A Generic Approach to Accelerating Belief Propagation based Incomplete Algorithms for DCOPs via A Branch-and-Bound Technique

Ziyu Chen, Xingqiong Jiang, Yanchen Deng*, Dingding Chen, Zhongshi He

College of Computer Science,

Chongqing University, Chongqing, China

Outline

- Background
  - Distributed Constraint Optimization Problems (DCOPs)
  - Max-sum
- Proposed Method
  - Motivation
  - Function Decomposing and State Pruning (FDSP)
- Experimental Evaluation

Distributed Constraint Optimization Problems (DCOPs)

- DCOPs are a fundamental framework for Multi-agent Systems in which agents need to coordinate their decisions to optimize a global objective

- Applications
  - Task scheduling
  - Power networks
  - Sensor networks

  ……

Formal Definition of DCOPs

- **Notations:**
  - Agents: $A = \{a_1, a_2, \ldots, a_h\}$
  - Variables: $X = \{x_1, x_2, \ldots, x_q\}$
  - Domains: $D = \{D_1, D_2, \ldots, D_q\}$
  - Constraints: $F = \{F_1, F_2, \ldots, F_r\}$
- **Note:**

  Each constraint is a n-ary function: $F_k : \mathbf{x_k} \to \mathbb{R}^+, \; \mathbf{x_k} \subseteq X \; and \; n = |\mathbf{x_k}|$
- **Goal :**

  Find an assignment X^* to maximize the total utility:

$$X^* = \arg\max_{X} \sum_{F_k(\mathbf{x_k}) \in F, \mathbf{x_k} \subseteq X} F_k(\mathbf{x_k})$$

Taxonomy of Algorithms for DCOPs

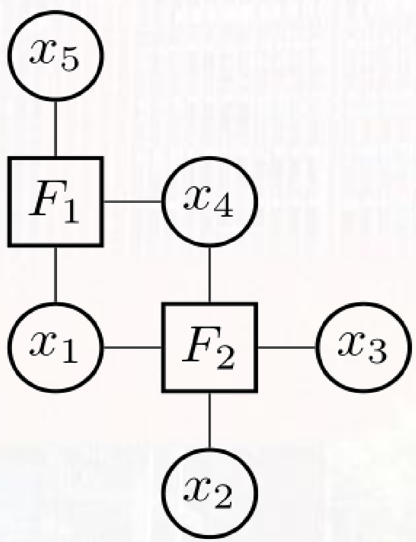
- DCOP Algorithms
  - Complete Algorithms
    - Search-based Algorithms → SynchBB、ADOPT、BnB-ADOPT、AFB……
    - Inference-based Algorithms → DPOP、Action-GDL……
  - Incomplete Algorithms
    - Local Search Algorithms → DSA、MGM、MGM2、GDBA……
    - Inference-based Algorithms → **Max-Sum、BMS、IBMS、Max-Sum_ADVP……**
    - Sample-based Algorithms → DUCT、D-Gibbs……

Taxonomy of Algorithms for DCOPs

- DCOP Algorithms
  - Complete Algorithms
    - Search-based Algorithms → SynchBB、ADOPT、BnB-ADOPT、AFB……
    - Inference-based Algorithms → DPOP、Action-GDL……
  - Incomplete Algorithms
    - Local Search Algorithms → DSA、MGM、MGM2、GDBA……
    - Inference-based Algorithms → **Max-Sum、BMS、IBMS、Max-Sum_ADVP……**
    - Sample-based Algorithms → DUCT、D-Gibbs……

Max-Sum Algorithm (belief propagation approach)

- **Factor Graph**
  - Variable-nodes: **variables** $X = \{x_1, x_2, \ldots, x_q\}$
  - Function-nodes: **constraint functions** $F = \{F_1, F_2, \ldots, F_r\}$
- Example



A factor graph

F_1 and F_2 are two function-nodes

x_1, x_2, x_3, x_4 and x_5 are variable-nodes

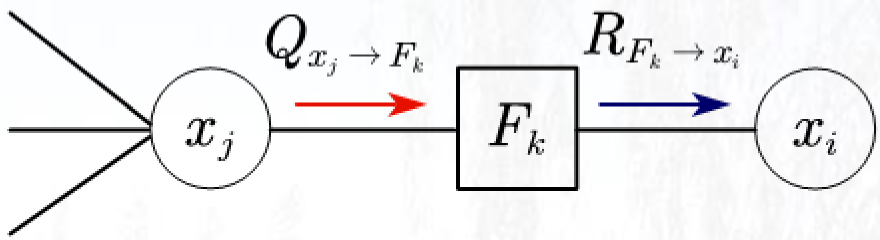
$F_1(\mathbf{x_1})$ is a 3-ary constraint

$$\mathbf{x_1} = \{x_1, x_4, x_5\}$$

$F_2(\mathbf{x_2})$ is a 4-ary constraint

$$\mathbf{x_2} = \{x_1, x_2, x_3, x_4\}$$

Max-Sum Algorithm



- Messages from variable-nodes: accumulating and forwarding utilities

$$Q_{x_i \to F_k}(x_i) = \alpha_{ik} + \sum_{F_j \in N_i \setminus F_k} R_{F_j \to x_i}(x_i) \qquad (1)$$

- Messages from function-nodes: maximizing utilities upon received utilities and local functions

$$R_{F_k \to x_i}(x_i) = \max_{\mathbf{x_k} \setminus x_i} \left(F_k(\mathbf{x_k}) + \sum_{x_j \in \mathbf{x_k} \setminus x_i} Q_{x_j \to F_k}(x_j) \right) \qquad (2)$$

- Decision-making strategy for variable-nodes: selecting a value to maximize the total utility:

$$x_i^* = \operatorname*{argmax}_{x_i} \sum_{F_k \in N_i} R_{F_k \to x_i}(x_i) \qquad (3)$$

Motivation

- **Problem:** the ***scalability*** of Max-sum and its variants
- **Reason:** the ***complexity*** $\boldsymbol{O(d^{n})}$ of computing message with Eq. (2) (*d*: domain size, *n*: arity)
- **Existing methods**
  - *Branch and bound* : BnB-MS and BnB-FMS
    - ? Exchange a number of messages in the preprocessing phase
    - ? Lack of ***generalization***
  - *Sorting* : G-FBP , GDP
    - ? Require ***prohibitively expensive sorting*** in the preprocessing phase
    - ? G-FBP may lead to a complete traverse to all possible combinations
    - ? GDP cannot prune the search space dynamically
- **Our goal: a generic, fast and easy-to-use approach**

Function Decomposing and State Pruning (FDSP)

- **Idea:** using the learned experience from the combinations explored to dynamically prune the search space
- **Scheme**
  - **Function Decomposing (FD) in the preprocessing phase**
    - Computing *function estimations* to provide upper bounds for the local function by means of *Dynamic Programming*
  - **State Pruning (SP)**
    - Pruning the search space by means of *Branch and Bound* in terms of the optimal upper bound from *function estimations* and *the received query message estimations*

Function Decomposing (FD)

- The function estimation of $F_k(\mathbf{x_k})$:

  $$FunEst_{\mathbf{x_{k,i}}}(PA|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}) = \max_{z=\{\mathbf{x_{k,j}}|j>i\}} F_k(PA|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}, z)$$

  $PA|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}$ is a partial assignment and $\{\mathbf{x_{k,w}} \in \mathbf{x_k} | 1 \le w \le i\}$

- Two types of function estimations:
  - *uninformed* function estimation for tight upper bounds

    $$FunEst_{\mathbf{x_{k,i}}} = \begin{cases} F_k(\mathbf{x_k}) & i=n \\ \max\limits_{\mathbf{x_{k,i+1}}} FunEst_{\mathbf{x_{k,i+1}}} & otherwise \end{cases}$$

  - *informed* function estimation for tighter upper bounds

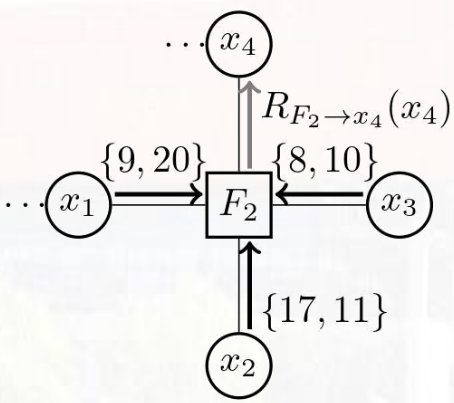
    $$FunEst_{\mathbf{x_{k,i}}}^{\mathbf{x_{k,j}}=v_{k,j}} = \begin{cases} FunEst_{\mathbf{x_{k,j}}}(v_{k,j}) & i=j-1 \\ \max\limits_{\mathbf{x_{k,i+1}}} FunEst_{\mathbf{x_{k,i+1}}}^{\mathbf{x_{k,j}}=v_{k,j}} & otherwise \end{cases}$$

Example of Function Decomposing (FD)

- The uninformed function estimations for variable x_1, x_2, x_3, and x_4

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1

(a) utility matrix of F_2



(b) messages exchange for F_2

Example of Function Decomposing (FD)

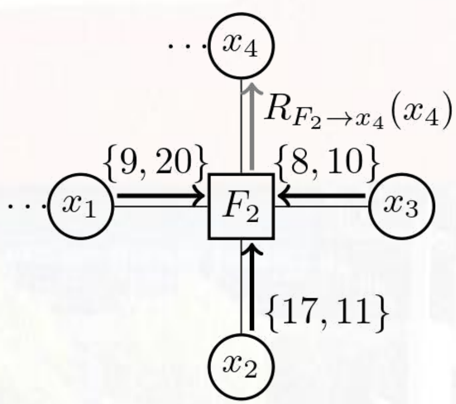
- The uninformed function estimations for variable x_1, x_2, x_3, and x_4

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1

(a) utility matrix of F_2

Step1: $FunEst_{x_4} = F_2$

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1



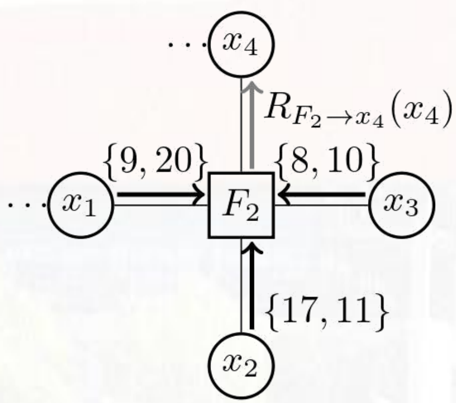
(b) messages exchange for F_2

Example of Function Decomposing (FD)

- The uninformed function estimations for variable x_1, x_2, x_3, and x_4

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1

(a) utility matrix of F_2



(b) messages exchange for F_2

Step1: $FunEst_{x_4} = F_2$

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1

Step2: $FunEst_{x_3} = \max_{x_4} FunEst_{x_4}$

x_1	x_2	x_3	F_2
R	R	R	13
R	R	G	26
R	G	R	5
R	G	G	4
G	R	R	10
G	R	G	9
G	G	R	11
G	G	G	12

Example of Function Decomposing (FD)

- The uninformed function estimations for variable x_1, x_2, x_3, and x_4

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1

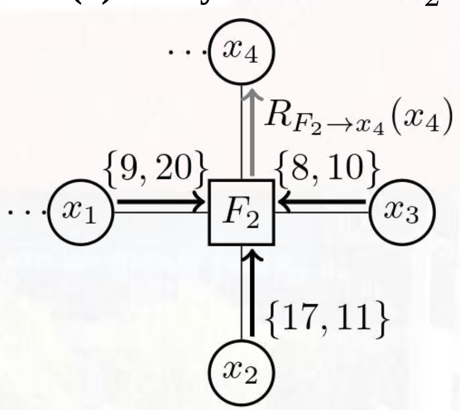
(a) utility matrix of F_2

Step1: $FunEst_{x_4} = F_2$

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1

Step2: $FunEst_{x_3} = \max_{x_4} FunEst_{x_4}$

x_1	x_2	x_3	F_2
R	R	R	13
R	R	G	26
R	G	R	5
R	G	G	4
G	R	R	10
G	R	G	9
G	G	R	11
G	G	G	12



(b) messages exchange for F_2

Step3: $FunEst_{x_2} = \max_{x_3} FunEst_{x_3}$

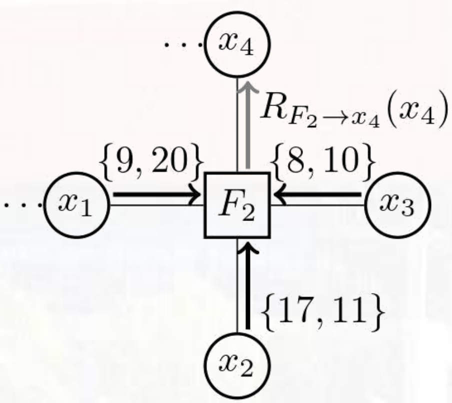
x_1	x_2	F_2
R	R	26
R	G	5
G	R	10
G	G	12

Example of Function Decomposing (FD)

- The uninformed function estimations for variable x_1, x_2, x_3, and x_4

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1

(a) utility matrix of F_2



(b) messages exchange for F_2

Step1: $FunEst_{x_4} = F_2$

x_1	x_2	x_3	x_4	F_2
R	R	R	R	4
R	R	R	G	13
R	R	G	R	26
R	R	G	G	5
R	G	R	R	1
R	G	R	G	5
R	G	G	R	2
R	G	G	G	4
G	R	R	R	10
G	R	R	G	7
G	R	G	R	6
G	R	G	G	9
G	G	R	R	11
G	G	R	G	8
G	G	G	R	12
G	G	G	G	1

Step2: $FunEst_{x_3} = \max_{x_4} FunEst_{x_4}$

x_1	x_2	x_3	F_2
R	R	R	13
R	R	G	26
R	G	R	5
R	G	G	4
G	R	R	10
G	R	G	9
G	G	R	11
G	G	G	12

Step3: $FunEst_{x_2} = \max_{x_3} FunEst_{x_3}$

x_1	x_2	F_2
R	R	26
R	G	5
G	R	10
G	G	12

Step4: $FunEst_{x_1} = \max_{x_2} FunEst_{x_2}$

x_1	F_2
R	26
G	12

Example of Function Decomposing (FD)

- The informed function estimations in terms of $x_4 = R$

Example of Function Decomposing (FD)

- The informed function estimations in terms of $x_4 = R$

Step1: $FunEst_{x_3}^{x_4=R} = FunEst_{x_4}(x_4 = R)$

x_1	x_2	x_3	F_2
R	R	R	4
R	R	G	26
R	G	R	1
R	G	G	2
G	R	R	10
G	R	G	6
G	G	R	11
G	G	G	12

Example of Function Decomposing (FD)

- The informed function estimations in terms of $x_4 = R$

Step1: $FunEst_{x_3}^{x_4=R} = FunEst_{x_4}(x_4 = R)$

x_1	x_2	x_3	F_2
R	R	R	4
R	R	G	26
R	G	R	1
R	G	G	2
G	R	R	10
G	R	G	6
G	G	R	11
G	G	G	12

Step2: $FunEst_{x_2}^{x_4=R} = \max_{x_3} FunEst_{x_3}^{x_4=R}$

x_1	x_2	F_2
R	R	26
R	G	2
G	R	10
G	G	12

Example of Function Decomposing (FD)

- The informed function estimations in terms of $x_4 = R$

Step1: $FunEst_{x_3}^{x_4=R} = FunEst_{x_4}(x_4 = R)$

x_1	x_2	x_3	F_2
R	R	R	4
R	R	G	26
R	G	R	1
R	G	G	2
G	R	R	10
G	R	G	6
G	G	R	11
G	G	G	12

Step2: $FunEst_{x_2}^{x_4=R} = \max_{x_3} FunEst_{x_3}^{x_4=R}$

x_1	x_2	F_2
R	R	26
R	G	2
G	R	10
G	G	12

Step3: $FunEst_{x_1}^{x_4=R} = \max_{x_2} FunEst_{x_2}^{x_4=R}$

x_1	F_2
R	26
G	12

State Pruning (SP)

- The received query message estimation:

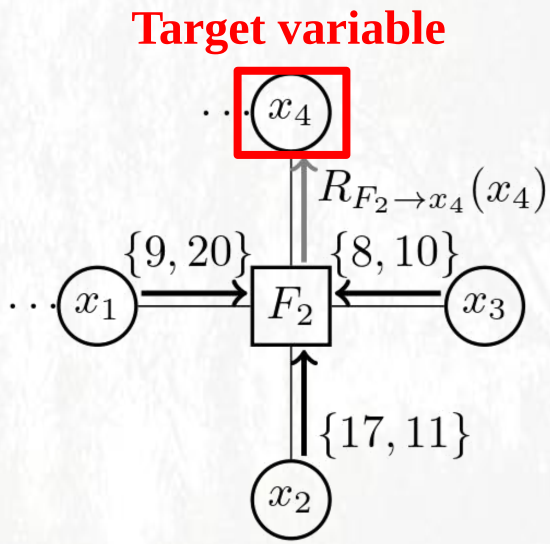
$$MsgEst_{\mathbf{x_{k,i}}} = \begin{cases} \sum\limits_{j>i \wedge j \neq t} \max(\mathcal{M}_{\mathbf{x_{k,j}}}) & i=n-1 \\ MsgEst_{\mathbf{x_{k,i+1}}} + \max(\mathcal{M}_{\mathbf{x_{k,i+1}}}) & otherwise \end{cases}$$

- Example

$$MsgEst_{x_3} = 0$$

$$MsgEst_{x_2} = MsgEst_{x_3} + \max\mathcal{M}_{x_3} = 10$$

$$MsgEst_{x_1} = MsgEst_{x_2} + \max\mathcal{M}_{x_2} = 10 + 17 = 27$$



State Pruning (SP)

- The upper bound $ub_{\mathbf{x_{k,i}}}$ for a partial assignment $Assign\big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}$ is

$$ub_{\mathbf{x_{k,i}}} = \begin{cases} msgUtil_{\mathbf{x_{k,i}}} + MsgEst_{\mathbf{x_{k,i}}} + FunEst_{\mathbf{x_{k,i}}}\left(Assign\big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}\right) & i > t \\ msgUtil_{\mathbf{x_{k,i}}} + MsgEst_{\mathbf{x_{k,i}}} + FunEst_{\mathbf{x_{k,i}}}^{\mathbf{x_{k,t}}=v_{k,t}}\left(Assign\big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}\right) & i < t \end{cases}$$

Here,

$$msgUtil_{\mathbf{x_{k,i}}} = \sum_{1 \leqslant w \leqslant i} \mathcal{M}_{\mathbf{x_{k,w}}}(v_{k,w})$$

- Discard the search space corresponding to the partial assignment when $ub_{\mathbf{x_{k,i}}} \leqslant lb$.

State Pruning (SP)

- The upper bound $ub_{\mathbf{x_{k,i}}}$ for a partial assignment $Assign\Big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}$ is

$$ub_{\mathbf{x_{k,i}}} = \begin{cases} msgUtil_{\mathbf{x_{k,i}}} + MsgEst_{\mathbf{x_{k,i}}} + FunEst_{\mathbf{x_{k,i}}}\left(Assign\Big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}\right) & i > t \\ msgUtil_{\mathbf{x_{k,i}}} + MsgEst_{\mathbf{x_{k,i}}} + FunEst_{\mathbf{x_{k,i}}}^{\mathbf{x_{k,t}} = v_{k,t}}\left(Assign\Big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}\right) & i < t \end{cases}$$

Message estimation

Here,

$$msgUtil_{\mathbf{x_{k,i}}} = \sum_{1 \leqslant w \leqslant i} \mathcal{M}_{\mathbf{x_{k,w}}}\left(v_{k,w}\right)$$

- Discard the search space corresponding to the partial assignment when $ub_{\mathbf{x_{k,i}}} \leqslant lb$.

State Pruning (SP)

- The upper bound $ub_{\mathbf{x_{k,i}}}$ for a partial assignment $Assign\big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}$ is

$$ub_{\mathbf{x_{k,i}}} = \begin{cases} msgUtil_{\mathbf{x_{k,i}}} + \boxed{MsgEst_{\mathbf{x_{k,i}}}} + \boxed{FunEst_{\mathbf{x_{k,i}}}\left(Assign\big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}\right)} & i > t \\ msgUtil_{\mathbf{x_{k,i}}} + \boxed{MsgEst_{\mathbf{x_{k,i}}}} + \boxed{FunEst_{\mathbf{x_{k,i}}}^{\mathbf{x_{k,t}} = v_{k,t}}\left(Assign\big|_{\mathbf{x_{k,1}}}^{\mathbf{x_{k,i}}}\right)} & i < t \end{cases}$$

Message estimation

Function estimation

Here,

$$msgUtil_{\mathbf{x_{k,i}}} = \sum_{1 \leqslant w \leqslant i} \mathcal{M}_{\mathbf{x_{k,w}}}\left(v_{k,w}\right)$$

- Discard the search space corresponding to the partial assignment when $ub_{\mathbf{x_{k,i}}} \leqslant lb$.

Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ $(i.e., R_{F_2 \rightarrow x_4}(x_4 = R))$

Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ $(i.e., R_{F_2 \to x_4}(x_4 = R))$

$$\{\emptyset, \emptyset, \emptyset, R\}$$

Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ ($i.e.$, $R_{F_2 \to x_4}(x_4 = R)$)

$\{\emptyset, \emptyset, \emptyset, R\}$

$[-\infty, 62]$

①

Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ $(i.e., R_{F_2 \to x_4}(x_4 = R))$

$$\{\emptyset, \emptyset, \emptyset, R\}$$

$$[-\infty, 62] \quad ①$$

$$\{R, \emptyset, \emptyset, R\}$$

Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ ($i.e.$, $R_{F_2 \rightarrow x_4}(x_4 = R)$)

$$\{\emptyset, \emptyset, \emptyset, R\}$$

$$[-\infty, 62] \quad ①$$

$$\{R, \emptyset, \emptyset, R\}$$

$$[-\infty, 62] \quad ②$$

Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ $(i.e., R_{F_2 \to x_4}(x_4 = R))$

$$\{\emptyset, \emptyset, \emptyset, R\}$$

$[-\infty, 62]$ ①

$$\{R, \emptyset, \emptyset, R\}$$

$[-\infty, 62]$ ②

$$\{R, R, \emptyset, R\}$$

Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ (i.e., $R_{F_2 \to x_4}(x_4 = R)$)

$$\{\emptyset, \emptyset, \emptyset, R\}$$

$$[-\infty, 62] \quad ①$$

$$\{R, \emptyset, \emptyset, R\}$$

$$[-\infty, 62] \quad ②$$

$$\{R, R, \emptyset, R\}$$

$$[-\infty, 38] \quad ③$$

Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ ($i.e.$, $R_{F_2 \to x_4}(x_4 = R)$)

$$\{\emptyset, \emptyset, \emptyset, R\}$$

$$[-\infty, 62] \quad ①$$

$$\{R, \emptyset, \emptyset, R\}$$

$$[-\infty, 62] \quad ②$$

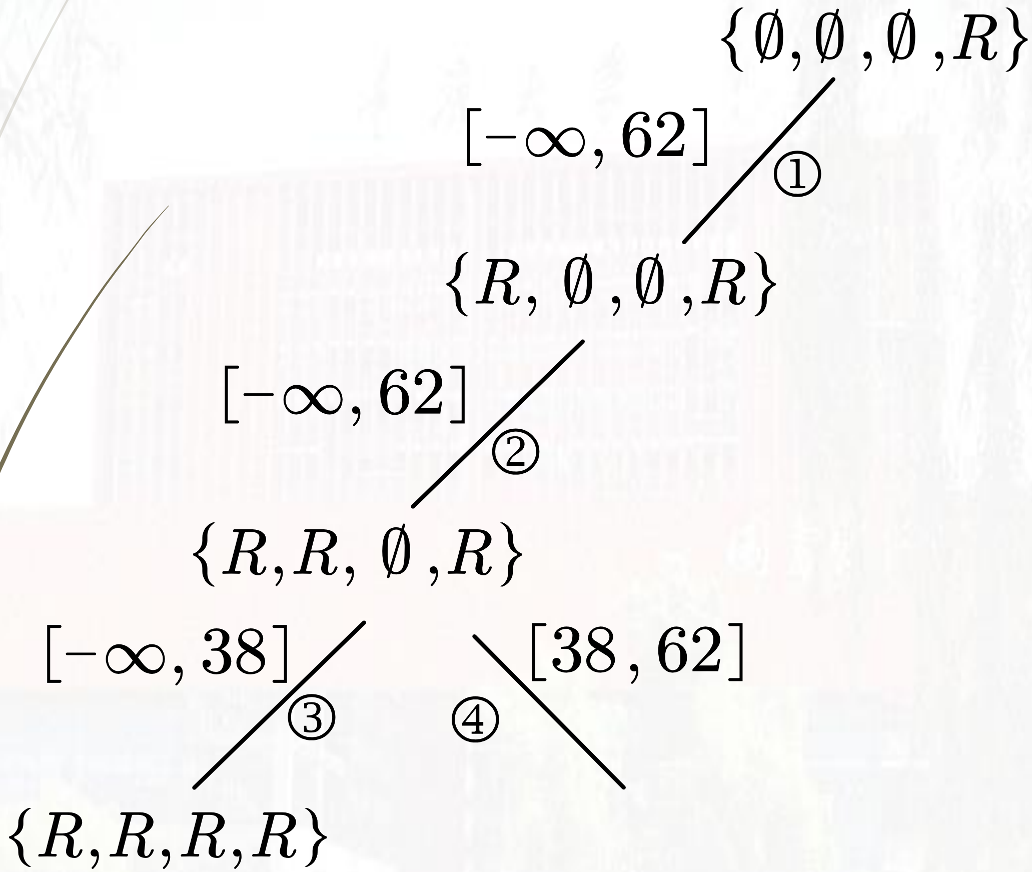
$$\{R, R, \emptyset, R\}$$

$$[-\infty, 38] \quad ③$$

$$\{R, R, R, R\}$$

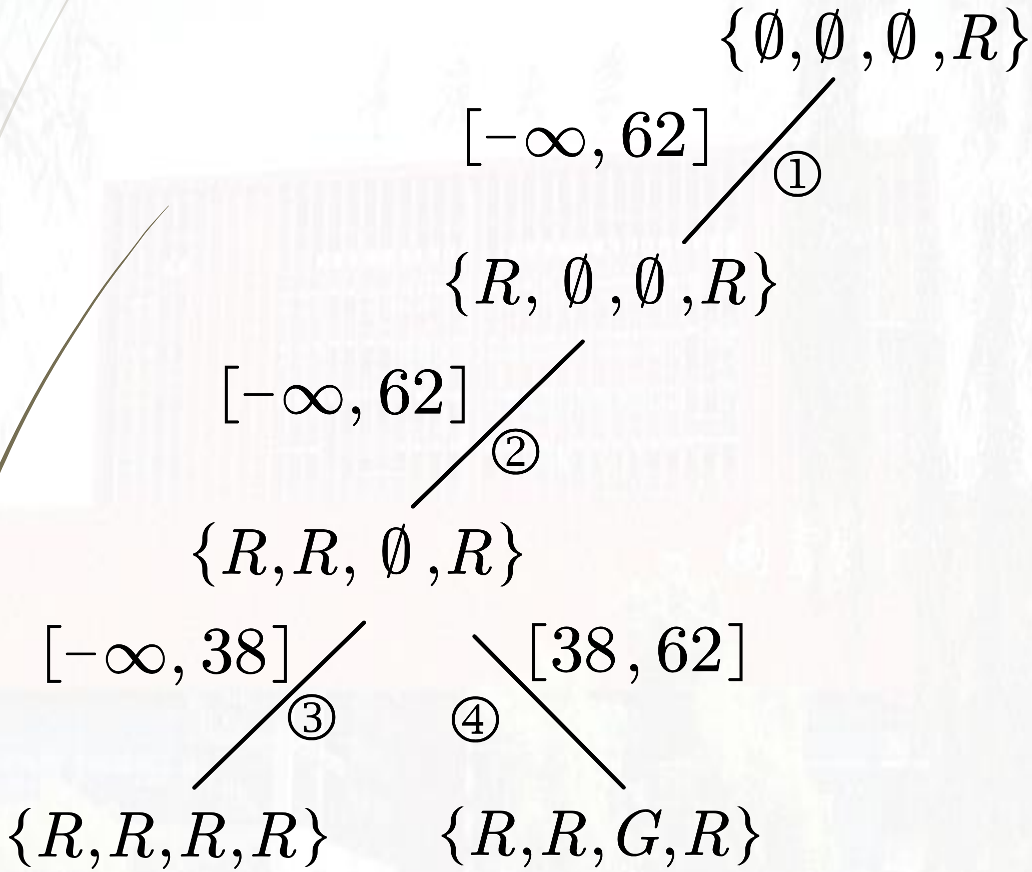
Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ (*i.e.*, $R_{F_2 \to x_4}(x_4 = R)$)



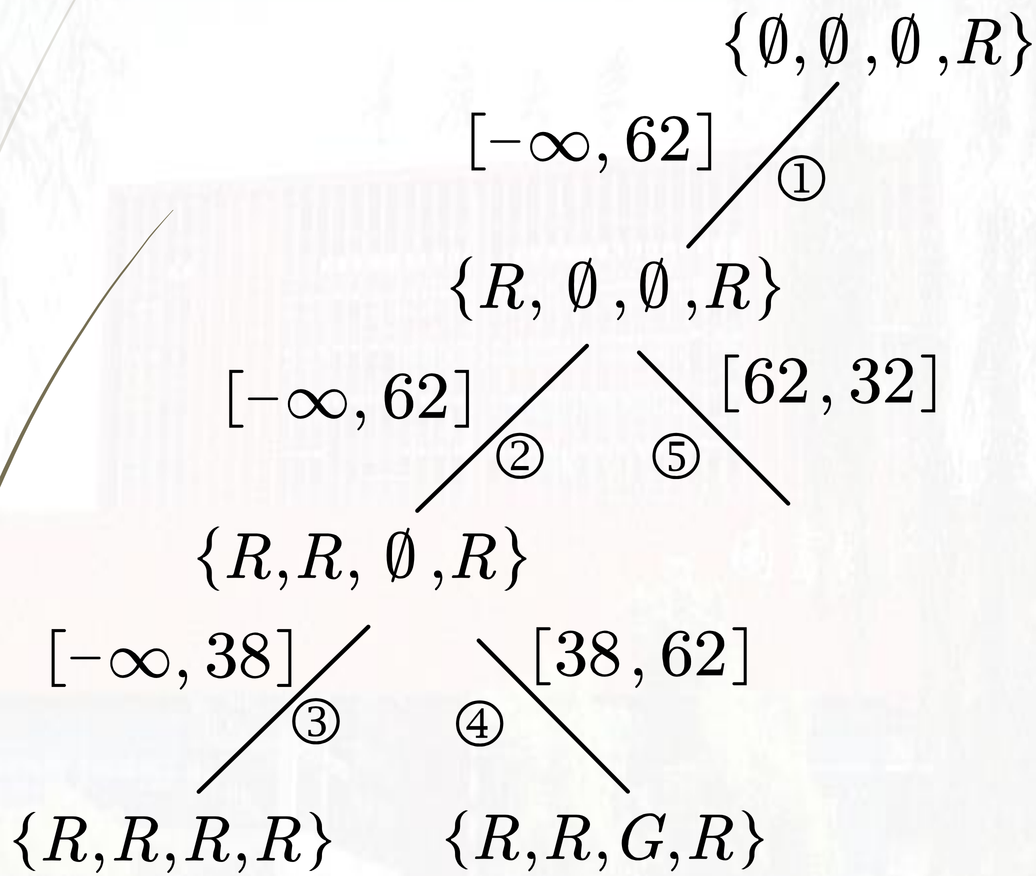
Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ (*i.e.*, $R_{F_2 \to x_4}(x_4 = R)$)



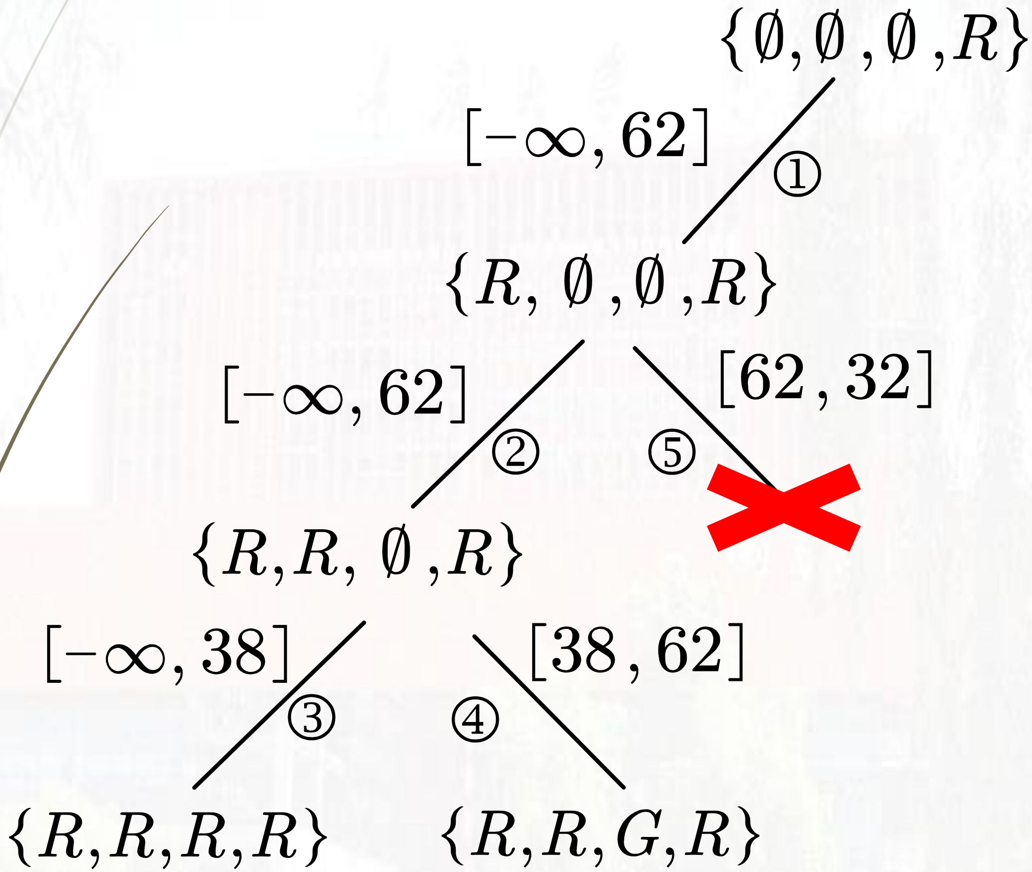
Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ (*i.e.*, $R_{F_2 \to x_4}(x_4 = R)$)



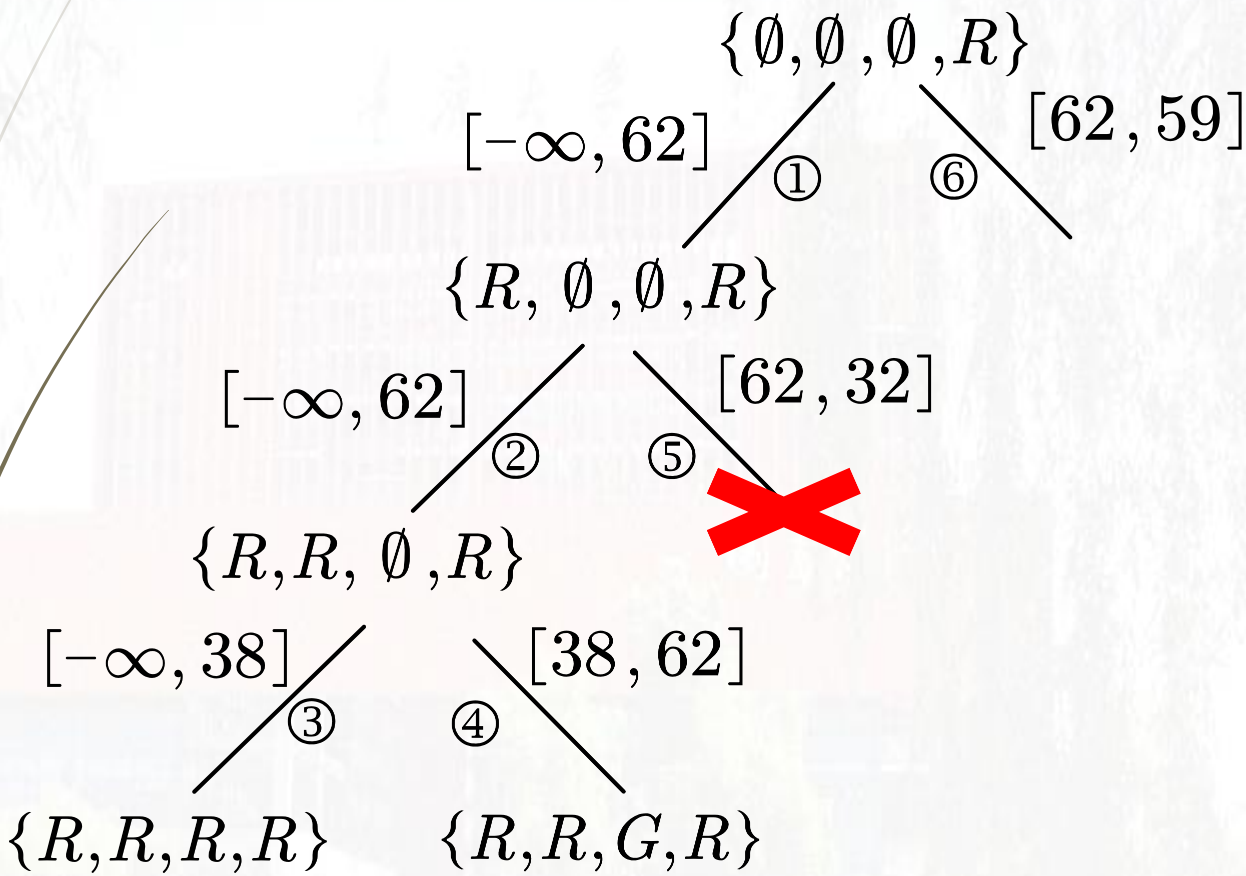
Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ (*i.e.*, $R_{F_2 \rightarrow x_4}(x_4 = R)$)



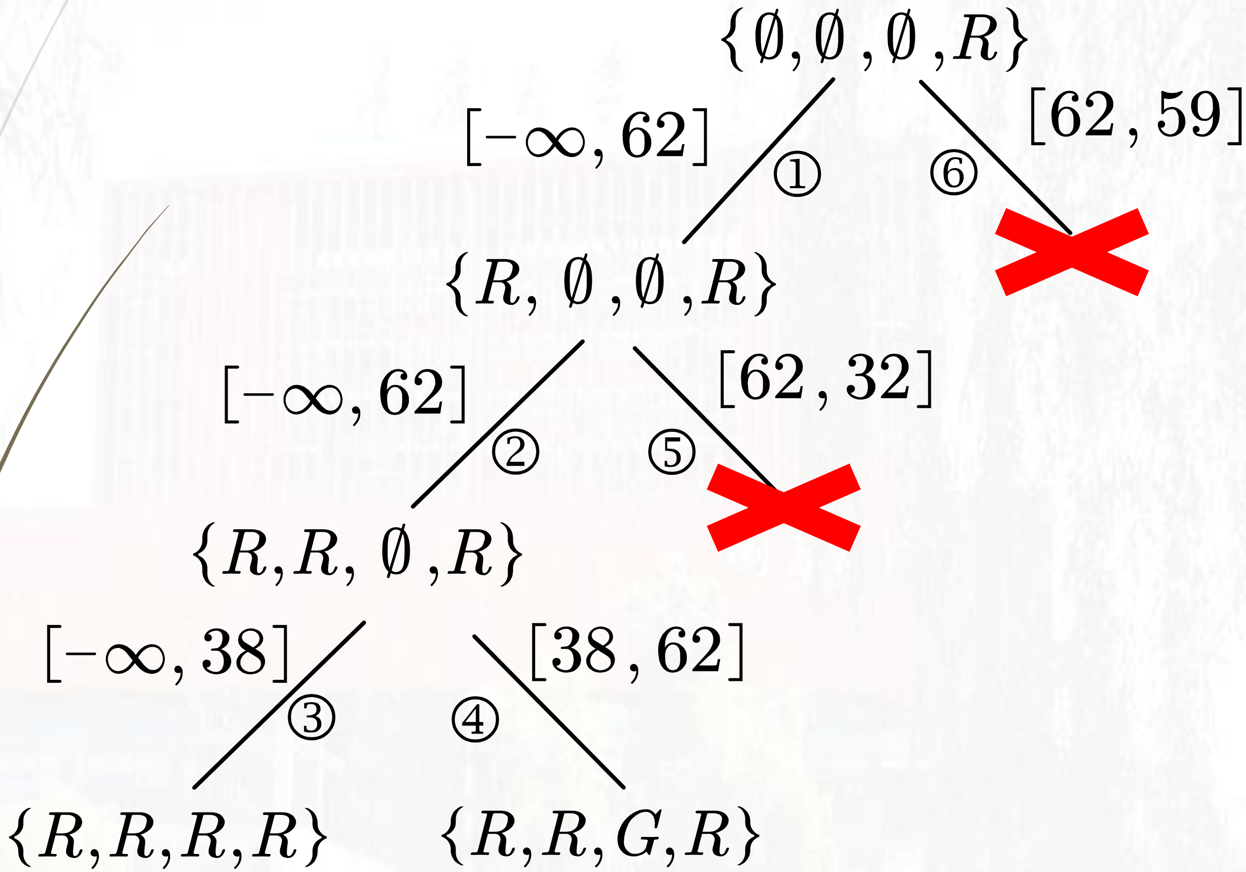
Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ (*i.e.*, $R_{F_2 \to x_4}(x_4 = R)$)



Example of State Pruning (SP)

- Calculate the message from function F_2 to variable x_4, when $x_4 = R$ (*i.e.*, $R_{F_2 \rightarrow x_4}(x_4 = R)$)



Experimental Evaluation

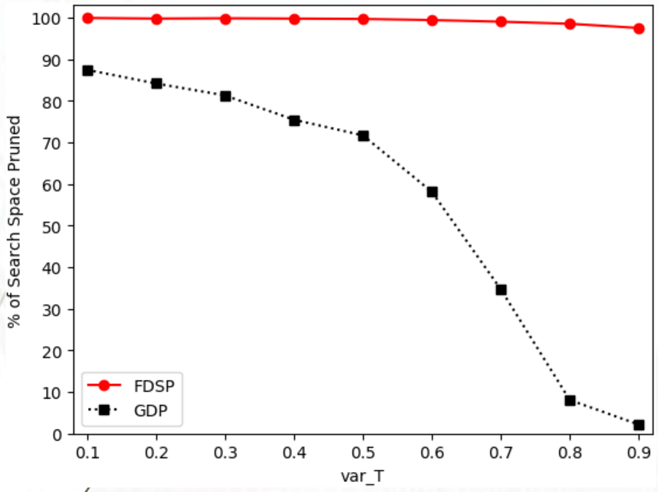
- **Evaluation criteria**
  - *Percentage of search space pruned*
  - *Runtime*
- **Experimental configuration**
  - Complexity of n-ary DCOPs
    - *Function-nodes number*
    - *Average/Maximal arity*
    - *Domain Size*
    - *Variable tightness* (*var_T*)

$$\mathrm{var\_T} = 1 - \frac{\text{number of variable} - \text{nodes}}{\text{total number of arities}}$$

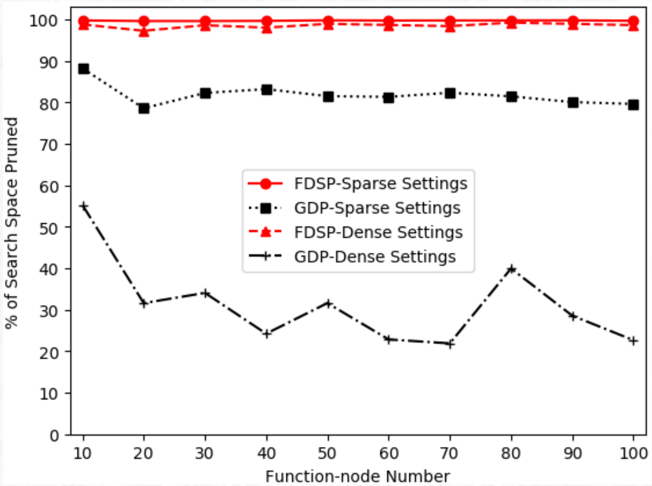
  - Random DCOPs
    - *Function-nodes number*: 100
    - *Maximal arity*: [2,7]
    - *Cost Range*: [1,100]
    - *Domain Size*: [2,10]
    - *var_T* $\in [0.1, 0.5]$ (sparse)

      var_T $\in (0.5, 0.9]$ (dense)

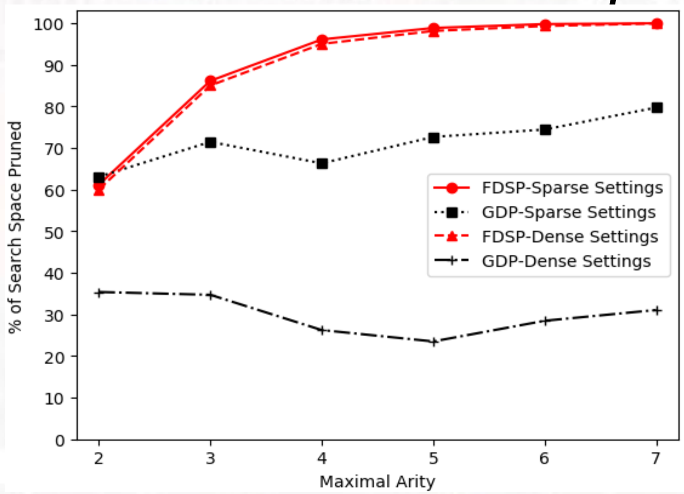
Experimental Results (Percentage of search space pruned)



Performance comparison on ***different var_T***

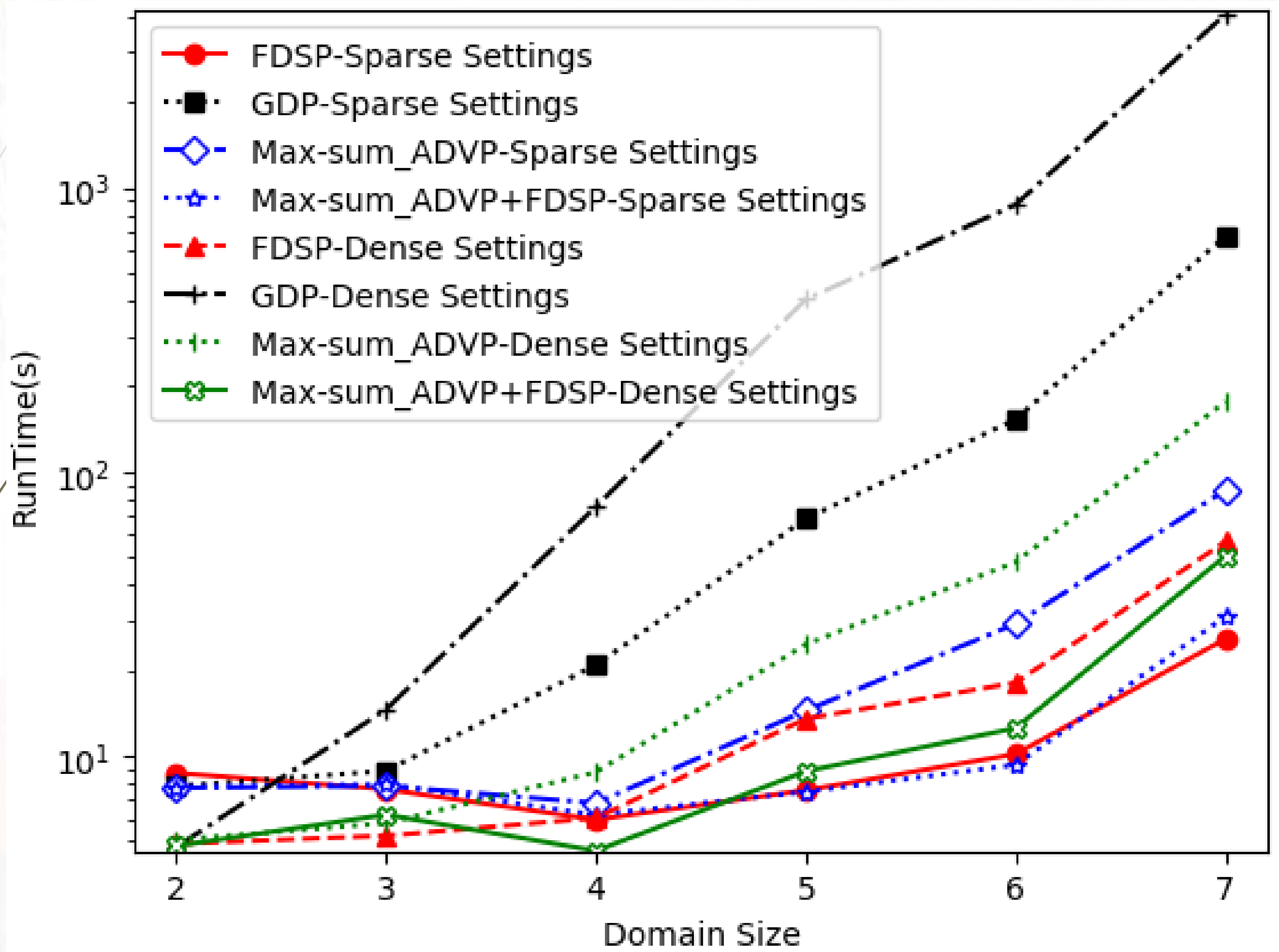


Performance comparison on ***different function-node numbers***



Performance comparison on ***different maximal arities***

Experimental Results (Runtime)



Runtime on ***different domain sizes***

Conclusion

- Propose a generic, fast and easy-to-use method, named FDSP
  - Function decomposing (FD) with *Dynamic Programming* to effectively compute the function estimations
  - State pruning (SP) based on *branch and bound* to reduce the search space
- *Theoretically* prove that FDSP provides monotonically non-increasing bounds and never prunes the assignment with the maximum utility
- *Empirical* evaluation shows that FDSP can reduce at least 97% of the search space and effectively accelerate Max-Sum and its variants

Thank you