The Missing Models: A Data-driven Approach to Learning How Networks Grow

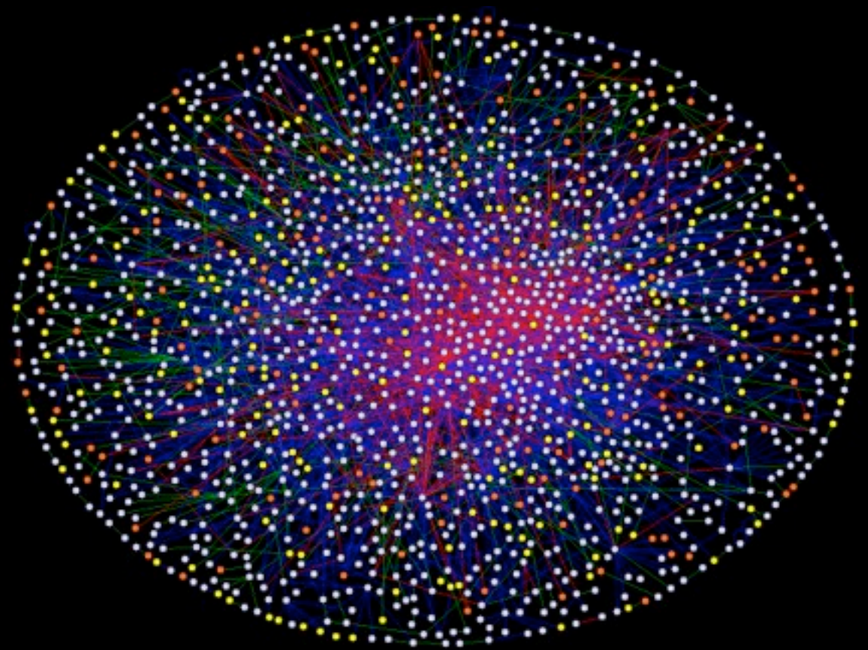
Carl Kingsford
Professor
Computational Biology Department
School of Computer Science
Carnegie Mellon University

Robert Patro, Geet Duggal, Emre Sefer, Hao Wang, Darya Filippova, Carl Kingsford (2012). The missing models: A data-driven approach for learning how networks grow. *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 42-50.

Computational Biology Department | Carnegie Mellon University School of Computer Science

Networks are everywhere

Biological

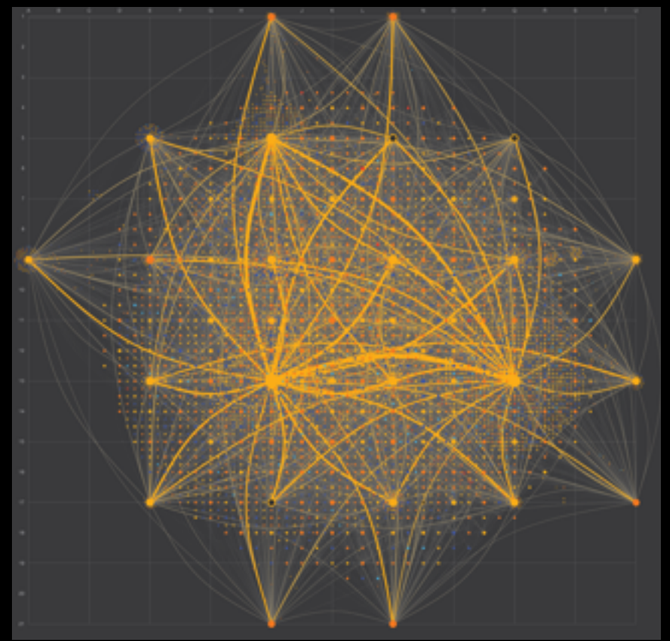


[Stelzl et al. 2005]

Social

[Bulik-Sullivan & Sullivan 2012]

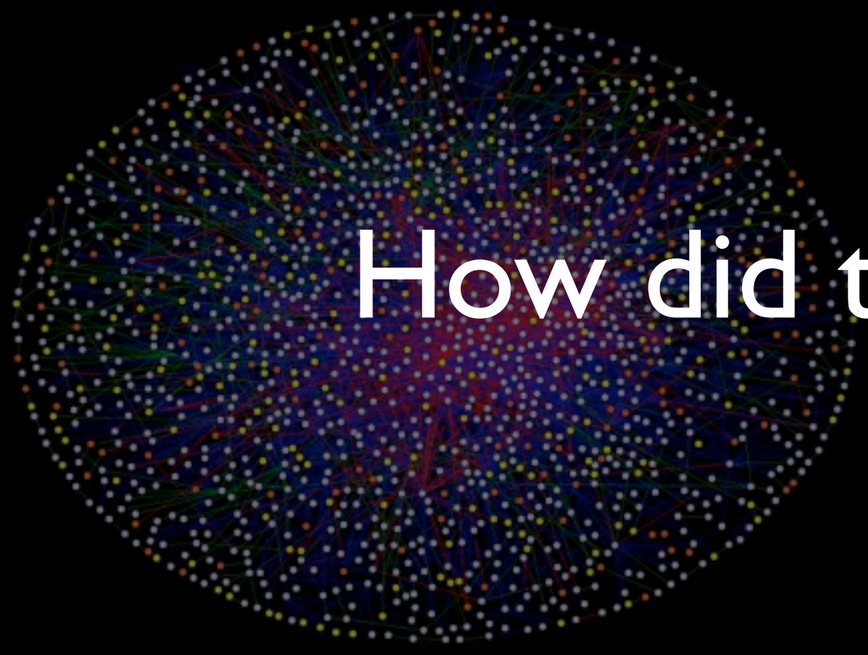
Technological



[Peer 1 2011]

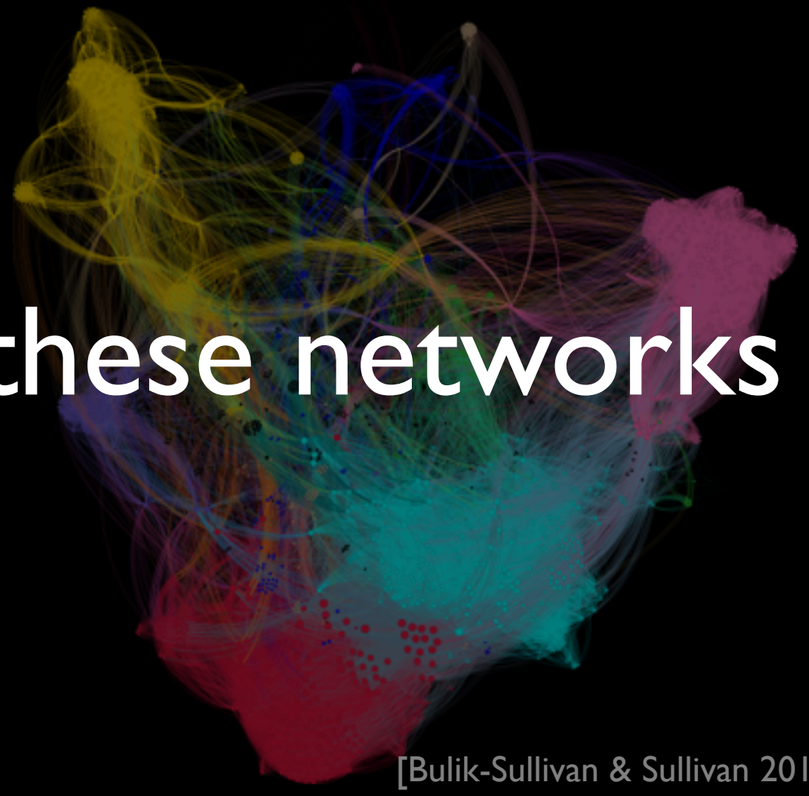
Networks are everywhere

Biological



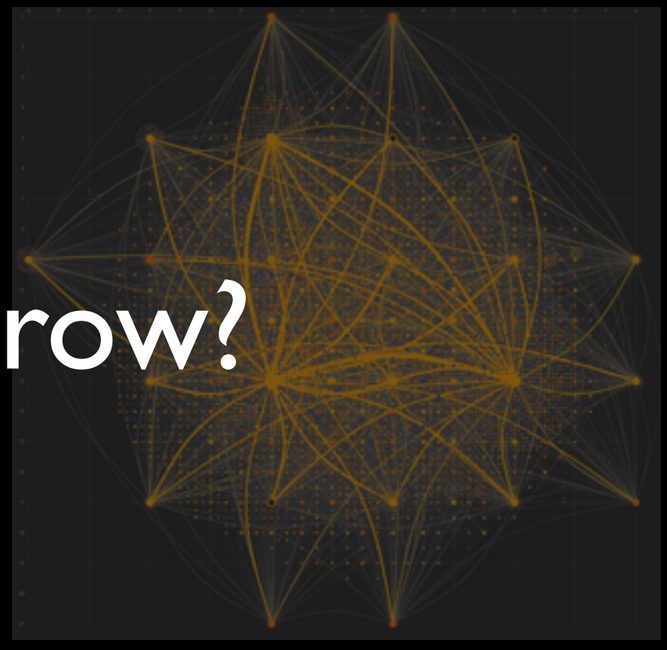
[Stelzl et al. 2005]

Social



[Bulik-Sullivan & Sullivan 2012]

Technological

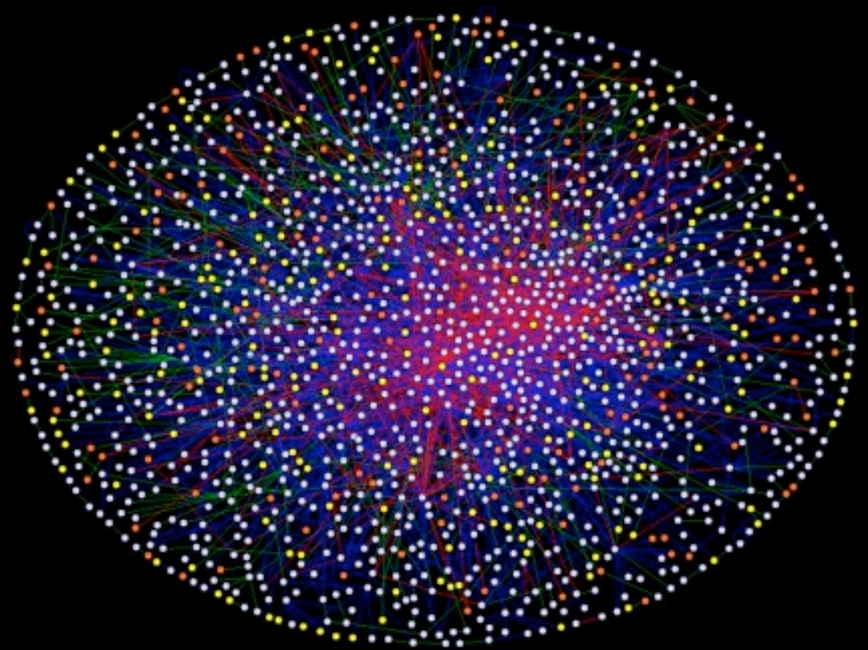


[Peer 1 2011]

How did these networks grow?

Enter Network Growth Models

Biological

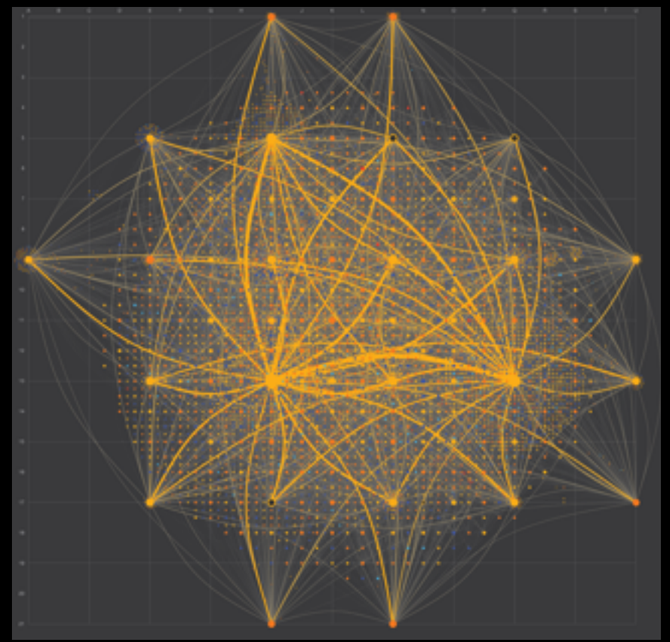


[Stelzl et al. 2005]

Social

[Bulik-Sullivan & Sullivan 2012]

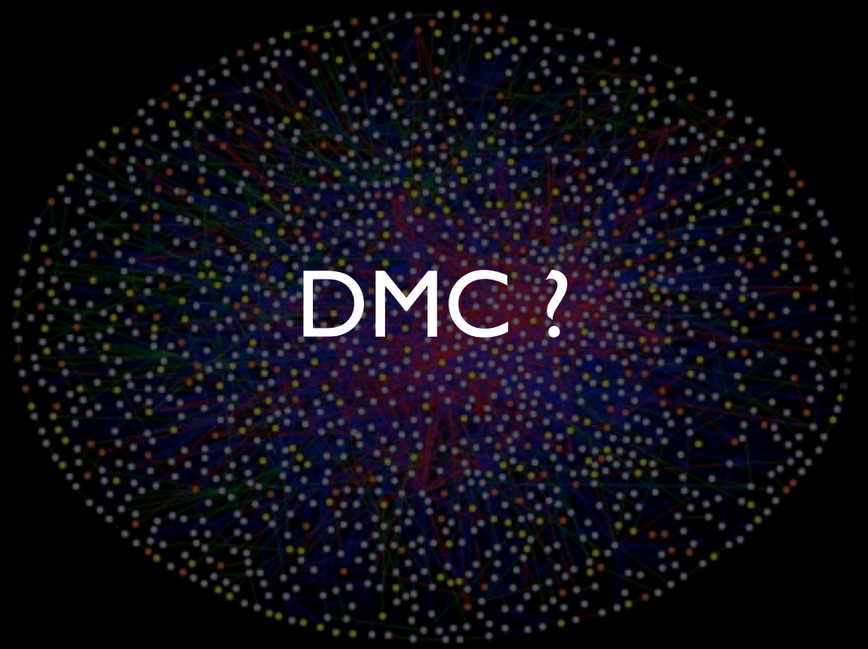
Technological



[Peer 1 2011]

Enter Network Growth Models

Biological



[Stelzl et al. 2005]

Social

[Bulik-Sullivan & Sullivan 2012]

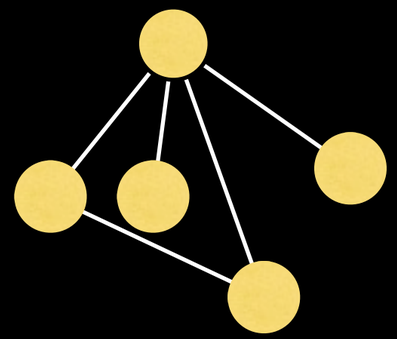
Technological

[Peer 1 2011]

Example : DMC Model

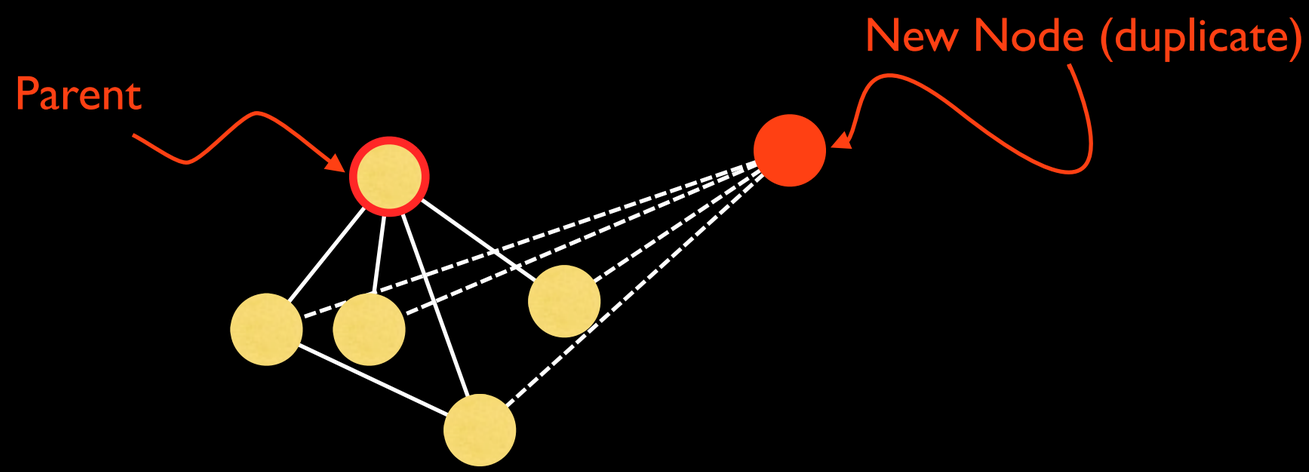
Plausible model of protein interaction network growth introduced by Vazquez et al. in 2001

Based on gene duplication & divergence



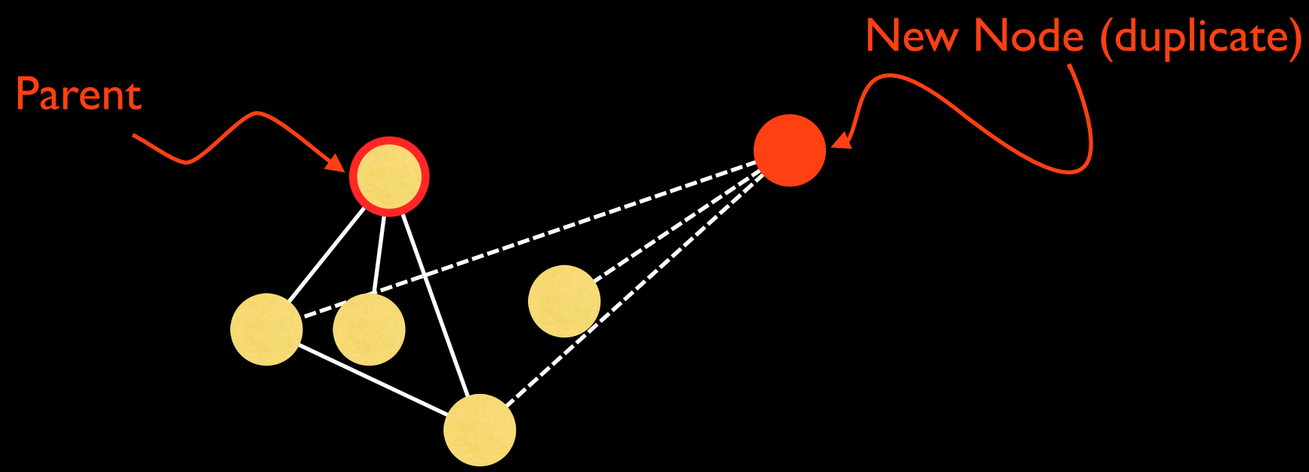
Network at time t

Example of DMC Model



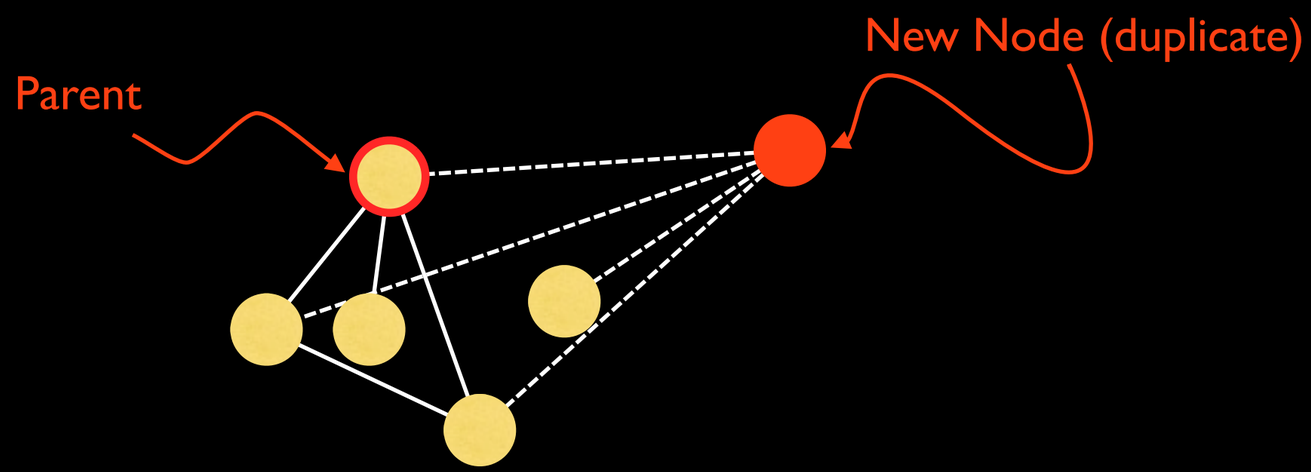
[Duplication, Mutation, Complementarity]

Example of DMC Model



[Duplication, Mutation, Complementarity]

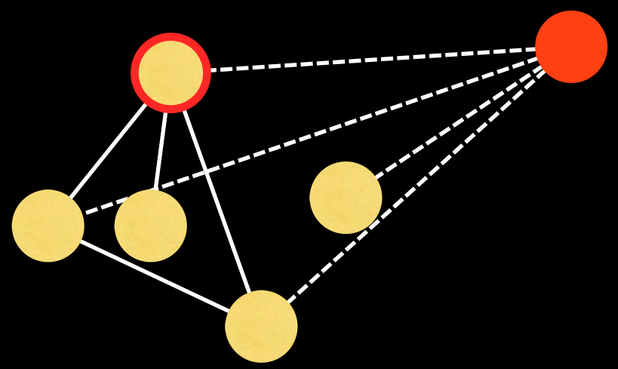
Example of DMC Model



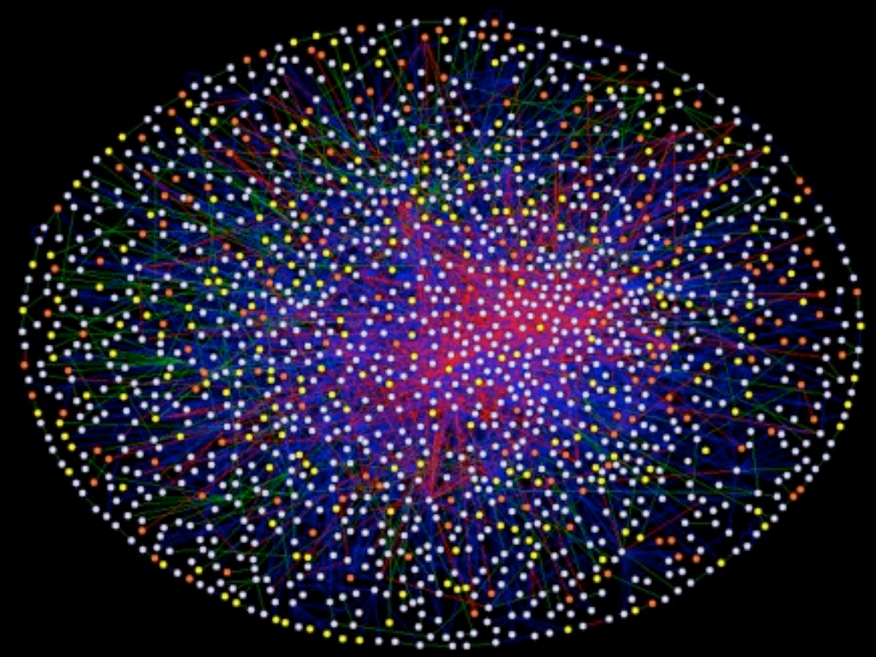
[Duplication, Mutation, Complementarity]

Network at time t+1

Example of DMC Model



Repeated for many steps



[Duplication, Mutation, Complementarity]

[Stelzl et al. 2005]

In addition to biologically plausible mechanism, can produce networks with similar degree distribution and clustering coeff. as real PPIs

Network Growth Models (NGMs)

"What I Cannot Create, I Do Not Understand" -- Richard Feynman

Bottom-up generative model of network growth process

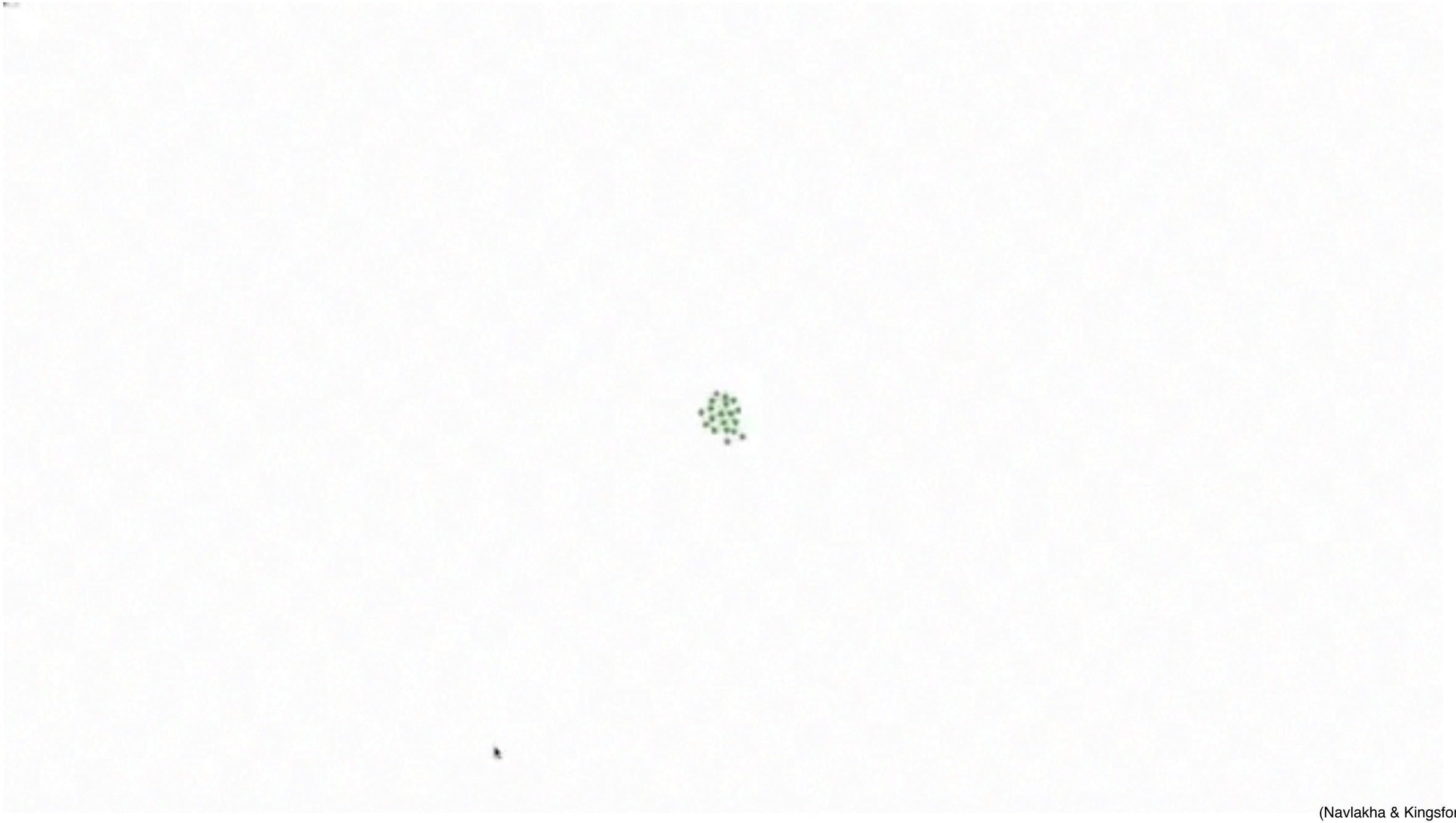
Creates "random" graphs with similar topological characteristics to the target

Practical

Evaluate statistical significance of observed features
Test algorithms in different contexts
- Varying topological characteristics
- Varying scales

Theoretical

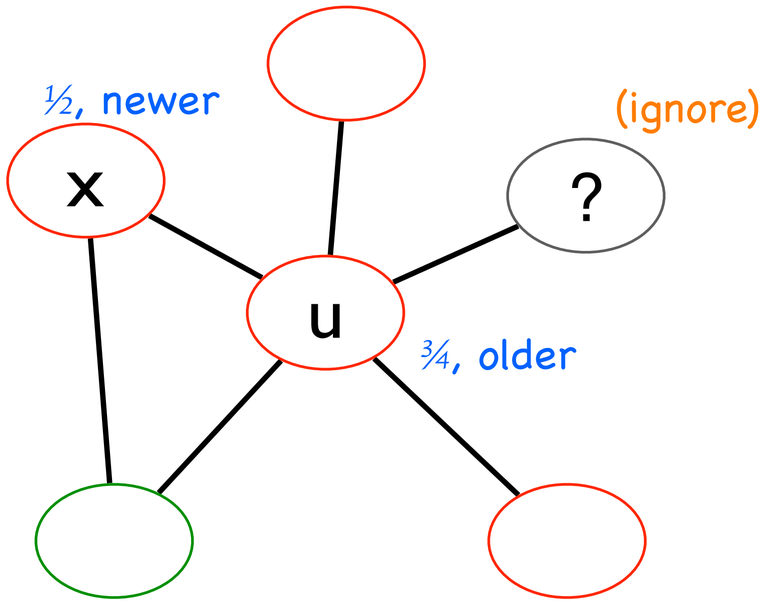
Discover reasons for observed structure
How does topology change over time?
How did the network look in the past?
How will it look in the future?



(Navlakha & Kingsford,
PLoS Comp. Biol., 2011)

Core vs. Peripheral Complex Members

Coreness of a protein = percentage of like-annotated neighbors

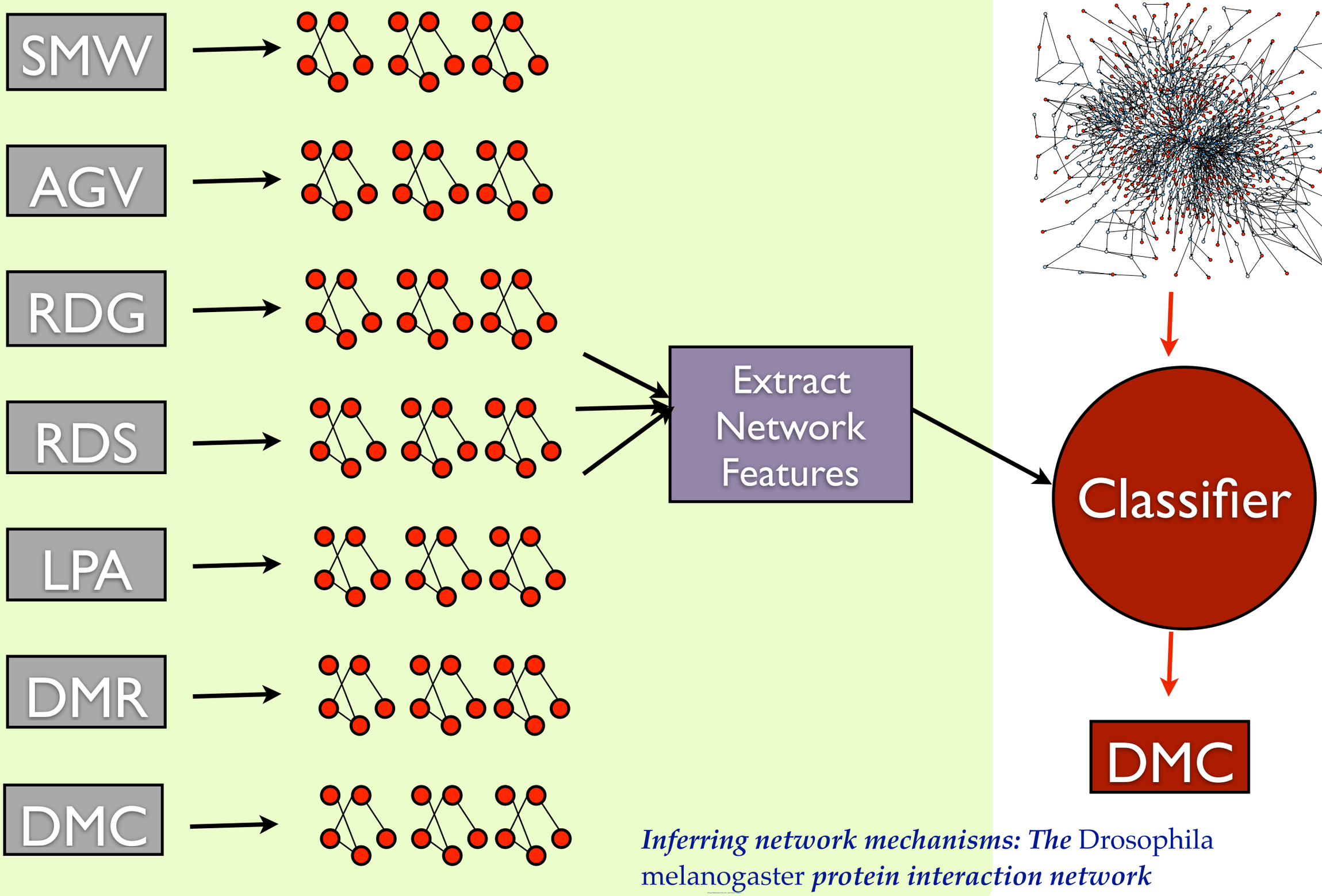


Are core members of a protein complex older than peripheral members?

Yes, somewhat: R = 0.37, P < 0.01

Agrees with 3D protein structure analysis (Kim & Marcotte, 2008) looking at age distribution of domains among eukaryotic species.

Supervised Learning → Predict Network Models



Inferring network mechanisms: The Drosophila melanogaster *protein interaction network*

Many Existing Growth Models

Varying complexity / accuracy

Erdös-Rényi [1960]

⋮

Barabási-Albert [1999]

DMC [Vazquez et al. 2001]

Duplication-divergence [Ispolatov et al. 2005]

⋮

RTG [Akoglu & Faloutsos 2009]

Forest Fire Model [Leskovec et al. 2010]

Kronecker Model [Leskovec et al. 2010]

Multifractal Network Generator [Palla et al. 2010]

⋮

Repeated application of simple growth rule

More complex but highly flexible models

Many Existing Growth Models

Erdös-Rényi [1960]

⋮

Barabási-Albert [1999]

DMC [Vazquez et al. 2001]

Duplication-divergence [Ispolatov et al. 2005]

⋮

RTG [Akoglu & Faloutsos 2009]

Forest Fire Model [Leskovec et al. 2005]

Kronecker Model [Leskovec et al. 2010]

Multifractal Network Generator [Palla et al. 2010]

⋮

Varying complexity / accuracy

Previous work focused on either

Repeated application of simple growth rule

Manually designed growth models

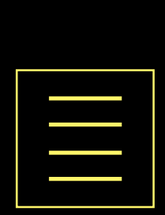
or

Parameterized family of models (possibly with parameter learning)

More complex but highly flexible models

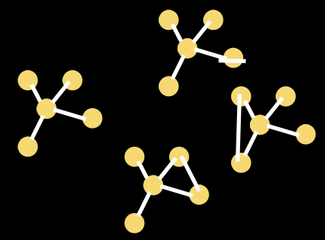
So What's New?

Method to automatically learn growth models which is nonparametric & data-driven

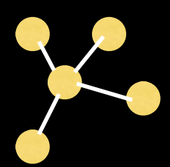


GrowCode program =
network growth model

GrowCode
Virtual
Machine



Random graphs



Target graph

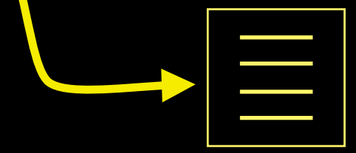
GrowCode
Optimization

Set of network growth models optimized to produce graphs similar to the target graph

So What's New?

Method to automatically learn growth models which is nonparametric & data-driven

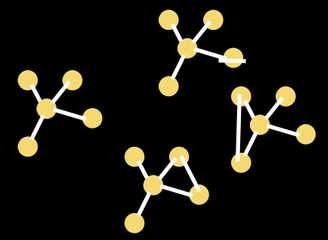
Growth model is a program in the GrowCode language



GrowCode program = network growth model

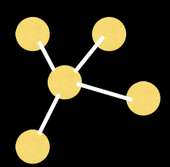
GrowCode Virtual Machine

Instructions represent basic topological operations



Random graphs

Pose finding NGMs as optimization over the space of programs



Target graph

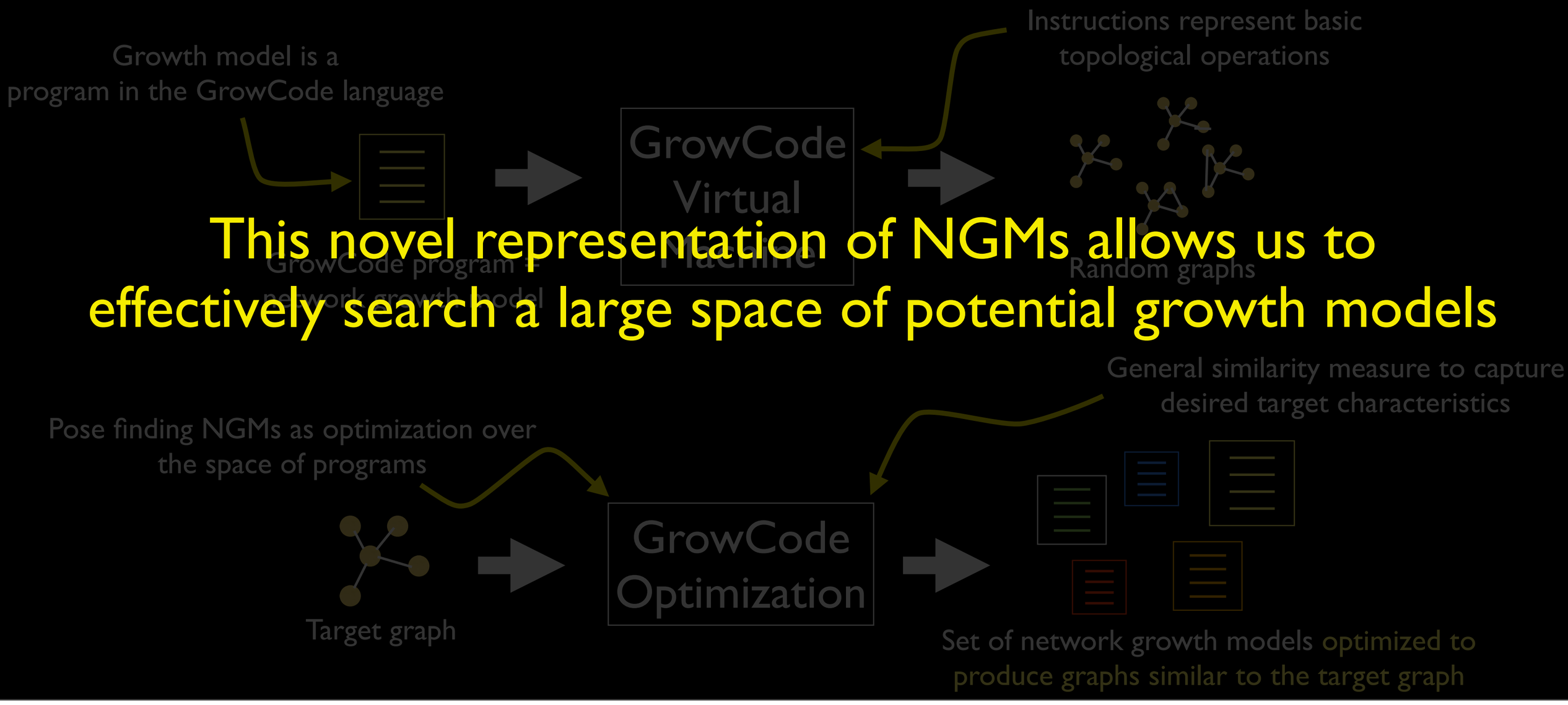
GrowCode Optimization

General similarity measure to capture desired target characteristics

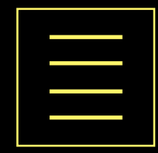
Set of network growth models optimized to produce graphs similar to the target graph

So What's New?

Method to automatically learn growth models which is nonparametric & data-driven

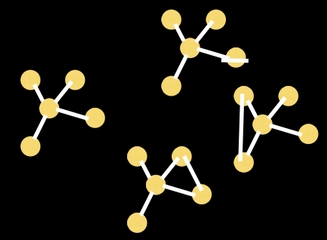


This novel representation of NGMs allows us to effectively search a large space of potential growth models

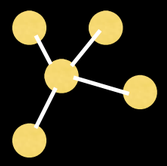


GrowCode program =
network growth model

GrowCode
Virtual
Machine



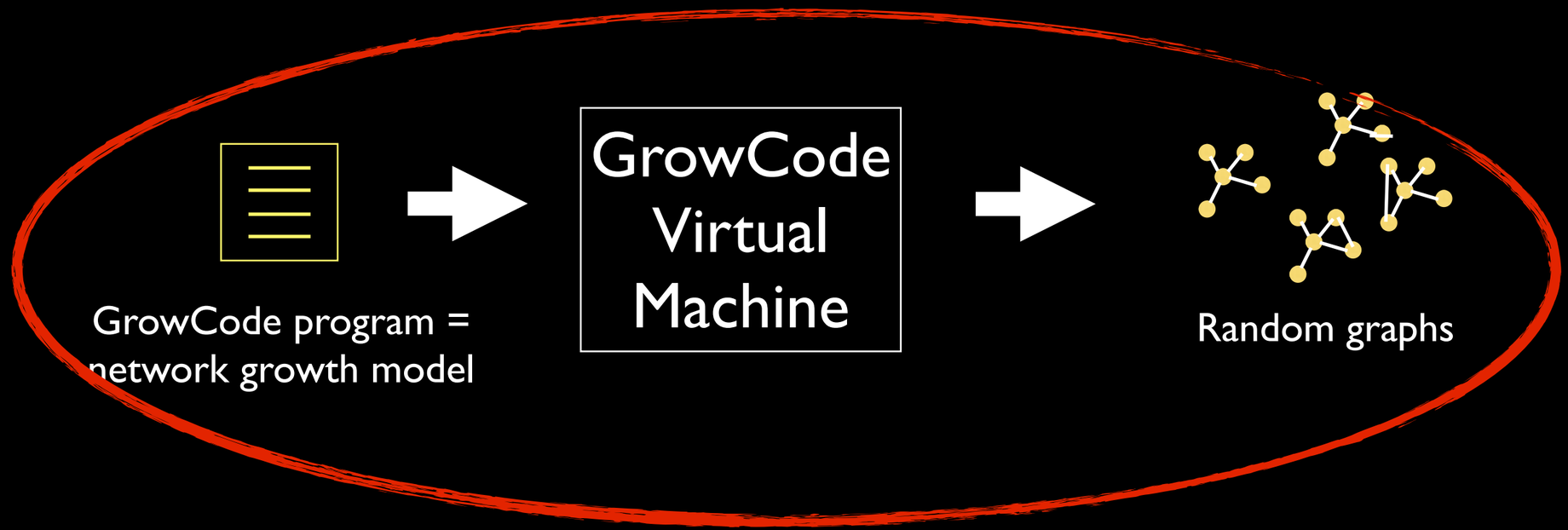
Random graphs



Target graph

GrowCode
Optimization

Set of network growth models
optimized to produce graphs
similar to the target graph


GrowCode program =
network growth model
GrowCode
Virtual
Machine
Random graphs

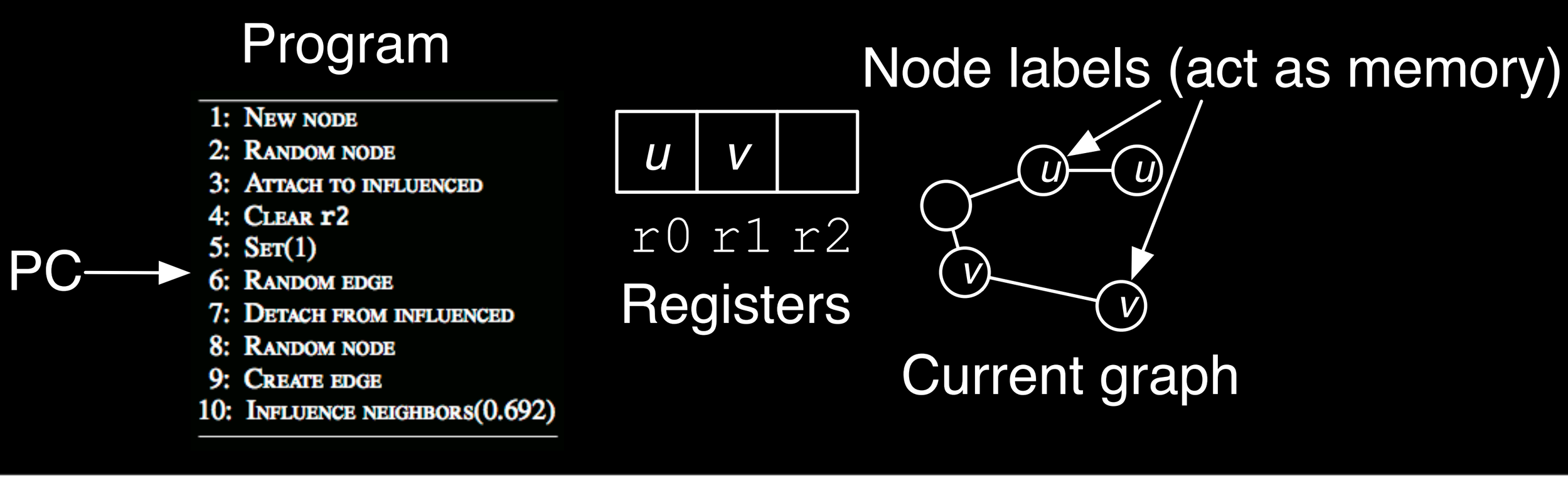

Target graph
GrowCode
Optimization
Set of network growth models
optimized to produce graphs
similar to the target graph

GrowCode Virtual Machine

Register-based virtual machine

Node label memory L : V→V

Runs program iteratively to grow a graph



Machine Instructions

Modify graph toplogy

Modify label memory

Program control flow

Manipulate machine registers

Name	Description
NEW NODE	create new node
CREATE EDGE	create new edge
RANDOM NODE	pick random node
RANDOM EDGE	pick random edge
INFLUENCE NEIGHBORS(p)	label neighbors with u
ATTACH TO INFLUENCED	add edges to neighbors labeled u
DETACH FROM INFLUENCED	remove edges to neighbors labeled u
CLEAR INFLUENCED	clear all labels from `L`
REWIND(r, i)	jump back r positions i times
SKIP INSTRUCTION(p)	skip next instruction
SET(i)	copy node ID to `r2`
SAVE	copy `r0` to `r2`
LOAD	copy `r2` to `r0`
SWAP	swap `r0` and `r1`
CLEAR `r2`	set `r2` to `NIL`

Machine Instructions

Modify graph toplogy

Modify label memory

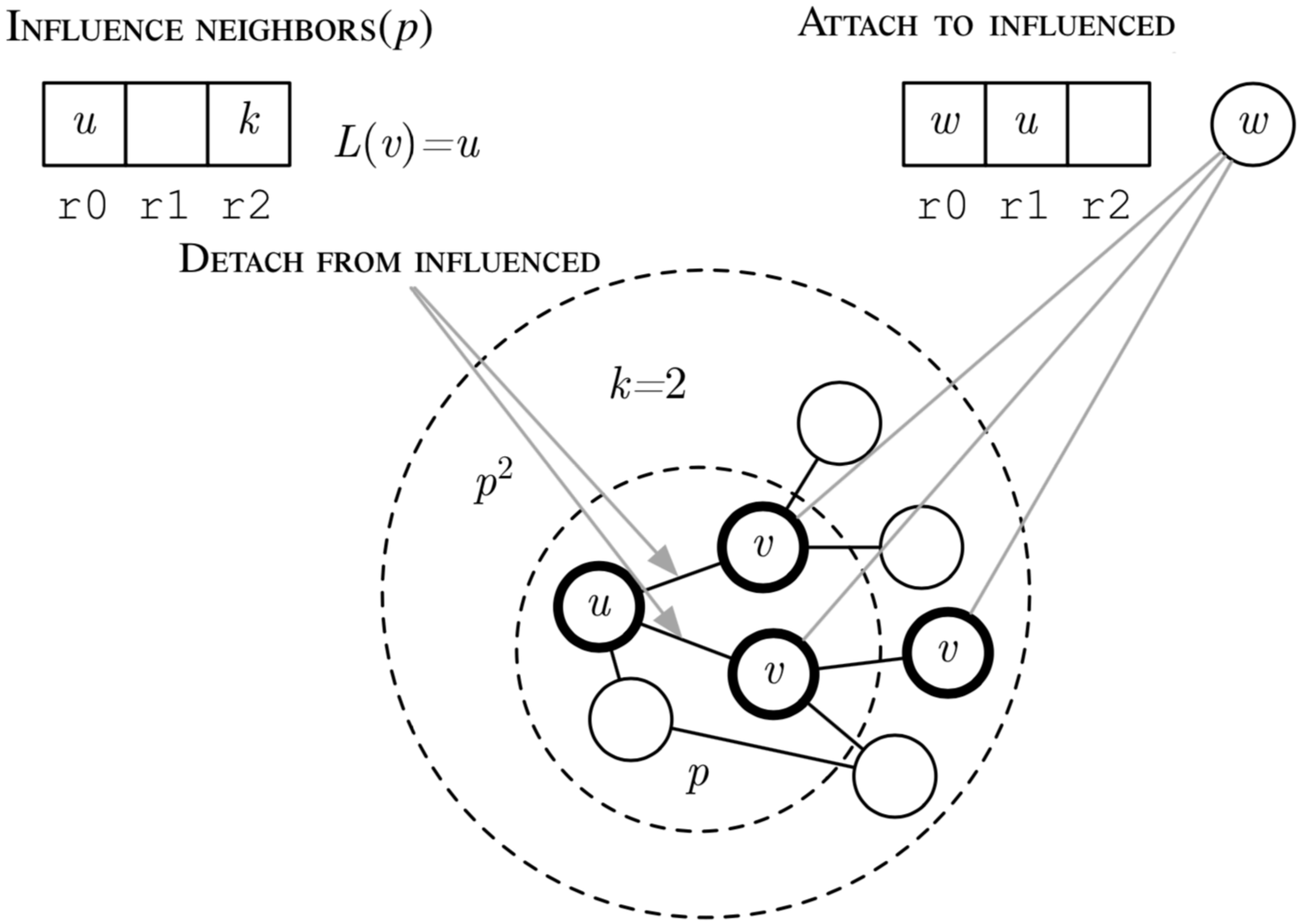
Program control flow

Manipulate machine registers

Name	Description
NEW NODE	create new node
CREATE EDGE	create new edge
RANDOM NODE	pick random node
RANDOM EDGE	pick random edge
INFLUENCE NEIGHBORS(p)	label neighbors with u
ATTACH TO INFLUENCED	add edges to neighbors labeled u
DETACH FROM INFLUENCED	remove edges to neighbors labeled u
CLEAR INFLUENCED	clear all labels from `L`
REWIND(r, i)	jump back r positions i times
SKIP INSTRUCTION(p)	skip next instruction
SET(i)	copy node ID to `r2`
SAVE	copy `r0` to `r2`
LOAD	copy `r2` to `r0`
SWAP	swap `r0` and `r1`
CLEAR `r2`	set `r2` to `NIL`

Every sequence of instructions is a semantically valid GrowCode Program

Influence Instructions



Example GrowCode Program

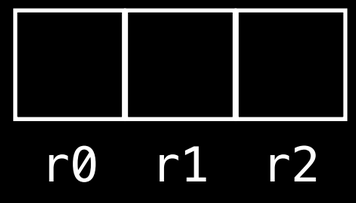
A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

Current graph:

Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

Registers:



Example GrowCode Program

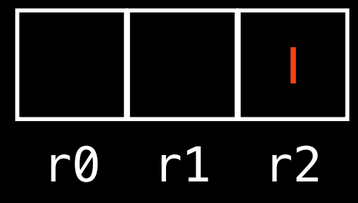
A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

Current graph:

Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

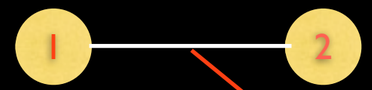
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

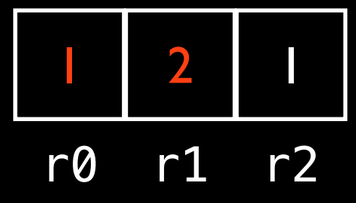
Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

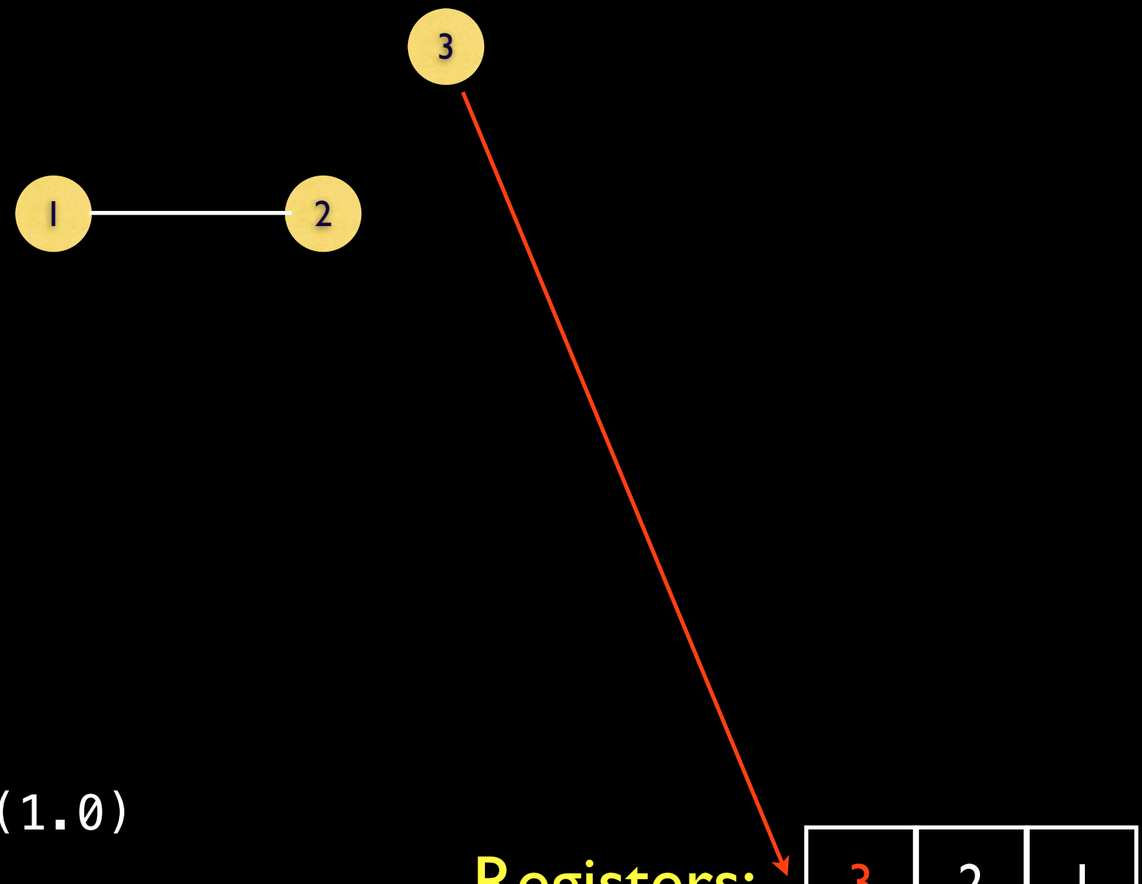
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

Current graph:



Program:

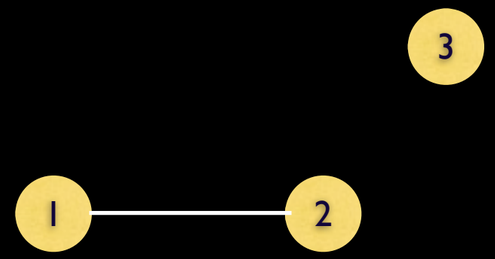
```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

Registers:

Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

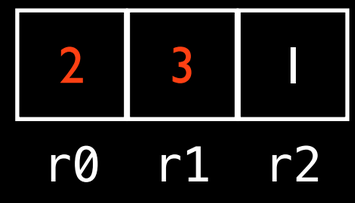
Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

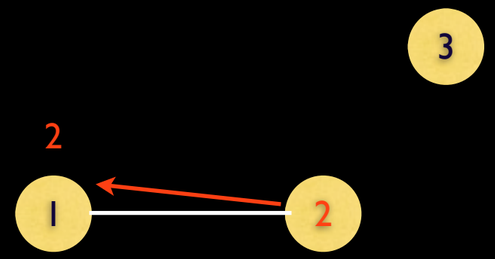
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

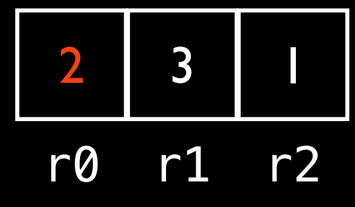
Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

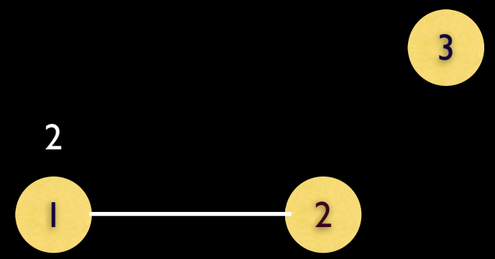
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

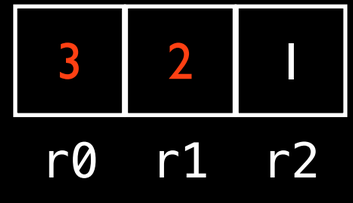
Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

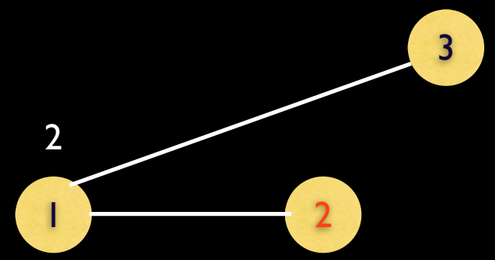
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

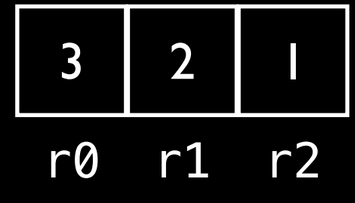
Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

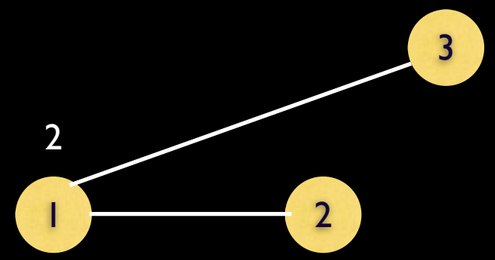
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

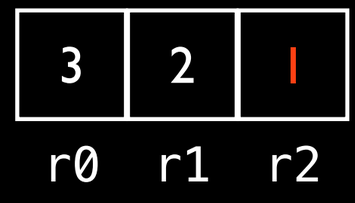
Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

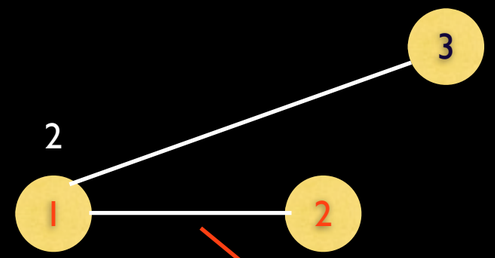
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

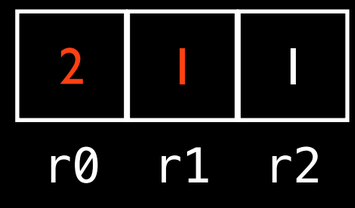
Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

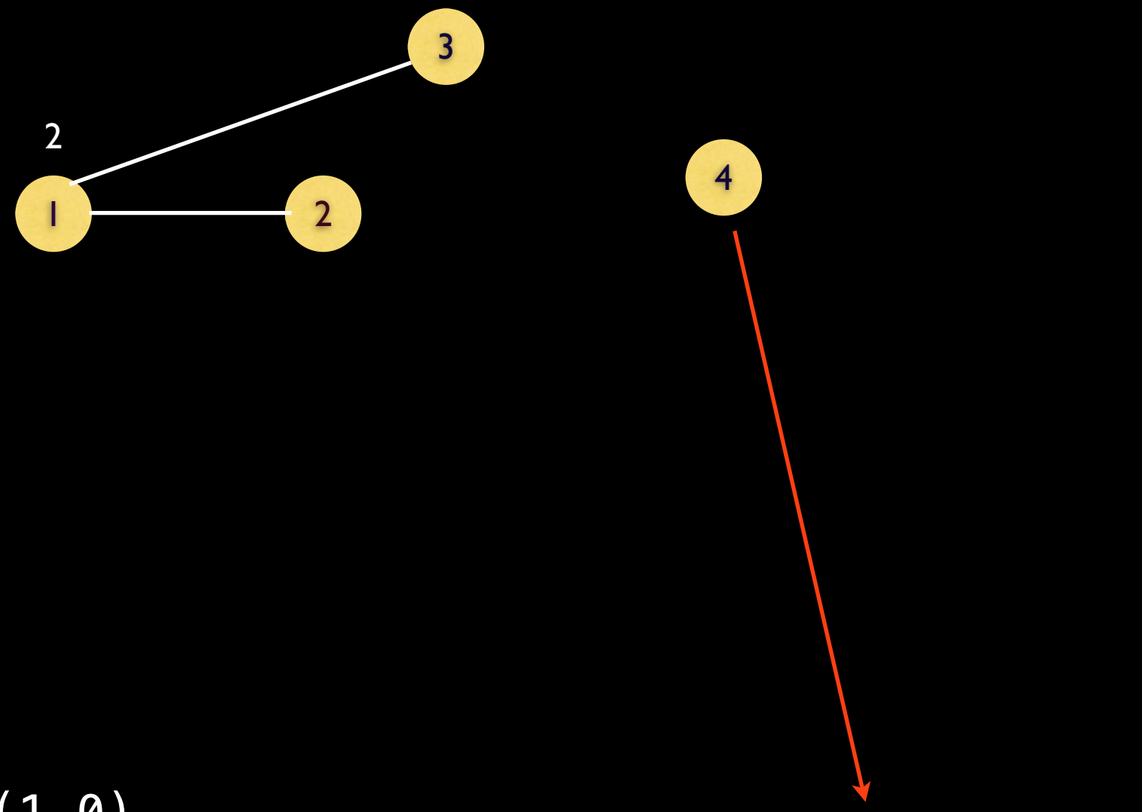
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

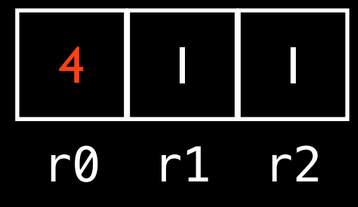
Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

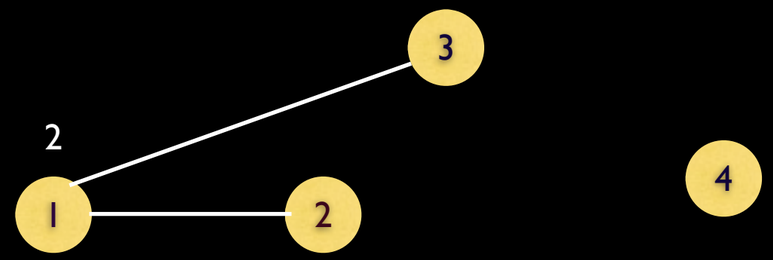
Registers:



Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

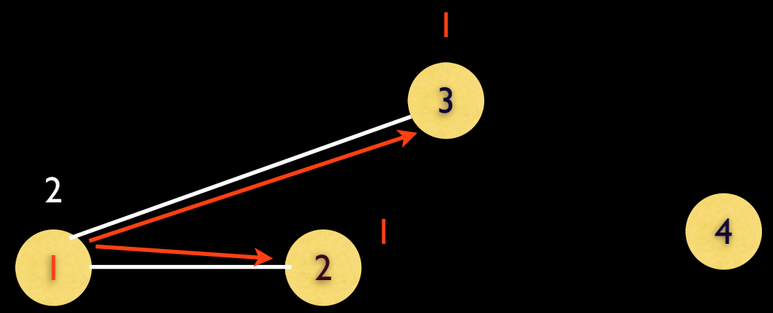
Registers:

r0	r1	r2
1	4	1

Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

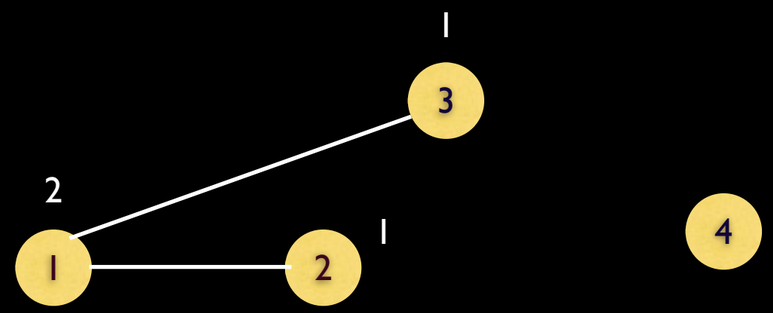
Registers:

r0	r1	r2
1	4	1

Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

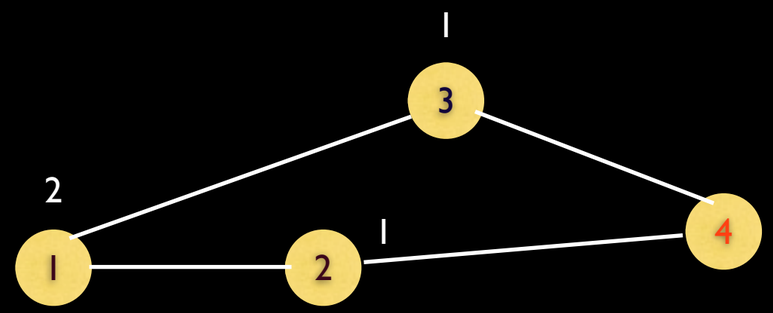
Registers:

r0	r1	r2
4	1	1

Example GrowCode Program

A new node duplicates an existing node *u* where *u* is selected proportional to its degree.

Current graph:



Program:

```
Set(1)
Random edge
New node
Swap
Influence neighbors(1.0)
Swap
Attach to influenced
```

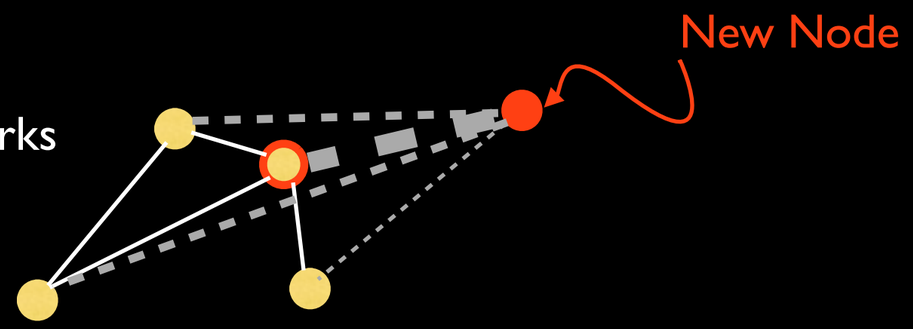
Registers:

r0	r1	r2
4	1	1

GrowCode Can Express Existing Models

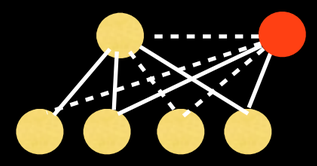
Barabási-Albert (models preferential attachment of new nodes)

Reproduces scale free distribution observed in large, real networks



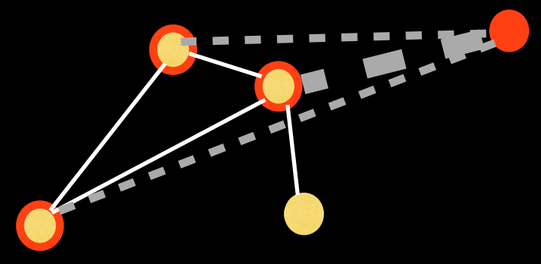
DMC (models protein duplication and functional divergence events)

Reproduces scale free distribution and clustering
properties of protein-protein interaction networks (PPI)



Forest Fire (new nodes connect to a set of topologically close nodes)

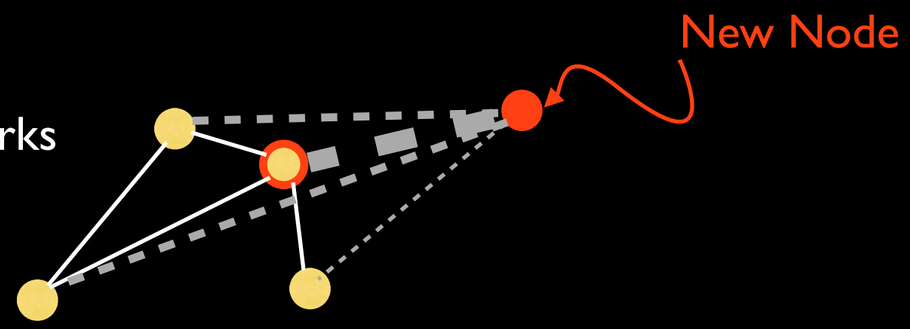
Reproduces scale free distribution, shrinking diameter,
and densification over time



GrowCode Can Express Existing Models

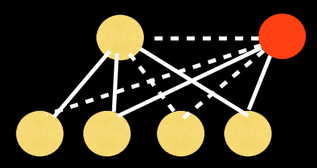
Barabási-Albert (models preferential attachment of new nodes)

Reproduces scale free distribution observed in large, real networks



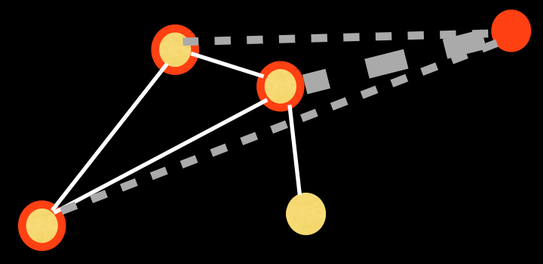
DMC (models protein duplication and functional divergence events)

Reproduces scale free distribution and clustering
properties of protein-protein interaction networks (PPI)



Forest Fire (new nodes connect to a set of topologically close nodes)

Reproduces scale free distribution, shrinking diameter,
and densification over time



GrowCode instructions are reused throughout all three models.

Existing Models in GrowCode

Algorithm 1 Barabási-Albert

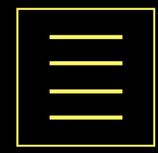
```
1: NEW NODE                          ▷ Create a new node, u
2: SAVE
3: RANDOM EDGE                       ▷ Choose a random edge, e
4: SKIP INSTRUCTION(0.5)             ▷ Choose random endpoint v of e
5: SWAP
6: LOAD
7: CREATE EDGE                       ▷ Create an edge between v and u
8: REWIND(5, i)                      ▷ Attach it to i random nodes
```

Algorithm 2 DMC

```
1: RANDOM NODE                       ▷ Put a random node v in r0
2: SET(1)                            ▷ Set r2 (k-hop for influence) to 1
3: INFLUENCE NEIGHBORS(1.0)          ▷ Influence v's neighbors
4: SWAP                              ▷ Swap r0 and r1
5: NEW NODE                          ▷ Create a new node u and put it in r0
6: ATTACH TO INFLUENCED              ▷ Connect u to influenced nodes
7: CLEAR INFLUENCED
8: INFLUENCE NEIGHBORS(q_MOD/2)      ▷ Influence u's neighbors
9: SWAP
10: INFLUENCE NEIGHBORS(q_MOD/2)     ▷ Influence v's neighbors
11: DETACH FROM INFLUENCED           ▷ Actually delete edges from v
12: SWAP
13: DETACH FROM INFLUENCED           ▷ Do the same for u
14: CLEAR INFLUENCED
15: SKIP INSTRUCTION(1.0 − q_CON)    ▷ Skip adding {u, v}
16: CREATE EDGE                      ▷ with probability q_CON
```

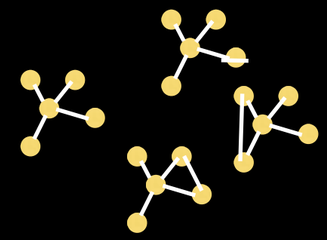
Algorithm 3 FF

```
1: RANDOM NODE                       ▷ Put a random node in r0
2: CLEAR r2                          ▷ Clear r2 to allow full graph influence
3: INFLUENCE NEIGHBORS(b)            ▷ Breadth-first recursive influence
4: SWAP                              ▷ Move the random node into r1
5: NEW NODE                          ▷ Create a new node, u
6: CREATE EDGE
7: ATTACH TO INFLUENCED              ▷ Connect u to influenced nodes
```

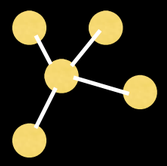


GrowCode program =
network growth model

GrowCode
Virtual
Machine



Random graphs

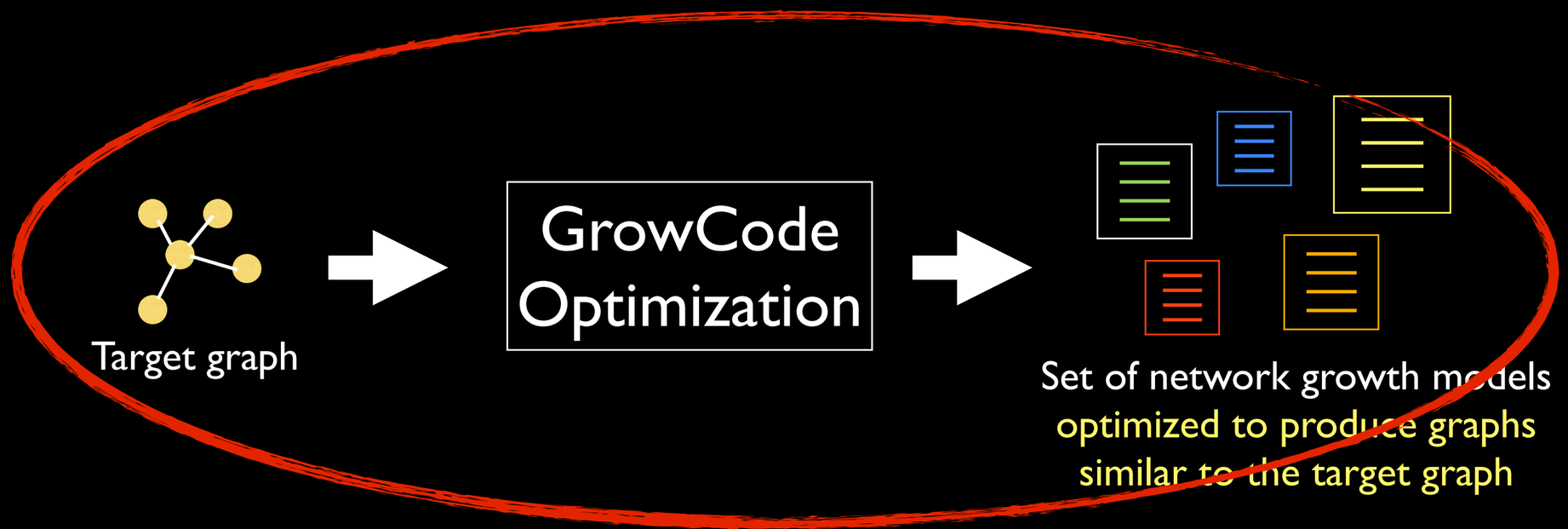


Target graph

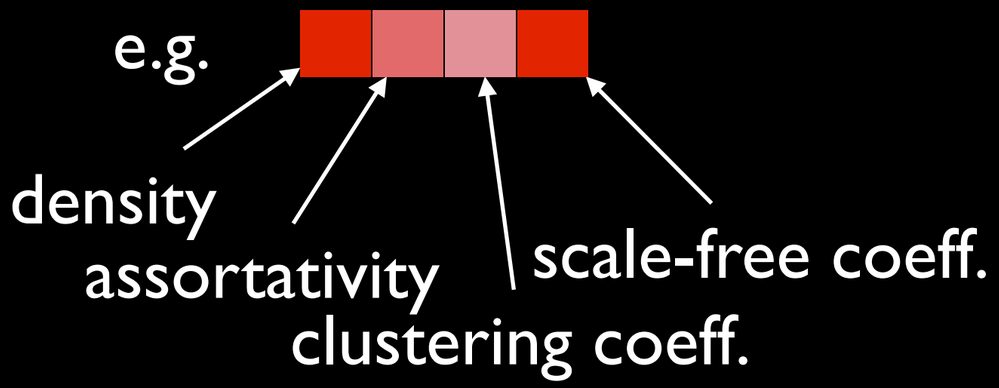
GrowCode
Optimization

Set of network growth models
optimized to produce graphs
similar to the target graph


GrowCode
Virtual
Machine
GrowCode program =
network growth model
Random graphs


GrowCode
Optimization
Target graph
Set of network growth models
optimized to produce graphs
similar to the target graph

Characterize a graph with a user-defined feature vector



Feature vector of "target" graph

$x^T =$

$x^P =$

Feature vector for graph generated by GrowCode Prog.

$$P^* = \arg\max_P \mathbb{E}[s(\mathbf{x}^P, \mathbf{x}^T)]$$

Finding a NGM becomes a search problem

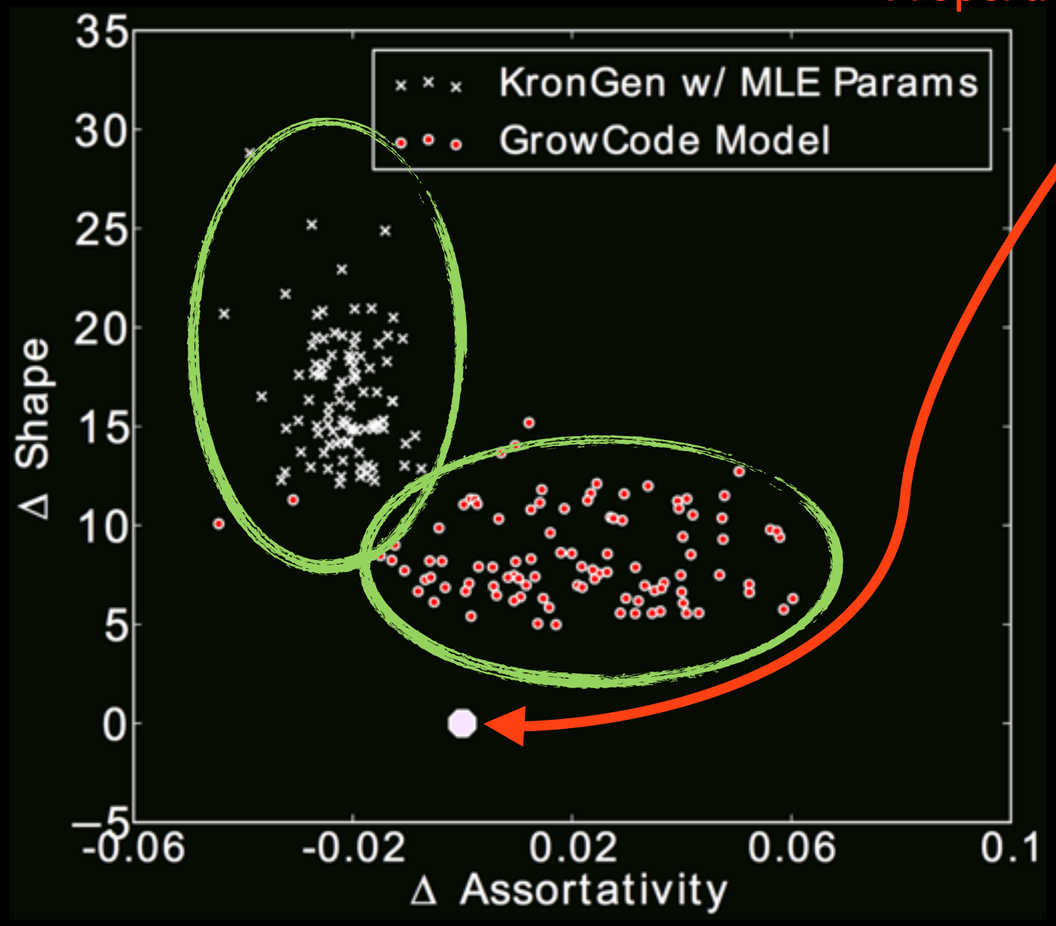
Want models (programs) that produce graphs whose feature vectors have high expected similarity with the target feature vector

Learning Models for Real-World Networks

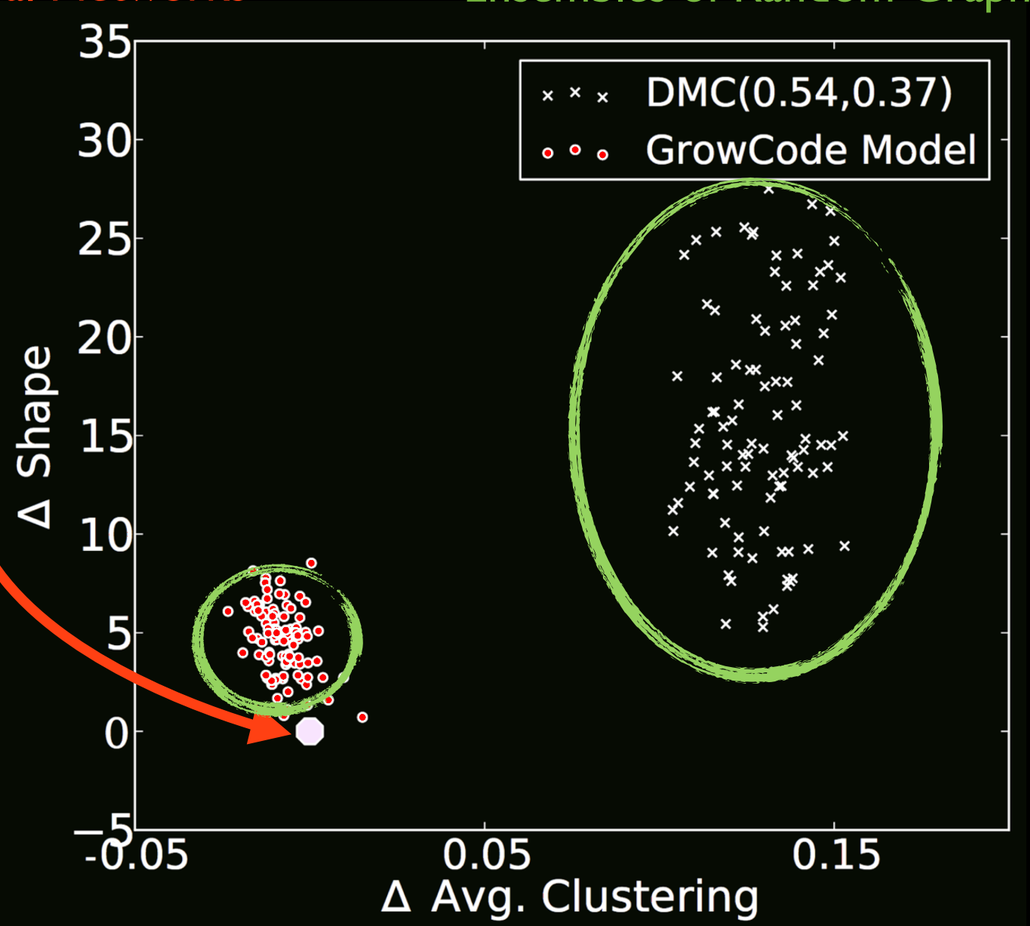
GrowCode Better Matches Pairs of Features

Properties of Real Networks

Ensembles of Random Graphs

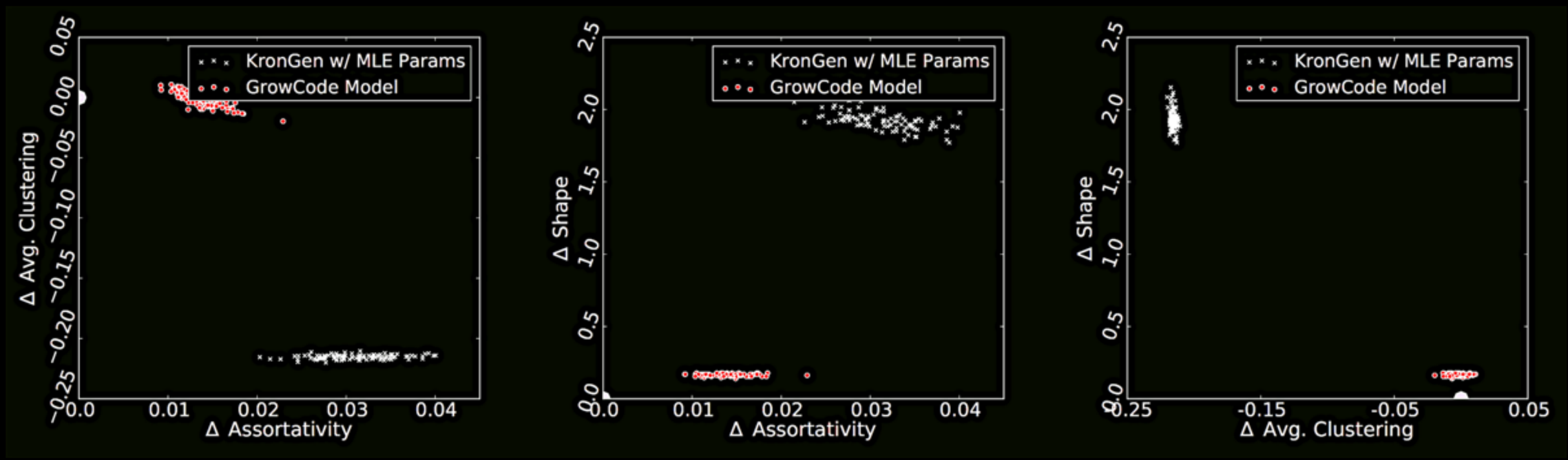


GWAS co-authorship network



PPI Network

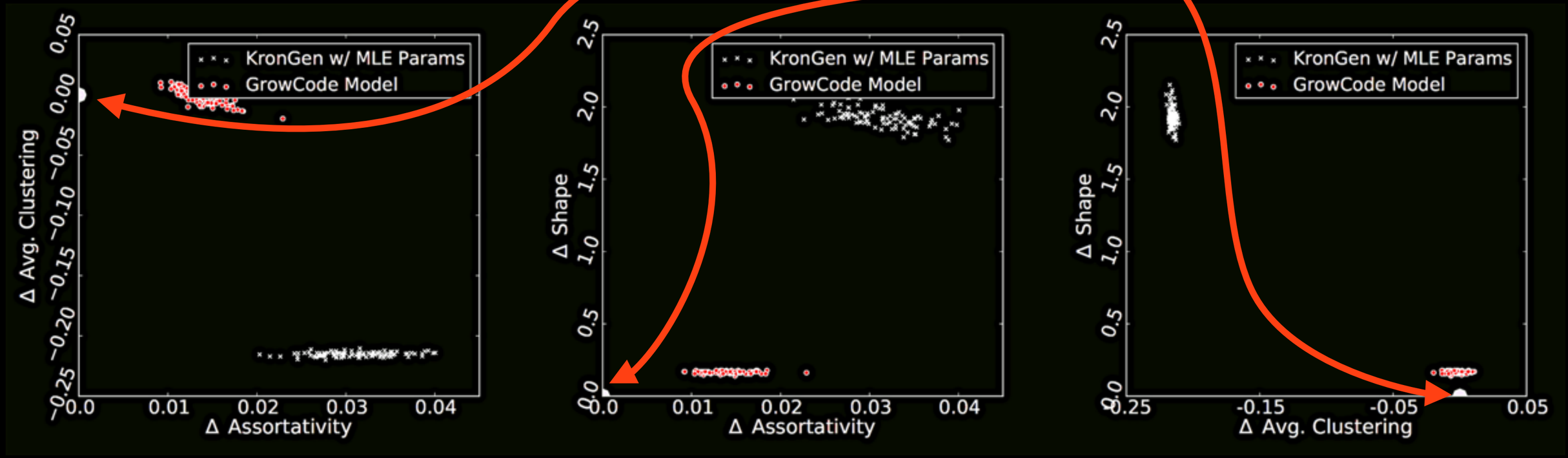
GrowCode Better Matches Triplets of Features



Internet Router Network

GrowCode Better Matches Triplets of Features

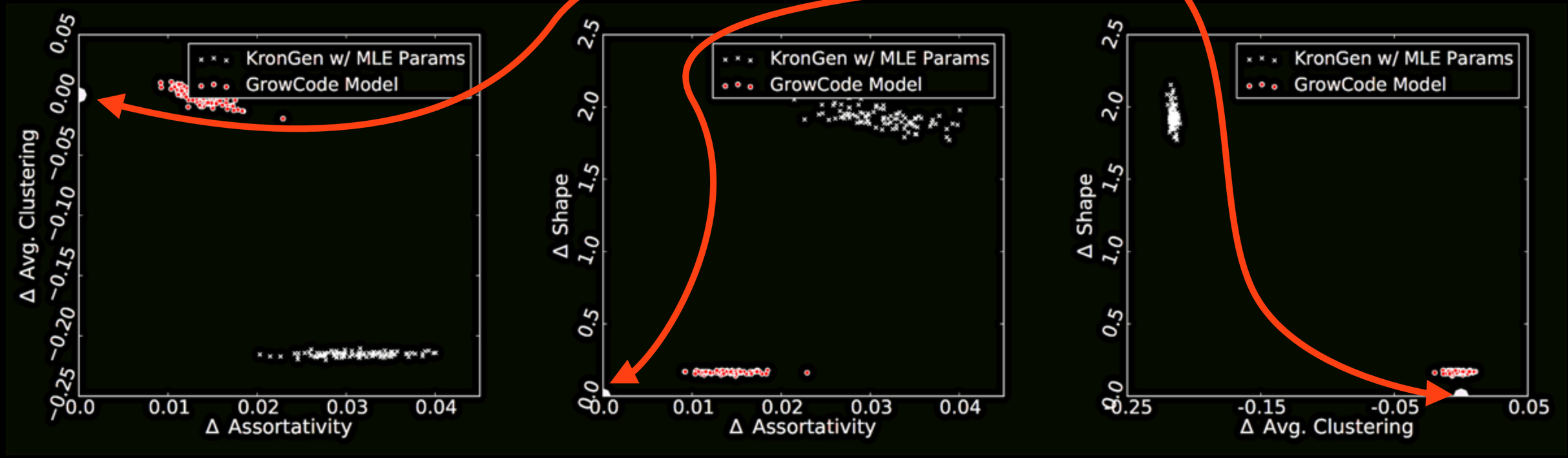
Properties of Real Networks



Internet Router Network

GrowCode Better Matches Triplets of Features

Properties of Real Networks



Social, technological, and biological!

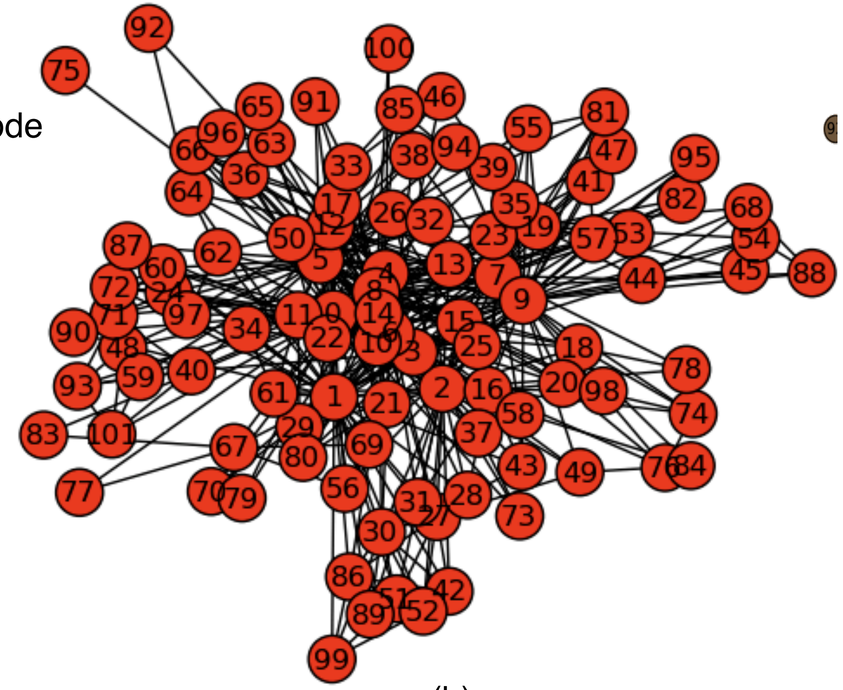
Internet Router Network

Random Walk with Restarts

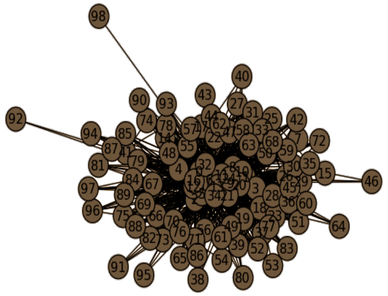
```
; RWR - Random Walk with Restart
; r3 is the start node; r2 is the new node; r1 is the current node
RNODE      ; r1 := random start node
SAVE       ; r3 := r1
COPY  ; copy r1 from an existing node
ZERO  ; remove all of r1s copied edge
SWAP  ; r2 := r1
STEP 100       ; do the random walk with restarts
   INC         ; increment attribute for node r1
   SIFA != 5   ; skip next statment if attribute for r1 != 50
   CREATE      ; create an edge between r1 and r2
   NEIGH       ; move to a random neighbor
   SCOIN 0.7   ; skip next statment if coin < r
   LOAD        ; r1 = r3 == start node
```

(a)

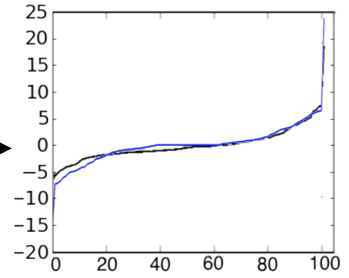
growcode model of network evolution



(b)



Spectra of adjacency matrix of (1) RWR model and (2) learned/optimized model



(c)

Conclusion

Novel representation of network growth models

Can express many existing models & define new ones

Automated process for learning models that grow graphs desired topological properties

Learned models can outperform (match better) than manually designed models with optimal parameters

Basic building blocks are interpretable -- entire model may not be

Direction for future work -- analyze programs to find interpretable motifs

Questions

- **Accuracy:** Can something be proved about the deviation between the distribution of graphs produced by a GrowCode* program and a hand-designed mechanism?
- **Completeness:** Can a set of properties and a GrowCode* language be defined such that one could show that any graph growth model with these properties can be expressed with GrowCode*?
- **Efficiency:** Is there a better optimization procedure than a GA?
- **Regularization:** How can one enforce "simple" models are determined?
- **Interpretation:** Can we mine learned programs to identify mechanism "motifs"?
- **Complexity**: Can we define a way to quantify how "easy" a property is to generate?

Thanks

[CCF-1053918, EF-0849899, and IIS-0812111]

[1R21AI085376]

Institute for Advanced Studies
New Frontiers Award

ALFRED P. SLOAN FOUNDATION

Research Fellowship to CK