Probabilistic Model Building GP with Belief Propagation

Hiroyuki Sato
Yoshihiko Hasegawa,
Danushka Bollegala
and Hitoshi Iba

THE UNIVERSITY OF TOKYO

June. 12, 2012

Outline

- Introduction
- POLE : Program Optimization with Linkage Estimation
- LBP : Loopy Belief Propagation
- POLE-BP : The proposed method
- Experiments & Discussion
- Conclusion

Outline

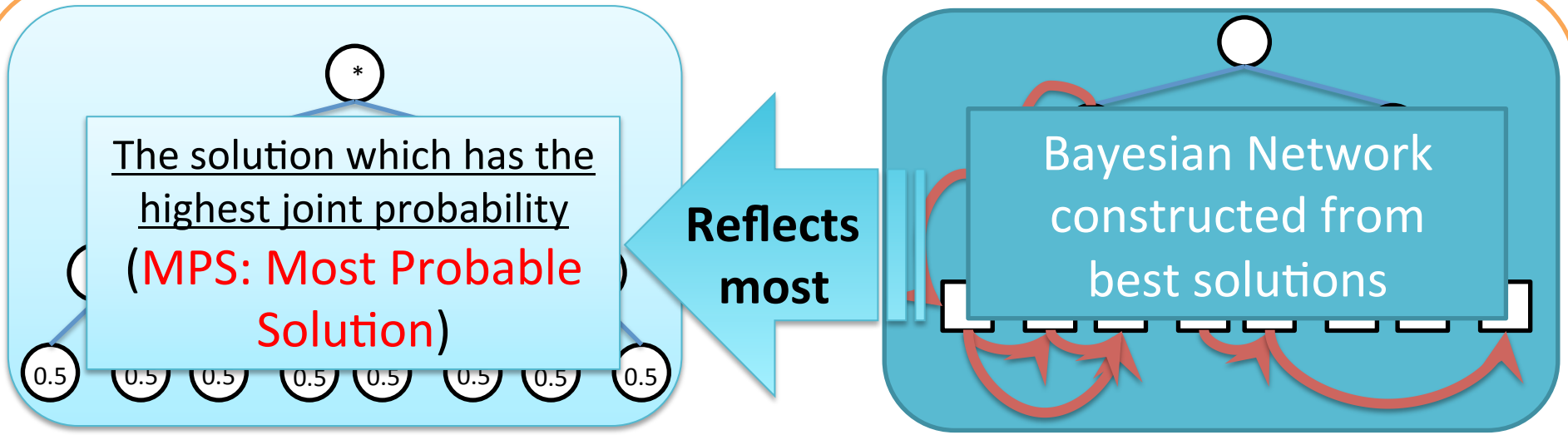
- Introduction
- POLE : Program Optimization with Linkage Estimation
- LBP : Loopy Belief Propagation
- POLE-BP : The proposed method
- Experiments & Discussion
- Conclusion

PMBGP: Probabilistic Model Building Genetic Programing (GP)

- EDAs deal with tree structures, like GP

1. PCFG (Probabilistic Context Free Grammar) based methods
   - Learn production rule probabilities
2. Prototype tree based methods (current research)
   - Convert full α-ary tree structures to 1 dimensional arrays in breadth-first order and apply GA-type EDAs

The weak point of prototype tree based PMBGP



traditional sampling does *not always* generate MPS

The same problem is pointed out in GA-type EDAs,
but sampling using Loopy Belief Propagation
is known as a way to overcome it

Motivation

- Improvement of sampling to generate always MPS will enhance PMBGP

Loopy Belief Propagation, which effects EDAs well, is the appropriate way to improve sampling of prototype tree based PMBGP

We propose POLE-BP, and compare its performance against POLE and Simple GP in 3 benchmark experiments

POLE-BP

POLE (a prototype tree based PMBGP) + **Loopy Belief Propagation**

Outline

- Introduction
- POLE : Program Optimization with Linkage Estimation
- LBP : Loopy Belief Propagation
- POLE-BP : The proposed method
- Experiments & Discussion
- Conclusion

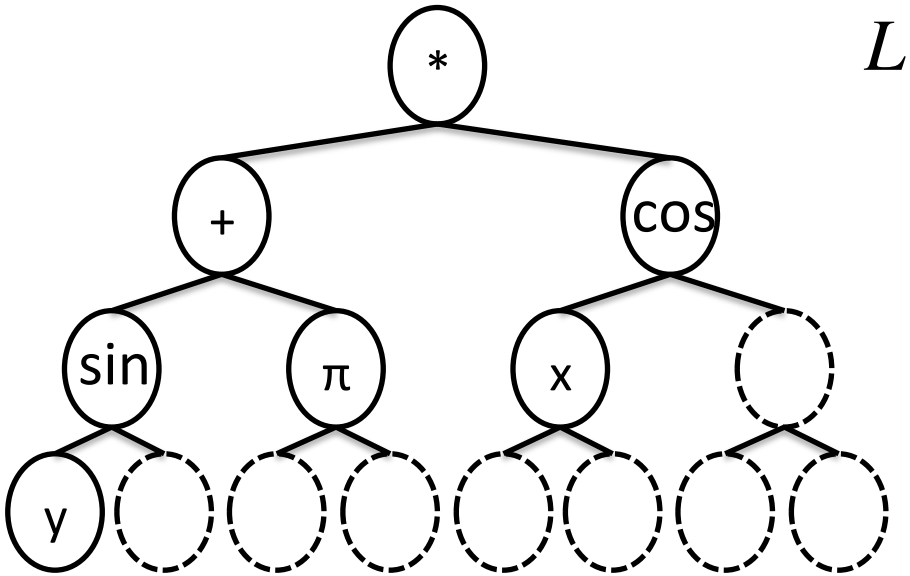
POLE : Program Optimization with Linkage Estimation

- Prototype tree based PMBGP
- Use Bayesian Network as probabilistic model
- Use *EPT (Expanded Parse Tree)* as chromosome
  - Guarantee syntactic correctness
  - Reduce symbols on trunk and conditional probability table size, and estimate probabilities more accurately

EPT : Expanded Parse Tree

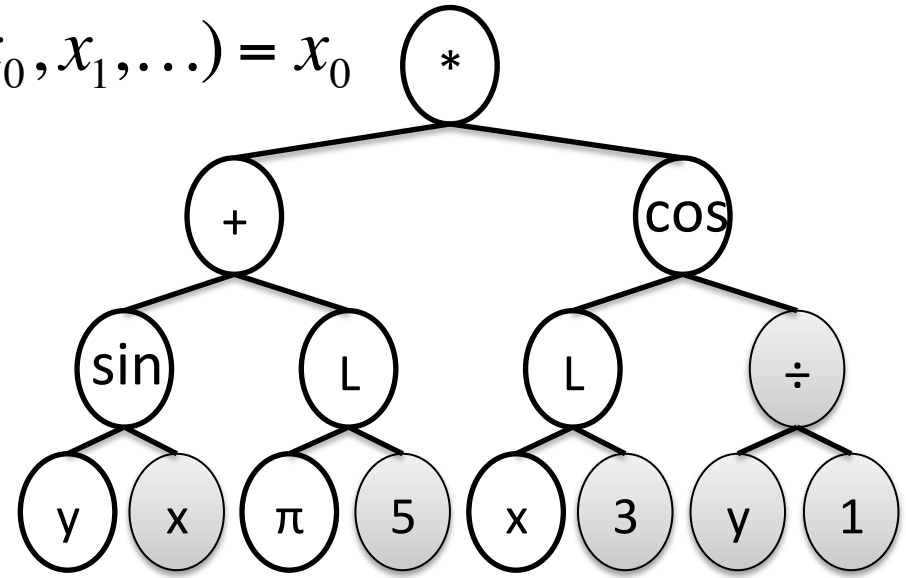
$$f(x,y) = (\sin(y) + \pi)\cos(x)$$

Normal GP tree



EPT (gray nodes are introns)

$$L(x_0, x_1, \ldots) = x_0$$



The details of POLE

1. Initialization
2. Evaluation
3. Selection
4. Construction of Bayesian Network by K2 algorithm
   - Bayesian information criteria (BIC) is used to evaluate networks
5. Generation of new individuals
   - Sampling ALL individuals by Probabilistic Logic Sampling (PLS)

Outline

- Introduction
- POLE : Program Optimization with Linkage Estimation
- LBP : Loopy Belief Propagation
- POLE-BP : The proposed method
- Experiments & Discussion
- Conclusion

LBP : Loopy Belief Propagation (Loopy max-sum)

- Infer MPS *approximately*, but succeeded in many applications: image processing, multi-agent system and *EDAs*
- Repeatedly update *Messages*, which are locally calculated joint probabilities
- Must be run on factor graph

Outline

- Introduction
- POLE : Program Optimization with Linkage Estimation
- LBP : Loopy Belief Propagation
- POLE-BP : The proposed method
- Experiments & Discussion
- Conclusion

POLE-BP: The proposed method

1. Initialization
2. Evaluation
3. Selection
4. Construction of Bayesian Network
5. Generation of new individuals by sampling
   - Sample M-1 individuals (M: population size)
6. Graph transformation
   - The Bayesian Network constructed at step 4 is transformed to the equivalent factor graph
7. Generate MPS by LBP on the factor graph

Comparison of individual generation

Conventional (POLE)

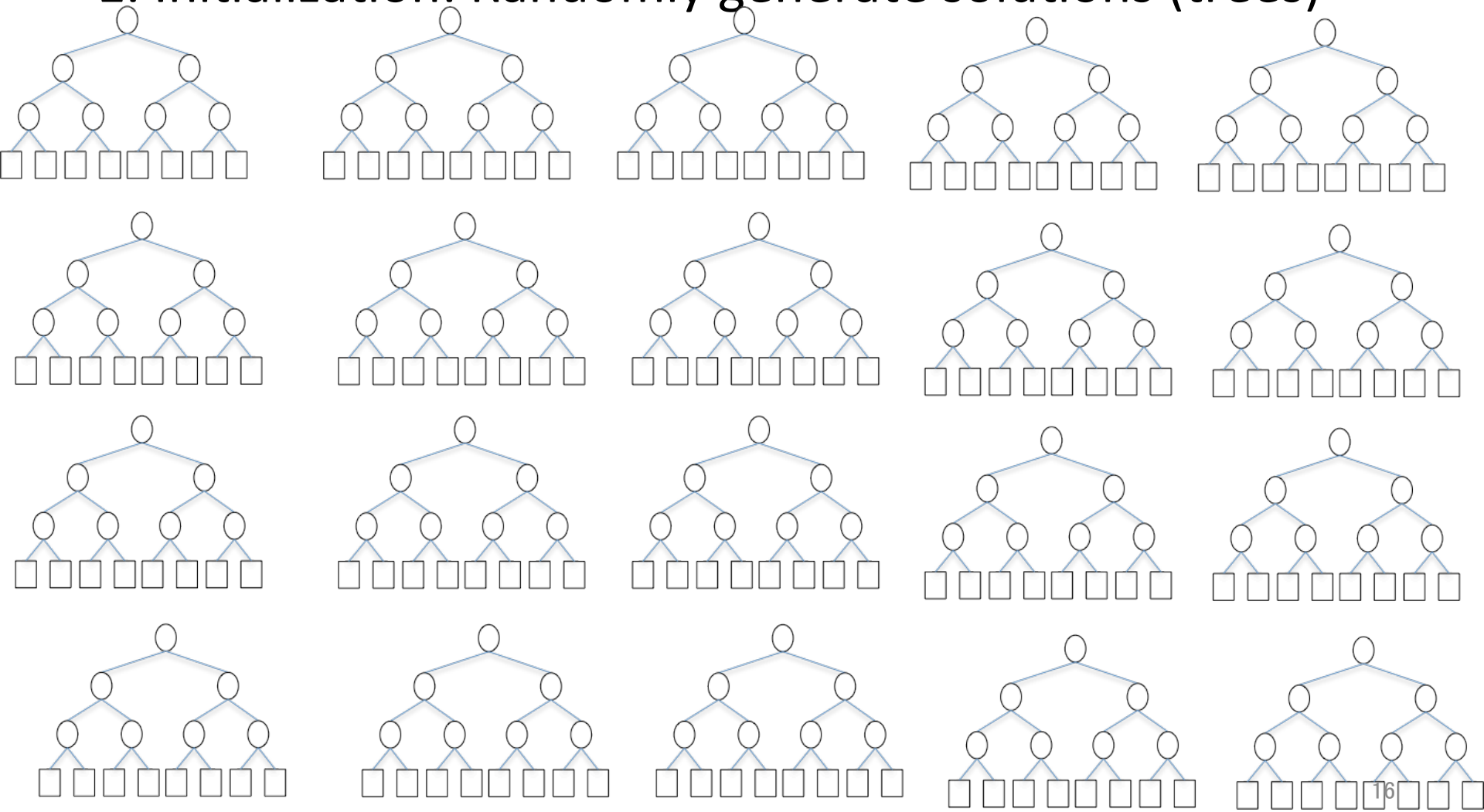
1. Building Bayesian Network
2. Sampling M individuals from Bayesian Network
   (M: population size)
3. M sampled individuals are carried to the next generation

Proposed (POLE-BP)

1. Building Bayesian Network
2. Sampling M-1 individuals from Bayesian Network
3. Transforming Bayesian Network to Factor Graph
4. Generating the individual having the highest joint probability (MPS) by LBP
5. M-1 sampled individuals and MPS generated by LBP are carried to the next generation

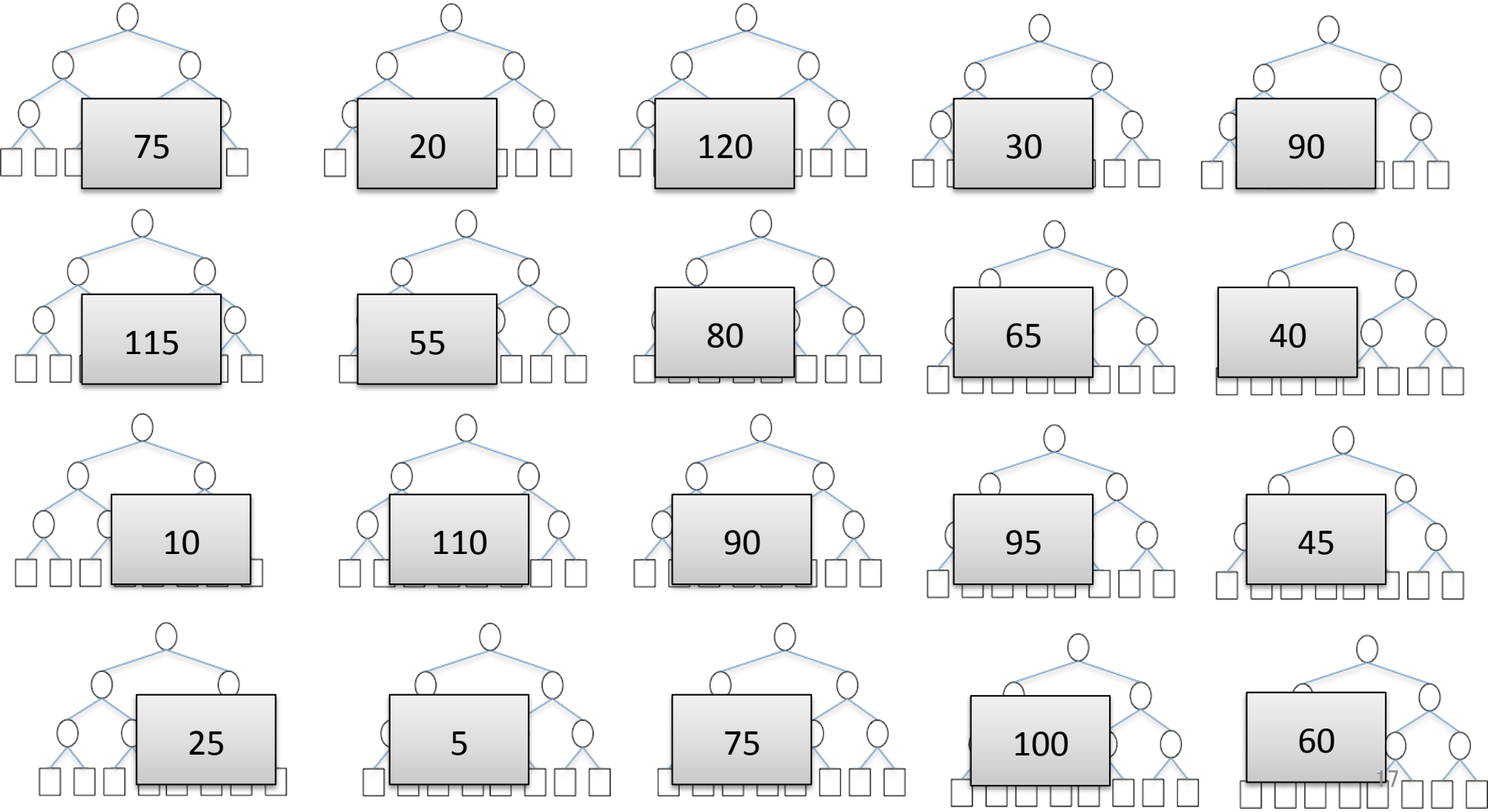
The details of POLE-BP

1. Initialization: Randomly generate solutions (trees)



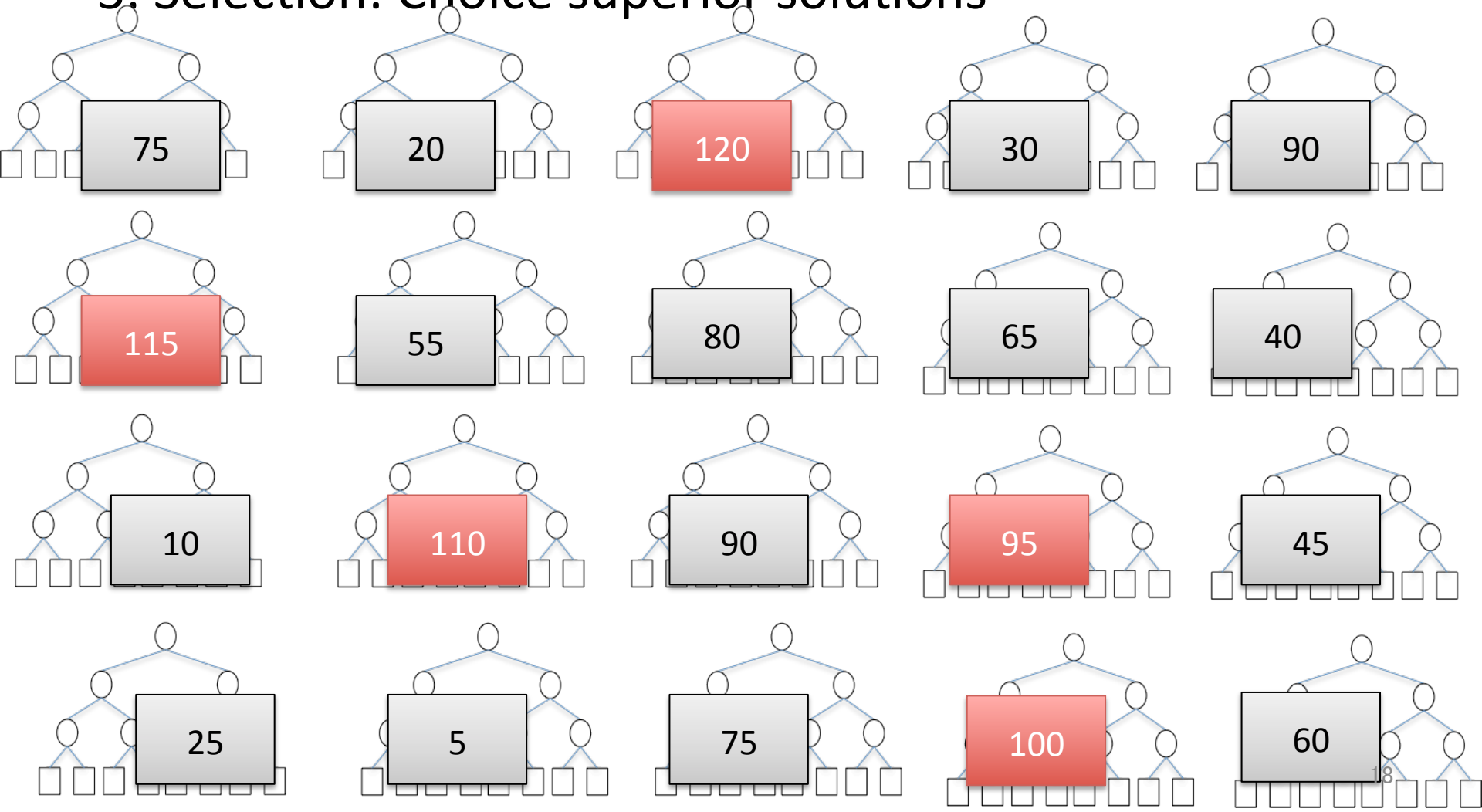
The details of POLE-BP

2. Evaluation: Evaluate solutions



The details of POLE-BP

3. Selection: Choice superior solutions



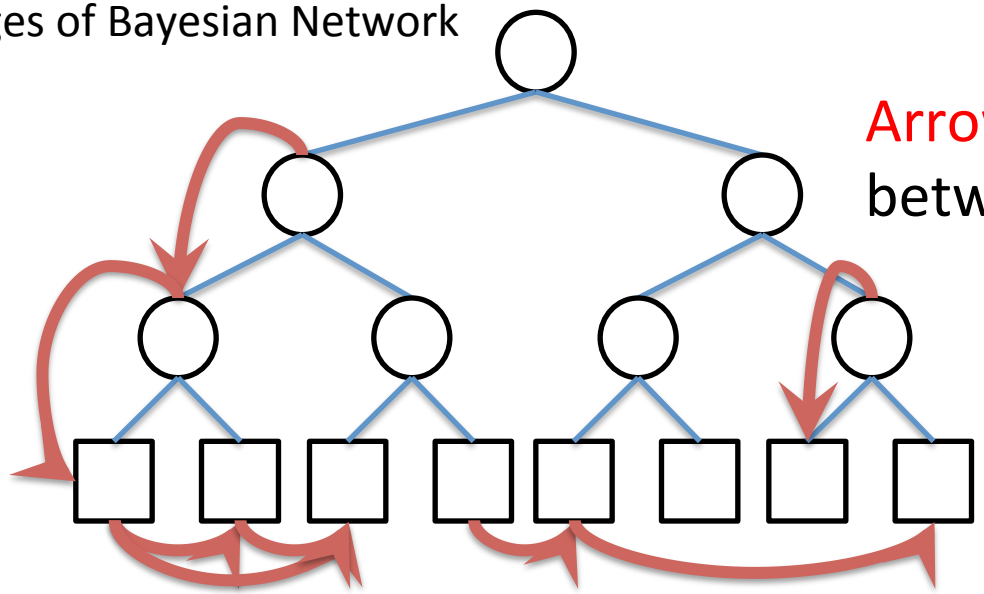
The details of POLE-BP

4. Construction of Bayesian Network on **tree structure**

Tree edge

K2 Algorithm

Edges of Bayesian Network



Arrows represent *dependencies* between nodes

Conditional Probabilities

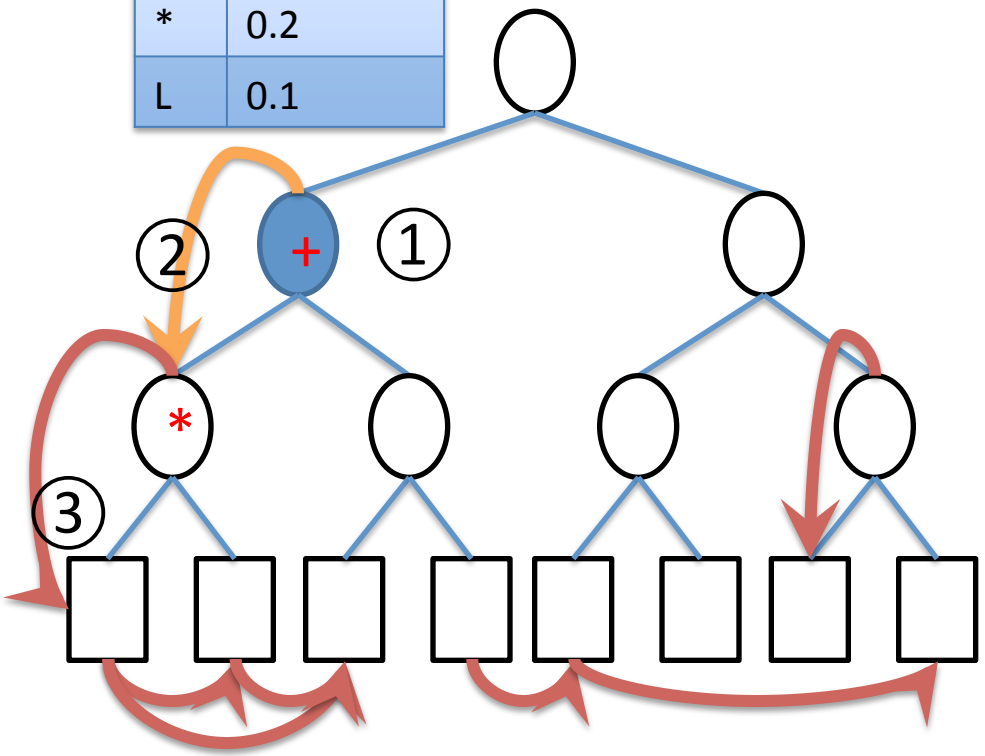
The details of POLE-BP

5. Sampling M-1 individuals (M: population size):

Sample from ancestors

Parent	Child	Conditional Prob
+	+	0.6
+	*	0.2
+	L	0.2
*	+	0.05
*	*	0.15
*	L	0.8
L	+	0.25
L	*	0.2
L	L	0.55

	Prior Prob
+	0.7
*	0.2
L	0.1



The details of POLE-BP

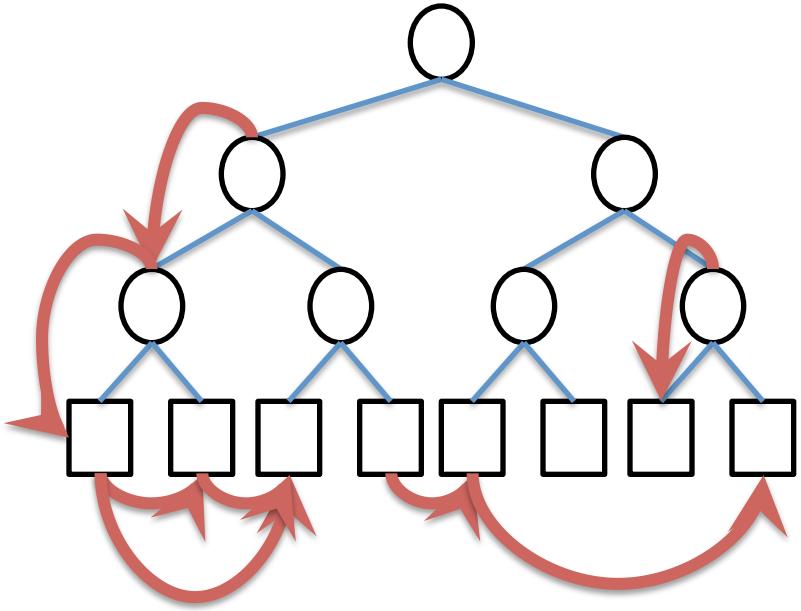
6. Graph Transformation

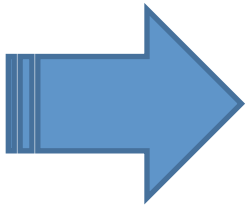
Bayesian Network : Directed Graph

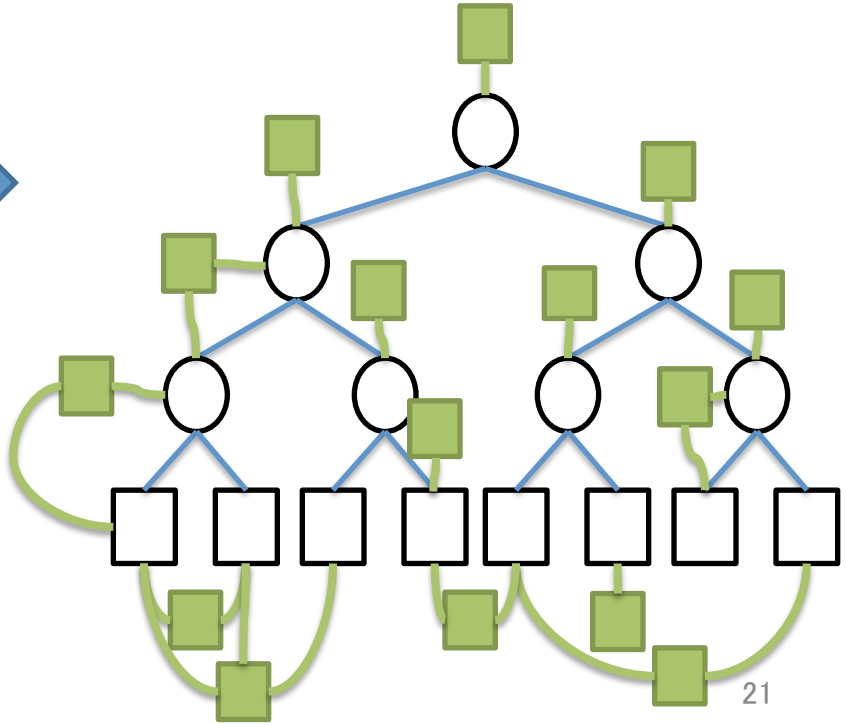
- Directed Edge: conditional probability
- Nodes without parents: prior probability

Factor Graph : Undirected Graph

- Factor: prior and conditional probability
- Undirected Edge: Only represent connection

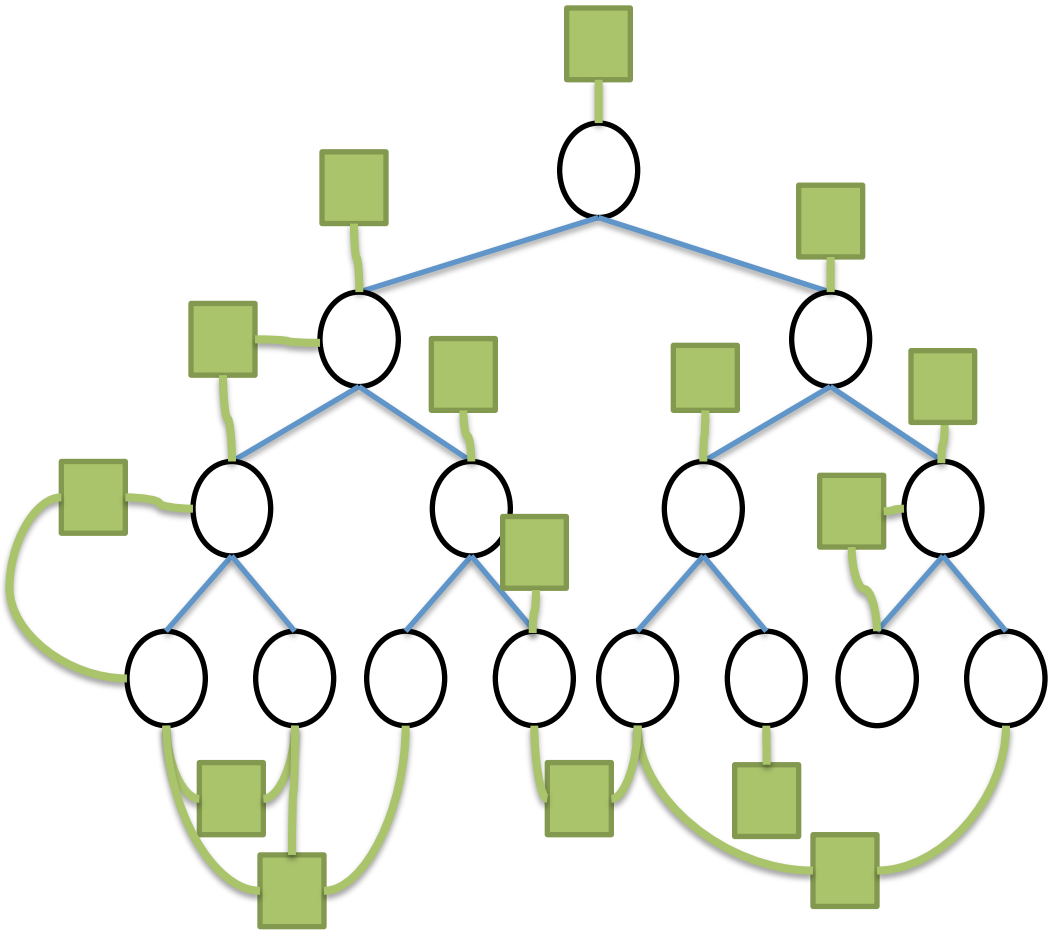






The detail of POLE-BP

7. Generate MPS by LBP on the factor graph



1. Initialize all messages to 0

2. Message Passing

factor f variable x

Messages from factor to variable

leafs $\mu_{f\to x}(x) = \ln f(x)$

The others $\mu_{f\to x}(x) = \max_{x,x_1,\ldots,x_M} \left[\ln f(x,x_1,\ldots x_M) + \sum_{m\in ne(f_s)\setminus x} \mu_{x_m\to f}(x_m) \right]$

Messages from variable to factor

leafs $\mu_{x\to f}(x) = 0$

The others $\mu_{x\to f}(x) = \alpha_{xf} + \sum_{l\in ne(x)\setminus f} \mu_{f_l\to x}(x)$

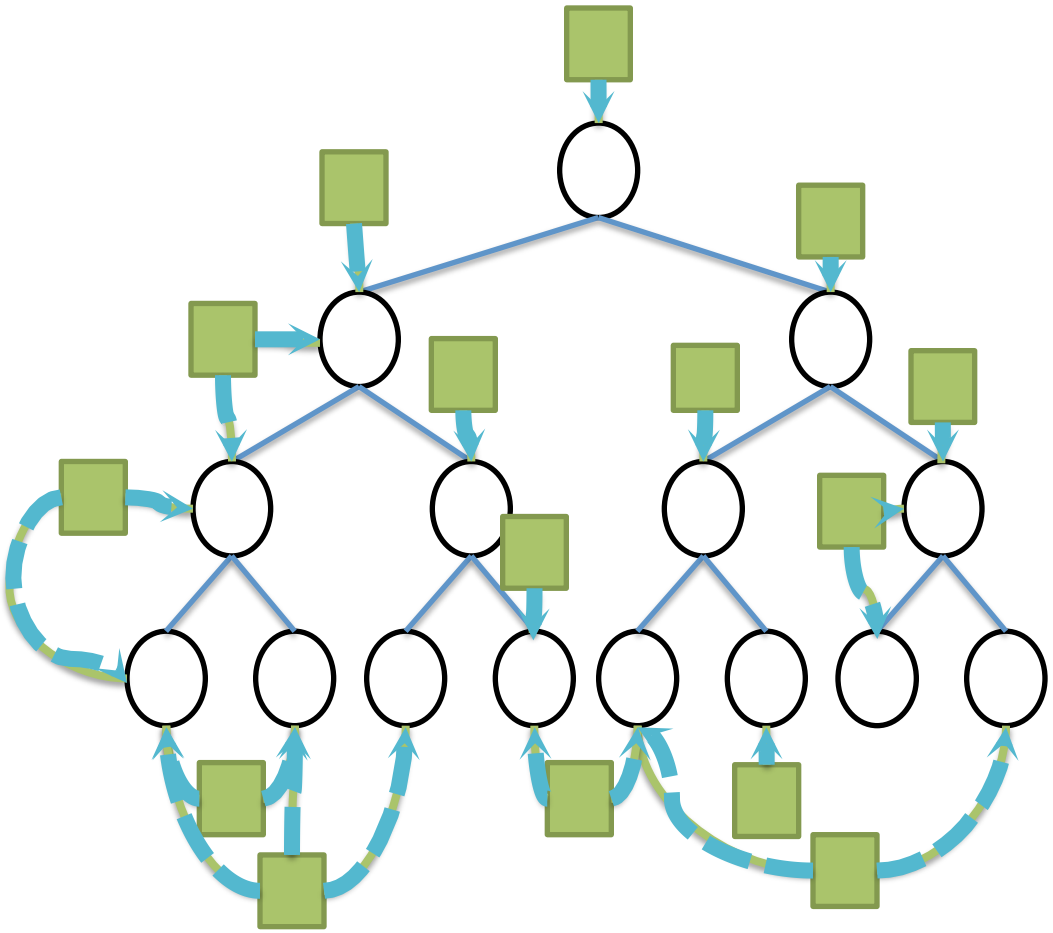
3. Get MPS

Joint probability

$$x^{\max} = \arg\max_x \left[\sum_{s\in ne(x)} \mu_{f_s\to x}(x) \right]$$

The detail of POLE-BP

7. Generate MPS by LBP on the factor graph



1. Initialize all messages to 0

2. Message Passing

factor f variable x

Messages from factor to variable

leafs $\mu_{f\to x}(x) = \ln f(x)$

The others $\mu_{f\to x}(x) = \max_{x,x_1,\ldots,x_M} \left[\ln f(x,x_1,\ldots x_M) + \sum_{m\in ne(f_s)\setminus x} \mu_{x_m\to f}(x_m) \right]$

Messages from variable to factor

leafs $\mu_{x\to f}(x) = 0$

The others $\mu_{x\to f}(x) = \alpha_{xf} + \sum_{l\in ne(x)\setminus f} \mu_{f_l\to x}(x)$

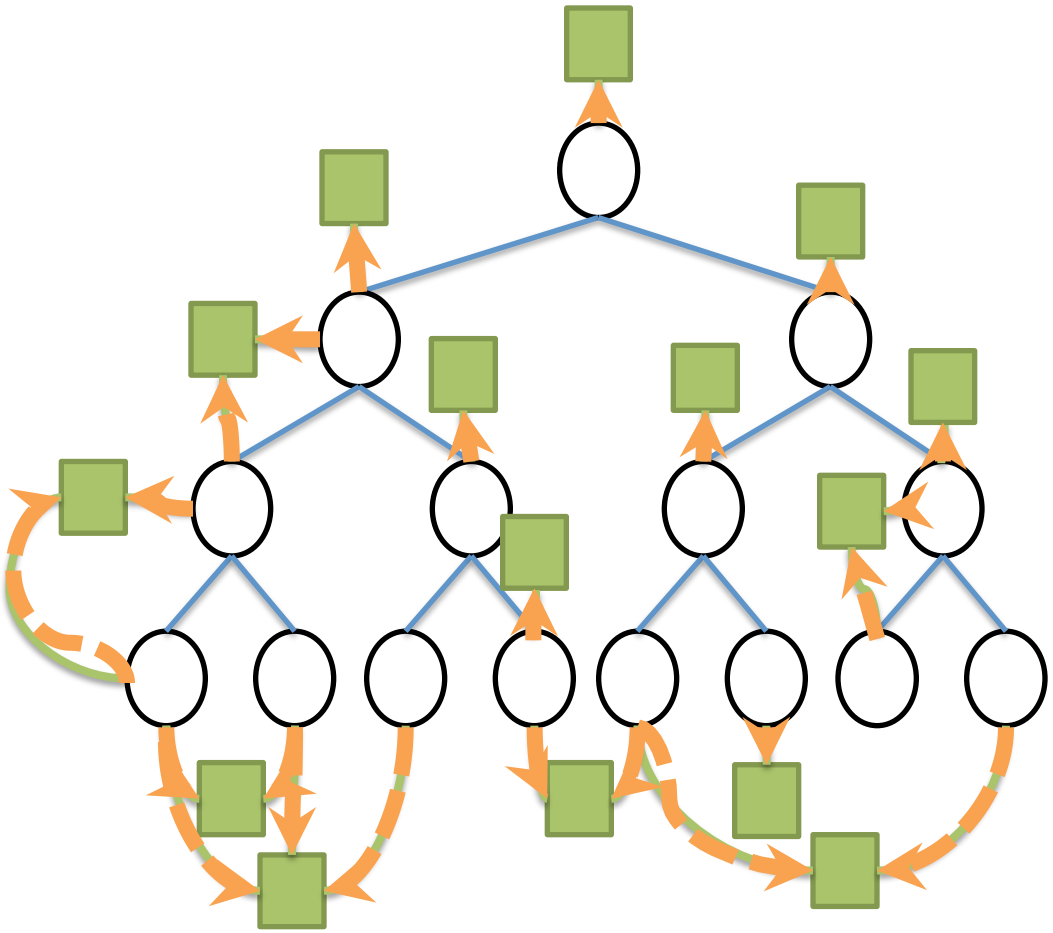
3. Get MPS

Joint probability

$$x^{\max} = \arg\max_{x} \left[\sum_{s\in ne(x)} \mu_{f_s\to x}(x) \right]$$

The detail of POLE-BP

7. Generate MPS by LBP on the factor graph



1. Initialize all messages to 0

2. Message Passing

factor f variable x

Messages from factor to variable

leafs $\mu_{f\to x}(x) = \ln f(x)$

The others $\mu_{f\to x}(x) = \max_{x,x_1,\ldots,x_M} \left[\ln f(x,x_1,\ldots x_M) + \sum_{m\in ne(f_s)\setminus x} \mu_{x_m\to f}(x_m) \right]$

Messages from variable to factor

leafs $\mu_{x\to f}(x) = 0$

The others $\mu_{x\to f}(x) = \alpha_{xf} + \sum_{l\in ne(x)\setminus f} \mu_{f_l\to x}(x)$

3. Get MPS

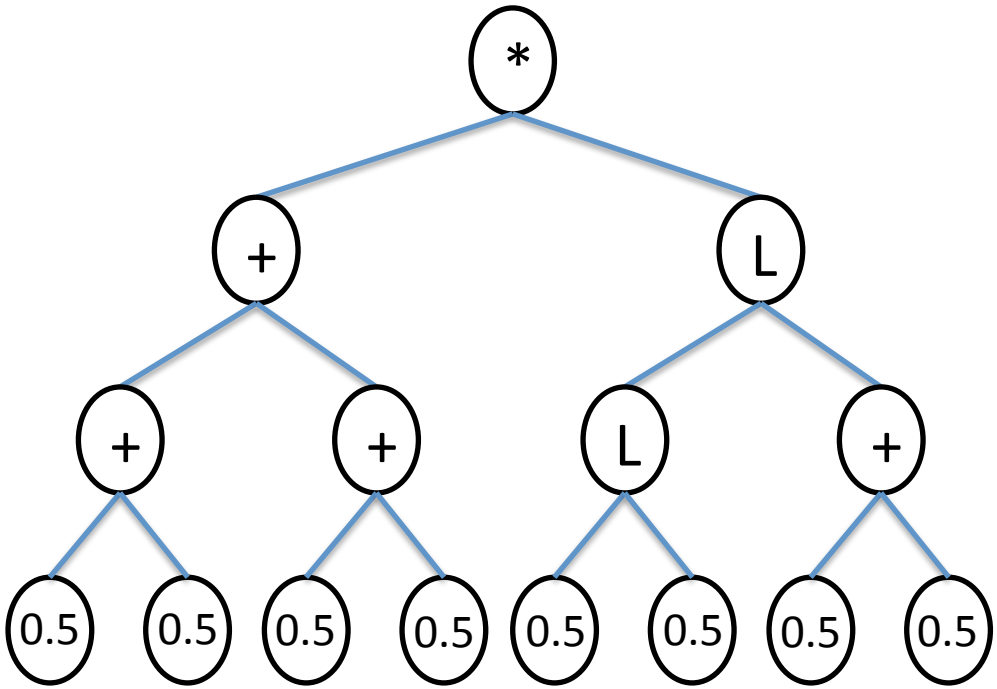
Joint probability

$$x^{\max} = \arg\max_x \left[\sum_{s\in ne(x)} \mu_{f_s\to x}(x) \right]$$

The detail of POLE-BP

7. Generate MPS by LBP on the factor graph

Most Probable Solution



1. Initialize all messages to 0

2. Message Passing

factor f variable x

Messages from factor to variable

leafs $\mu_{f\to x}(x) = \ln f(x)$

The others $\mu_{f\to x}(x) = \max_{x,x_1,\ldots,x_M}\left[\ln f(x,x_1,\ldots x_M) + \sum_{m\in ne(f_s)\setminus x}\mu_{x_m\to f}(x_m)\right]$

Messages from variable to factor

leafs $\mu_{x\to f}(x) = 0$

The others $\mu_{x\to f}(x) = \alpha_{xf} + \sum_{l\in ne(x)\setminus f}\mu_{f_l\to x}(x)$

3. Get MPS

Joint probability

$$x^{\max} = \arg\max_x\left[\sum_{s\in ne(x)}\mu_{f_s\to x}(x)\right]$$

Outline

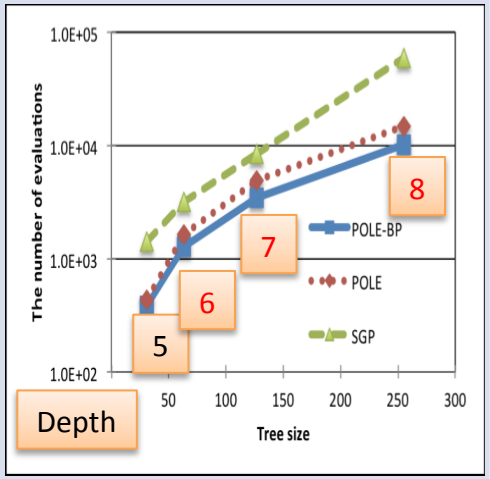
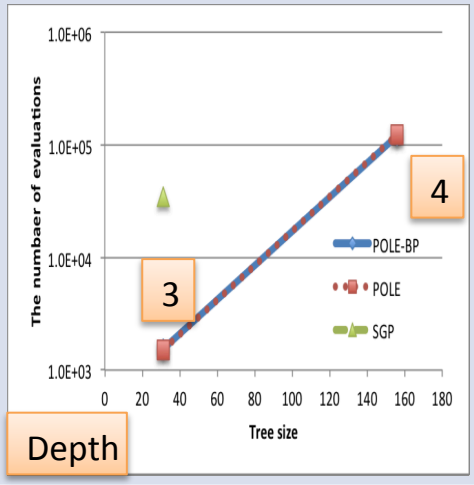
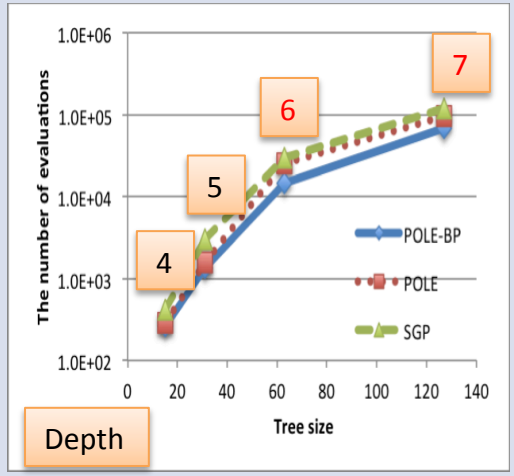
- Introduction
- POLE : Program Optimization with Linkage Estimation
- LBP : Loopy Belief Propagation
- POLE-BP : The proposed method
- Experiments & Discussion
- Conclusion

Experiments

- Benchmark on 3 problems with different characters:
  - MAX
  - Deceptive MAX (DMAX) (strong deceptiveness)
  - Royal Tree (many symbols)
- Compared the performance of POLE-BP(proposed), POLE and Simple GP(no mutation)
  - The relation between the average number of evaluations and tree size (the number of nodes)

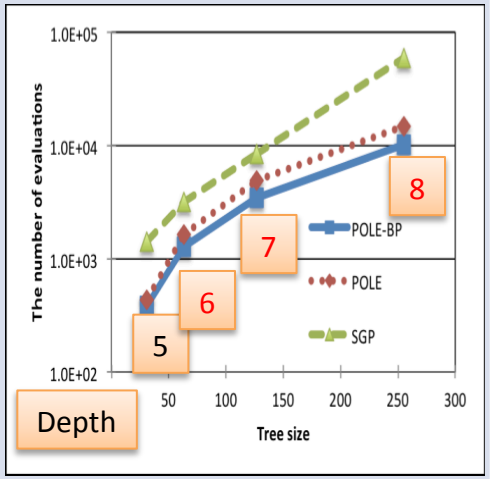
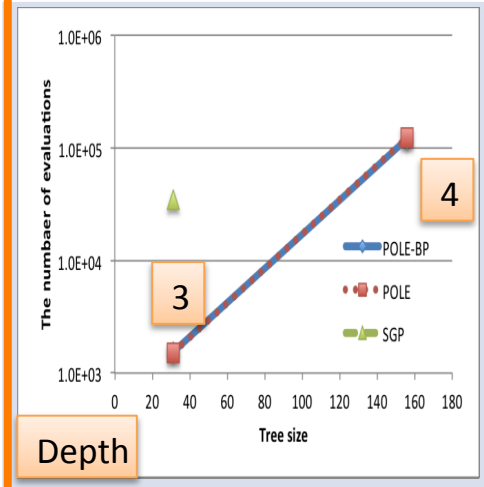
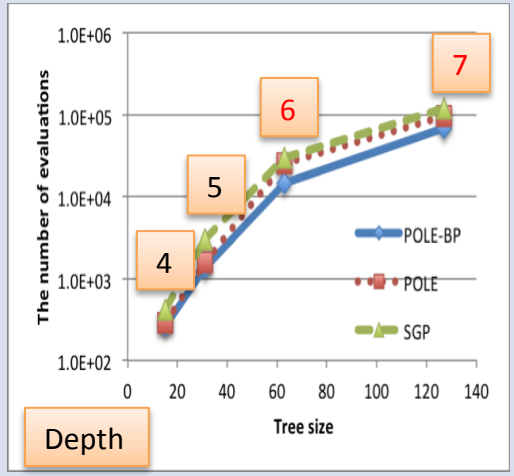
The whole result (The number of evaluations)

POLE-BP(Proposed) | POLE(Conventional) | Simple GP

	MAX (Normal)	DMAX (Deceptiveness)	Royal Tree (Many symbols)
The number of evaluations Versus Tree size(the number of nodes)			
Statistical significance & Decreasing rate	• POLE-BP is better for depth = 6, 7, 8 at 1% significance level. • Average 23.7% decreased.	• No difference.	• POLE-BP is better for depth = 6, 7 at 1% significance level. • Average 23.6% decreased.

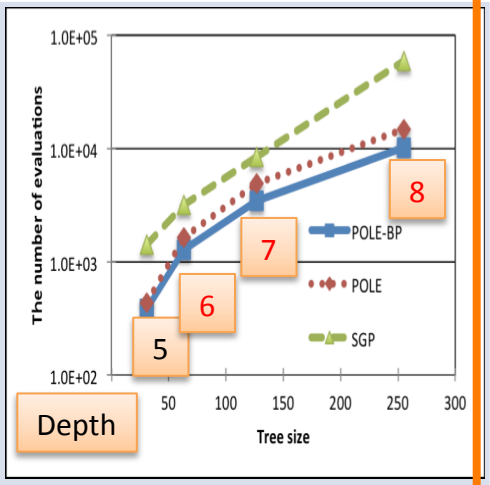
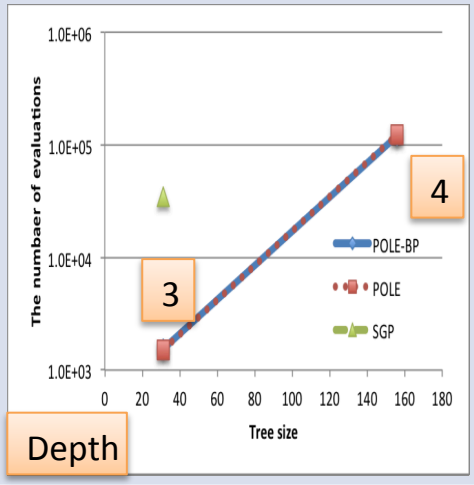
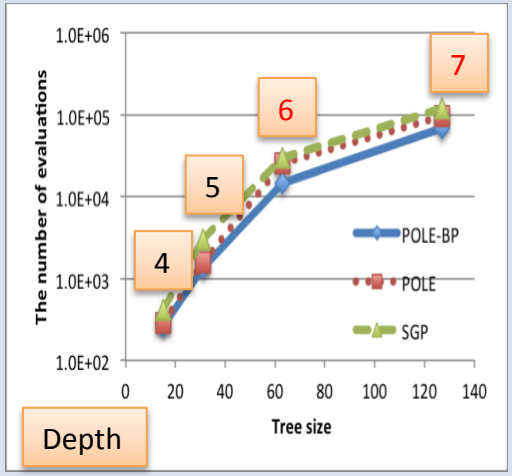
The whole result (The number of evaluations)

POLE-BP(Proposed) | POLE(Conventional) | Simple GP

	MAX (Normal)	DMAX (Deceptiveness)	Royal Tree (Many symbols)
The number of evaluations Versus Tree size(the number of nodes)			
Statistical significance & Decreasing rate	• POLE-BP is better for depth = 6, 7, 8 at 1% significance level. • Average 23.7% decreased.	• No difference.	• POLE-BP is better for depth = 6, 7 at 1% significance level. • Average 23.6% decreased.

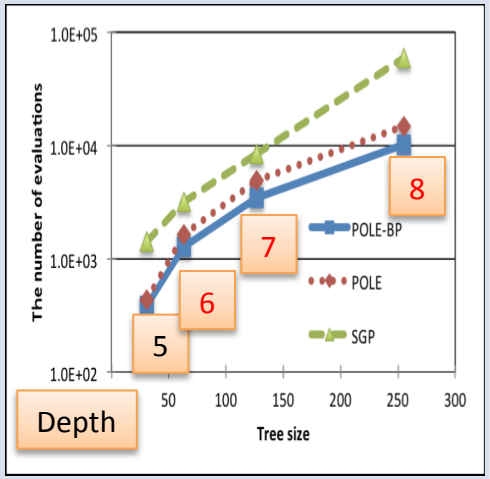
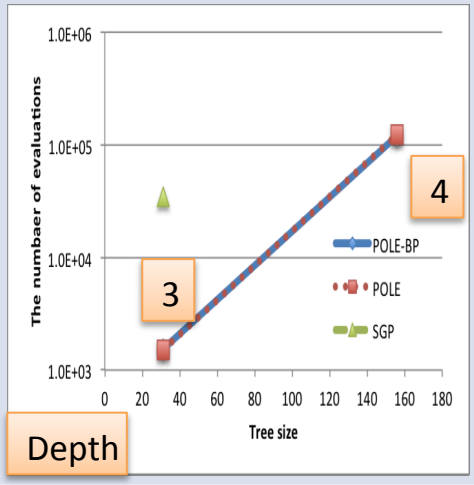
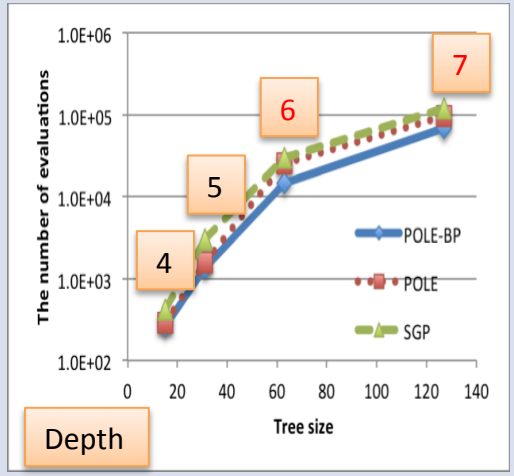
The whole result (The number of evaluations)

POLE-BP(Proposed) | POLE(Conventional) | Simple GP

	MAX (Normal)	DMAX (Deceptiveness)	Royal Tree (Many symbols)
The number of evaluations Versus Tree size(the number of nodes)			
Statistical significance & Decreasing rate	• POLE-BP is better for depth = 6, 7, 8 at 1% significance level. • Average 23.7% decreased.	• No difference.	• POLE-BP is better for depth = 6, 7 at 1% significance level. • Average 23.6% decreased.

The whole result (The number of evaluations)

POLE-BP(Proposed) | POLE(Conventional) | Simple GP

	MAX (Normal)	DMAX (Deceptiveness)	Royal Tree (Many symbols)
The number of evaluations Versus Tree size(the number of nodes)			
Statistical significance & Decreasing rate	• POLE-BP is better for depth = 6, 7, 8 at 1% significance level. • Average 23.7% decreased.	• No difference.	• POLE-BP is better for depth = 6, 7 at 1% significance level. • Average 23.6% decreased.

Outline

- Introduction
- POLE : Program Optimization with Linkage Estimation
- LBP : Loopy Belief Propagation
- POLE-BP : The proposed method
- Experiments & Discussion
- Conclusion

Conclusion

- We proposed the hybrid PMBGP combining POLE and Loopy Belief Propagation
- LBP reduces the average number of evaluations to get the optimum solution in PMBGP as well as GA-type EDAs
- LBP enhances the search ability in small population

Thank you for your attention

Hiroyuki SATO
sato@iba.t.u-tokyo.ac.jp

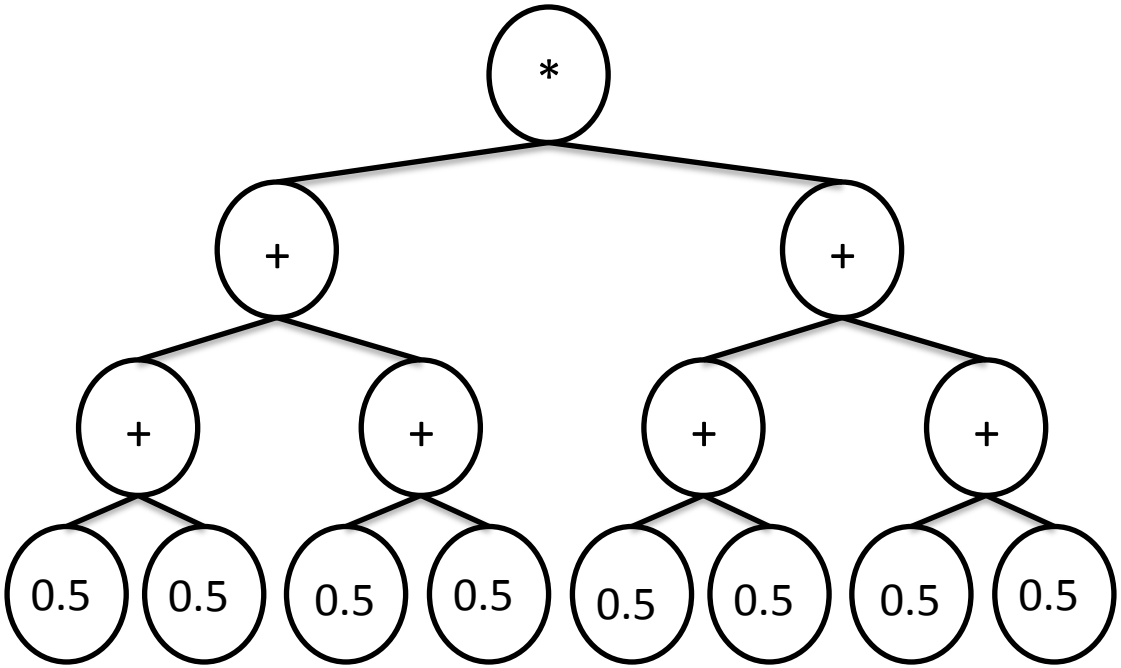
Future work

- Analyze how LBP works in PMBGP
- Real world application

MAX problem

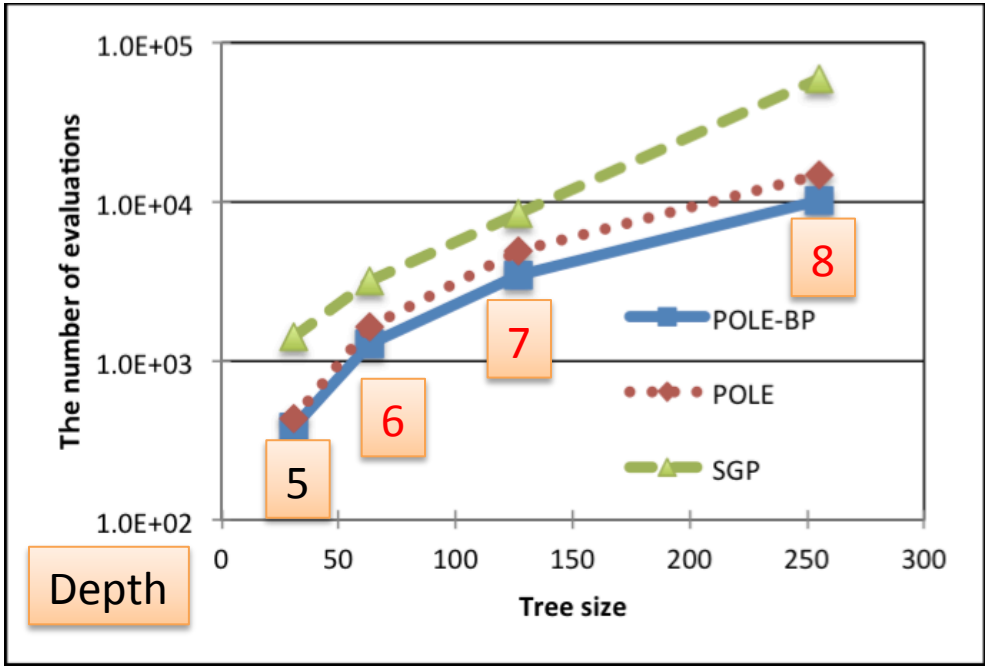
- Search the function which returns the biggest value $F = \{+, *\}, \quad T = \{0.5\}$

The optimum solution : depth 4



MAX : Results and discussion

- Difference between POLE-BP and POLE for depth = 6, 7, 8 is statically significant at 1% significant level
- MAX problem has no deceptiveness
  - MPS always accelerates to get an optimum solution

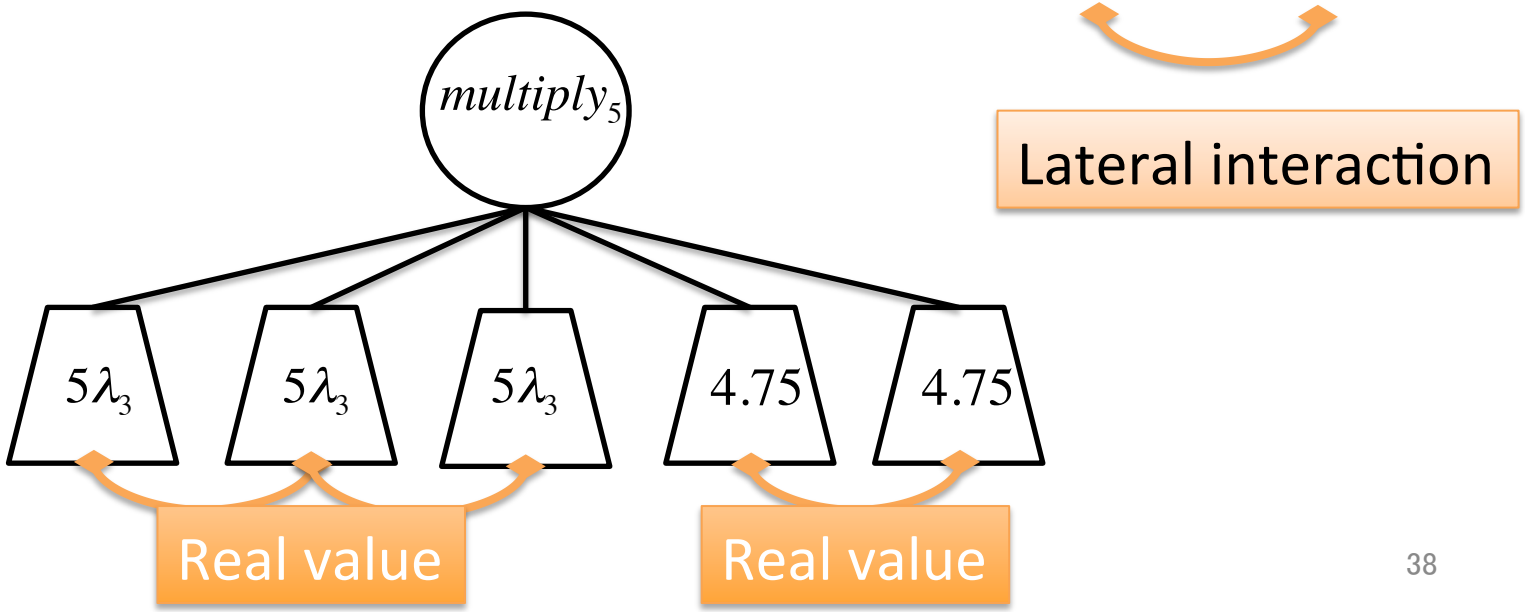


DMAX : Deceptive MAX problem

- Search the function which returns the biggest *real* value

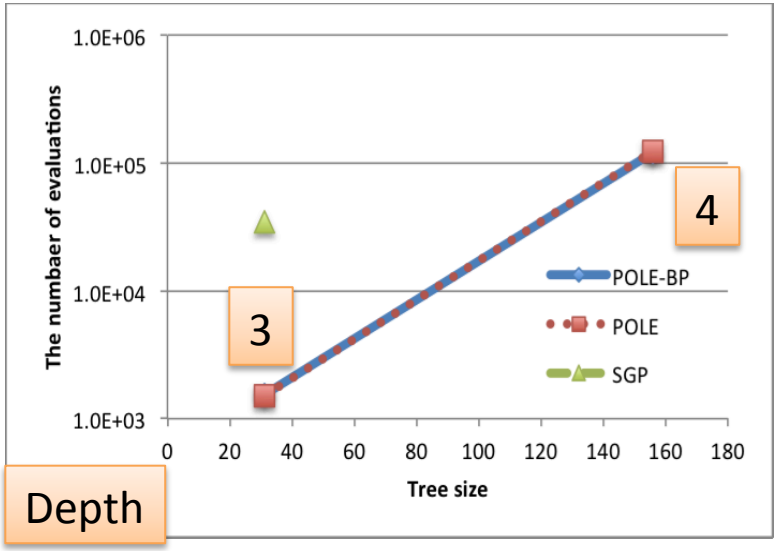
$$F = \{add_5, multiply_5\}$$
$$T = \{\lambda_3, 0.95\}$$
$$\lambda_3 = \left(-\frac{1}{2} + \frac{i\sqrt{3}}{2}\right)$$

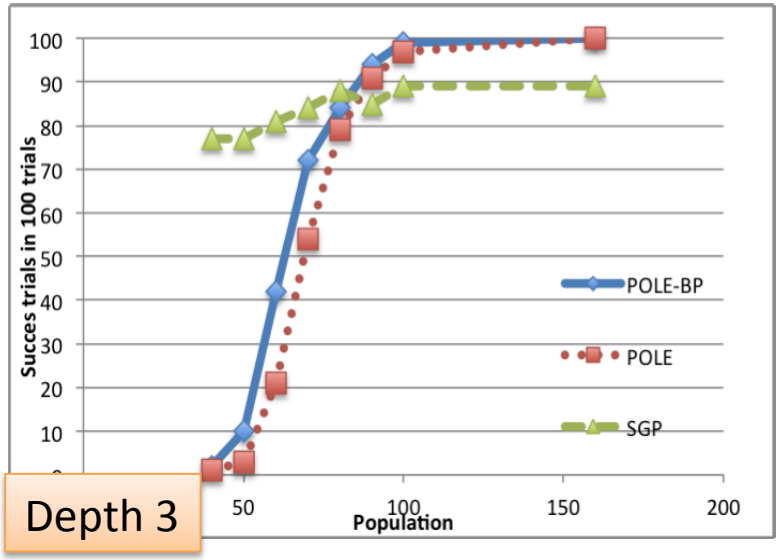
An optimum solution : depth 3



DMAX : Results and discussion

- No difference between POLE-BP and POLE in the average number of evaluations at 1% significant level
  - Because of deceptiveness, MPS does not work well to get an optimum
- In small population size, POLE-BP tends to get an optimal solution more than POLE
  - Because of sampling bias, sampling does not work well in small population. Therefore, LBP is effective in small population.



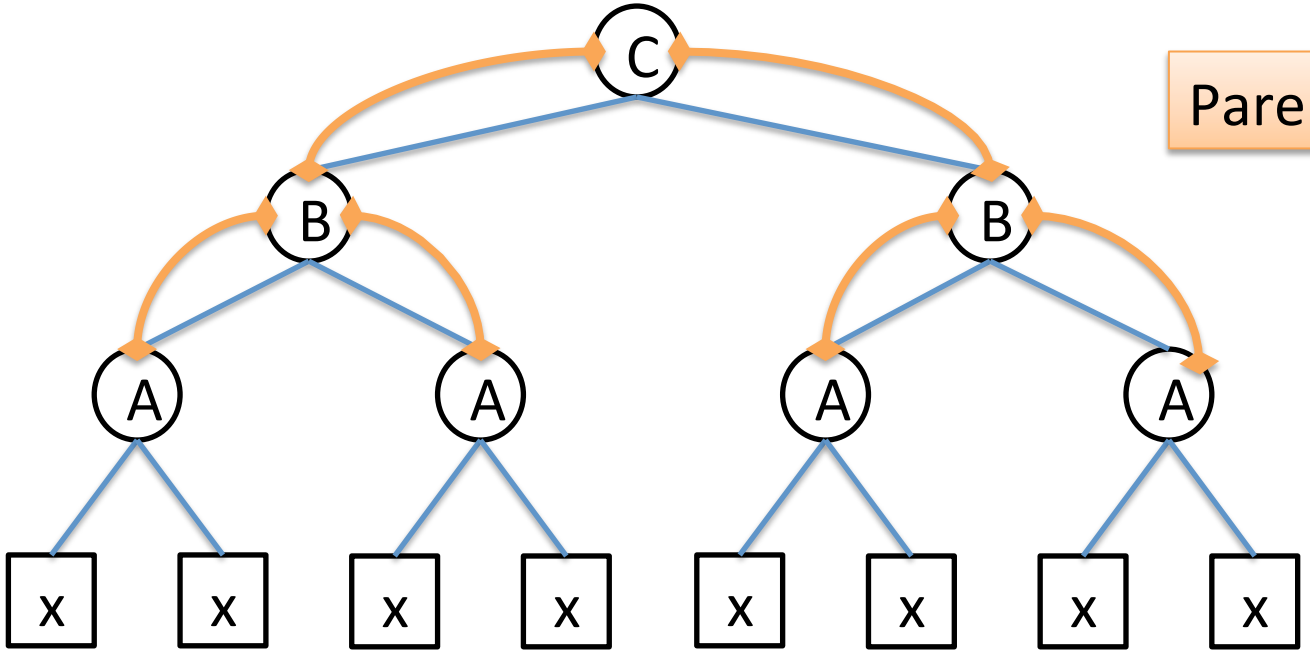


Royal tree problem

- Search *Perfect Tree* $F = \{A, B, C, D, E, F\}$

$$T = \{x\}, \quad x = 1$$

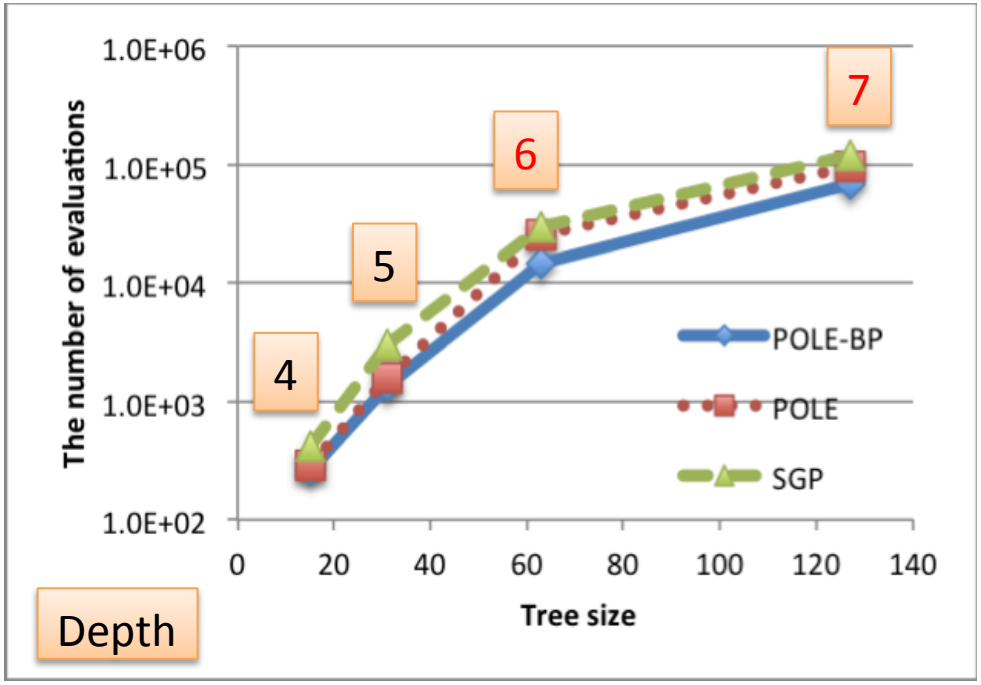
The optimum solution(Perfect Tree) : depth 4



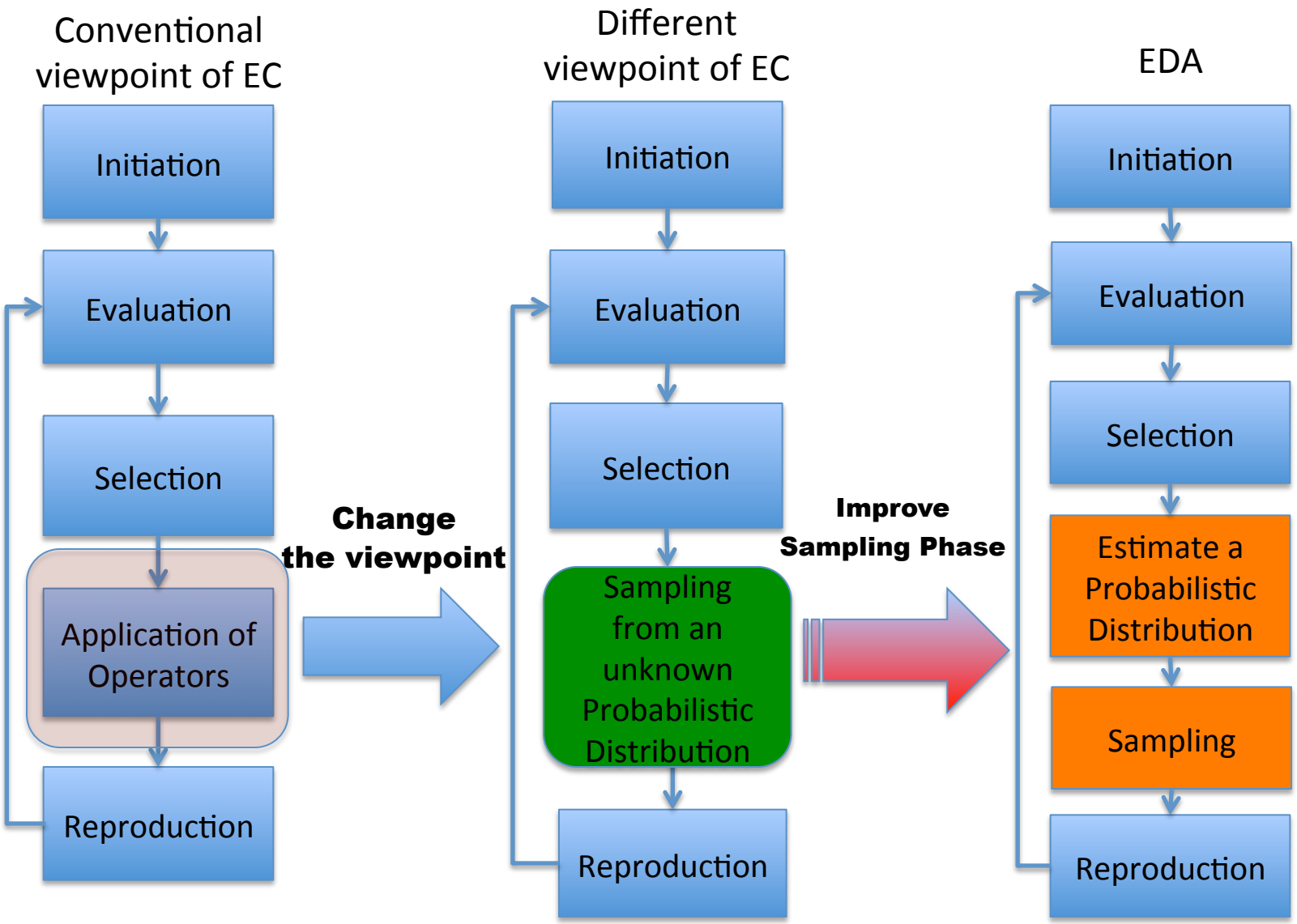
Parent-child interaction

Royal tree : Results and discussion

- Difference between POLE-BP and POLE for depth 6, 7 is statically significant at 1% significant level.
- Royal tree uses more symbols (=depth) than MAX (3) and DMAX (4)
  - MPS generation by sampling is difficult, so MPS generation by LBP at each generation is effective

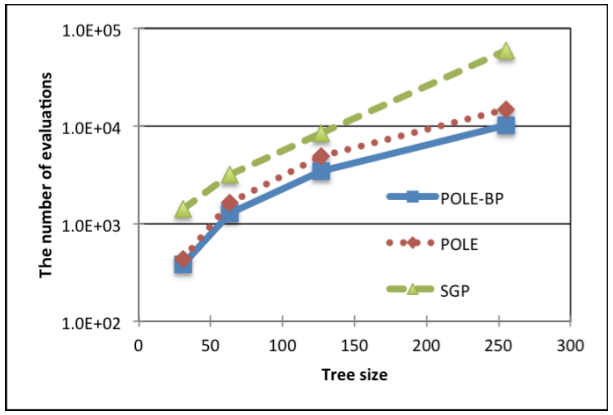


EDAs: Estimation Distribution Algorithms

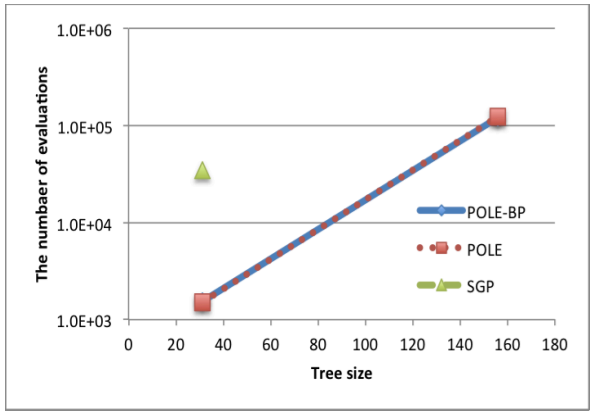


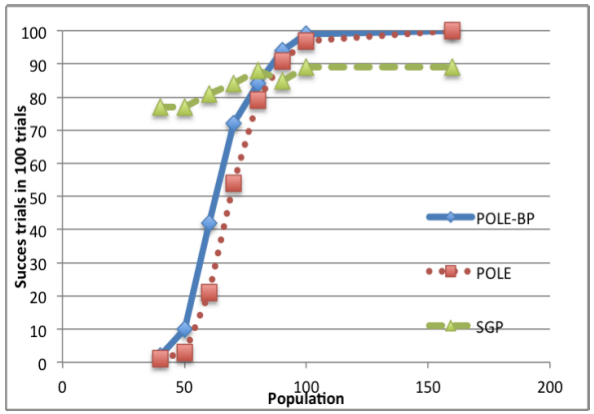
Experiments on 3 benchmarks

MAX

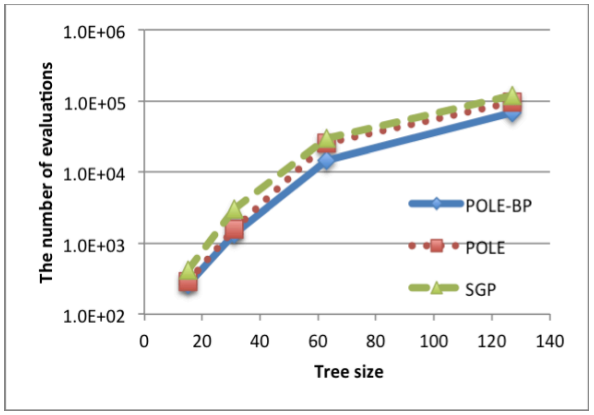


DMAX





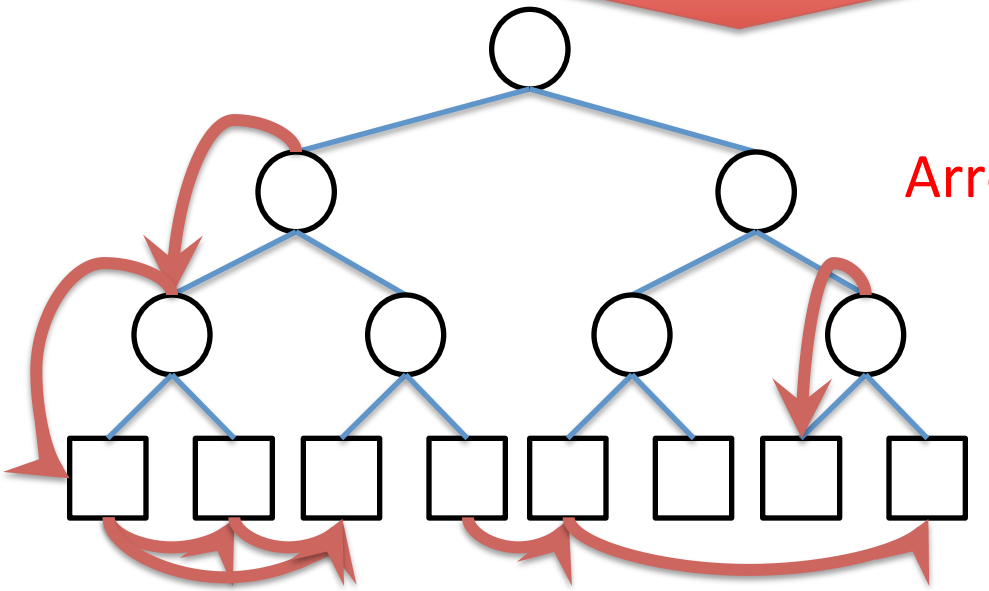
Royal tree



Estimation of PMBGP(Probabilistic Model Building GP) (GP version EDA, Bayesian Network)

1. Learning Graph Structures: Construct Bayesian Network

K2 Algorithm



Arrows represent dependencies

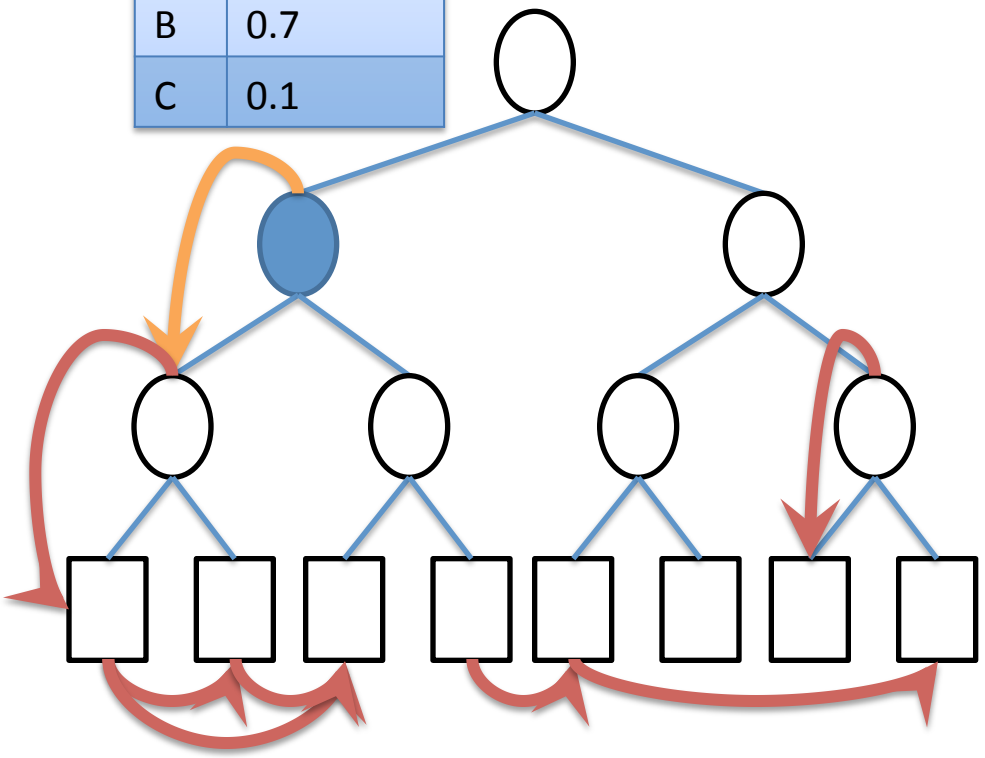
Conditional Probabilities

Estimation of PMBGP

2. Parameter Estimation: Estimate conditional probabilities

Parent	Child	Conditional Prob
A	A	0.1
A	B	0.7
A	C	0.2
B	A	0.05
B	B	0.15
B	C	0.8
C	A	0.25
C	B	0.2
C	C	0.55

	Prior Prob
A	0.2
B	0.7
C	0.1

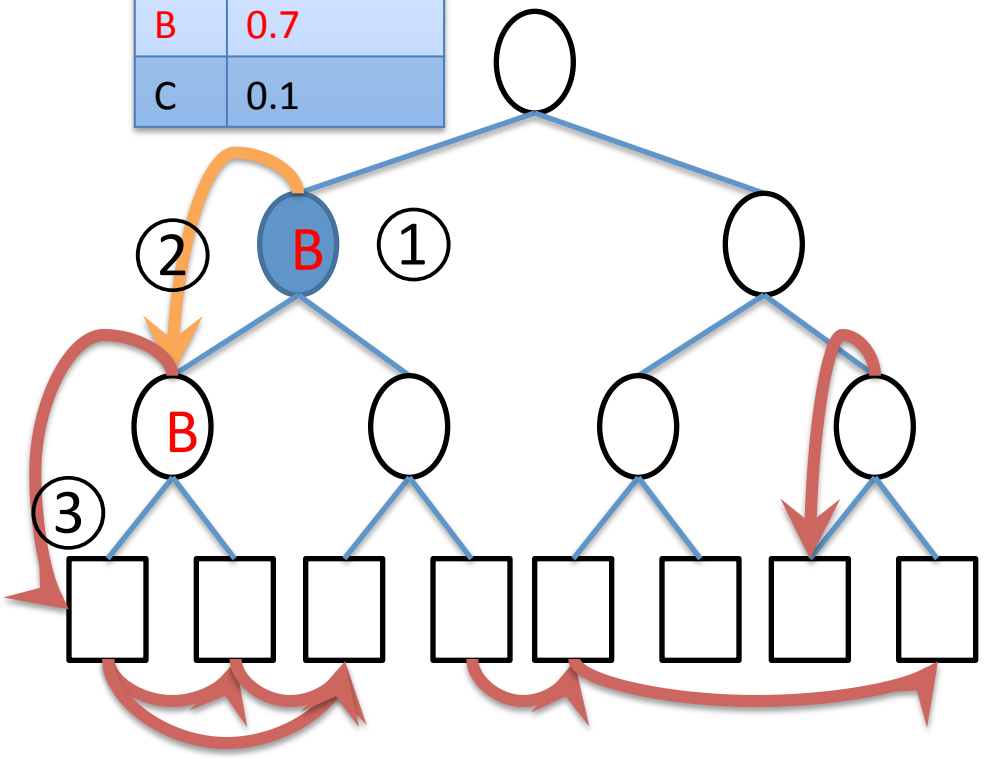


Sampling of PMBGP

Probabilistic Logic Sampling (PLS): Sample from ancestors

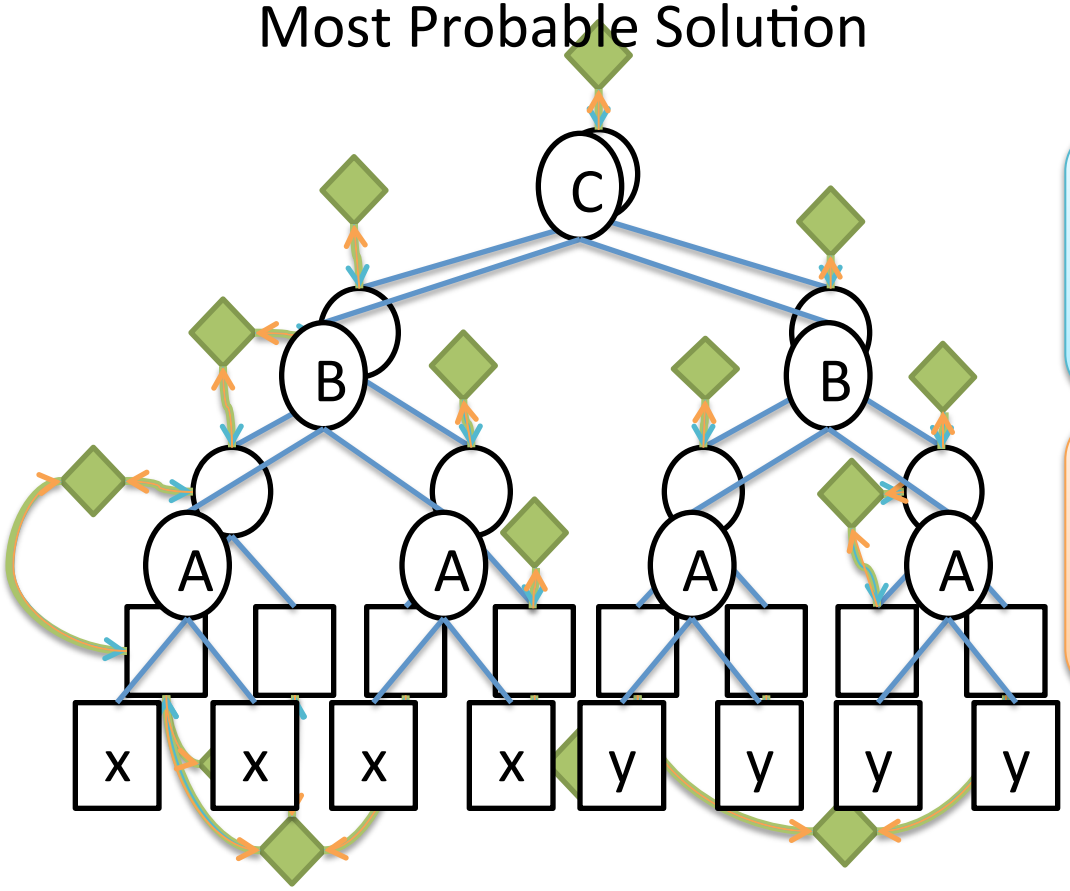
Parent	Child	Conditional Prob
A	A	0.1
A	B	0.7
A	C	0.2
B	A	0.05
B	B	0.15
B	C	0.8
C	A	0.25
C	B	0.2
C	C	0.55

	Prior Prob
A	0.2
B	0.7
C	0.1



Get MPS by Loopy Belief Propagation

Most Probable Solution



1. Initialize all messages to 0

2. Message Passing

Messages from factor to variable

leafs $\mu_{f\to x}(x) = \ln f(x)$

The others $\mu_{f\to x}(x) = \max_{x,x_1,\ldots,x_M} \left[\ln f(x,x_1,\ldots x_M) + \sum_{m\in ne(f_s)\setminus x} \mu_{x_m\to f}(x_m) \right]$

Messages from variable to factor

leafs $\mu_{x\to f}(x) = 0$

The others $\mu_{x\to f}(x) = \alpha_{xf} + \sum_{l\in ne(x)\setminus f} \mu_{f_l\to x}(x)$

3. Get MPS

Joint probability

$$x^{\max} = \arg\max_x \left[\sum_{s\in ne(x)} \mu_{f_s\to x}(x) \right]$$

Graph transformation

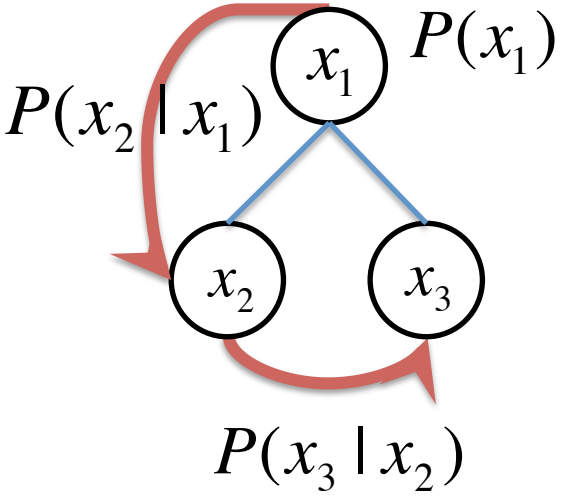
Bayesian Network : Directed Graph

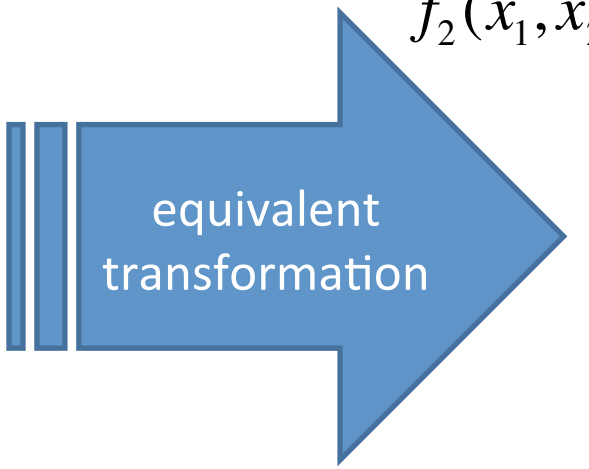
Directed Edge: conditional probability

Nodes without parents: prior probability

Blue edge: Tree structure

Bayesian Network constructed on tree (depth=2)

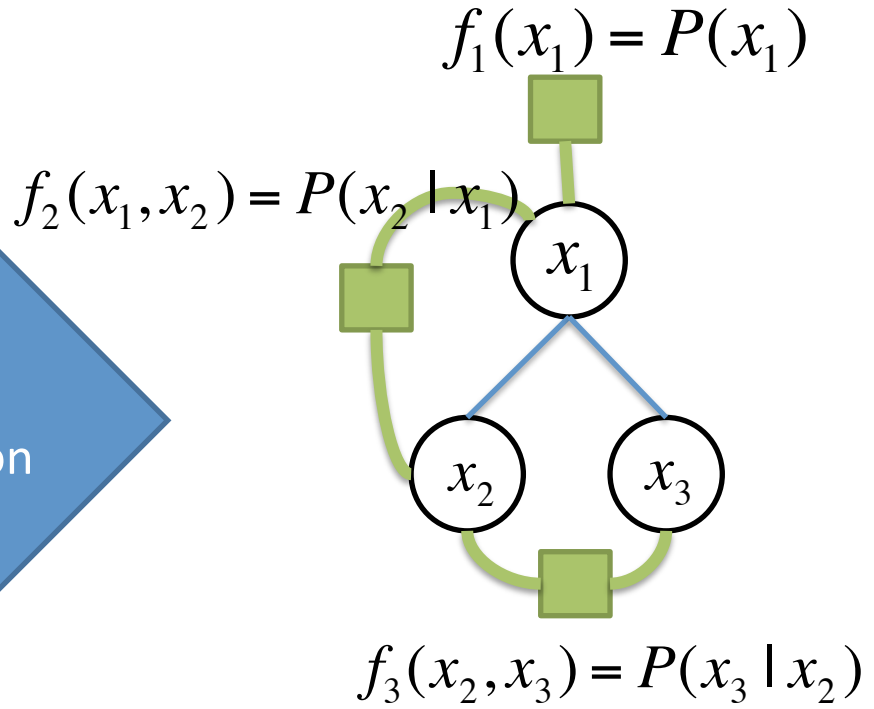




Factor Graph : Undirected Graph

Factor: prior and conditional probability

Undirected Edge: Only represent connection

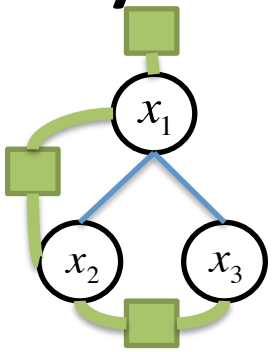


Joint Probability is calculated as

$$P(x_1, x_2, x_3) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_2) = f_1(x_1)f_2(x_1, x_2)f_3(x_2, x_3)$$

Effective calculation of the highest joint probability

- Let the number of cardinalities of each node be $N_{x_1} = |F|, N_{x_2} = N_{x_3} = |T|$



$$\max_{x_1,x_2,x_3}(P(x_1,x_2,x_3)) = \max_{x_1,x_2,x_3}(P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_2))$$

$$= \max_{x_1,x_2,x_3}(f_1(x_1)f_2(x_1,x_2)f_3(x_2,x_3))$$

$$= \max_{x_3}\left[\max_{x_2}\left[\max_{x_1}\left[f_1 f_2\right] f_3\right]\right]$$

The number of multiplication

$$N_{x_1}N_{x_2} + N_{x_1}N_{x_2}N_{x_3}$$

$$N_{x_1}N_{x_2} + N_{x_2}N_{x_3}$$

Permuting max and multiplication reduces the number of multiplication

More intuitional representation of this effective procedure?

→ *Message Passing*

Message Passing on a *chain graph* (~~Loopy~~ Belief Propagation)

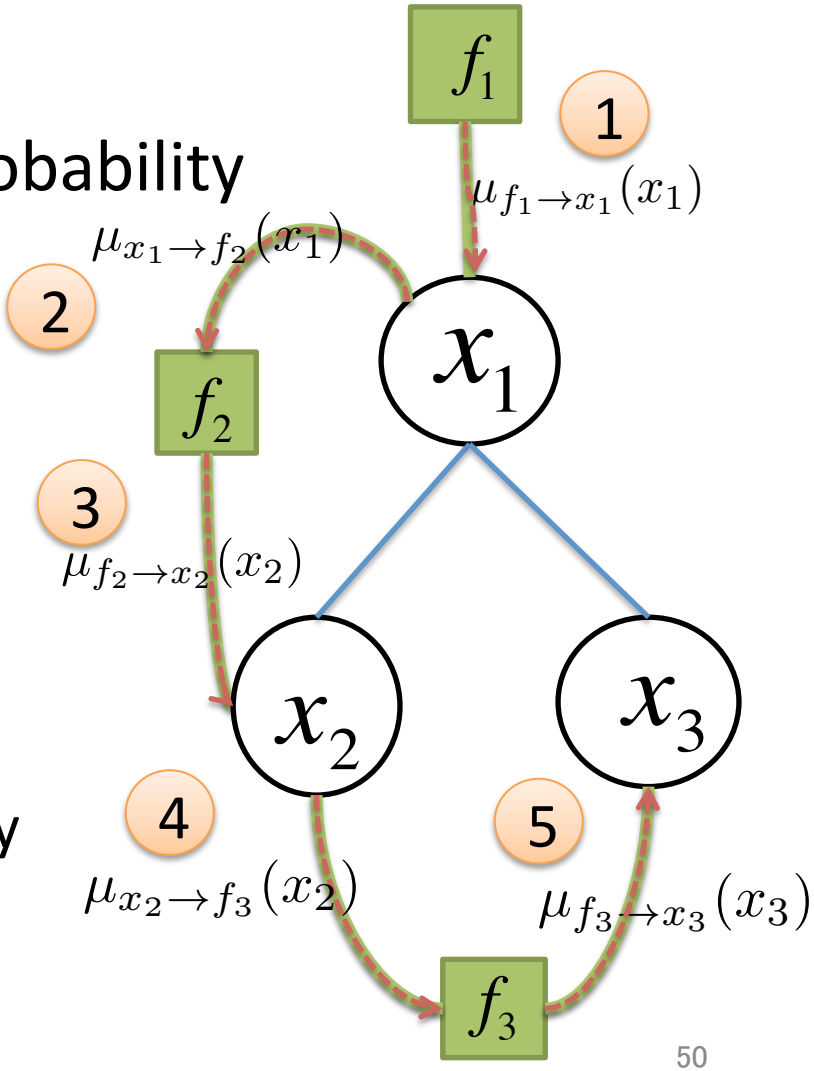
- Send *messages* in order, finally get the highest joint probability

Message definition

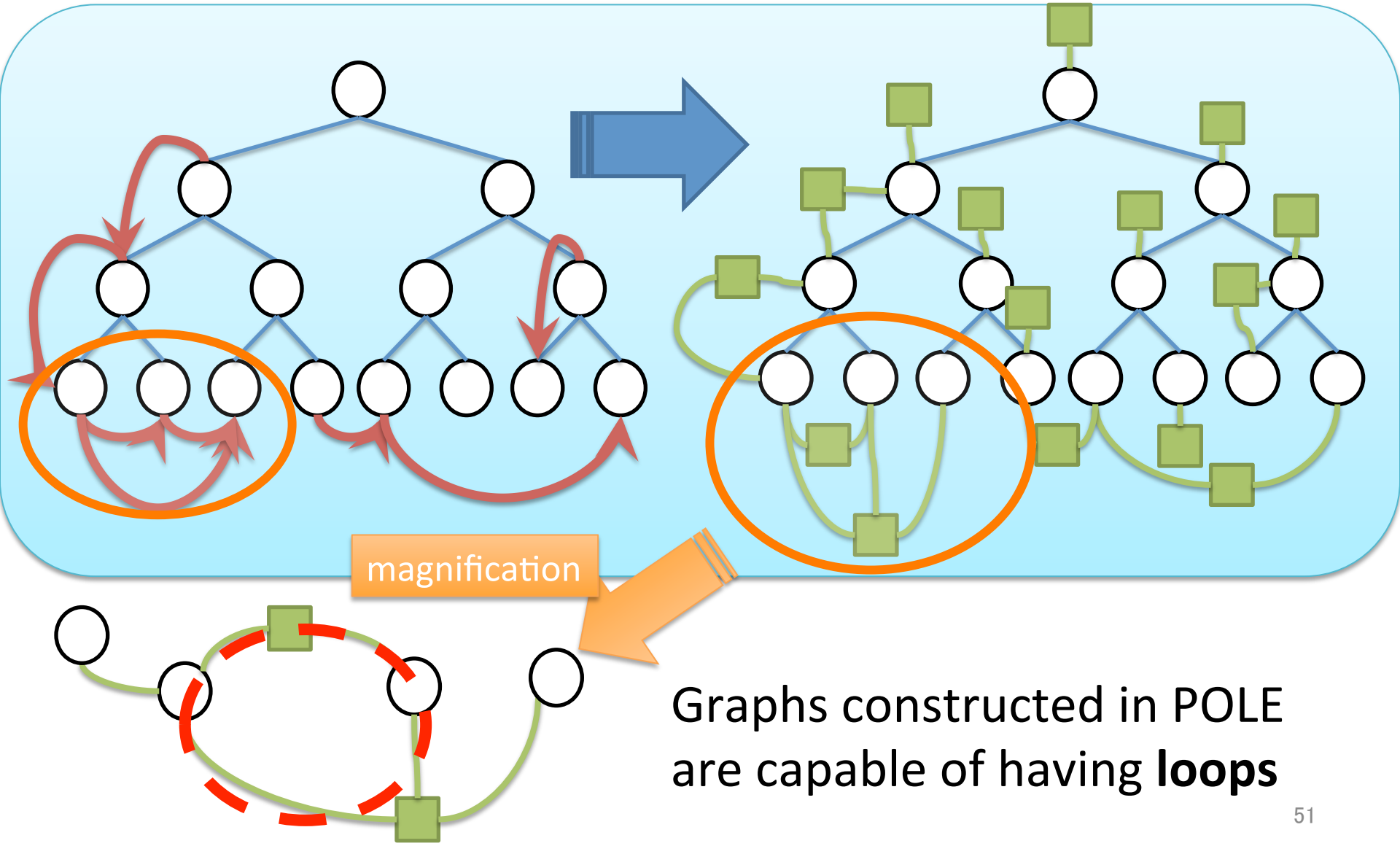
$$
\begin{aligned}
\mu_{f_1 \to x_1}(x_1) &= f_1(x_1), \\
\mu_{x_1 \to f_2}(x_1) &= \mu_{f_1 \to x_1}(x_1), \\
\mu_{f_2 \to x_2}(x_2) &= \max_{x_1} \left[f_2(x_1, x_2) \mu_{x_1 \to f_2}(x_1) \right], \\
\mu_{x_2 \to f_3}(x_2) &= \mu_{f_2 \to x_2}(x_2), \\
\mu_{f_3 \to x_3}(x_3) &= \max_{x_2} \left[f_3(x_2, x_3) \mu_{x_2 \to f_3}(x_2) \right],
\end{aligned}
$$

Calculation of the highest joint probability

$$
\begin{aligned}
\max(P(x_1, x_2, x_3)) &= \max_{x_3} \left[\mu_{f_3 \to x_3} \right] \\
&= \max_{x_3} \left[\max_{x_2} \left[\max_{x_1} \left[f_1 f_2 \right] f_3 \right] \right]
\end{aligned}
$$



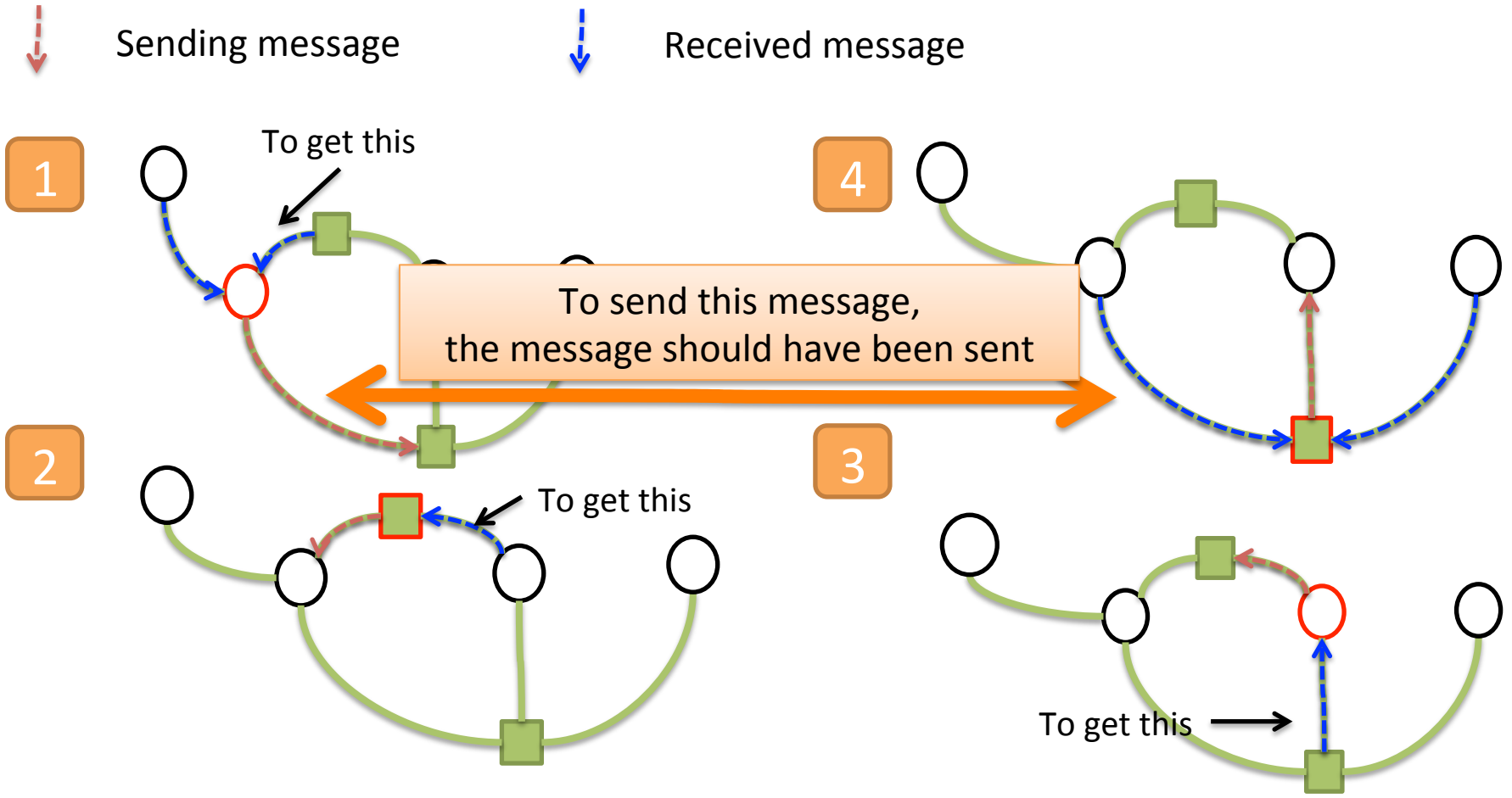
Problems of Message Passing in practice



Graphs constructed in POLE are capable of having **loops**

Contradiction of Message Passing on graphs with loops

To send the message, the node should receive the other messages



Generalization of Message Passing

- To adapt loopy graphs,
  1. Set all messages with proper initial values
  2. Repeatedly send messages in arbitrary order
  3. Generally, messages converge to the right value
- To prevent underflow, substitute log and summation for multiplication in message
  - This substitution doesn't effect max

Graph Transformation

Bayesian Network : Directed Graph

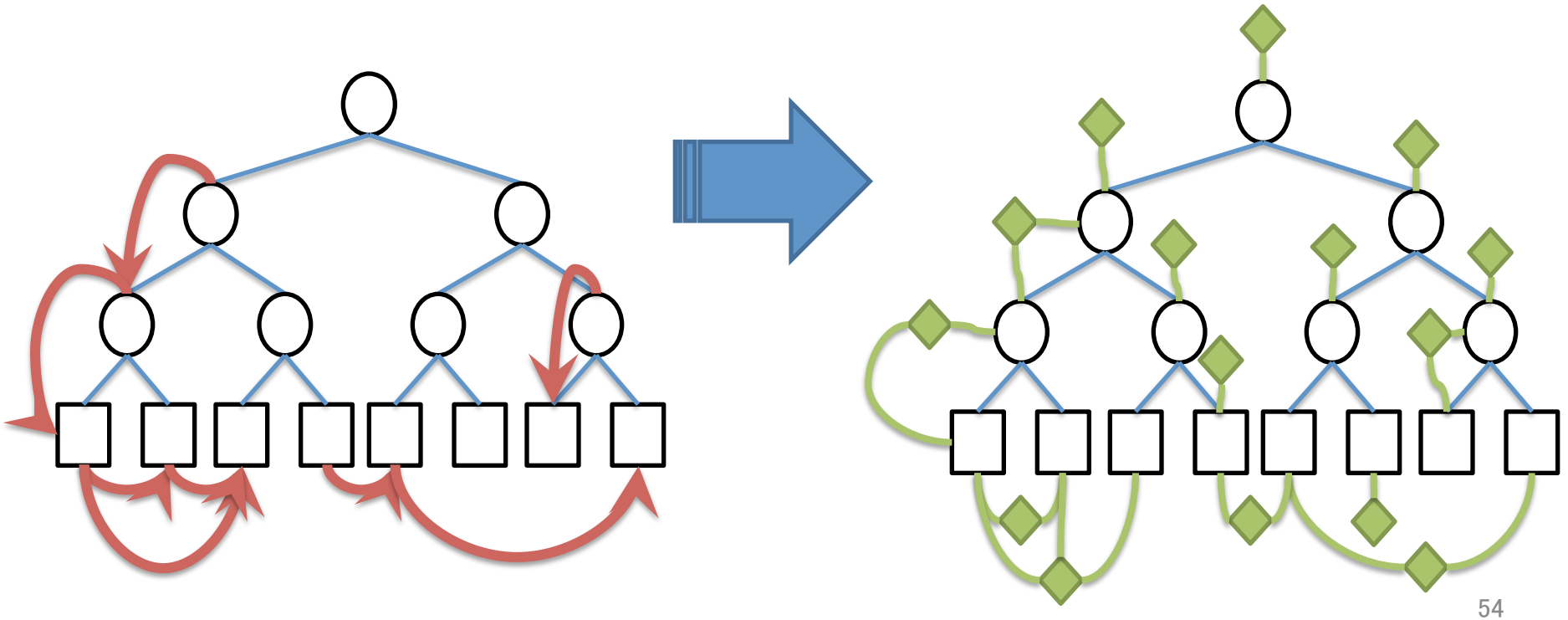
Directed Edge: conditional probability

Nodes without parents: prior probability

Factor Graph : Undirected Graph

Factor: prior and conditional probability

Undirected Edge: Only represent connection



Message Passing on loopy graphs
LBP : Loopy Belief Propagation

1. Initialize all messages to 0

2. Message Passing repeatedly

factor f variable x

Messages from factor to variable

Leafs

$$\mu_{f\to x}(x) = \ln f(x)$$

The others

$$\mu_{f\to x}(x) = \max_{x,x_1,\ldots,x_M}\left[\ln f(x,x_1,\ldots x_M) + \sum_{m\in ne(f_s)\setminus x}\mu_{x_m\to f}(x_m)\right]$$

Messages from variable to factor

Leafs

$$\mu_{x\to f}(x) = 0$$

The others

$$\mu_{x\to f}(x) = \alpha_{xf} + \sum_{l\in ne(x)\setminus f}\mu_{f_l\to x}(x)$$

3. Get MPS

$$x^{\max} = \arg\max_{x}\left[\text{Joint probability}\ \sum_{s\in ne(x)}\mu_{f_s\to x}(x)\right]$$

The weak point of prototype tree based PMBGP

1. The solution which has the highest joint probability (MPS: Most Probable Solution) reflects the constructed Bayesian Network, which is usually used as probabilistic model.
2. However, traditional sampling does *not always* generate MPS
3. This problem wastes a part of learning and increases the number of evaluations to get an optimum solution.

The same problem is pointed out in GA-type EDAs, but sampling using Loopy Belief Propagation is known as a way to overcome it