Metaheuristic Optimization Methods: Variable Neighborhood Search

Variable Neighborhood Search

- Variable neighborhood search (VNS) is a metaheuristic, proposed by Mladenovic and Hansen, in 1997.
- It's based on a simple principle: **a systematic change of neighborhood within the search.**
- Many extensions have been made, to allow solving large problem instances.

VNS is based on the following central observations:

1. A local minimum w.r.t. one neighborhood structure is not necessarily locally minimal w.r.t. another neighborhood structure
2. A global minimum is locally optimal w.r.t. *all* neighborhood structures
3. For many problems, local minima with respect to one or several neighborhoods are relatively close to each other

N_k, (for $k = 1,2,\ldots, k_{max}$)

- a finite set of pre-determined neighborhood structures
- $N_k(x)$ is the set of solutions in the k^{th} neighborhood of x

Basic principle: change the neighborhood during the search

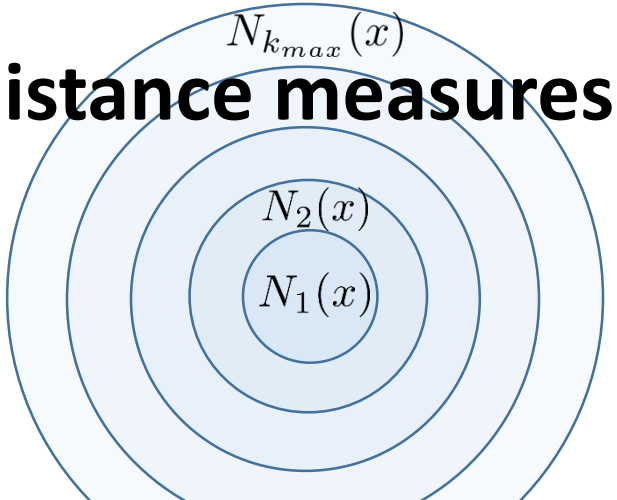
Some neighborhood structures:

- **For many problems certain common different neighborhood structures already exists**

  E.g., Vehicle Routing Problem: Cross, Swap, Exchange
  (I uploaded a few papers relating to VNS+VRP in D2L)

- **Find neighborhoods that depend on a parameter**
  - k-Opt (k=2,3,…)
  - Flip-neighborhoods can be extended to double-flip, triple-flip, etc…

- **Some neighborhoods are associated with distance measures: can increase the distance**
  - $$N_1(x) \subset N_2(x) \ldots \subset N_{k_{max}}(x)$$



VNS Versions...

Variable Neighborhood Descent

- VND method is obtained if change of neighborhood is performed determinstically.

Reduced Variable Neighborhood Search

- RVNS method is obtained if random points are selected from $N_k(x)$, without being followed by descent.

Basic Variable Neighborhood Search

- has both deterministic and stochastic changes of neighborhood.

Variable Neighborhood Descent

1: input: starting solution, s_0
2: input: neighborhood operators, $\{N_k\}$, $k = 1, \ldots, k_{max}$
3: input: evaluation function, f
4: $current \Leftarrow s_0$
5: $k \Leftarrow 1$;
6: **while** $k \leq k_{max}$ **do**
7: $s \Leftarrow$ the best neighbor in $N_k(current)$
8: **if** $f(s) < f(current)$ **then**
9: $current \Leftarrow s$
10: $k \Leftarrow 1$
11: **else**
12: $k \Leftarrow k + 1$
13: **end if**
14: **end while**

Variable Neighborhood Descent

- The final solution is locally optimal with respect to all neighborhoods, N_1, N_2,…, $N_{k\text{-}max}$
- "First Improvement" could be used instead of "Best Improvement"
- Typically, neighborhoods are ordered from smallest to largest

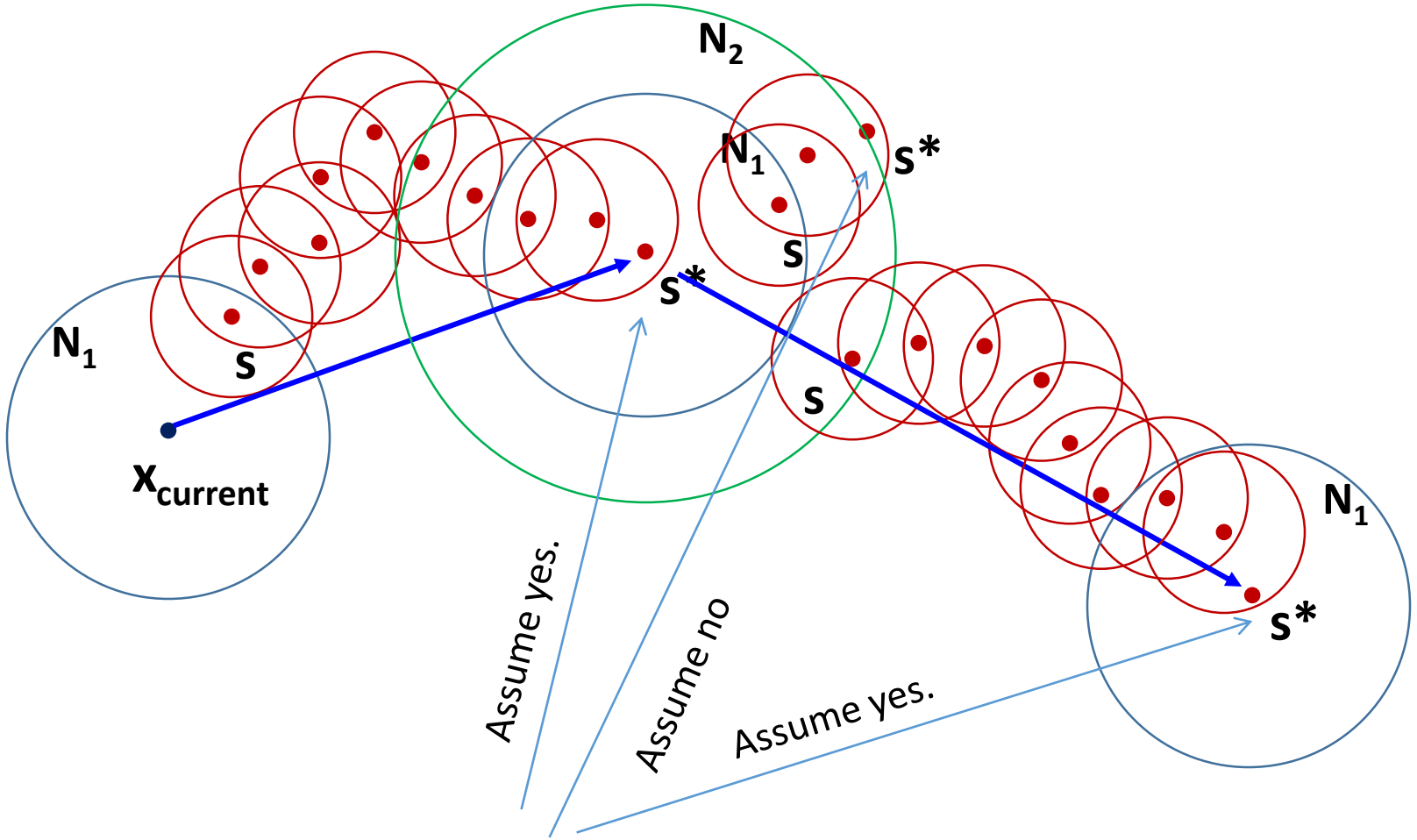
~~Variable Neighborhood Descent~~ **Reduced VNS**

1: input: starting solution, s_0
2: input: neighborhood operators, $\{N_k\}$, $k = 1, \ldots, k_{max}$
3: input: evaluation function, f
4: $current \Leftarrow s_0$
5: $k \Leftarrow 1$;
6: **while** $k \leq k_{max}$ **do**
7: $s \Leftarrow$ ~~the best~~ **random** neighbor in $N_k(current)$ ← *this is called "shaking"*
8: **if** $f(s) < f(current)$ **then**
9: $current \Leftarrow s$
10: $k \Leftarrow 1$
11: **else**
12: $k \Leftarrow k + 1$
13: **end if**
14: **end while**

Stochastic search

Basic Variable Neighborhood Search

1: input: starting solution, s_0
2: input: neighborhood operators, $\{N_k\}$, $k = 1, \ldots, k_{max}$
3: input: Local Search procedure LS, using a different neighborhood operator
4: input: evaluation function, f
5: $current \Leftarrow s_0$
6: **while** stopping criterion not met **do**
7: $k \Leftarrow 1$;
8: **while** $k \leq k_{max}$ **do**
9: $s \Leftarrow$ a random solution in $N_k(current)$ ← "shaking"
10: $s* \Leftarrow LS(s)$ ← "local search for local optima"
11: **if** $f(s*) < f(current)$ **then**
12: $current \Leftarrow s*$
13: $k \Leftarrow 1$
14: **else**
15: $k \Leftarrow k + 1$
16: **end if**
17: **end while**
18: **end while**



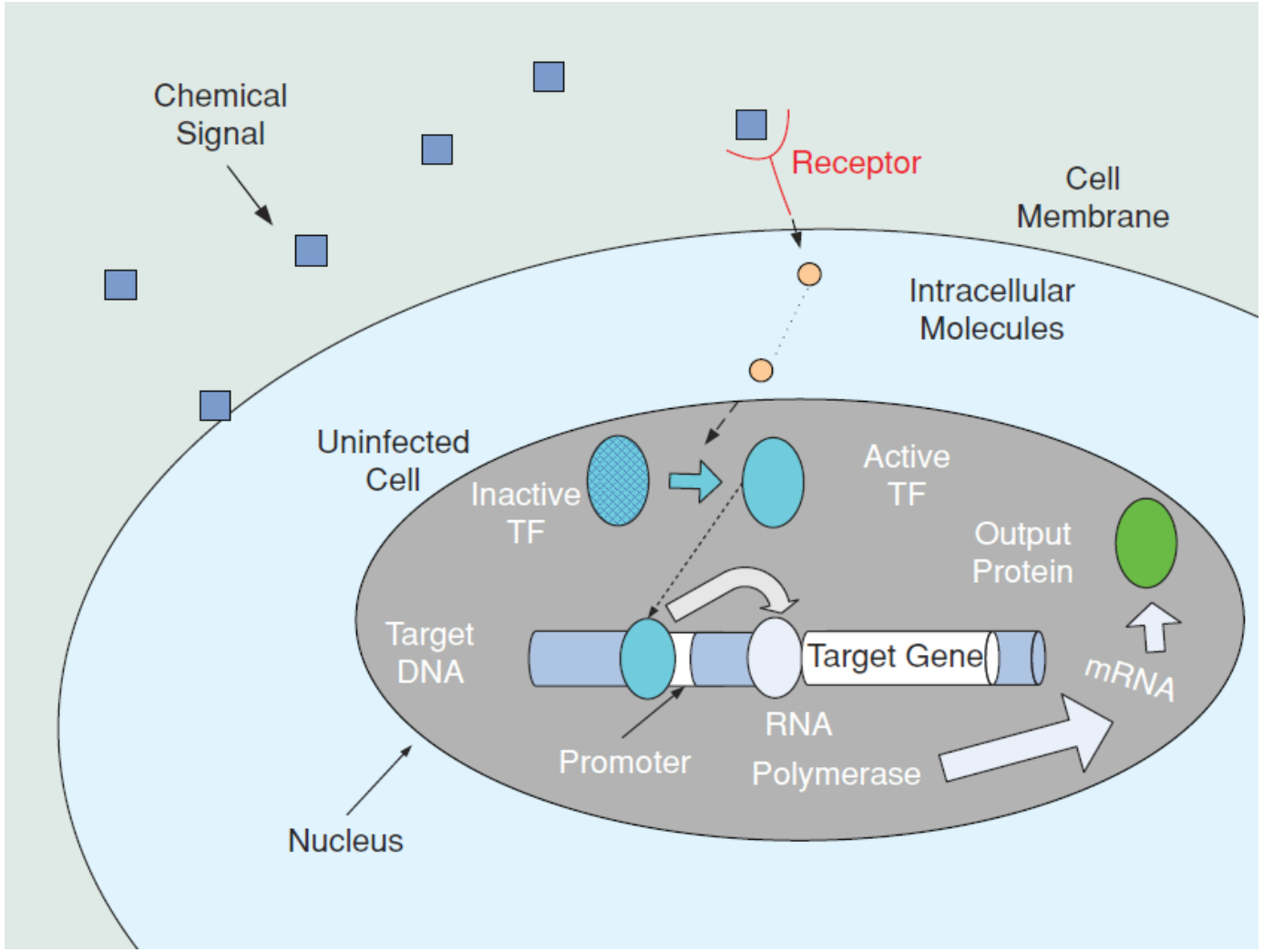
Is $f(s^*) < f(x_{current})$?

Conclusions VNS

- Based on simple principles
- Easy to understand
- Basic versions are easy to implement
- Robust
- Highly effective
- Example application:
  - C. Avanthay, A. Hertz and N. Zufferey, A variable neighborhood search for graph coloring, *European Journal of Operation Research* 151, pp. 379-388, 2003.
  - http://mat.gsia.cmu.edu/COLOR2
  - They designed 12 different large neighborhoods.
  - Outperformed Tabu Search – our next method to discuss!

VNS Application: Reverse Engineering Gene Regulatory Network

(L. Goodwin, C. Nicholson, C. Clark)

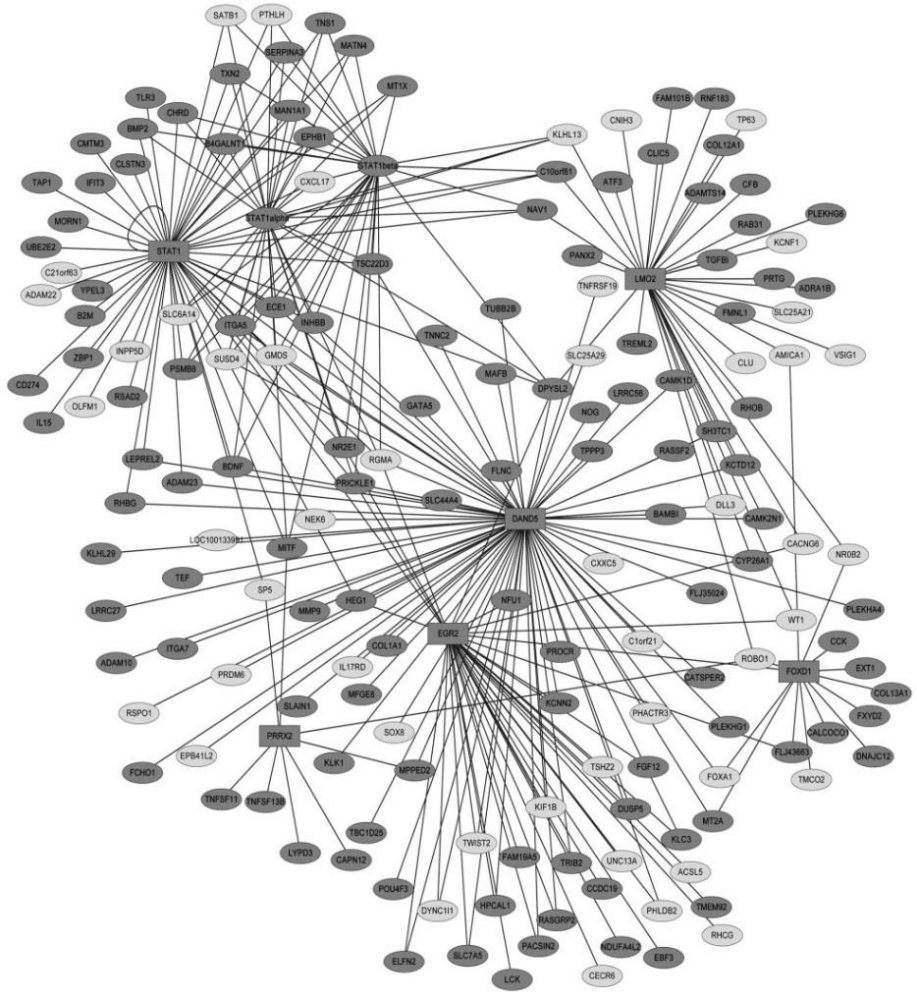


Reverse Engineering

- Computational systems biology
- Microarray data

	Exp_1	Exp_2	$\cdots$	Exp_m
$Gene_1$	0.12	-0.3	$\cdots$	0.01
$Gene_2$	0.50	0.41		
$\vdots$	$\vdots$		$\ddots$	
$Gene_n$	-0.02	-0.07		

- Inference Algorithms
  - Bayesian Networks
  - Search Heuristics



(Filkov, 2005; Huang et al., 2009)

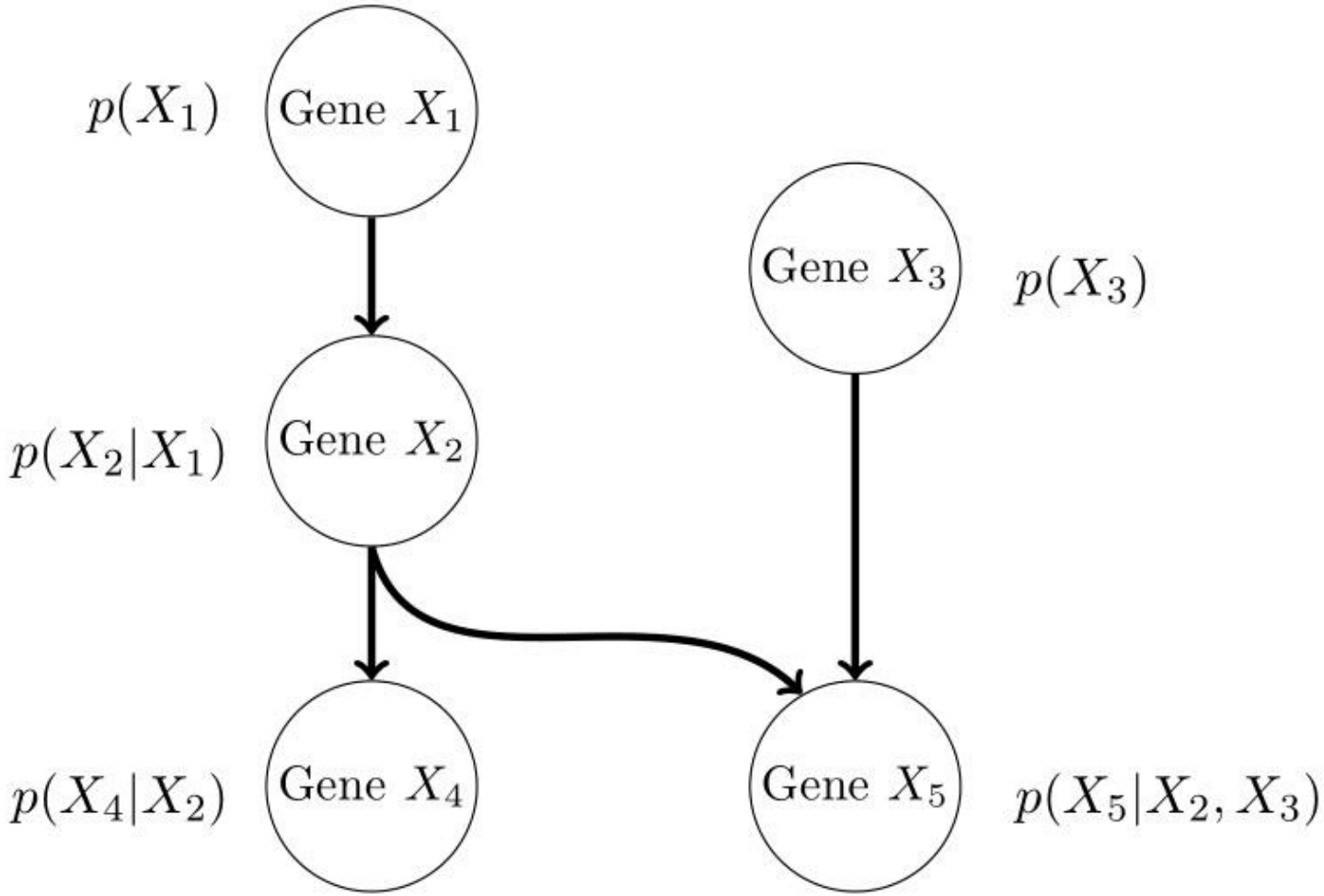
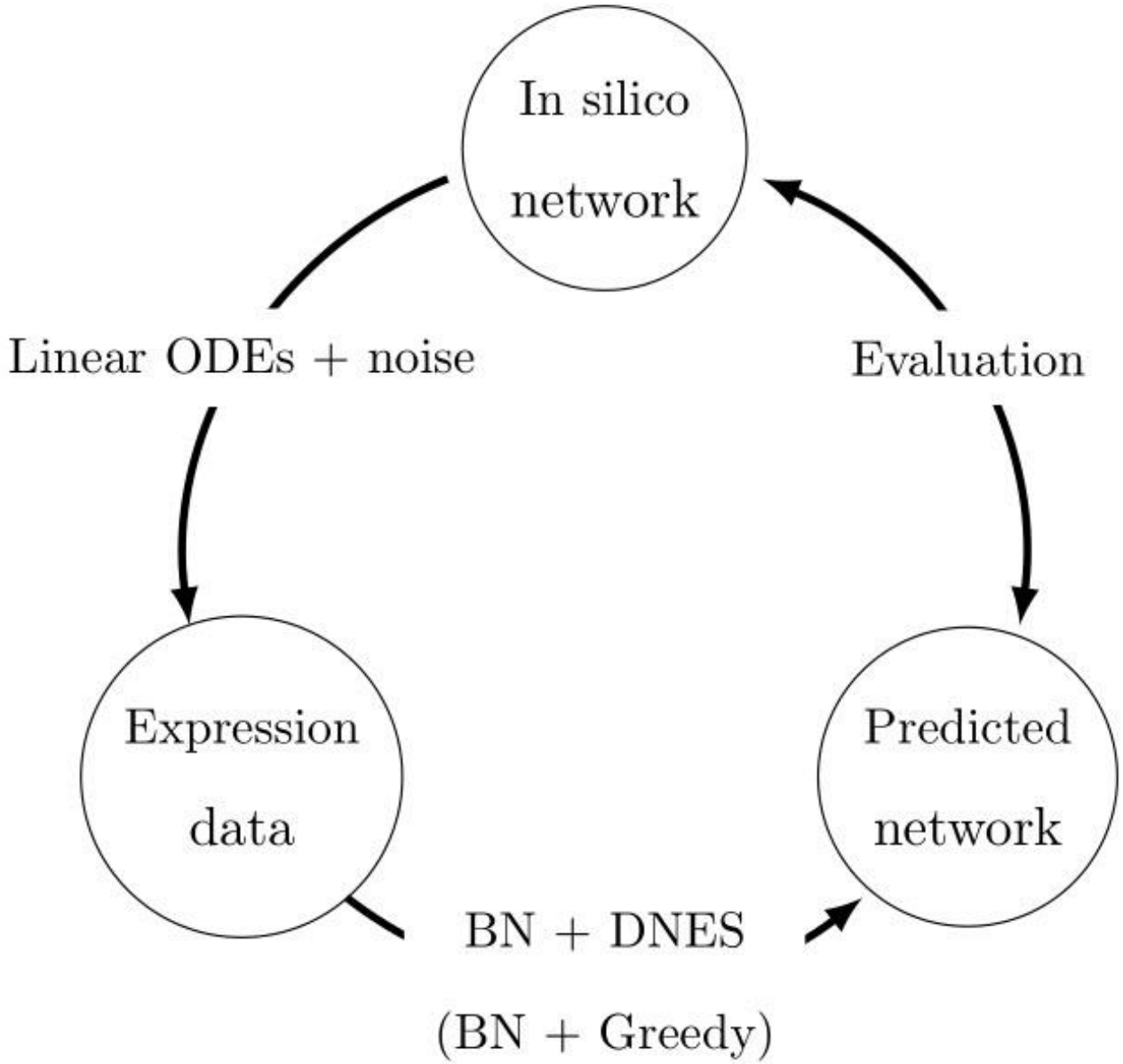
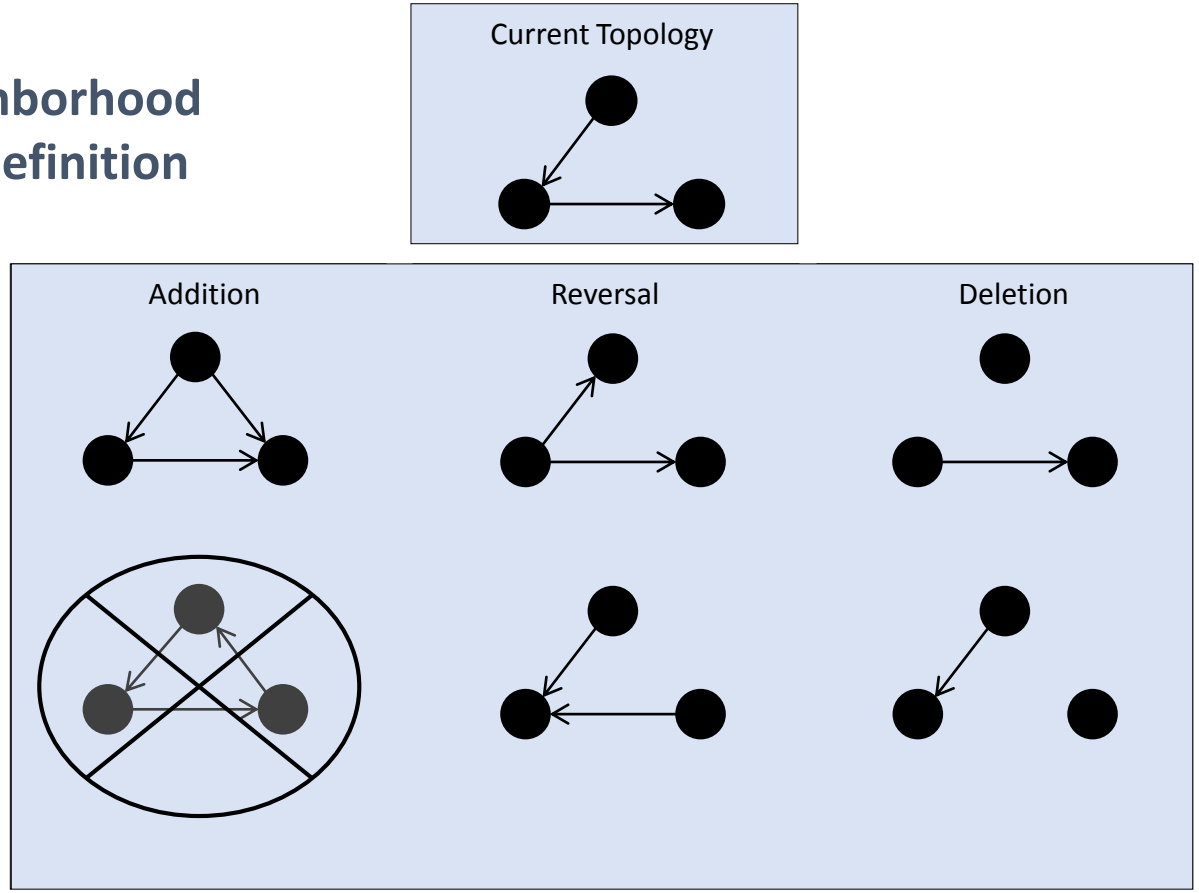


Figure 1: Bayesian network example

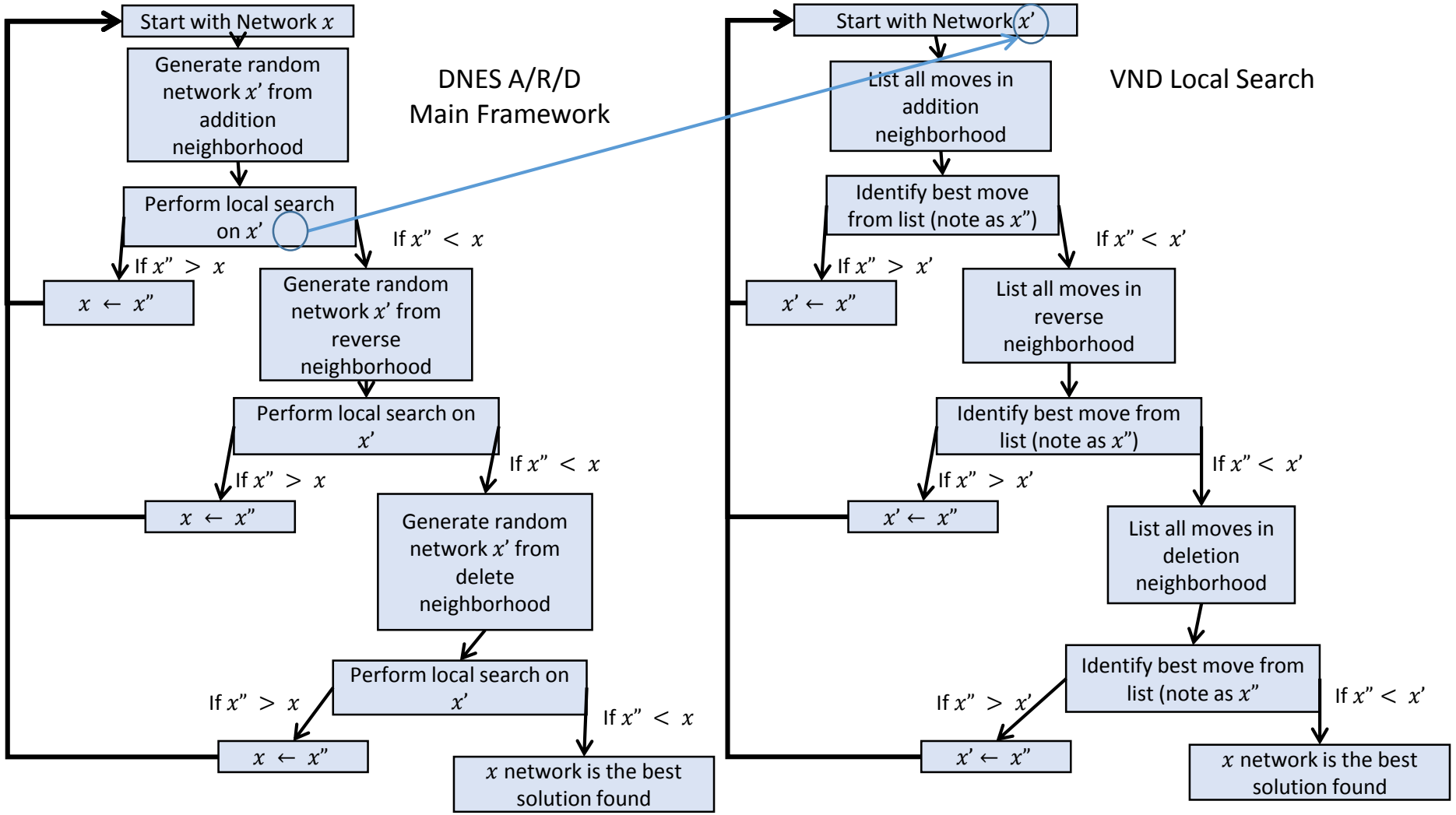

In silico network
Linear ODEs + noise
Evaluation
Expression data
Predicted network
BN + DNES
(BN + Greedy)

Neighborhood Structures

DNES Neighborhood Structure Definition



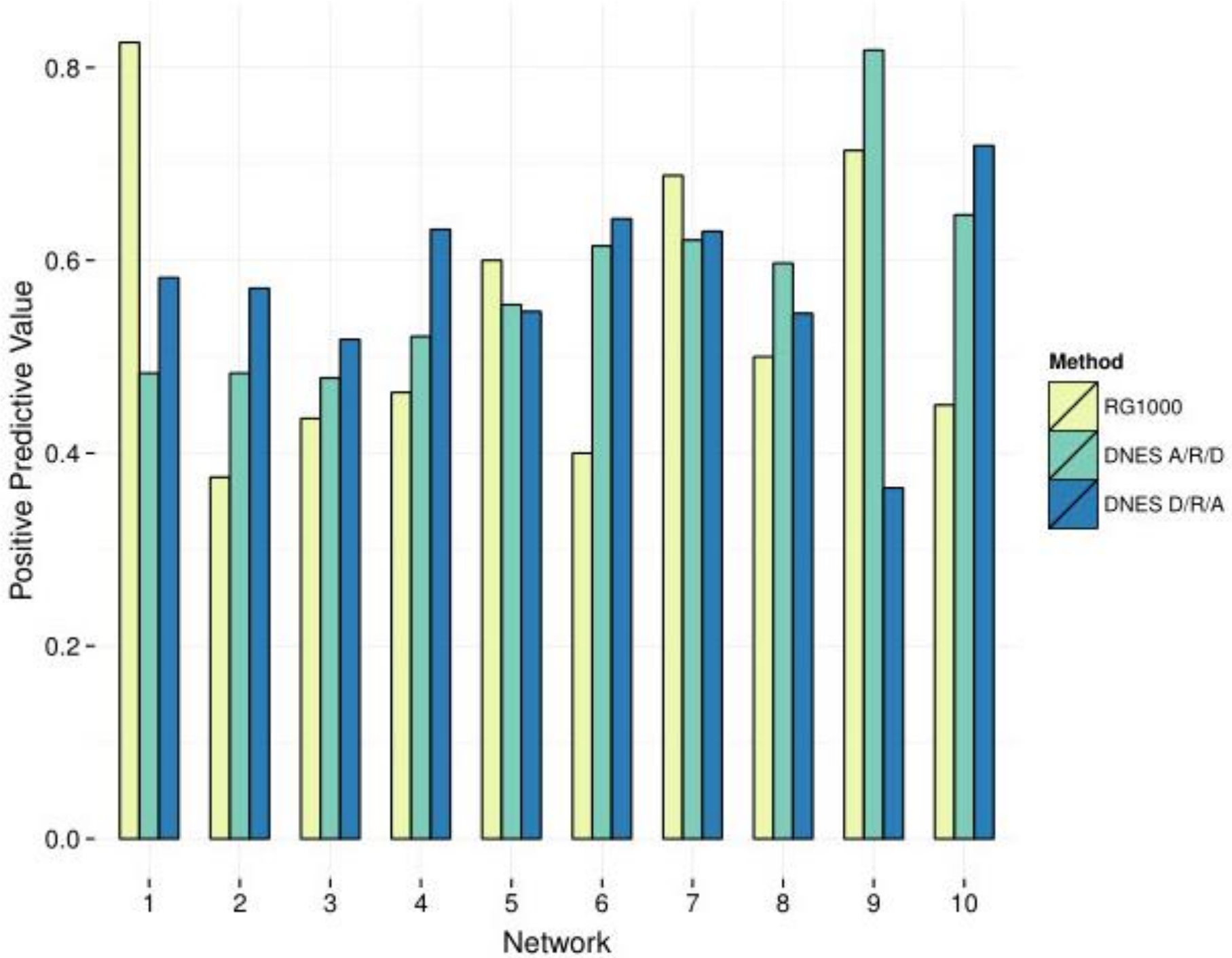
DNES Framework

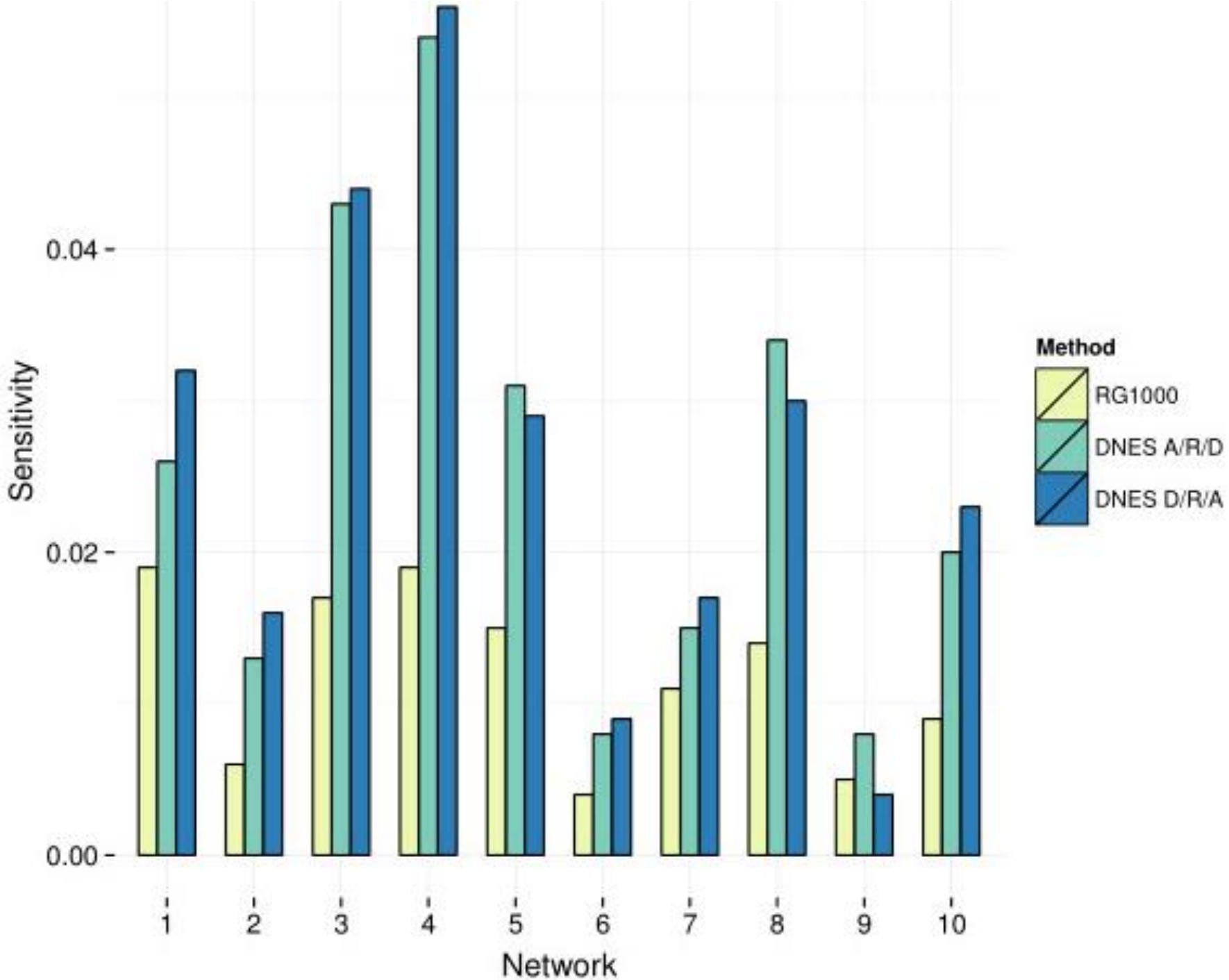


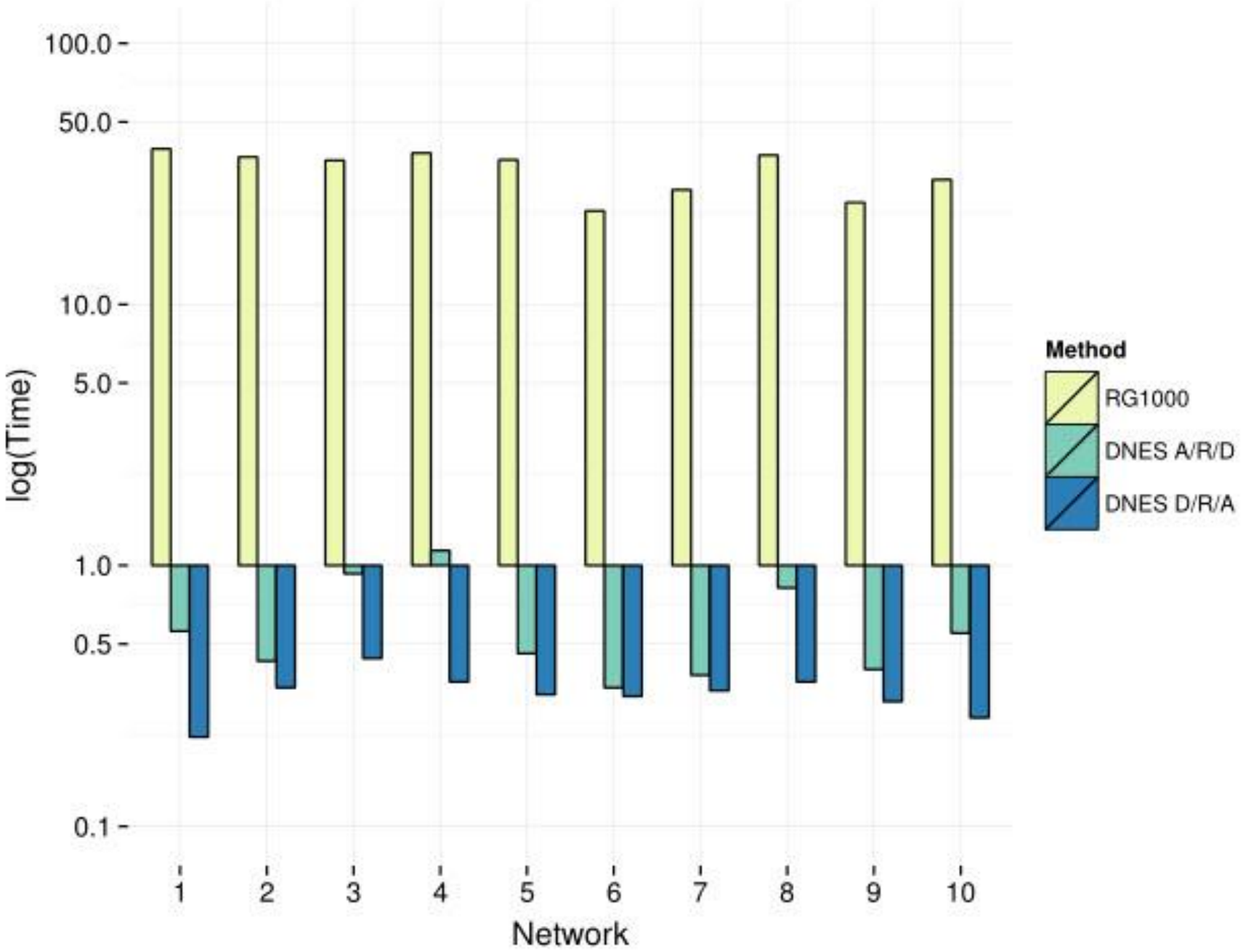
Experimental Design

Searcher	Proposer	Number of Restarts	Name
Greedy	Random Moves	1000	RG1000
Greedy	Random Moves	500	RG500
Greedy	All Local Moves	1000	LG1000
Greedy	All Local Moves	500	LG500
DNES	A/R/D	0	DNES A/R/D
DNES	D/R/A	0	DNES D/R/A

	Directed					
	PPV_D	SE_D	Correct	BDe	Time	Networks
RG1000	0.545	0.012	11.9	-1,118	32.75	4,388,590
RG500	0.540	0.010	11.5	-1,123	16.03	2,158,000
LG1000	0.090	0.010	8.4	-1,309	107.8	10,110,343
LG500	0.090	0.010	8.5	-1,309	53.8	5,055,125
DNES A/R/D	0.582	0.025	25.7	-1,042	0.60	197,919
DNES D/R/A	0.575	0.026	25.8	-1,043	0.32	94,124


Positive Predictive Value
0.0
0.2
0.4
0.6
0.8
1
2
3
4
5
6
7
8
9
10
Network
Method
RG1000
DNES A/R/D
DNES D/R/A


0.04
0.02
0.00
Sensitivity
1
2
3
4
5
6
7
8
9
10
Network
Method
RG1000
DNES A/R/D
DNES D/R/A


100.0
50.0
10.0
5.0
1.0
0.5
0.1
log(Time)
1
2
3
4
5
6
7
8
9
10
Network
Method
RG1000
DNES A/R/D
DNES D/R/A

Metric	DNES A/R/D − RG1000	DNES D/R/A − RG1000
SE_D	**0.013**	**0.014**
	(0.003)	(0.003)
PPV_D	0.036	0.030
	(0.492)	(0.660)
Total Edges	**48.8**	**44.0**
	(0.002)	(0.002)
Edges Correct	**24.8**	**22.2**
	(0.002)	(0.002)
BDe	**75.64**	**74.52**
	(< 0.001)	(< 0.001)
Time	**-32.15**	**-32.43**
	(< 0.001)	(< 0.001)