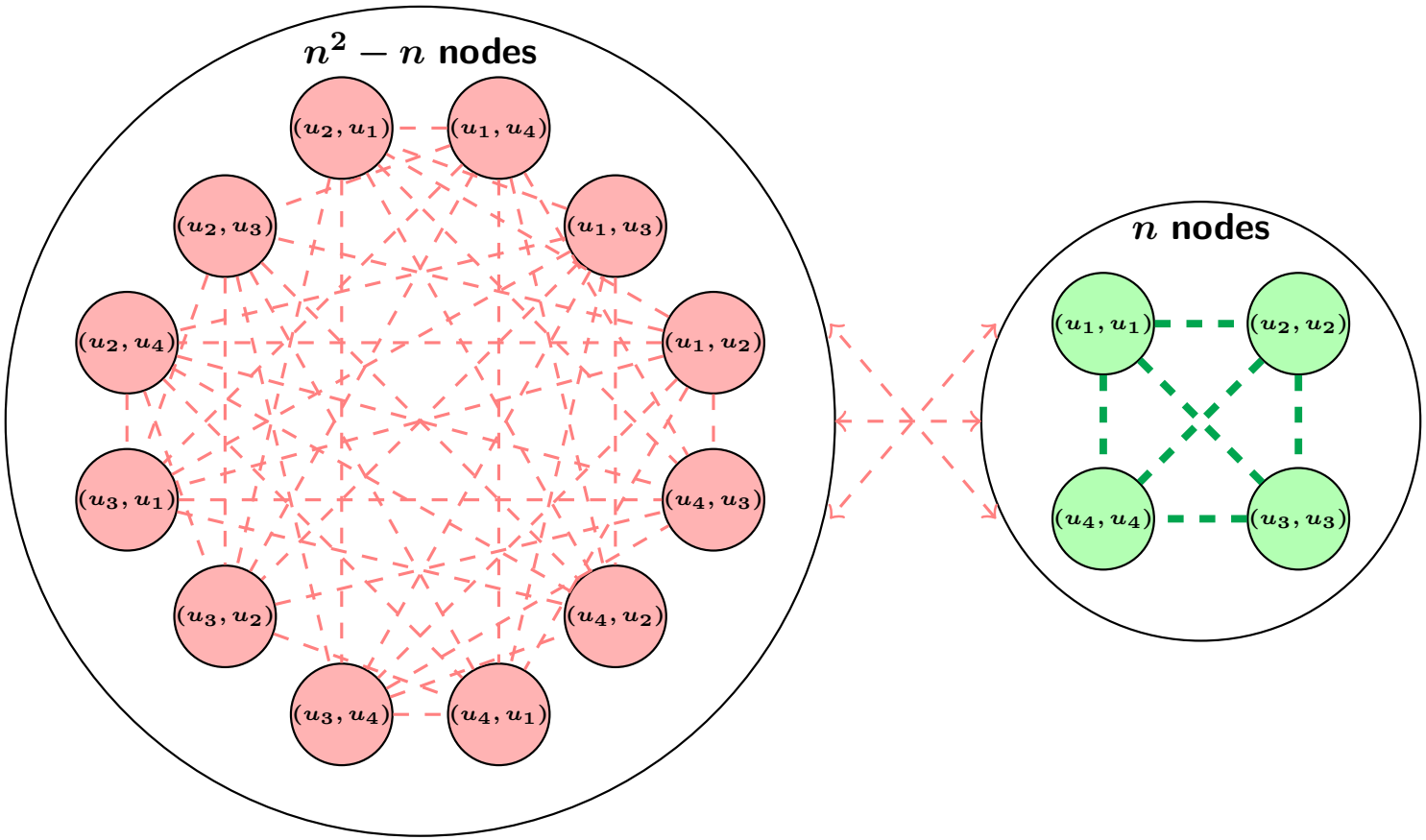


- Subcritical regime: **dies young**



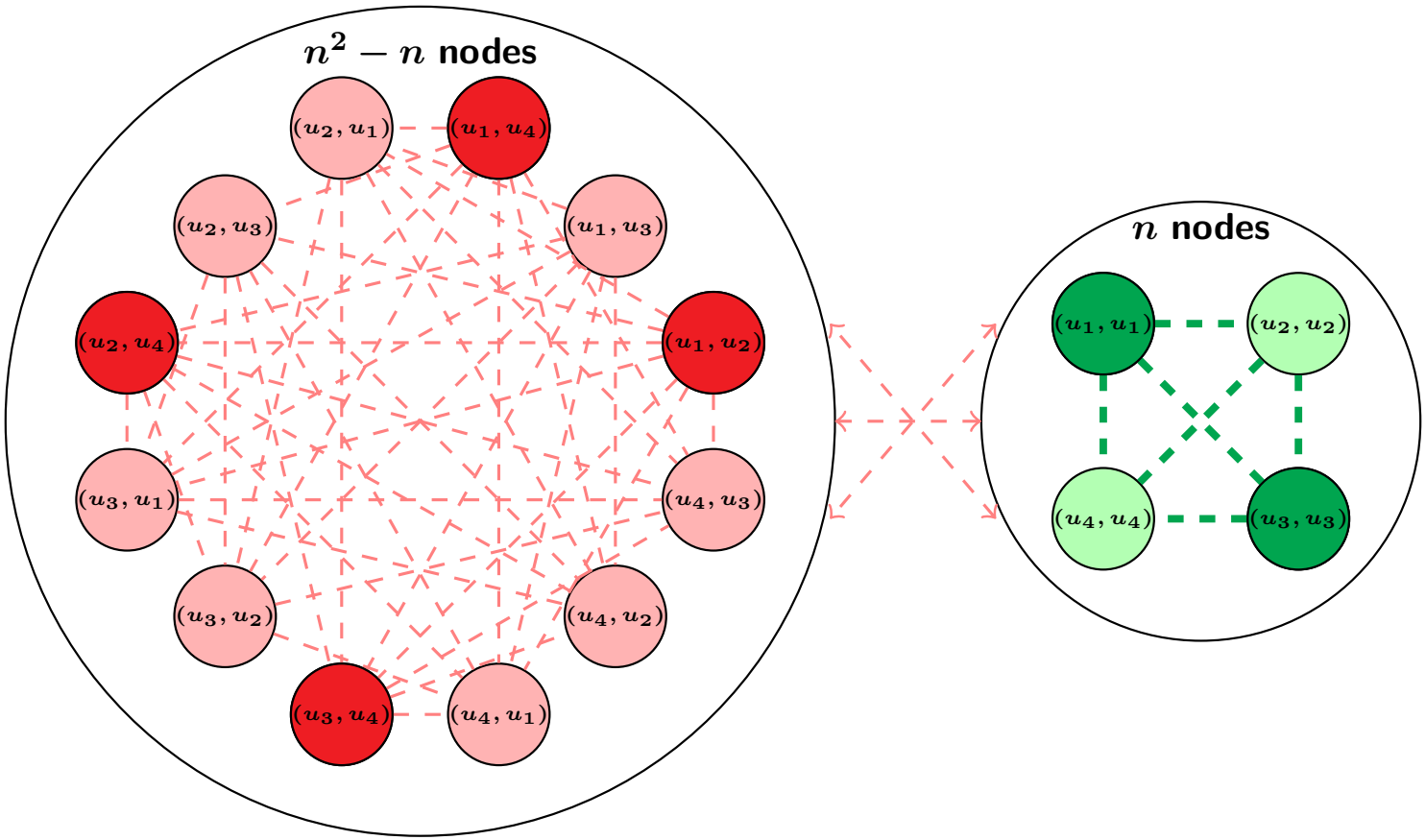
NoisySeeds Algorithms

- State-of-the-art PGM algorithms needs **many seeds**: with even moderate number of seeds percolation stuck in early steps
- Finding many seeds is **difficult** and **expensive**
- **Observation:** PGM is robust to the noise



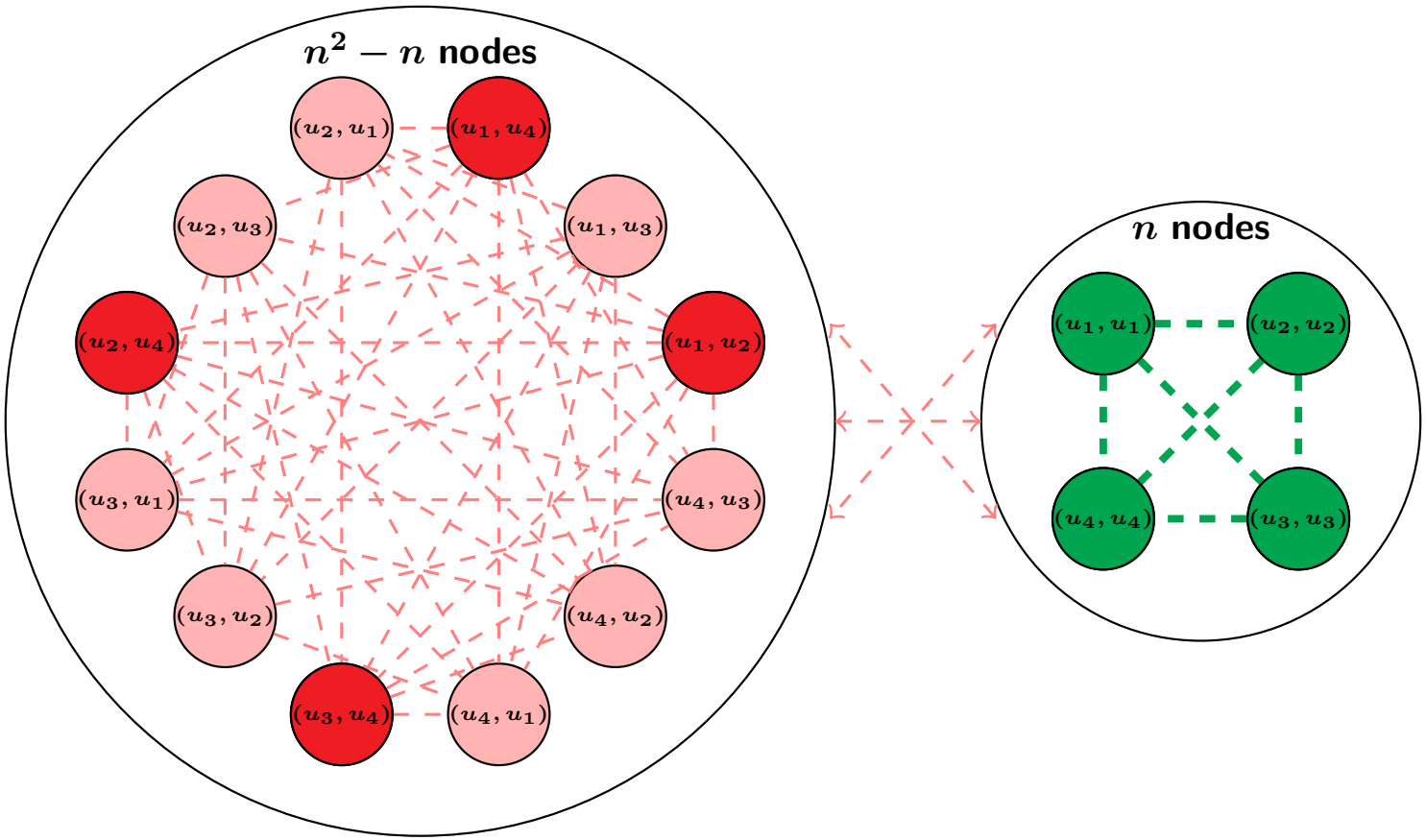
NoisySeeds Algorithms

- State-of-the-art PGM algorithms needs **many seeds**: with even moderate number of seeds percolation stuck in early steps
- Finding many seeds is **difficult** and **expensive**
- **Observation:** PGM is robust to the noise



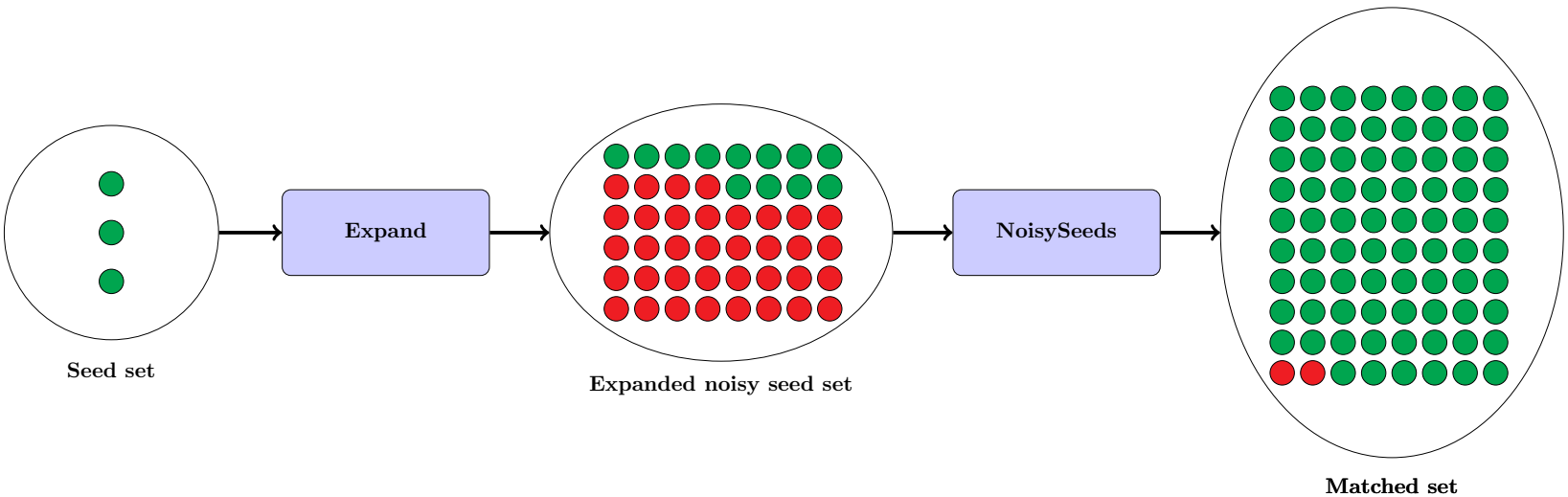
NoisySeeds Algorithms

- State-of-the-art PGM algorithms needs **many seeds**: with even moderate number of seeds percolation stuck in early steps
- Finding many seeds is **difficult** and **expensive**
- **Observation:** PGM is robust to the noise



NoisySeeds Algorithms

- Addition of **many wrong pairs** to the initial candidate set have a **negligible effect** on the performance of NoisySeeds



NoisySeeds: Performance Guarantee

Theorem (Performance Guarantee over $\mathrm{Bi}(G(n,p);t,s)$)

For $\mathrm{Bi}(G(n,p);t,s)$ with fixed s and t assume $n^{-1} \ll p \leq n^{-\frac{5}{6}-\epsilon}$, provided a seed set of

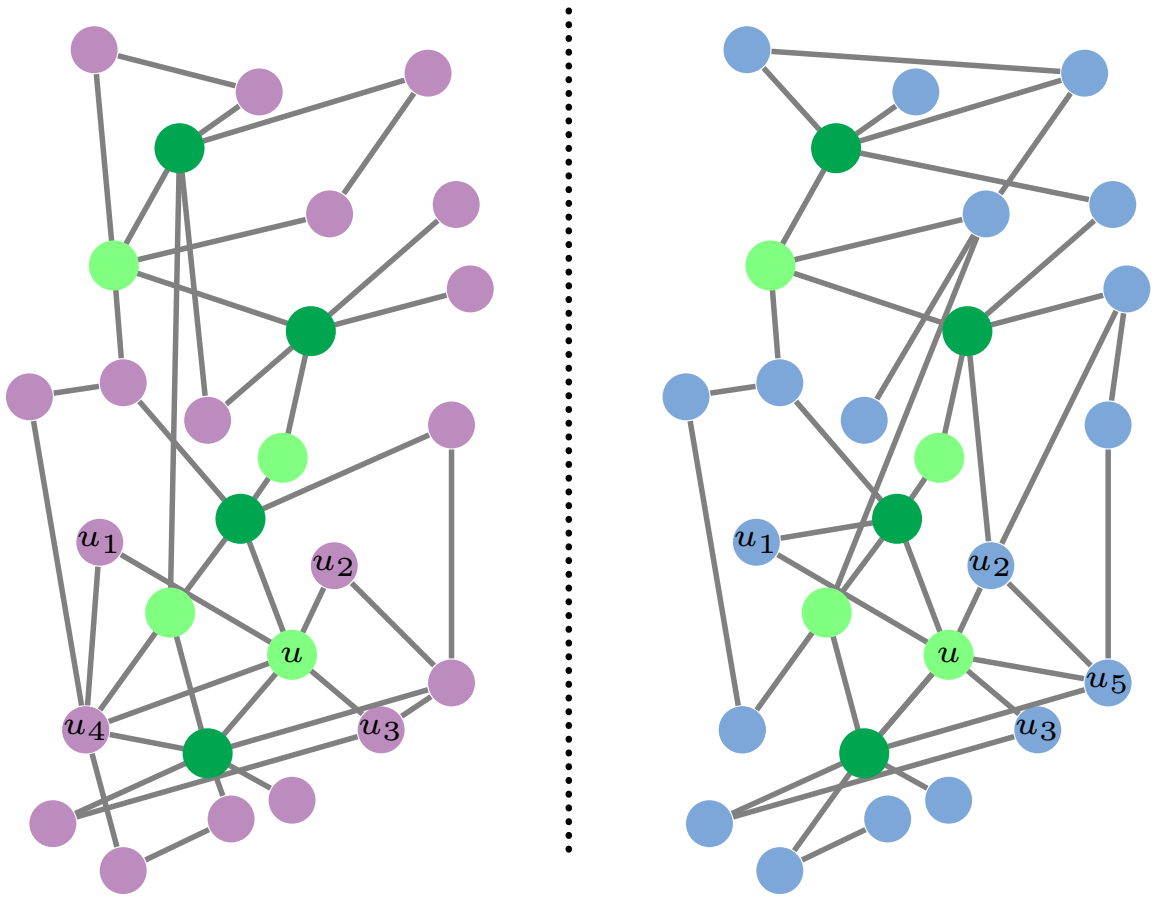
- $a_{t,s,r} = (1 - \frac{1}{r}) \left[\frac{(r-1)!}{nt^2 (ps^2)^r} \right]^{\frac{1}{r-1}}$ *correct pairs*
- $O(n)$ *wrong pairs,*

with high probability `NoisySeeds` *percolates and outputs*

- $\boldsymbol{nt^2 \pm o(n)}$ **correct pairs**
- $\boldsymbol{o(n)}$ **wrong pairs**

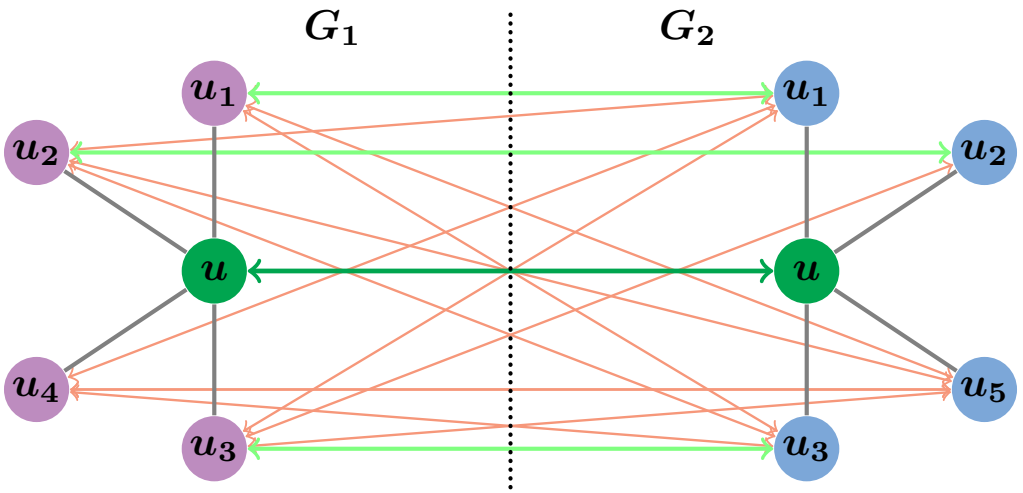
ExpandWhenStuck

- A heuristic based on the idea of robustness to noisy pairs
  - Percolation process is **stuck**
  - Node u is matched (correctly)



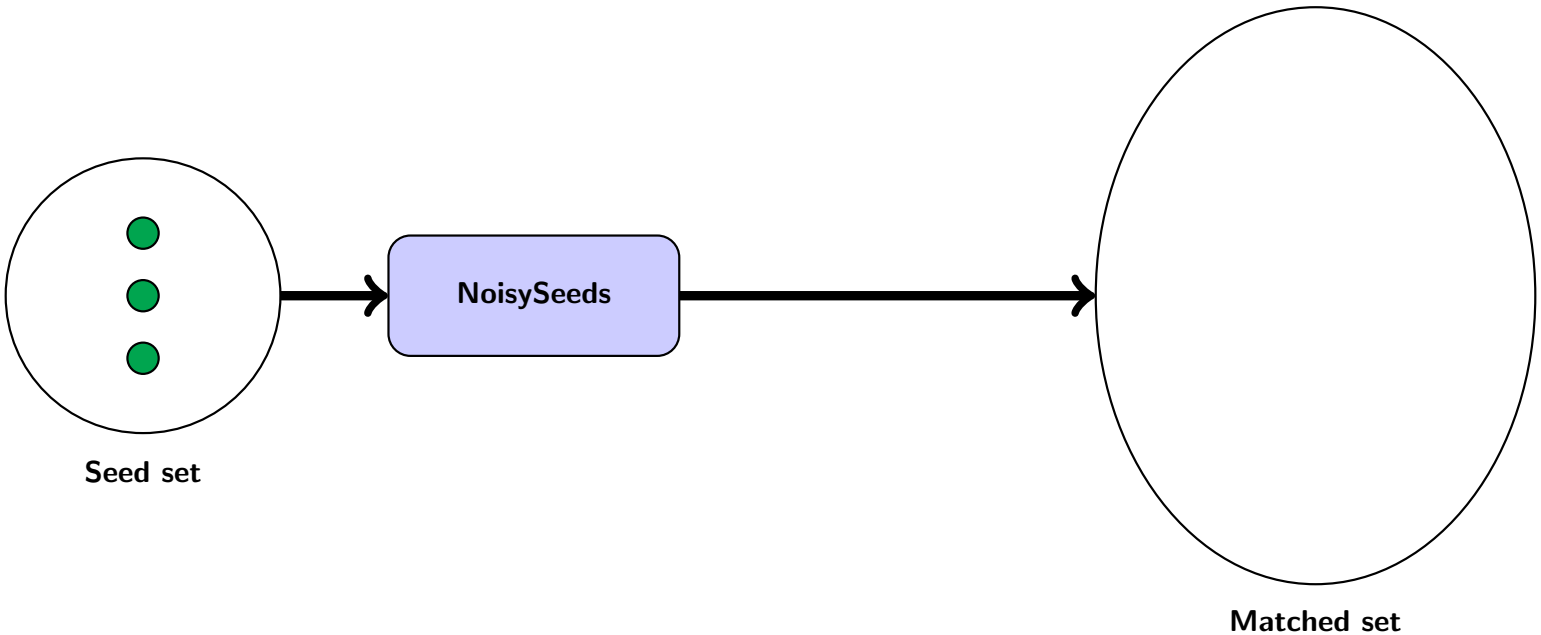
ExpandWhenStuck

- Unmatched neighbouring pairs of node-pair $[u, u]$ are **new candidate pairs**
- Two graphs are **correlated**: among new candidate pairs a small fraction is **correct**, e.g, $[u_1, u_1]$
- PGM is **robust** to the noise in candidate pairs



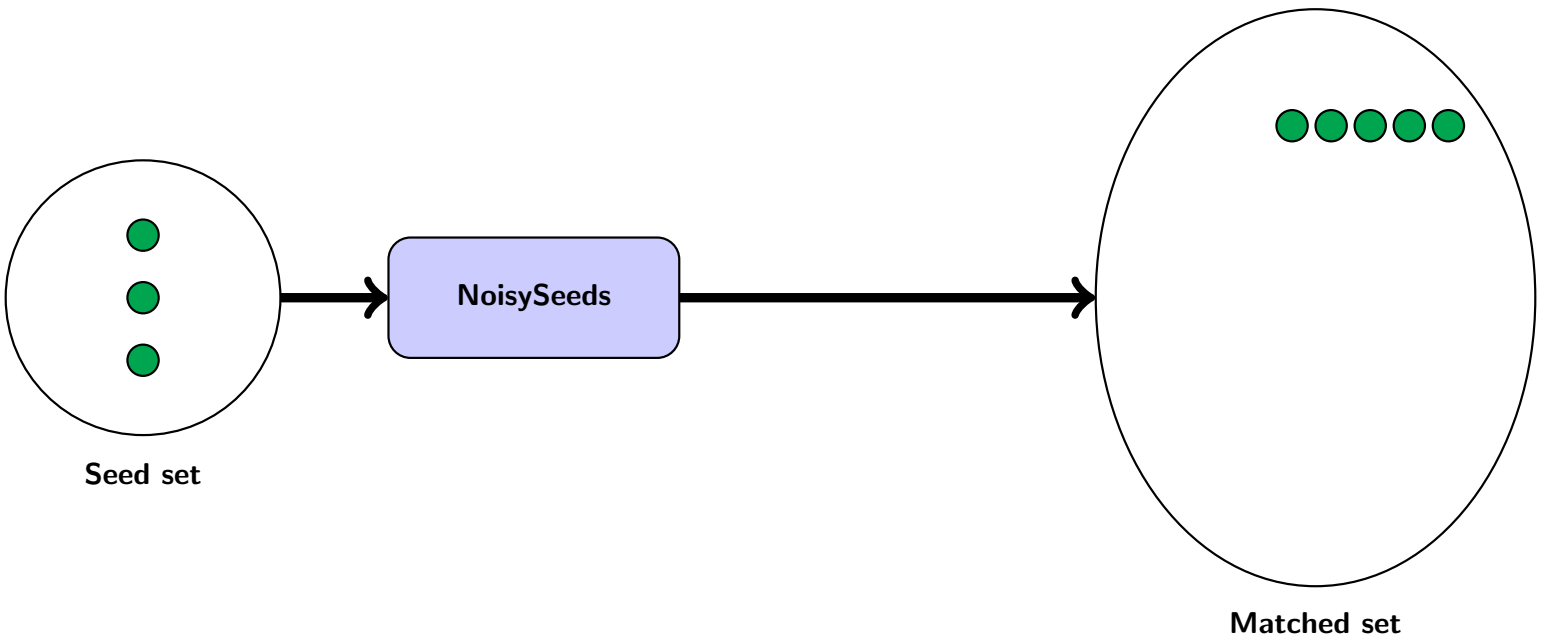
ExpandWhenStuck

- Expand the candidate pairs by many noisy pairs whenever the percolation process stuck



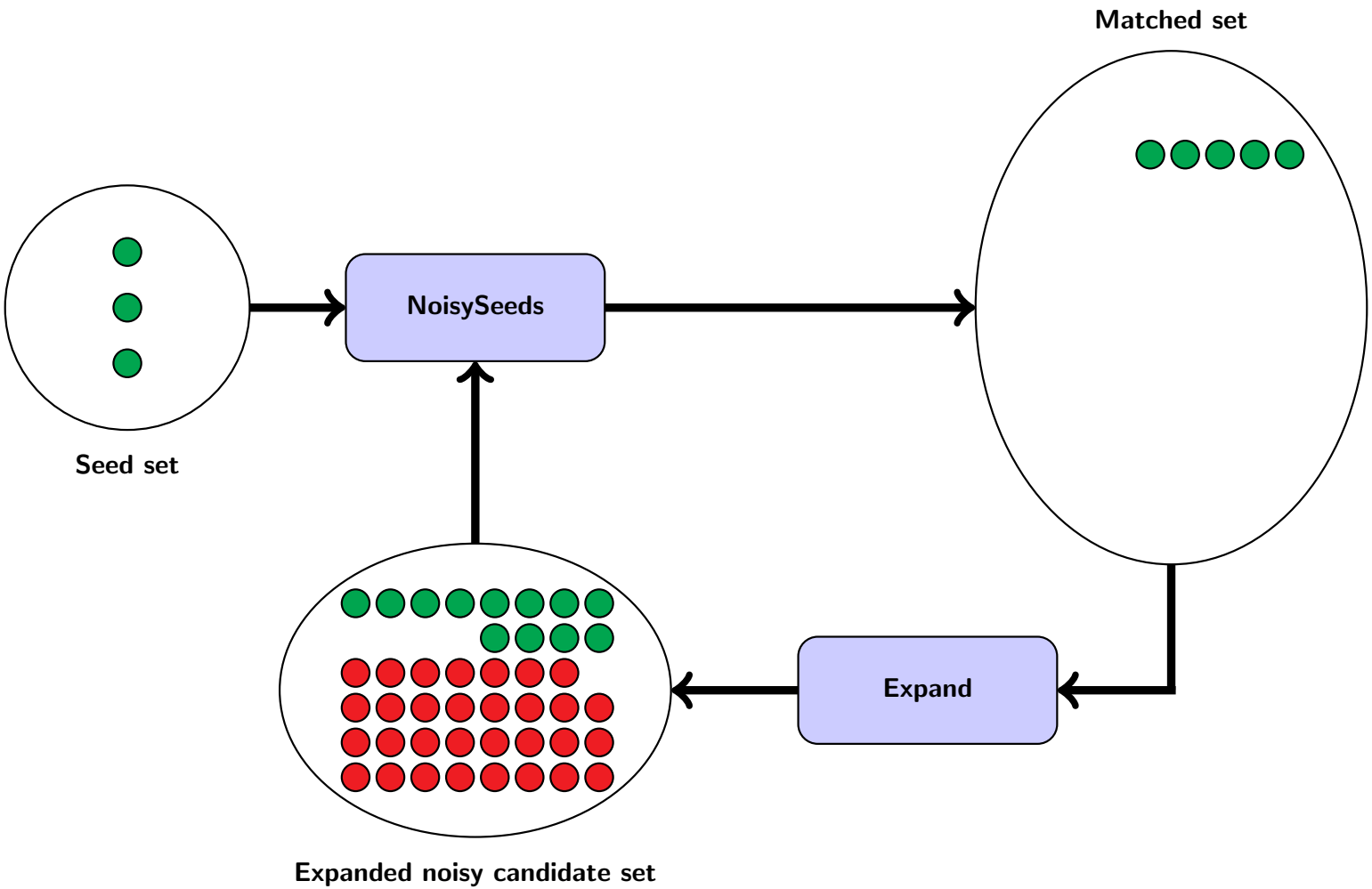
ExpandWhenStuck

- Expand the candidate pairs by many noisy pairs whenever the percolation process stuck



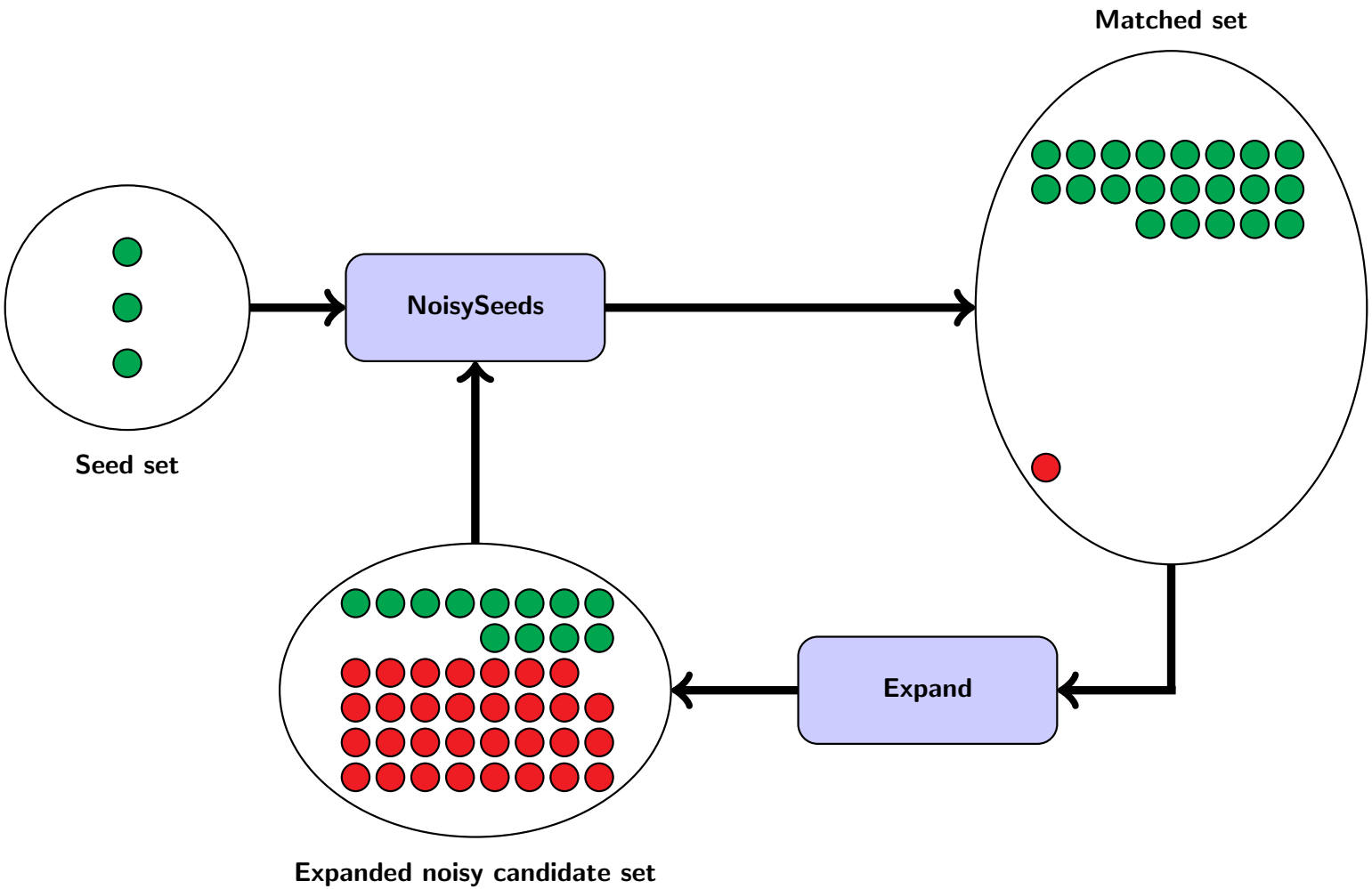
ExpandWhenStuck

- Expand the candidate pairs by many noisy pairs whenever the percolation process stuck



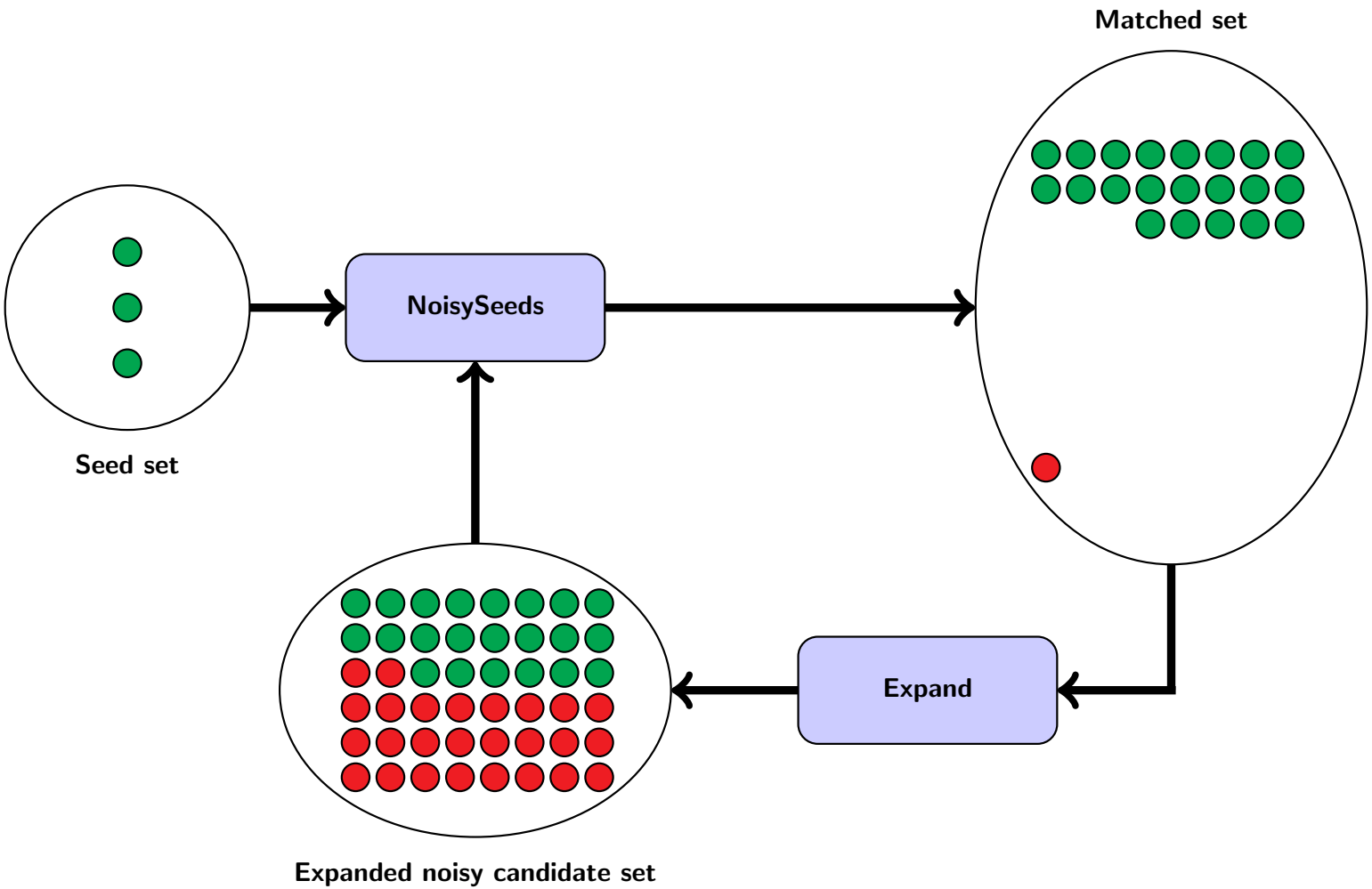
ExpandWhenStuck

- Expand the candidate pairs by many noisy pairs whenever the percolation process stuck



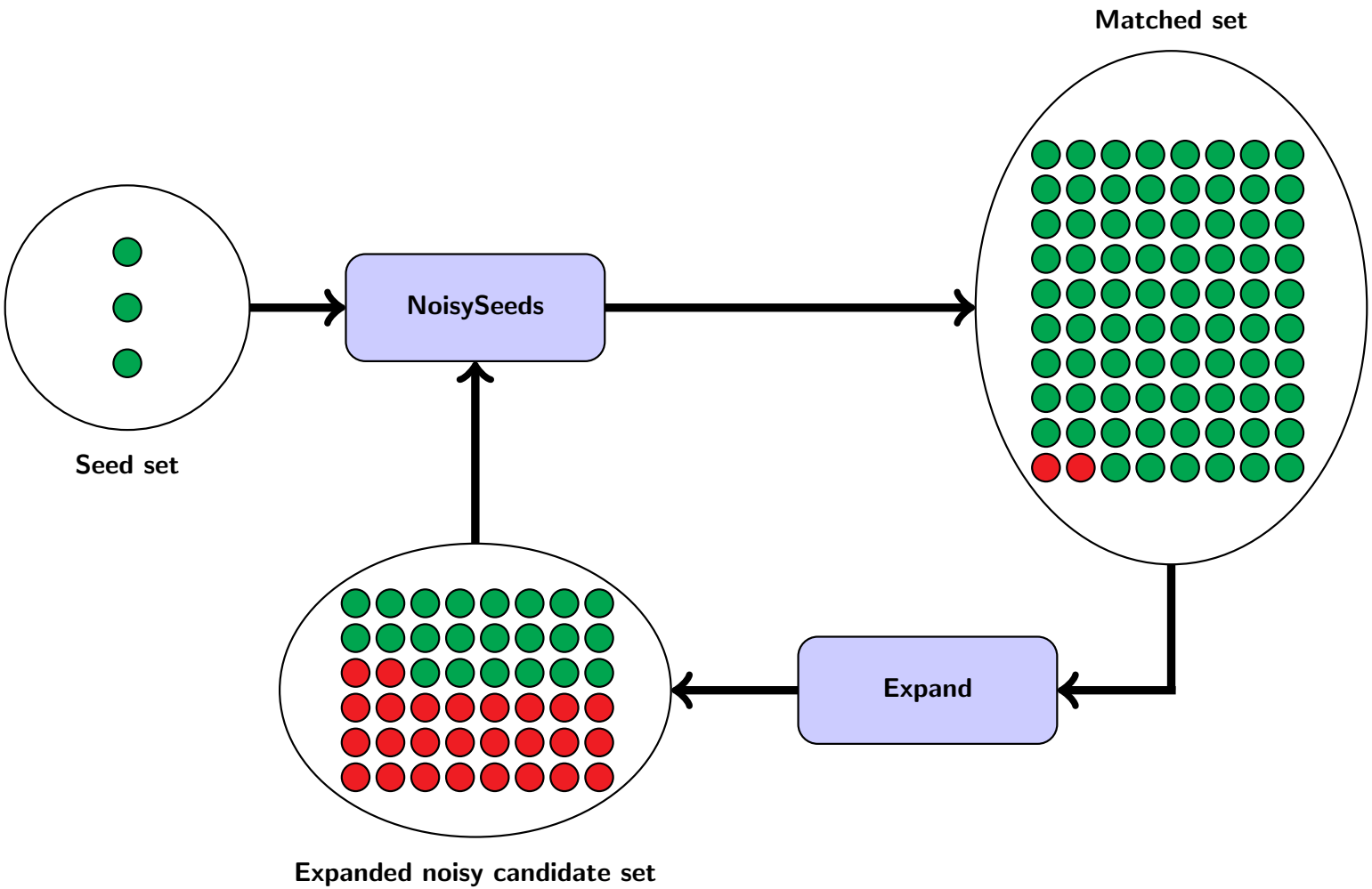
ExpandWhenStuck

- Expand the candidate pairs by many noisy pairs whenever the percolation process stuck



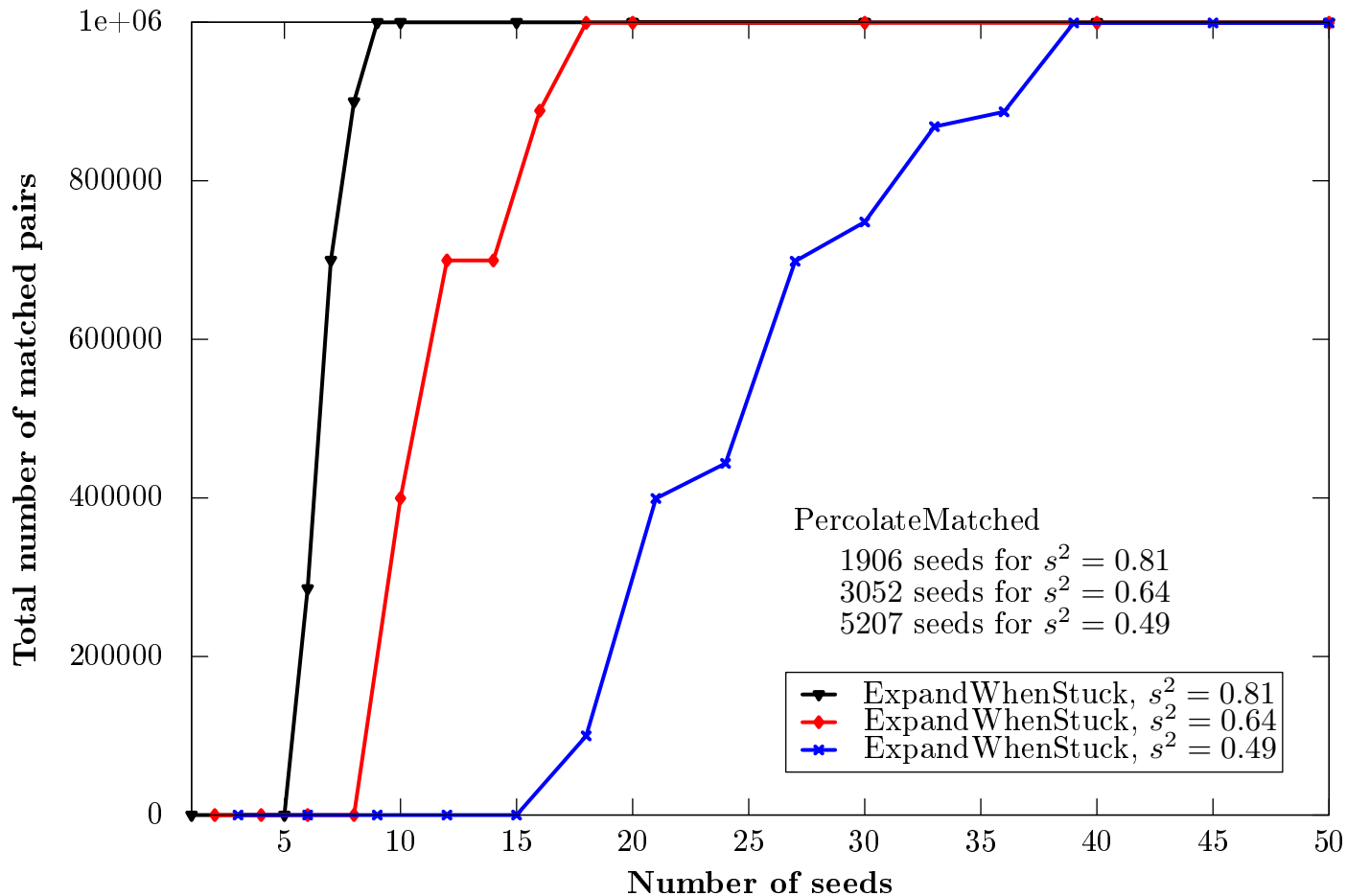
ExpandWhenStuck

- Expand the candidate pairs by many noisy pairs whenever the percolation process stuck



Experiment 1: Random Graphs

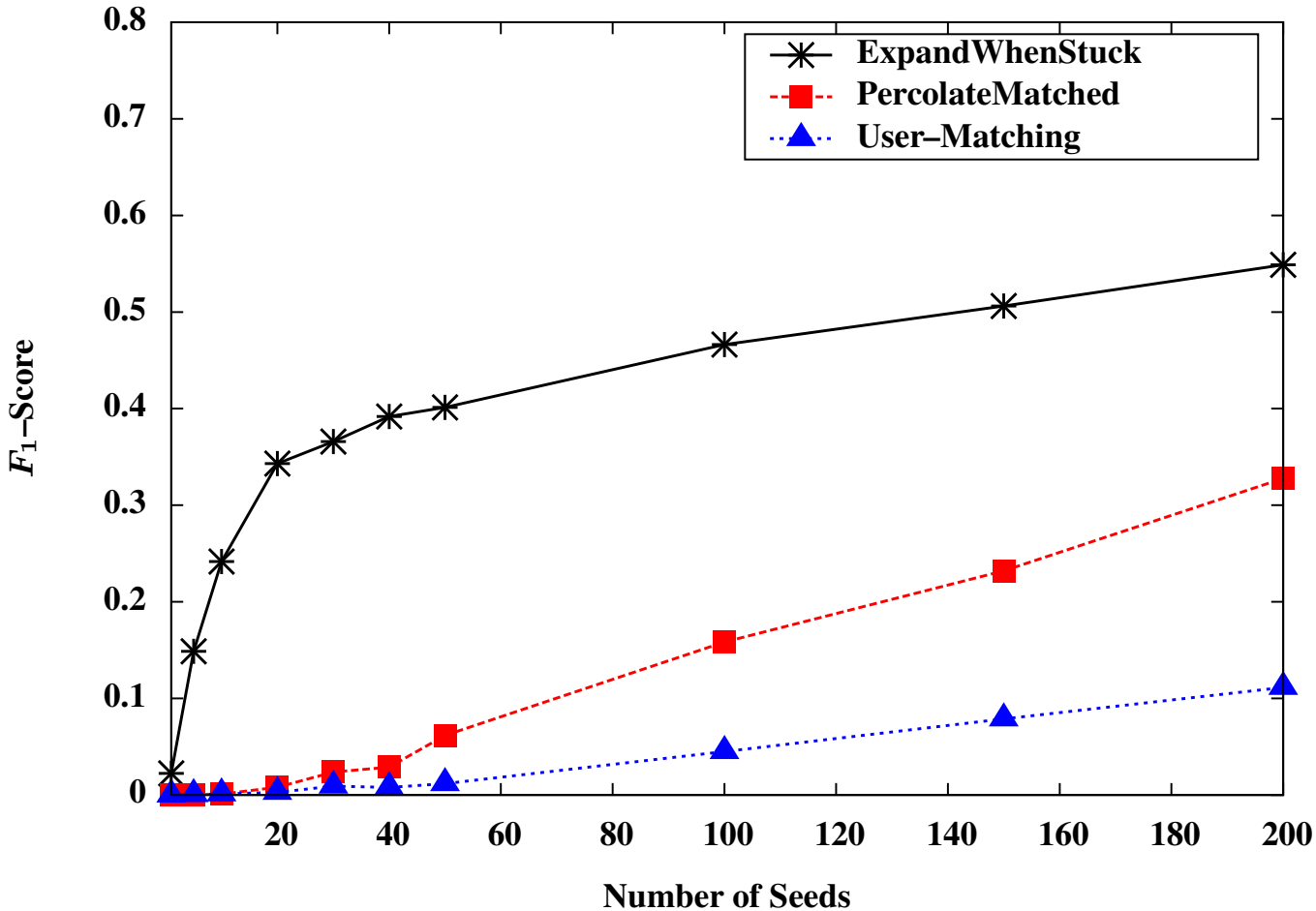
- **ExpandWhenStuck** vs. **PercolateMatche** [Yartseva and Grossglauser, 2013] over $\mathrm{Bi}(G(n,p);t,s)$ with $n = 10^6$, $p = \frac{20}{n}$ and $t^2 = 1.0$



- **238 times** improvement for $s^2 = 0.81$

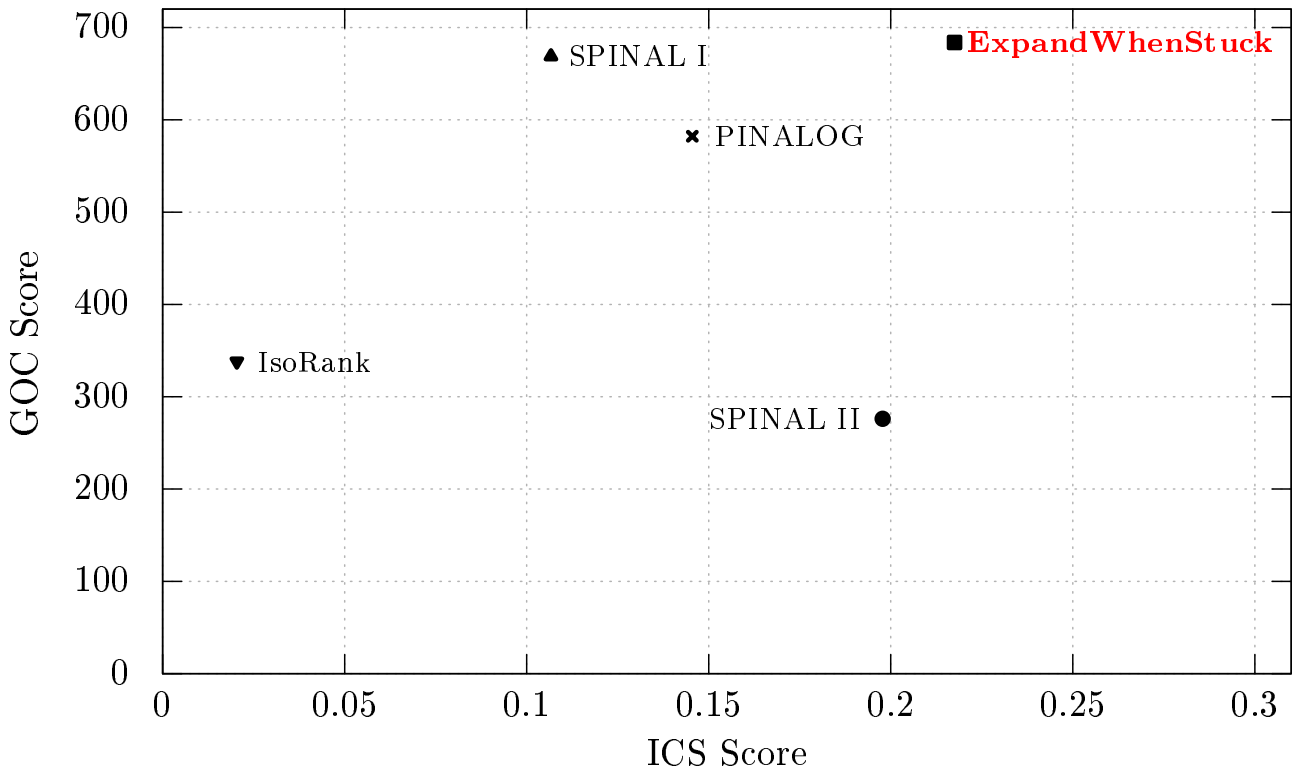
Experiment 2: Gowalla Network

- **ExpandWhenStuck** vs. **PercolateMatched**[Yartseva and Grossglauser, 2013] and **User–Matching** [Korula and Lattanzi, 2014] over Gowalla network.



Experiment 3: Network Alignment in Bioinformatics

- **ExpandWhenStuck** vs. state-of-the-art PPI network alignment algorithms*



Access: `http://proper.epfl.ch`

* E. Kazemi, S. H. Hassani, H. Pezeshgi Modarres and M. Grossglauser. "ProPer: Global Protein-Protein Interaction Network Alignment with Percolation Graph-Matching." Submitted to Bioinformatics.

Conclusion

- Graph matching has applications in many fields
- Percolation graph matching
  - ExpandWhenStuck: a fast and accurate algorithm
    - only a handful of seeds is enough for percolation
    - MapReduce implementation: a variant of ExpandWhenStuck
  - analysis of a simplified version of ExpandWhenStuck
  - a phase transition result

Mahalo!

ExpandWhenStuck

Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct pairs
Output: The set of matched pairs $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed pairs, $\mathcal{M} \leftarrow \mathcal{A}_0$;
$\mathcal{Z} \leftarrow \emptyset$ is the set of used pairs;
while $|\mathcal{A}| > 0$ **do**
    for *all pairs* $[i, j] \in \mathcal{A}$ **do**
        add the pair $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring pairs;
    while *there exists an unmatched pair with score at least* 2 **do**
        among the pairs with the highest score select the unmatched pair $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$;
        add $[i, j]$ to the set $\mathcal{M}$;
        if $[i, j] \notin \mathcal{Z}$ **then**
            add one mark to all of its neighbouring pairs and add the pair $[i, j]$ to $\mathcal{Z}$;
    $\mathcal{A} \leftarrow$ all neighbouring pairs $[i, j]$ of matched pairs $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$;
return $\mathcal{M}$;

ExpandWhenStuck

Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct pairs
Output: The set of matched pairs $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed pairs, $\mathcal{M} \leftarrow \mathcal{A}_0$; **Initial candidate set**
$\mathcal{Z} \leftarrow \emptyset$ is the set of used pairs;
while $|\mathcal{A}| > 0$ **do**
    for *all pairs* $[i, j] \in \mathcal{A}$ **do**
        add the pair $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring pairs;
    while *there exists an unmatched pair with score at least* 2 **do**
        among the pairs with the highest score select the unmatched pair $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$;
        add $[i, j]$ to the set $\mathcal{M}$;
        if $[i, j] \notin \mathcal{Z}$ **then**
            add one mark to all of its neighbouring pairs and add the pair $[i, j]$ to $\mathcal{Z}$;
    $\mathcal{A} \leftarrow$ all neighbouring pairs $[i, j]$ of matched pairs $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$;
return $\mathcal{M}$;

ExpandWhenStuck

Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct pairs
Output: The set of matched pairs $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed pairs, $\mathcal{M} \leftarrow \mathcal{A}_0$;
$\mathcal{Z} \leftarrow \emptyset$ is the set of used pairs;
while $|\mathcal{A}| > 0$ **do**

Spread marks from the candidate set

for *all pairs* $[i, j] \in \mathcal{A}$ **do**
add the pair $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring pairs;

while *there exists an unmatched pair with score at least* 2 **do**
among the pairs with the highest score select the unmatched pair $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$;
add $[i, j]$ to the set $\mathcal{M}$;
if $[i, j] \notin \mathcal{Z}$ **then**
add one mark to all of its neighbouring pairs and add the pair $[i, j]$ to $\mathcal{Z}$;

$\mathcal{A} \leftarrow$ all neighbouring pairs $[i, j]$ of matched pairs $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$;

return $\mathcal{M}$;

ExpandWhenStuck

Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct pairs
Output: The set of matched pairs $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed pairs, $\mathcal{M} \leftarrow \mathcal{A}_0$;
$\mathcal{Z} \leftarrow \emptyset$ is the set of used pairs;
while $|\mathcal{A}| > 0$ **do**
    for *all pairs* $[i, j] \in \mathcal{A}$ **do**
        add the pair $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring pairs;
    while *there exists an unmatched pair with score at least* 2 **do**
        among the pairs with the highest score select the unmatched pair $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$;
        add $[i, j]$ to the set $\mathcal{M}$;
        if $[i, j] \notin \mathcal{Z}$ **then**
            add one mark to all of its neighbouring pairs and add the pair $[i, j]$ to $\mathcal{Z}$;
    $\mathcal{A} \leftarrow$ all neighbouring pairs $[i, j]$ of matched pairs $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$;
return $\mathcal{M}$;

Percolation Graph Matching

ExpandWhenStuck

Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct pairs
Output: The set of matched pairs $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed pairs, $\mathcal{M} \leftarrow \mathcal{A}_0$;
$\mathcal{Z} \leftarrow \emptyset$ is the set of used pairs;
while $|\mathcal{A}| > 0$ **do**
    for *all pairs* $[i, j] \in \mathcal{A}$ **do**
        add the pair $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring pairs;
    while *there exists an unmatched pair with score at least* 2 **do**
        among the pairs with the highest score select the unmatched pair $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$;
        add $[i, j]$ to the set $\mathcal{M}$;
        if $[i, j] \notin \mathcal{Z}$ **then**
            add one mark to all of its neighbouring pairs and add the pair $[i, j]$ to $\mathcal{Z}$;
    oops! Stuck!:-(
    $\mathcal{A} \leftarrow$ all neighbouring pairs $[i, j]$ of matched pairs $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$;
return $\mathcal{M}$;

ExpandWhenStuck

Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct pairs
Output: The set of matched pairs $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed pairs, $\mathcal{M} \leftarrow \mathcal{A}_0$;
$\mathcal{Z} \leftarrow \emptyset$ is the set of used pairs;
while $|\mathcal{A}| > 0$ **do**
    for *all pairs* $[i, j] \in \mathcal{A}$ **do**
        add the pair $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring pairs;
    while *there exists an unmatched pair with score at least* 2 **do**
        among the pairs with the highest score select the unmatched pair $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$;
        add $[i, j]$ to the set $\mathcal{M}$;
        if $[i, j] \notin \mathcal{Z}$ **then**
            add one mark to all of its neighbouring pairs and add the pair $[i, j]$ to $\mathcal{Z}$;
    $\mathcal{A} \leftarrow$ all neighbouring pairs $[i, j]$ of matched pairs $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$; **Expand When Stuck!:-)**
return $\mathcal{M}$;

ExpandWhenStuck

Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct pairs
Output: The set of matched pairs $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed pairs, $\mathcal{M} \leftarrow \mathcal{A}_0$;
$\mathcal{Z} \leftarrow \emptyset$ is the set of used pairs;
while $|\mathcal{A}| > 0$ **do**

Fuel the percolation process!:-)

    for *all pairs* $[i, j] \in \mathcal{A}$ **do**
        add the pair $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring pairs;
    while *there exists an unmatched pair with score at least* 2 **do**
        among the pairs with the highest score select the unmatched pair $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$;
        add $[i, j]$ to the set $\mathcal{M}$;
        if $[i, j] \notin \mathcal{Z}$ **then**
            add one mark to all of its neighbouring pairs and add the pair $[i, j]$ to $\mathcal{Z}$;
    $\mathcal{A} \leftarrow$ all neighbouring pairs $[i, j]$ of matched pairs $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$;

Expand When Stuck!:-)

return $\mathcal{M}$;

ExpandWhenStuck

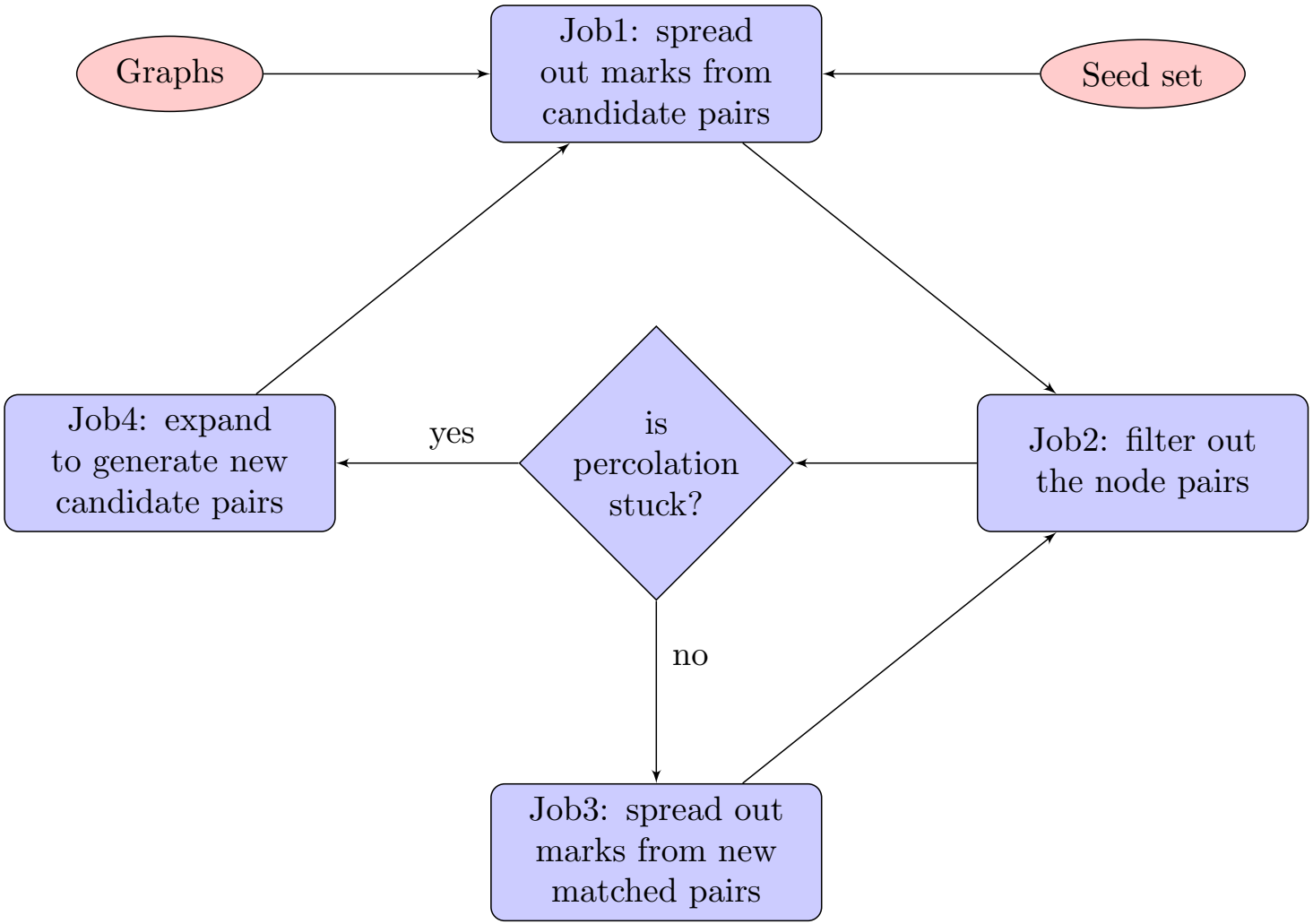
Input: $G_1(V_1, E_1), G_2(V_2, E_2)$, seed set $\mathcal{A}_0$ of correct pairs
Output: The set of matched pairs $\mathcal{M}$

$\mathcal{A} \leftarrow \mathcal{A}_0$ is the initial set of seed pairs, $\mathcal{M} \leftarrow \mathcal{A}_0$;
$\mathcal{Z} \leftarrow \emptyset$ is the set of used pairs;
while $|\mathcal{A}| > 0$ **do**
    for *all pairs* $[i, j] \in \mathcal{A}$ **do**
        add the pair $[i, j]$ to $\mathcal{Z}$ and add one mark to all of its neighbouring pairs;
    while *there exists an unmatched pair with score at least* 2 **do**
        among the pairs with the highest score select the unmatched pair $[i, j]$ with the minimum $|d_{1,i} - d_{2,j}|$;
        add $[i, j]$ to the set $\mathcal{M}$;
        if $[i, j] \notin \mathcal{Z}$ **then**
            add one mark to all of its neighbouring pairs and add the pair $[i, j]$ to $\mathcal{Z}$;
    $\mathcal{A} \leftarrow$ all neighbouring pairs $[i, j]$ of matched pairs $\mathcal{M}$ s.t. $[i, j] \notin \mathcal{Z}$, $i \notin V_1(\mathcal{M})$ and $j \notin V_2(\mathcal{M})$;
return $\mathcal{M}$;

Percolation Graph Matching

ExpandWhenStuck: MapReduce Implementation

- A parallelized variant of ExpandWhenStuck



Sketch of the Proof

1. In the beginning of NoisySeeds, all the pairs in the noisy seed set spread out marks to their neighbouring pairs.

Lemma

At the completion time of initial phase, the expected number of wrongly matched pairs is $o(a_{t,s,r})$.

2. Percolation graph matching process continues in at most $\min(|V_1|, |V_2|)$ more steps.

Lemma

When PGM stops, the expected number of wrongly matched pairs is $o(a_{t,s,r})$.

Sketch of the Proof (Continued)

3. Using Markov's inequality, we find an upper bound for the number of wrongly matched pairs.

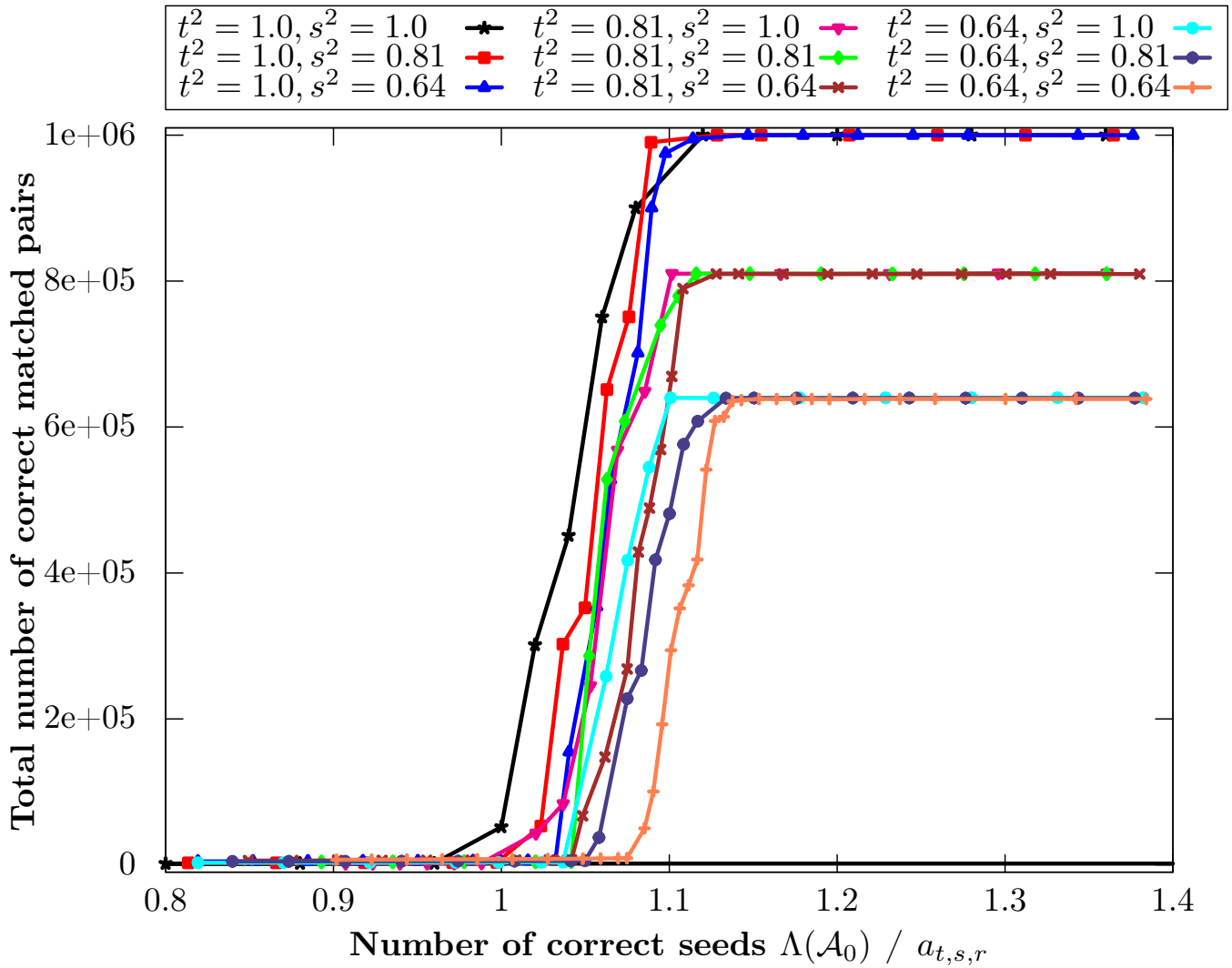
Lemma

With high probability, the total number of wrongly matched pairs at any time step is $o(a_{t,s,r})$.

Idea: To apply the theory of bootstrap percolation in $G(nt^2, ps^2)$ [Janson et al, 2010] over $\mathrm{Bi}(G(n,p);t,s)$ graphs.

Experiment 4: Phase Transition

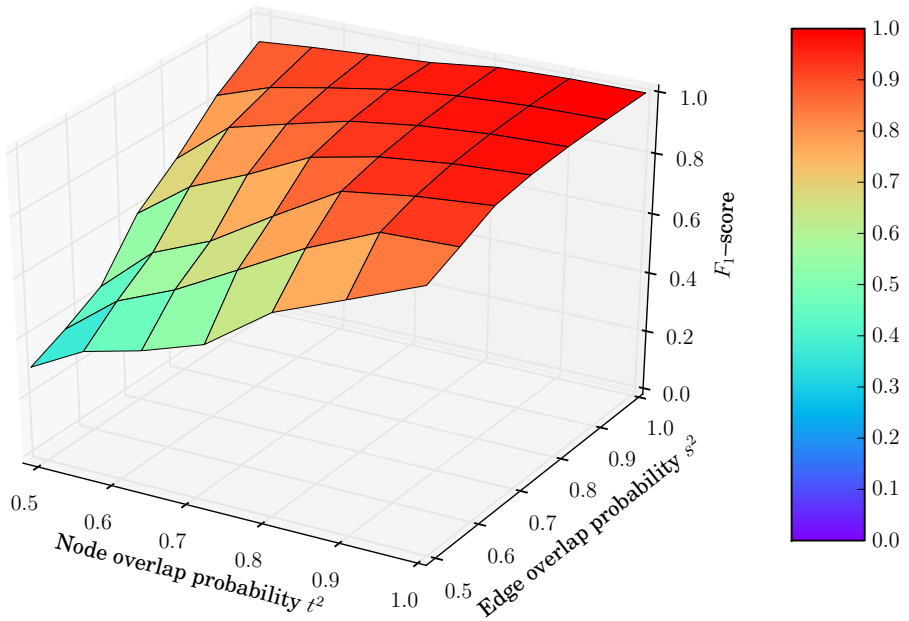
- $\mathrm{Bi}(G(n,p);t,s)$ with $n = 10^6$ and $p = \frac{20}{n}$



Experiment 5: Slashdot Network

- **ExpandWhenStuck** over $\mathrm{Bi}(\text{Slashdot network}; t, s)$ when the number of seeds is 10.
- Combine precision and recall in one metric:

$$F_1\text{–score} = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$



For results regarding random graph models such as Barabási-Albert, Chung-Lu and Erdös–Rényi please refer to the paper.

Experiment 6: EPFL e-mail Network

- **ExpandWhenStuck** vs. **PercolateMatched** over EPFL e-mail exchange network.