Distributed Constraint Optimization

Gauthier Picard

ONERA/DTIS

gauthier.picard@onera.fr

— Some contents taken from OPTMAS 2011 and OPTMAS-DCR 2014 Tutorials—

Contents

Constraint Optimization Problems

Sometimes satisfaction is not possible

- Overconstrained problem
- Solution is not binary

Switch from satisfaction to optimization

- Minimizing the number of violated constraints
- Minimizing the cost of violated constraints
- Maximizing the overall utility of the system
- ...

DCOP Framework

Motivations

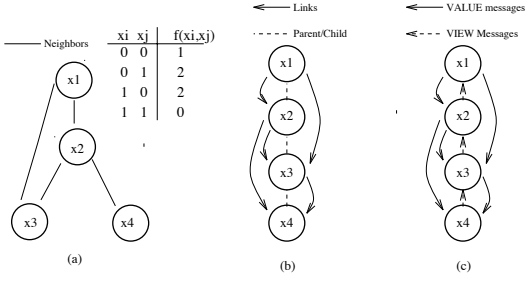
- In dynamic and complex environments not all constraints can be satisfied completely
- Satisfaction $\rightarrow$ **Optimisation** (combinatorial)
  - ex: minimizing the number of unchecked constraints, minimizing the sum of the costs of violated constraints, etc.

Definition (DCOP)

A *DCOP* is a DCSP $\langle A, X, D, C, \phi \rangle$ with

- a cost function $f_{ij} : D_i \times D_j \mapsto \mathbb{N} \cup \infty$ for each pair x_i, x_j
- an objective function $F : D \mapsto \mathbb{N} \cup \infty$ evaluating an assignment $\mathcal{A}$ with $f_{ij}(d_i, d_j)$ for each pair x_i, x_j

DCOP Framework (cont.)



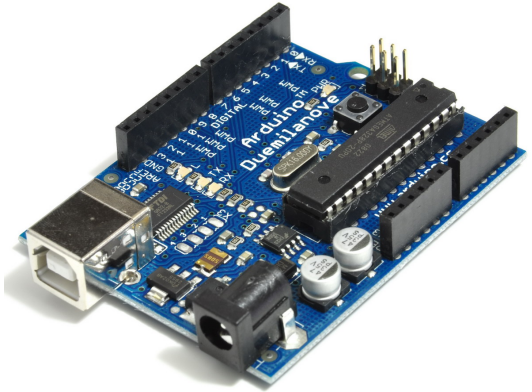
Objective Function

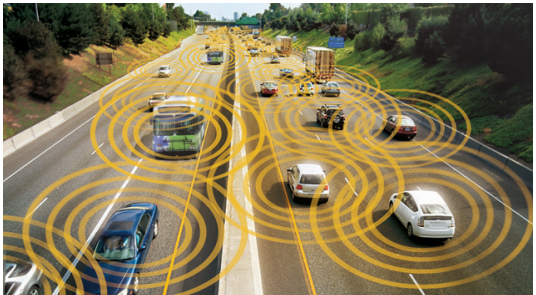
$$F(\mathcal{A}) = \sum_{x_i, x_j \in X} f_{ij}(d_i, d_j) \text{ where } x_i \leftarrow d_i \text{ and } x_i \leftarrow d_i \text{ in } \mathcal{A}$$

- $F(\{(x_1, 0), (x_2, 0), (x_3, 0), (x_4, 0)\}) = 4$
- $F(\{(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 1)\}) = 0$

$$\mathcal{A}^* = \{(x_1, 1), (x_2, 1), (x_3, 1), (x_4, 1)\}$$

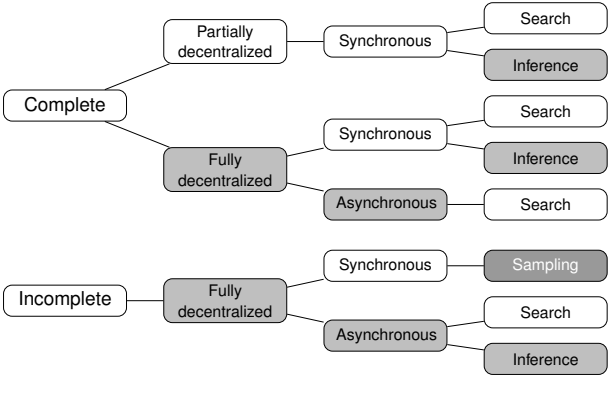
Application Domains





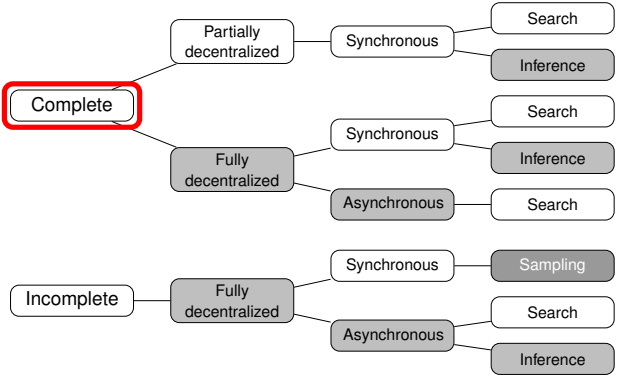
DCOP Algorithms

See (FIORETTO et al., 2018)



DCOP Algorithms

See (FIORETTO et al., 2018)

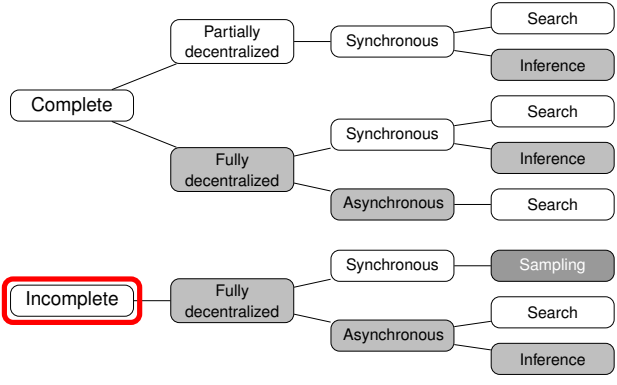


Important metrics

- Agent complexity
- Network loads
- Message size

DCOP Algorithms

See (FIORETTO et al., 2018)



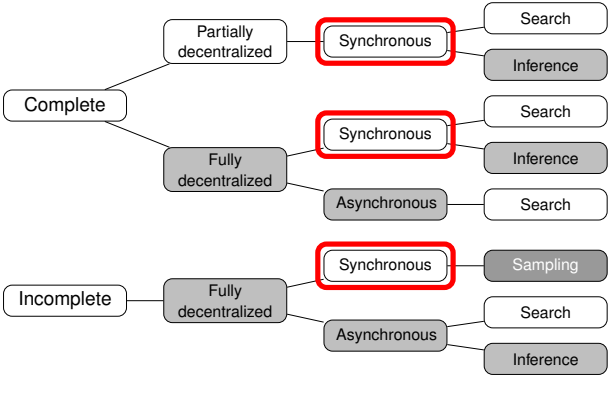
Important metrics

- Agent complexity
- Network loads
- Message size

- Anytime
- Quality guarantees
- Execution time vs. solution quality

DCOP Algorithms

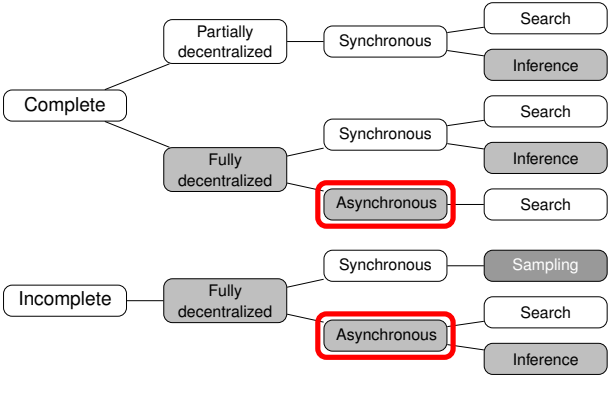
See (FIORETTO et al., 2018)



- Systematic process, divided in steps
- Each agent waits for particular messages before acting
- Consistent view of the search process
- Typically, increases idle-time

DCOP Algorithms

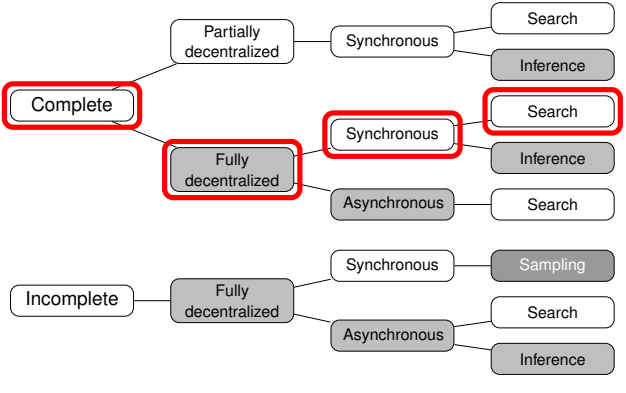
See (FIORETTO et al., 2018)



- Decision based on agents' local state
- Agents' actions do not depend on sequence of received messages
- Minimizes idle-time
- No guarantees on validity of local views

DCOP Algorithms

See (FIORETTO et al., 2018)



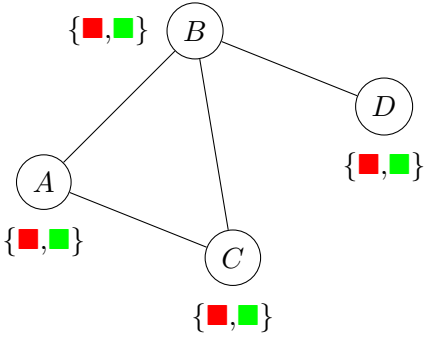
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)

Contents

Synchronous Branch-and-Bound (SBB)

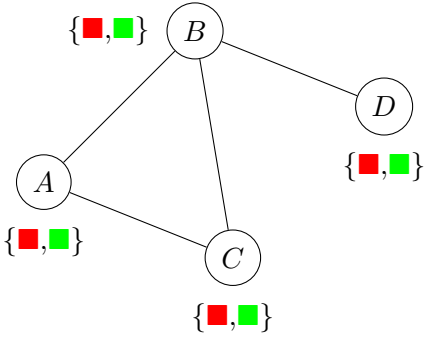
(HIRAYAMA and YOKOO, 1997)



x_i	x_j	(A,B)	(A,C)	(B,C)	(B,C)
■	■	5	5	5	3
■	■	8	10	4	8
■	■	20	20	3	10
■	■	3	3	3	3

Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



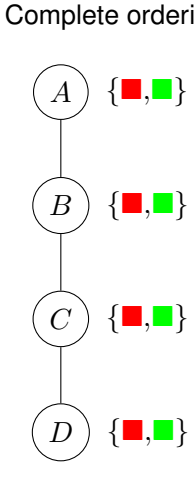
x_i	x_j	(A,B)	(A,C)	(B,C)	(B,C)
■	■	5	5	5	3
■	■	8	10	4	8
■	■	20	20	3	10
■	■	3	3	3	3

How do we solve this distributedly?

Synchronous Branch-and-Bound (SBB)

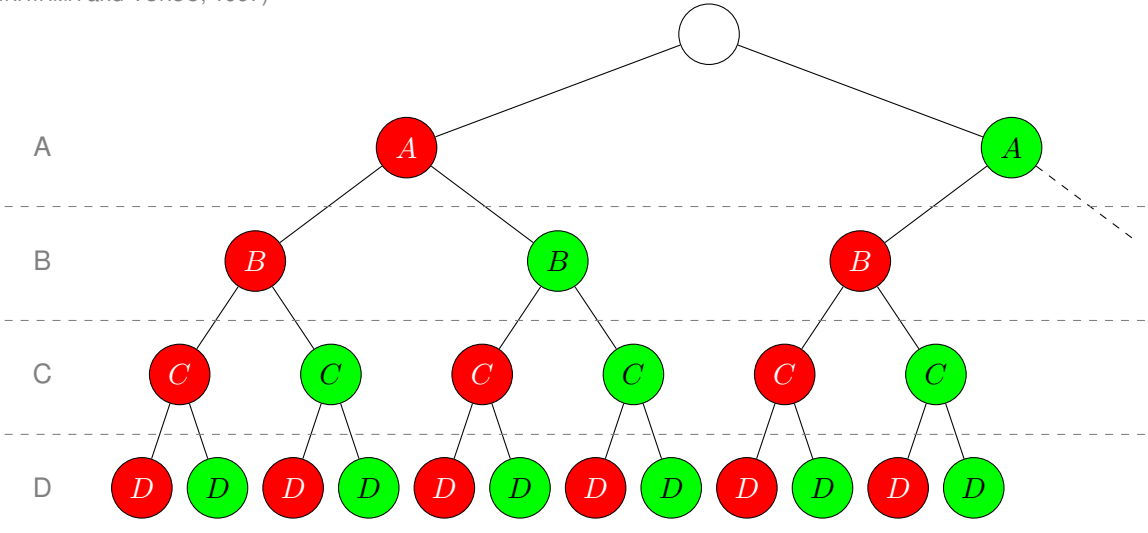
(HIRAYAMA and YOKOO, 1997)

- Agents operate on a complete ordering
- Agents exchange CPA messages containing partial assignments
- When a solution is found, its solution cost as an UB is broadcasted to all agents
- The UB is used for branch pruning



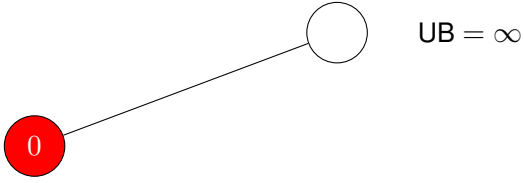
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



A

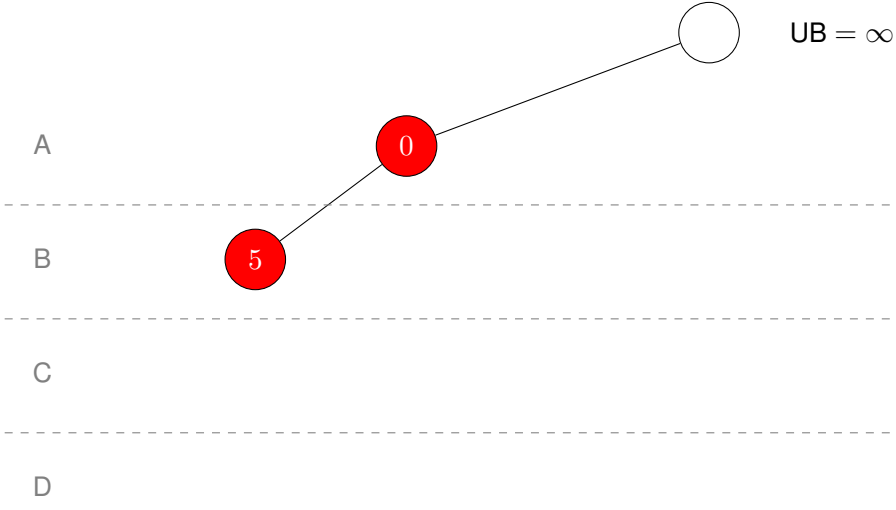
B

C

D

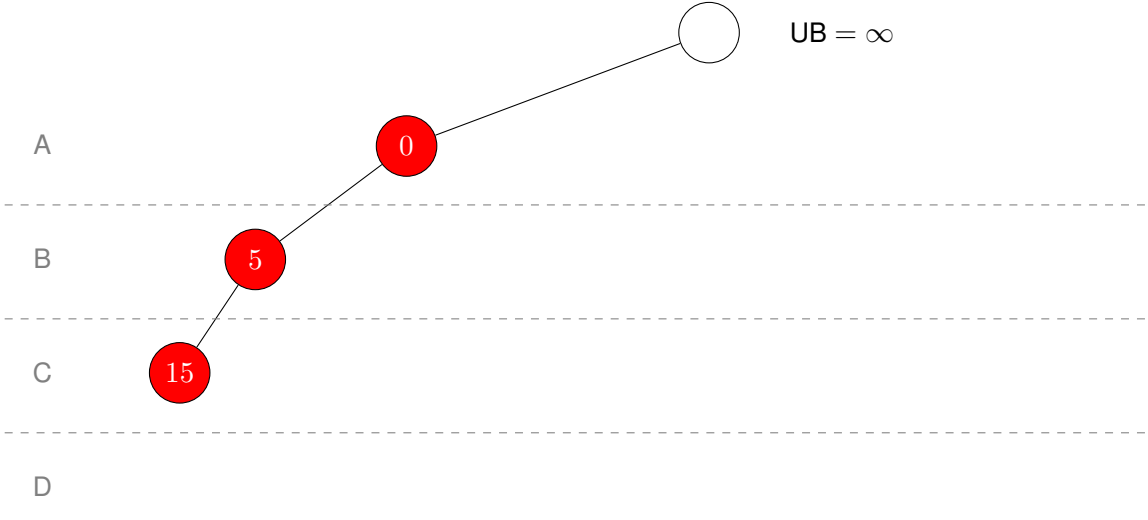
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



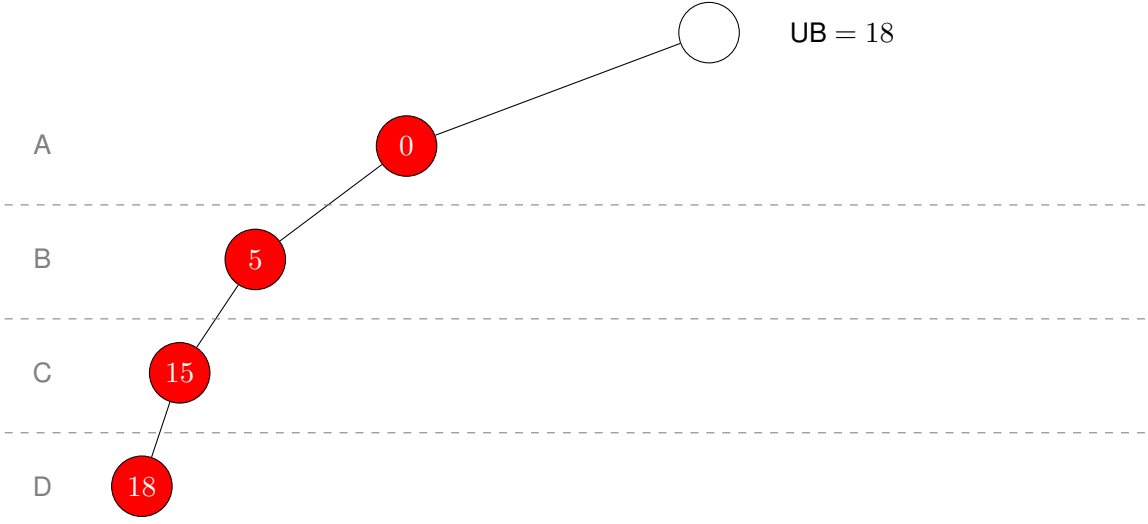
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



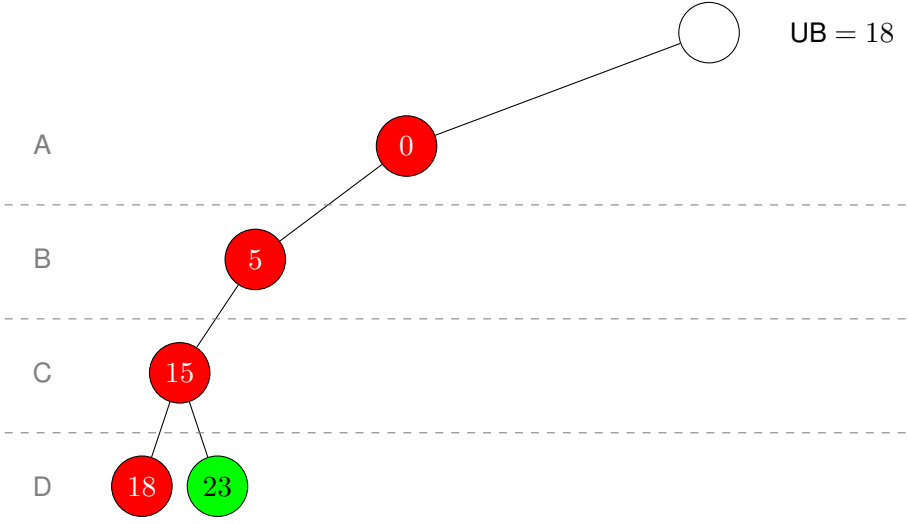
Synchronous Branch-and-Bound (SBB)

(Hirayama and Yokoo, 1997)



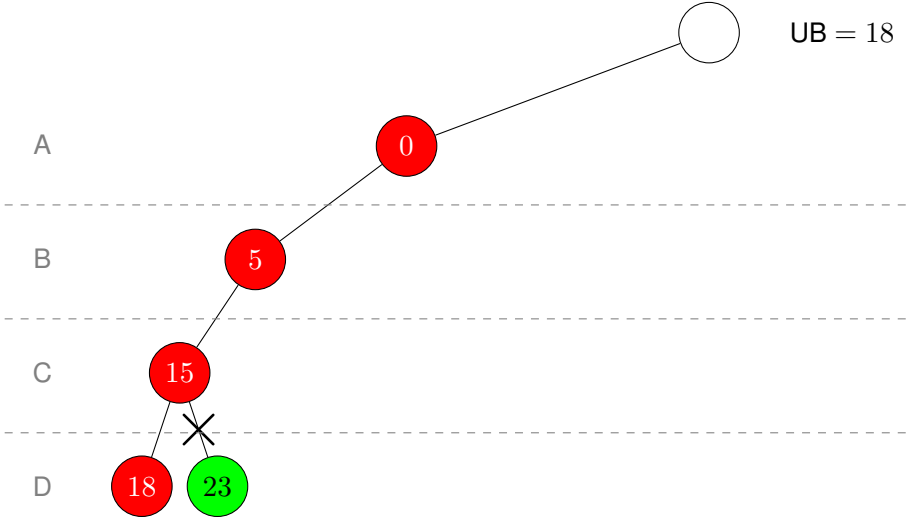
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



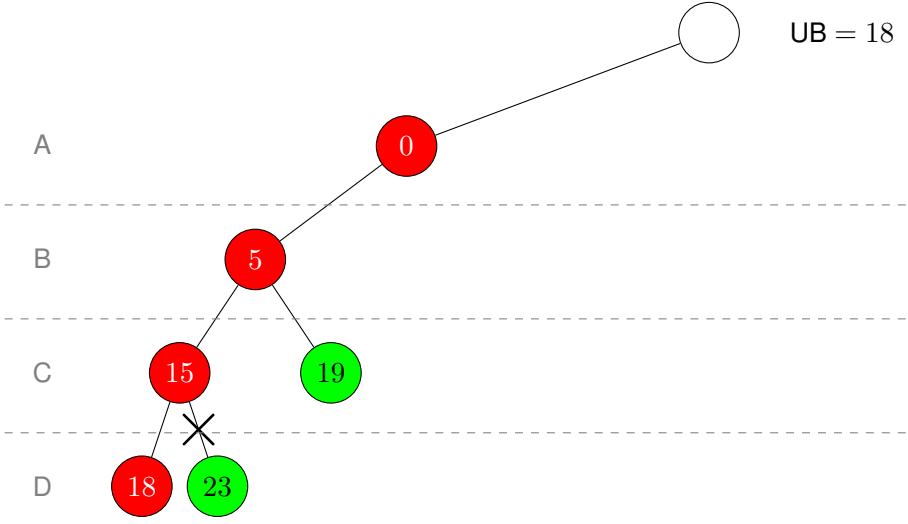
Synchronous Branch-and-Bound (SBB)

(Hirayama and Yokoo, 1997)



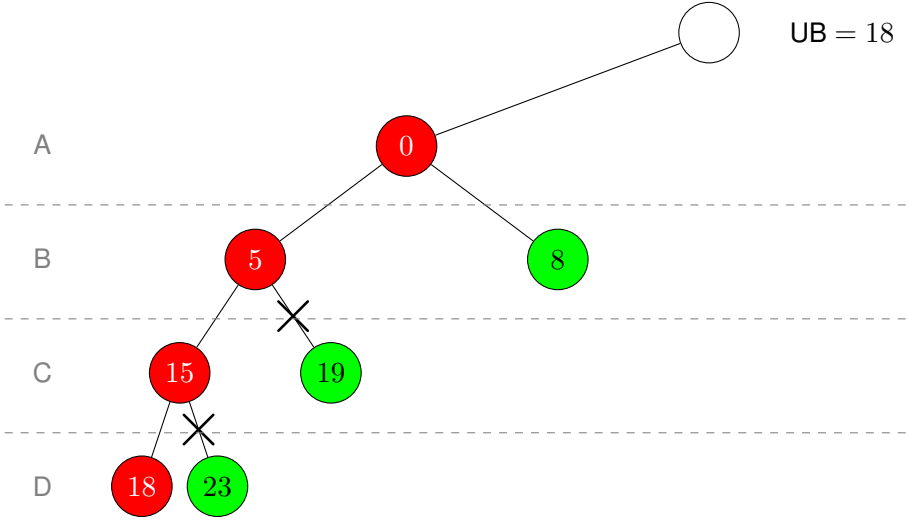
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



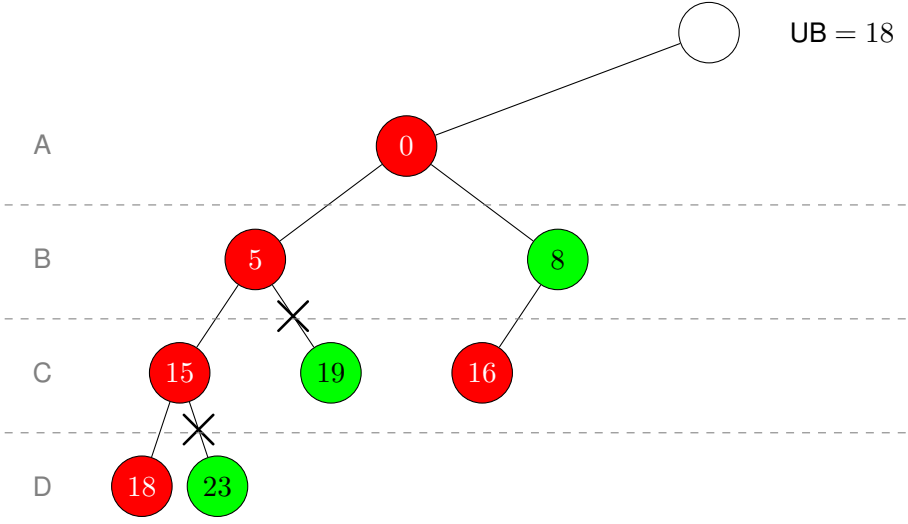
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



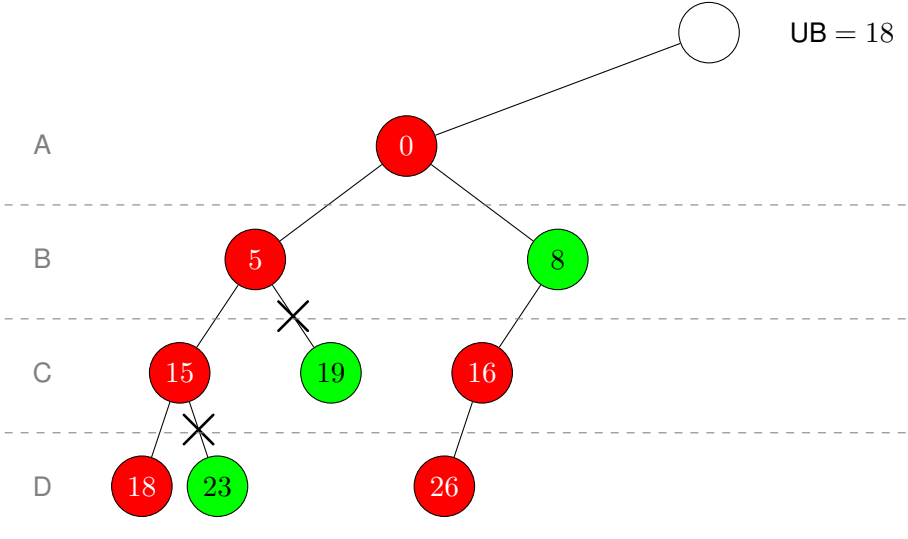
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



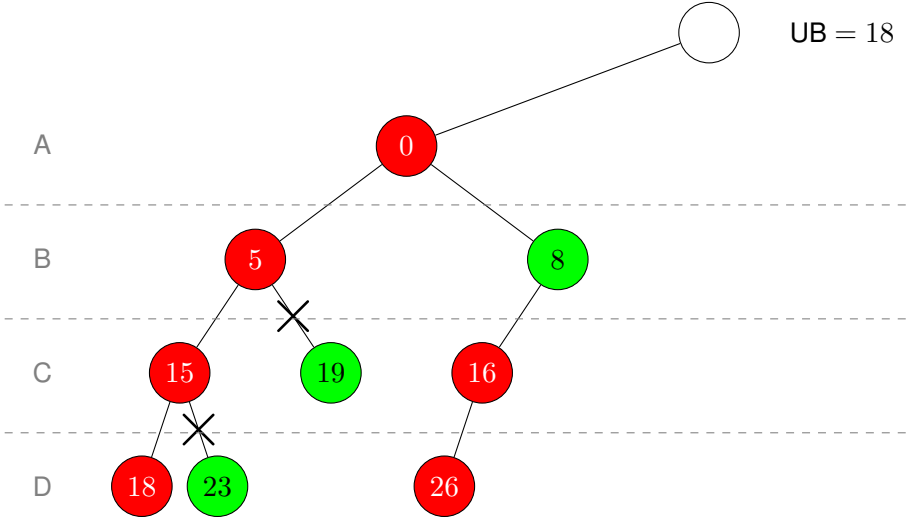
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



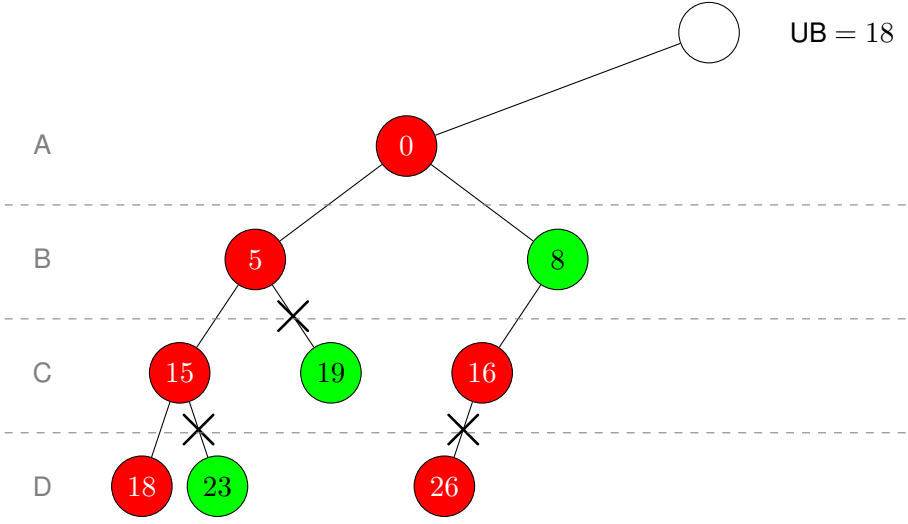
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



Synchronous Branch-and-Bound (SBB)

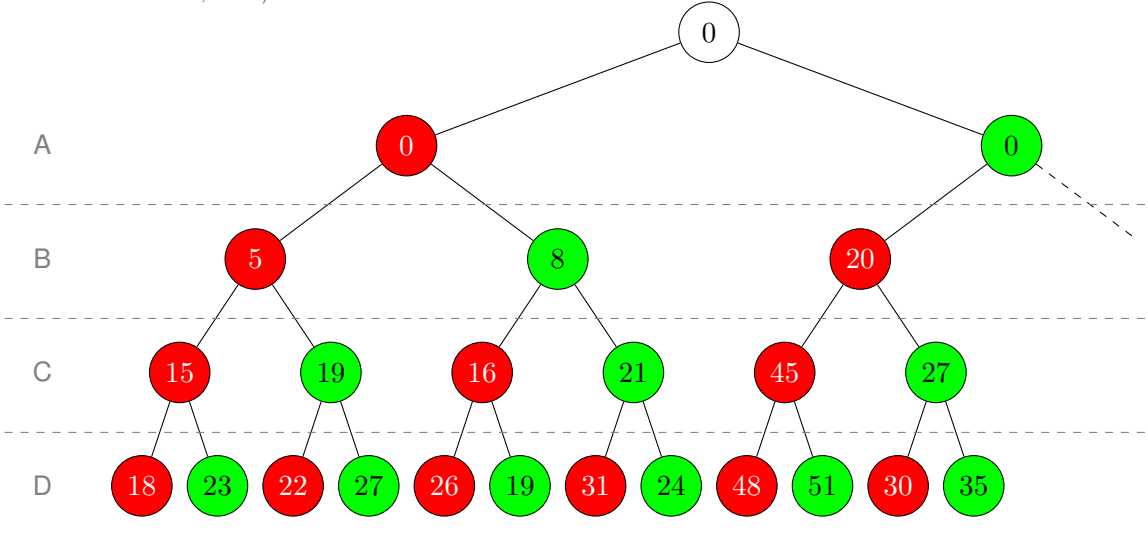
(HIRAYAMA and YOKOO, 1997)

	SBB
Correct the solution it finds is optimal	Yes
Complete it terminates	Yes
Message complexity max size of messages	$\mathcal{O}(n)$
Network load max number of messages	$\mathcal{O}(d^n)$
Runtime how long it takes	$\mathcal{O}(d^n)$

branching factor = b
num variables = n
domain size = d

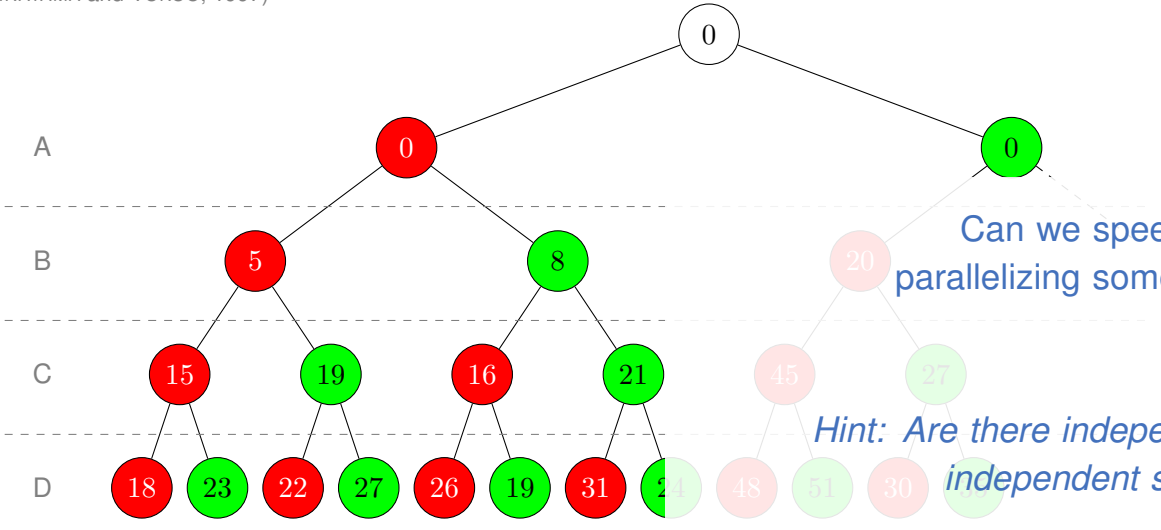
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)



Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)

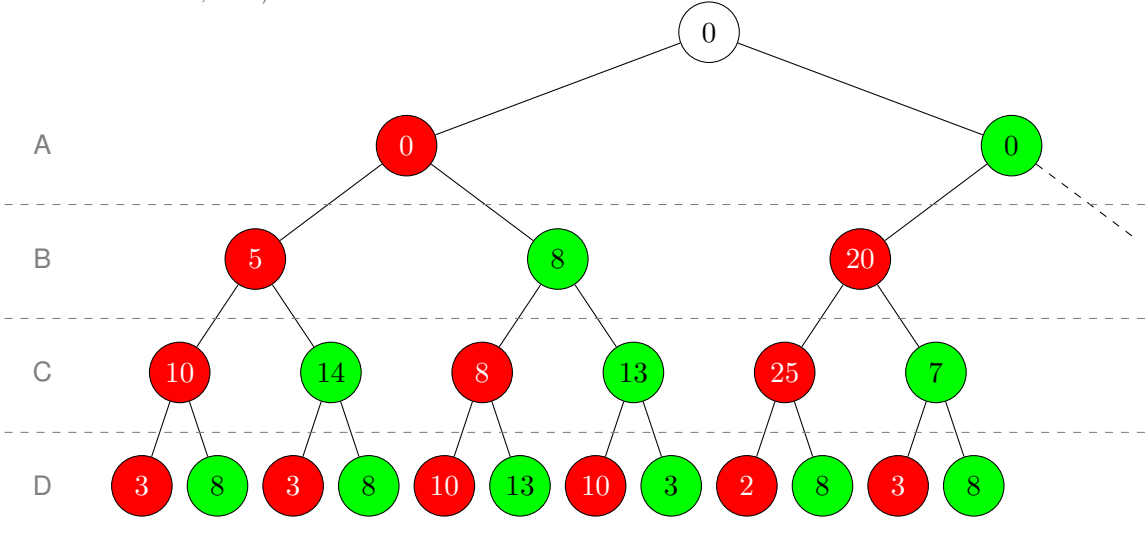


Can we spee
parallelizing som

Hint: Are there indepe
independent s

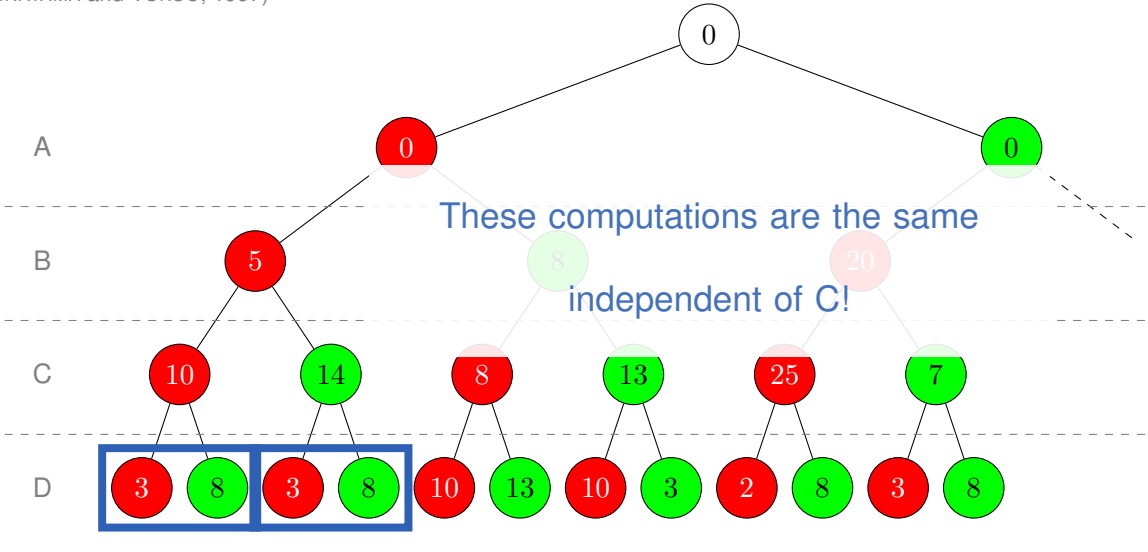
Synchronous Branch-and-Bound (SBB)

(HIRAYAMA and YOKOO, 1997)

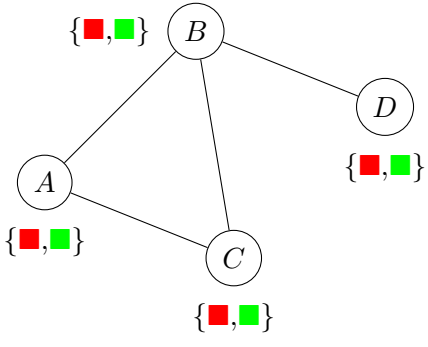


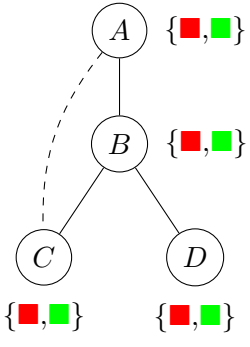
Synchronous Branch-and-Bound (SBB)

(Hirayama and Yokoo, 1997)

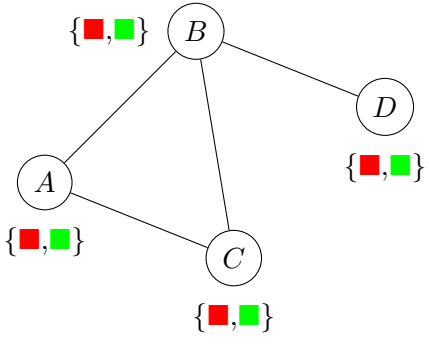


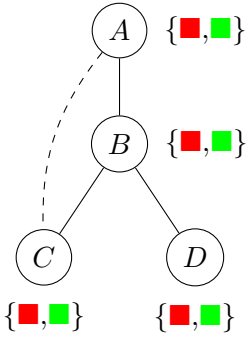
Pseudo-Tree





Pseudo-Tree

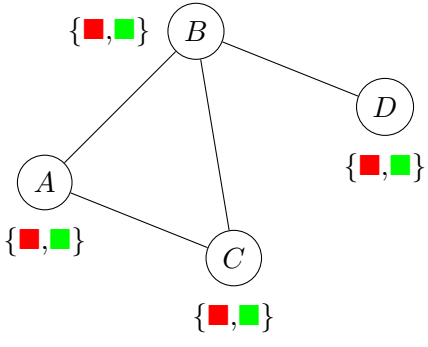


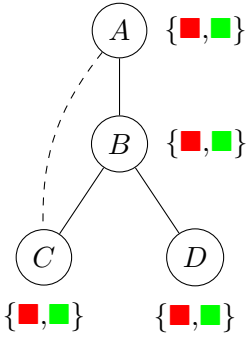


Definition (Pseudo-Tree)

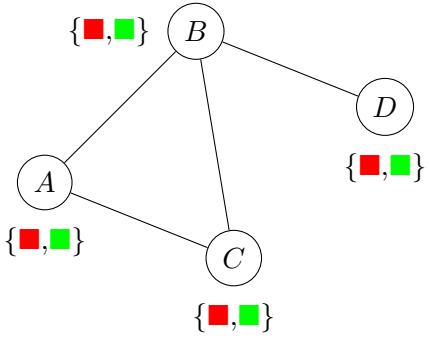
A spanning tree of the constraint graph such that no two nodes in sibling subtrees share a constraint in the constraint graph

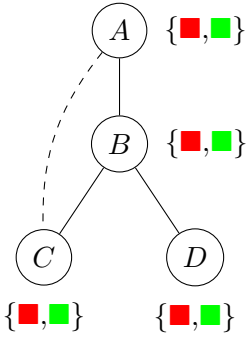
Pseudo-Tree





Pseudo-Tree



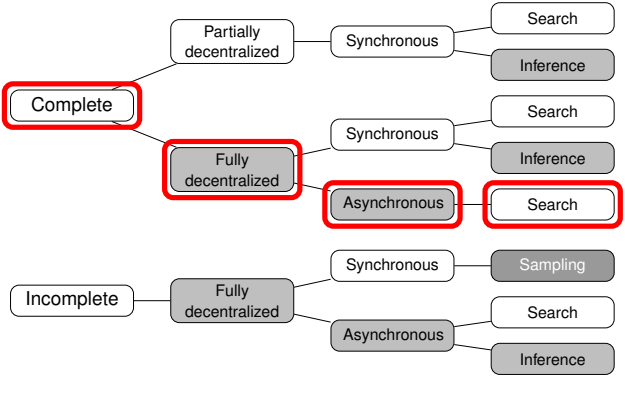


Definition (Pseudo-Tree)

A spanning tree of the constraint graph such that no two nodes in sibling subtrees share a constraint in the constraint graph

DCOP Algorithms

See (FIORETTO et al., 2018)



Asynchronous Distributed Optimisation (ADOPT)

(MODI et al., 2005)

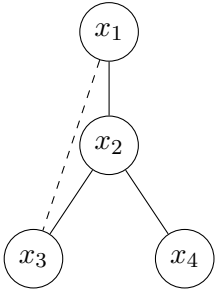
Asynchronous Distributed Optimisation (ADOPT) (Modi et al., 2005)

ADOPT: DFS tree (pseudotree)

ADOPT assumes that agents are arranged in a DFS tree:

- constraint graph $\rightarrow$ rooted graph (select a node as root)
- some links form a tree / others are backedges
- two constrained nodes must be in the same path to the root by tree links (same branch)

Every graph admits a DFS tree: DFS graph traversal



ADOPT Features

- Asynchronous algorithm
- Each time an agent receives a message:
  - Processes it (the agent may take a new value)
  - Sends VALUE messages to its children and pseudochildren
  - Sends a COST message to its parent
- Context: set of (variable value) pairs (as ABT agent view) of ancestor agents (in the same branch)
- Current context:
  - Updated by each VALUE message
  - If current context is not compatible with some child context, the later is initialized (also the child bounds)

ADOPT Procedures

```
Initialize
(1)   threshold ← 0; CurrentContext ← {};
(2)   forall d ∈ D_i, x_l ∈ Children do
(3)     lb(d, x_l) ← 0; t(d, x_l) ← 0;
(4)     ub(d, x_l) ← Inf; context(d, x_l) ← {}; enddo;
(5)   d_i ← d that minimizes LB(d);
(6)   backTrack;

when received (THRESHOLD, t, context)
(7)   if context compatible with CurrentContext:
(8)     threshold ← t;
(9)     maintainThresholdInvariant;
(10)    backTrack; endif;

when received (TERMINATE, context)
(11)  record TERMINATE received from parent;
(12)  CurrentContext ← context;
(13)  backTrack;

when received (VALUE, (x_j, d_j))
(14)  if TERMINATE not received from parent:
(15)    add (x_j, d_j) to CurrentContext;
(16)    forall d ∈ D_i, x_l ∈ Children do
(17)      if context(d, x_l) incompatible with CurrentContext:
(18)        lb(d, x_l) ← 0; t(d, x_l) ← 0;
(19)        ub(d, x_l) ← Inf; context(d, x_l) ← {}; endif; enddo;
(20)    maintainThresholdInvariant;
(21)    backTrack; endif;

when received (COST, x_k, context, lb, ub)
(22)  d ← value of x_i in context;
(23)  remove (x_i, d) from context;
(24)  if TERMINATE not received from parent:
(25)    forall (x_j, d_j) ∉ context and x_j is not my neighbor do
(26)      add (x_j, d_j) to CurrentContext;enddo;
(27)    forall d' ∈ D_i, x_l ∈ Children do
(28)      if context(d', x_l) incompatible with CurrentContext:
(29)        lb(d', x_l) ← 0; t(d', x_l) ← 0;
(30)        ub(d', x_l) ← Inf; context(d', x_l) ← {};endif;enddo;endif;
(31)  if context compatible with CurrentContext:
(32)    lb(d, x_k) ← lb;
(33)    ub(d, x_k) ← ub;
(34)    context(d, x_k) ← context;
(35)    maintainChildThresholdInvariant;
(36)    maintainThresholdInvariant; endif;
(37)  backTrack;

 procedure backTrack
(38)  if threshold == UB:
(39)    d_i ← d that minimizes UB(d);
(40)  else if LB(d_i) > threshold:
(41)    d_i ← d that minimizes LB(d);endif;
(42)  SEND (VALUE, (x_i, d_i))
(43)      to each lower priority neighbor;
(44)  maintainAllocationInvariant;
(45)  if threshold == UB:
(46)    if TERMINATE received from parent
(47)    or x_i is root:
(48)      SEND (TERMINATE,
(49)      CurrentContext ∪ {(x_i, d_i)})
(50)        to each child;
(51)      Terminate execution; endif;endif;
(52)  SEND (COST, x_i, CurrentContext, LB, UB)
        to parent;
```

Algorithm 1: ADOPT Procedures

ADOPT Messages

- value$(parent \rightarrow children \cup pseudochildren, a)$: parent informs descendants that it has taken value a
- cost$(child \rightarrow parent, lowerbound, upperbound, context)$: child informs parent of the best cost of its assignement; attached context to detect obsolescence
- threshold$(parent \rightarrow child, t)$: minimum cost of solution in child is at least t
- termination$(parent \rightarrow children)$: sent when $LB = UB$

ADOPT Data Structures

1. **Current context** (agent view): values of higher priority constrained agents

x_i	x_j	...
a	c	...

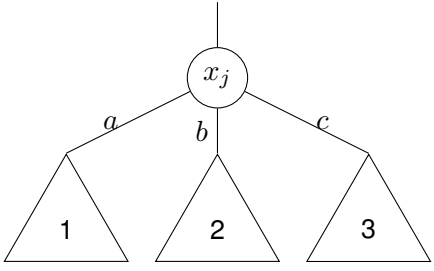
2. **Bounds** (for each value, child)

- lower bounds
- upper bounds
- thresholds
- contexts

x_j	a	b	c	d
$lb(x_k)$	3	0	0	0
$ub(x_k)$	∞	∞	∞	∞
$th(x_k)$	1	0	0	0
$context(x_k)$				

- Stored contextes must be active: $context \in currentcontext$
- If a context becomes no active, it is removed ($lb \leftarrow 0, th \leftarrow 0, ub \leftarrow \infty$)

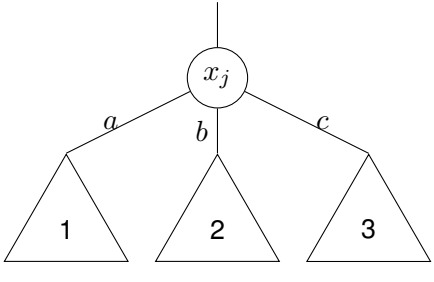
ADOPT Bounds



ADOPT Bounds

$\delta(value) =$ cost with higher agents

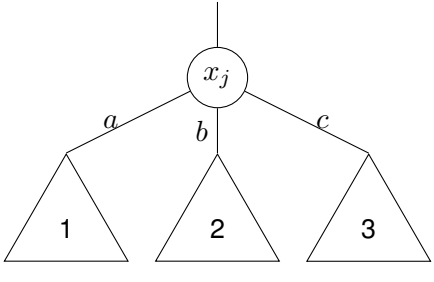
$$\delta(b) = \sum_{i \in curctx} c_{ij}(a, b)$$



ADOPT Bounds

$\delta(value) =$ cost with higher agents

$$\delta(b) = \sum_{i \in curctx} c_{ij}(a, b)$$

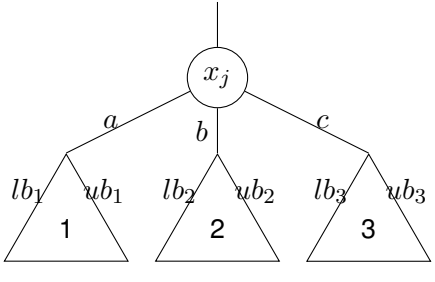


$$OPT(x_j, ctx) = \min_{d \in d_j} \delta(d) + \sum_{x_k \in children} OPT(x_k, ctx \cup (x_j, d))$$

ADOPT Bounds

$\delta(value) =$ cost with higher agents

$$\delta(b) = \sum_{i \in curctx} c_{ij}(a, b)$$

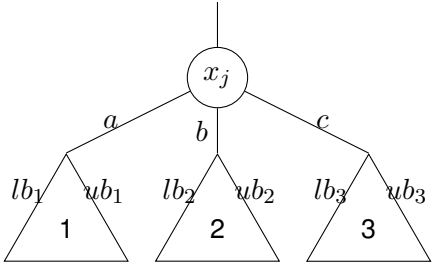


$$OPT(x_j, ctx) = \min_{d \in d_j} \delta(d) + \sum_{x_k \in children} OPT(x_k, ctx \cup (x_j, d))$$

ADOPT Bounds

$\delta(value)$ = cost with higher agents

$$\delta(b) = \sum_{i \in curctx} c_{ij}(a, b)$$



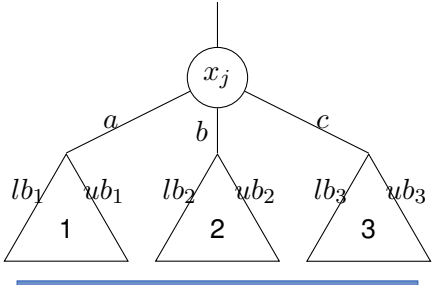
$[lb_k, ub_k]$ = cost of lower agents

$$OPT(x_j, ctx) = \min_{d \in d_j} \delta(d) + \sum_{x_k \in children} OPT(x_k, ctx \cup (x_j, d))$$

ADOPT Bounds

$\delta(value)$ = cost with higher agents

$$\delta(b) = \sum_{i \in curctx} c_{ij}(a, b)$$



$[lb_k, ub_k]$ = cost of lower agents

$$OPT(x_j, ctx) = \min_{d \in d_j} \delta(d) + \sum_{x_k \in children} OPT(x_k, ctx \cup (x_j, d))$$

$$LB(b) = \delta(b) + \sum_{x_k \in children} lb(b, x_k)$$

$$LB = \min_{b \in d_j} LB(b)$$

$$UB(b) = \delta(b) + \sum_{x_k \in children} ub(b, x_k)$$

$$UB = \min_{b \in d_j} UB(b)$$

ADOPT Value Assignment

- An ADOPT agent takes the value with minimum LB
- Eager behavior:
  - Agents may constantly change value
  - Generates many context changes
- Threshold:
  - lower bound of the cost that children have from previous search
  - parent distributes threshold among children
  - incorrect distribution does not cause problems: the child with minor allocation would send a COST to the parent later, and the parent will rebalance the threshold distribution

ADOPT Properties

- For any x_i, $LB \leq OPT(x_l, ctx) \leq UB$
- For any x_i, its threshold reaches UB
- For any x_i, its final threshold is equal to $OPT(x_l, ctx)$

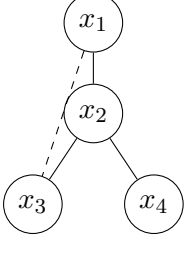
$\rightarrow$ ADOPT terminates with the optimal solution

ADOPT Example

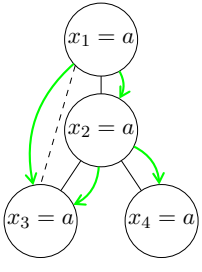
- 4 variables (4 agents) x_1, x_2, x_3 and x_4 with $D = \{a, b\}$
- 4 binary identical cost functions

x_i	x_j	cost
a	a	1
a	b	2
b	a	2
b	b	0

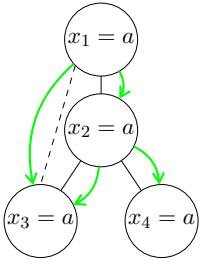
- Constraint graph

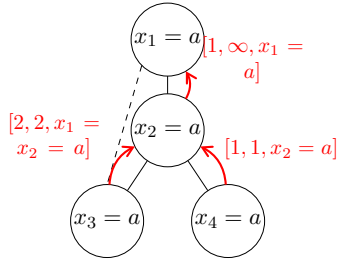


ADOPT Example (cont.)

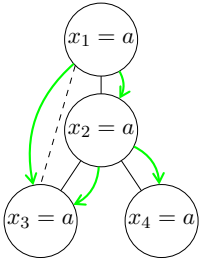


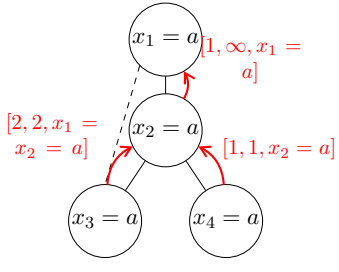
ADOPT Example (cont.)

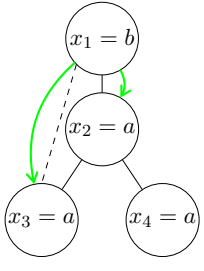




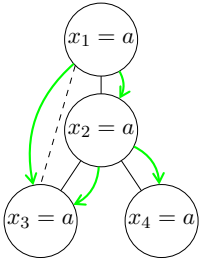
ADOPT Example (cont.)

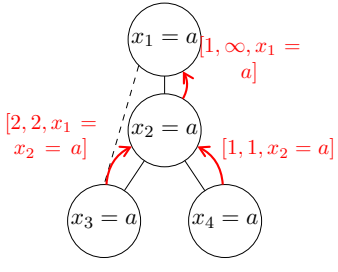


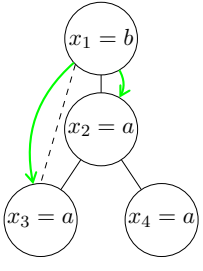


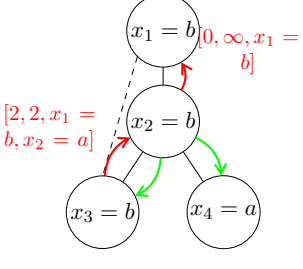


ADOPT Example (cont.)

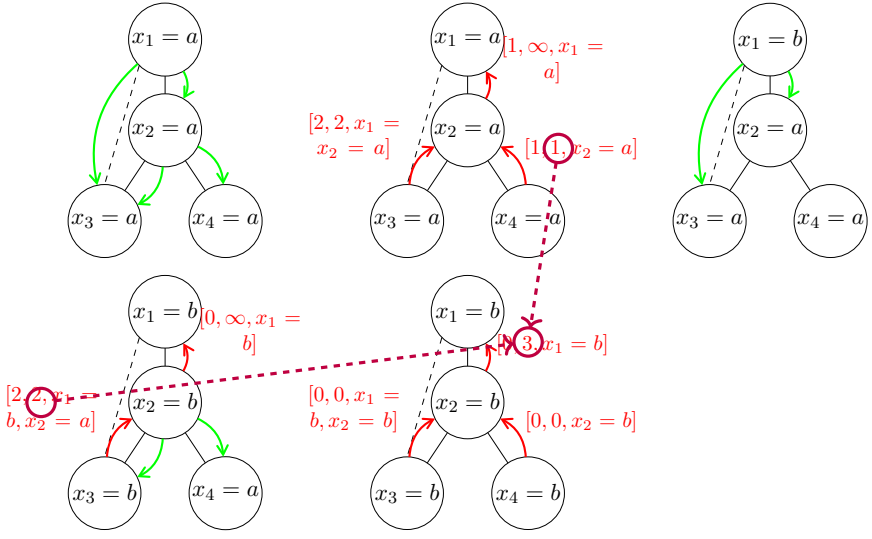




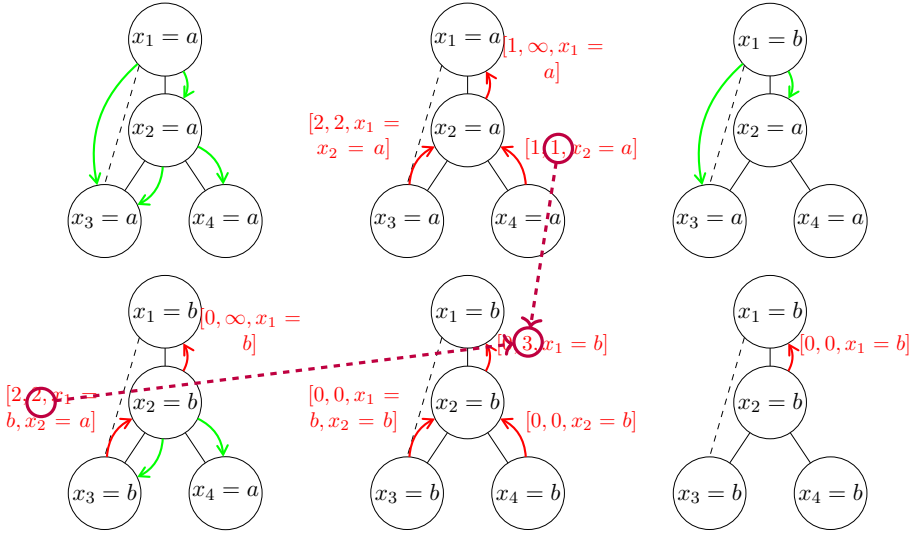




ADOPT Example (cont.)



ADOPT Example (cont.)



ADOPT

(MODI et al., 2005)

	SBB	ADOPT
Correct the solution it finds is optimal	Yes	Yes
Complete it terminates	Yes	Yes
Message complexity max size of messages	$\mathcal{O}(n)$	$\mathcal{O}(n)$
Network load max number of messages	$\mathcal{O}(d^n)$	$\mathcal{O}(d^n)$
Runtime how long it takes	$\mathcal{O}(d^n)$	$\mathcal{O}(d^n)$

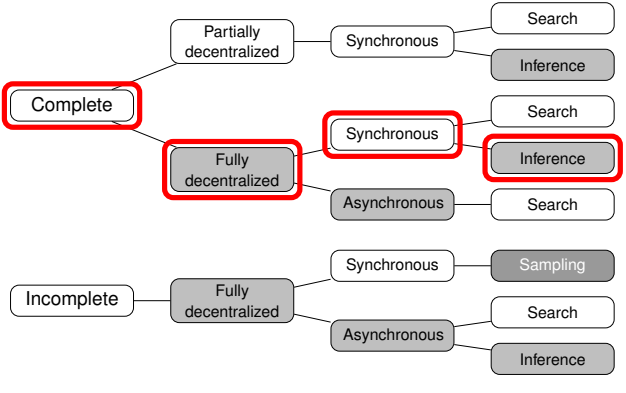
branching factor = b

num variables = n

domain size = d

DCOP Algorithms

See (FIORETTO et al., 2018)



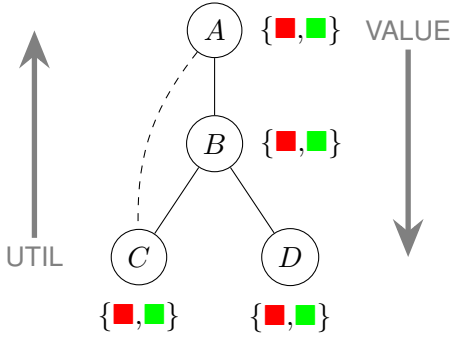
Distributed Pseudotree Optimization Procedure (DPOP)

(PETCU and FALTINGS, 2005b)

DPOP

(PETCU and FALTINGS, 2005b)

- Extension of the Bucket Elimination (BE)
- Agents operate on a pseudo-tree ordering
- UTIL phase: Leaves to root
- VALUE phase: Root to leaves



DPOP

(PETCU and FALTINGS, 2005b)

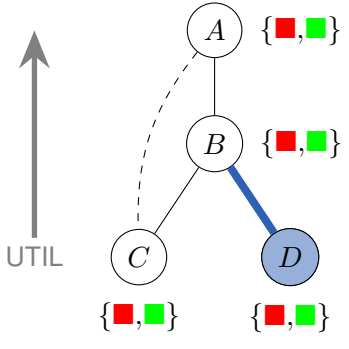
B	D	(B, D)
r	r	3
r	g	8
g	r	10
g	g	3

$\min\{3, 8\} = 3$

$\min\{10, 3\} = 3$

Message to B

B	cost
r	3
g	3



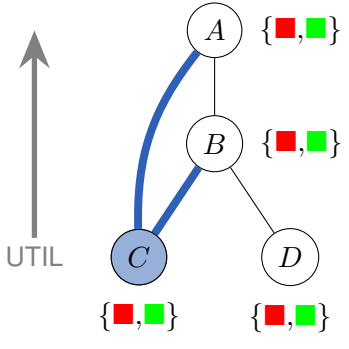
DPOP

(PETCU and FALTINGS, 2005b)

A	B	C	(B,C)	(A,C)	cost
r	r	r	5	5	10
r	r	g	4	8	12
r	g	r	3	5	8
r	g	g	3	8	11
g	r	r	5	10	15
g	r	g	4	3	7
g	g	r	3	10	13
g	g	g	3	3	6

Message to B

A	B	cost
r	r	10
r	g	8
g	r	7
g	g	6



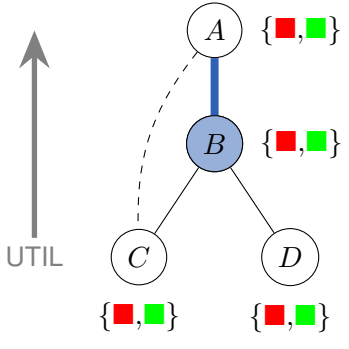
DPOP

(PETCU and FALTINGS, 2005b)

A	B	(A, B)	Util C	Util D	cost
r	r	5	10	53	18
r	g	8	8	3	19
g	r	20	7	3	30
g	g	3	6	3	12

Message to A

A	cost
r	18
g	12

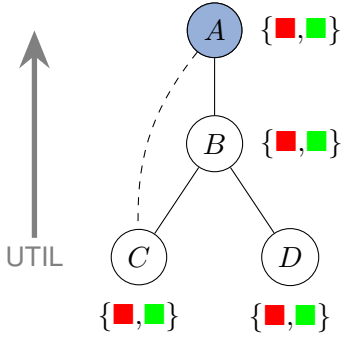


DPOP

(PETCU and FALTINGS, 2005b)

A	cost
r	18
g	12

optimal cost $= 12$

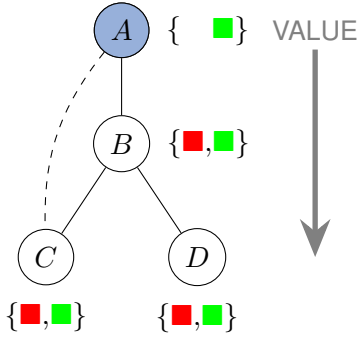


DPOP

(PETCU and FALTINGS, 2005b)

A	cost
r	18
g	12

- Select value for $A = g$
- Send MSG "$A = g$" to agents B and C

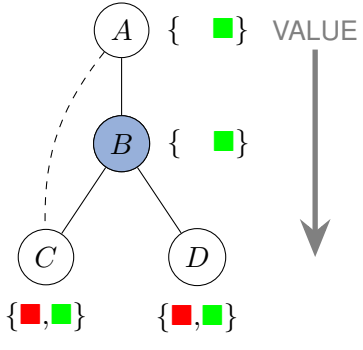


DPOP

(PETCU and FALTINGS, 2005b)

A	B	(A, B)	Util C	Util D	cost
r	r	5	10	53	18
r	g	8	8	3	19
g	r	20	7	3	30
g	g	3	6	3	12

- Select value for $B = g$
- Send MSG "$B = g$" to agents C and D

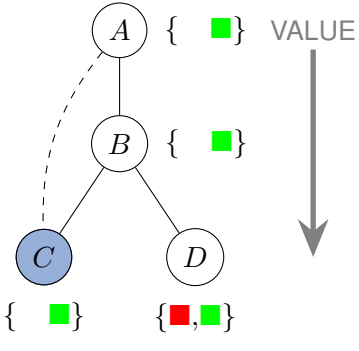


DPOP

(PETCU and FALTINGS, 2005b)

A	B	C	(B,C)	(A,C)	cost
r	r	r	5	5	10
r	r	g	4	8	12
r	g	r	3	5	8
r	g	g	3	8	11
g	r	r	5	10	15
g	r	g	4	3	7
g	**g**	r	3	10	13
g	**g**	g	3	3	6

- Select value for $C = g$



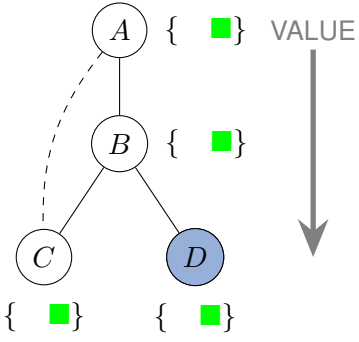
DPOP

(PETCU and FALTINGS, 2005b)

B	D	(B, D)
r	r	3
r	g	8
g	r	10
g	g	3

$\min\{3, 8\} = 3$

$\min\{10, 3\} = 3$



- Select value for $D = g$

DPOP

(PETCU and FALTINGS, 2005b)

	SBB	ADOPT	DPOP
Correct the solution it finds is optimal	Yes	Yes	Yes
Complete it terminates	Yes	Yes	Yes
Message complexity max size of messages	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(b^d)$
Network load max number of messages	$\mathcal{O}(d^n)$	$\mathcal{O}(d^n)$	$\mathcal{O}(n)$
Runtime how long it takes	$\mathcal{O}(d^n)$	$\mathcal{O}(d^n)$	$\mathcal{O}(b^n)$

branching factor = b
num variables = n
domain size = d

Contents

Approximate Algorithms for DCOPs

Complete algorithms

- e.g. ADOPT (MODI et al., 2005) and DPOP (PETCU and FALTINGS, 2005b)
  - ✓ complete
  - ✗ slow

Aproximate algorithms exist (fast, but sub-optimal in many cases)

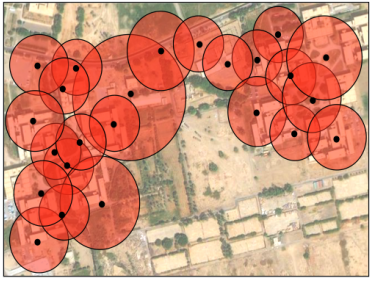
- Search algorithms
  - DBA (YOKOO, 2001), DSA (ZHANG et al., 2005), MGM (MAHESWARAN et al., 2004)
- Inference algorithms
  - Max-sum (FARINELLI et al., 2008)

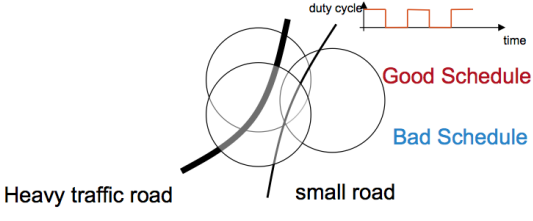
Why Approximate Algorithms

- Motivations
  - Often optimality in practical applications is not achievable
  - Fast good enough solutions are all we can have
- Example – Graph coloring
  - Medium size problem (about 20 nodes, three colors per node)
  - Number of states to visit for optimal solution in the worst case $3^{20} = 3M$ states
- Key problem
  - Provides guarantees on solution quality

Exemplar Application: Surveillance

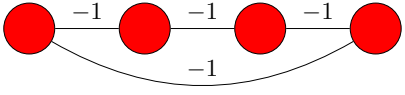
- Event Detection
  - Vehicles passing on a road
- Energy Constraints
  - Sense/Sleep modes
  - Recharge when sleeping
- Coordination
  - Activity can be detected by single sensor
  - Roads have different traffic loads
- Aim
  - Focus on road with more traffic load





Centralized Local Greedy approaches

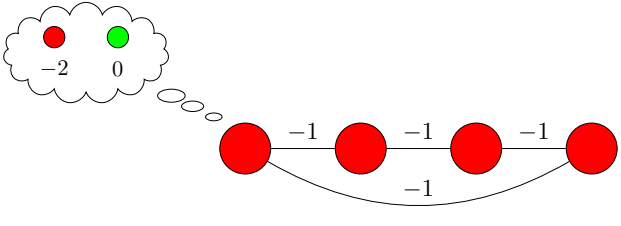
- Greedy local search
  - Start from random solution
  - Do local changes if global solution improves
  - Local: change the value of a subset of variables, usually one



-4

Centralized Local Greedy approaches

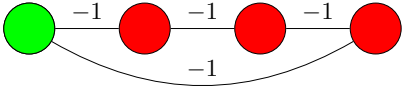
- Greedy local search
  - Start from random solution
  - Do local changes if global solution improves
  - Local: change the value of a subset of variables, usually one



-4

Centralized Local Greedy approaches

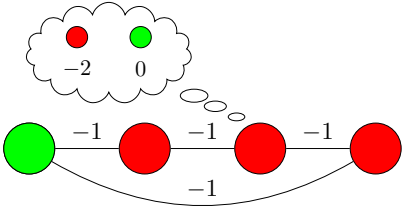
- Greedy local search
  - Start from random solution
  - Do local changes if global solution improves
  - Local: change the value of a subset of variables, usually one



-2

Centralized Local Greedy approaches

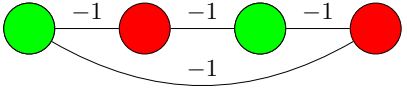
- Greedy local search
  - Start from random solution
  - Do local changes if global solution improves
  - Local: change the value of a subset of variables, usually one



-2

Centralized Local Greedy approaches

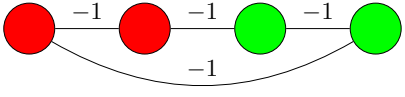
- Greedy local search
  - Start from random solution
  - Do local changes if global solution improves
  - Local: change the value of a subset of variables, usually one



-0

Centralized Local Greedy approaches

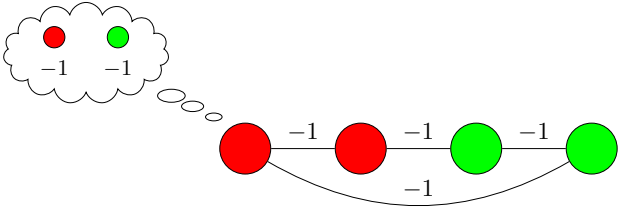
- Problems
  - Local minima
  - Standard solutions: Random Walk, Simulated Annealing



-2

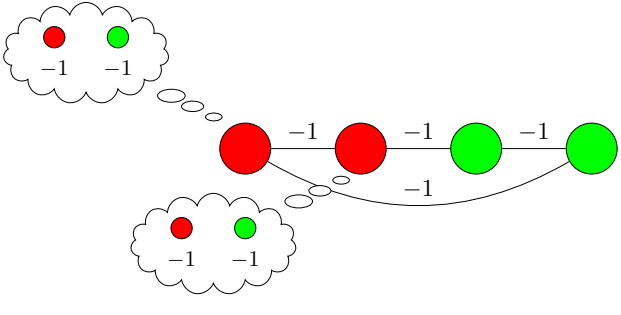
Centralized Local Greedy approaches

- Problems
  - Local minima
  - Standard solutions: Random Walk, Simulated Annealing



Centralized Local Greedy approaches

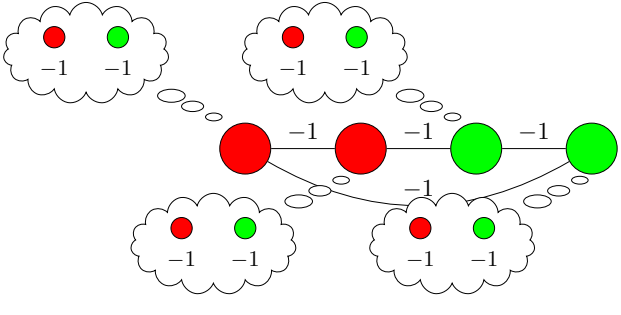
- Problems
  - Local minima
  - Standard solutions: Random Walk, Simulated Annealing



-2

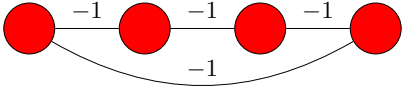
Centralized Local Greedy approaches

- Problems
  - Local minima
  - Standard solutions: Random Walk, Simulated Annealing



Distributed Local Greedy approaches

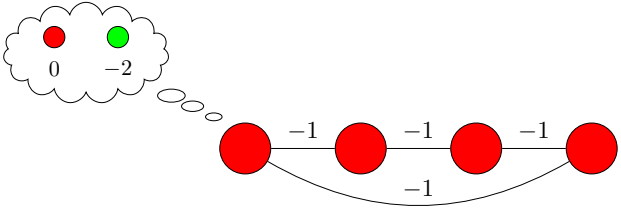
- Local knowledge
- Parallel execution
  - A greedy local move might be harmful/useless
  - Need coordination



-4

Distributed Local Greedy approaches

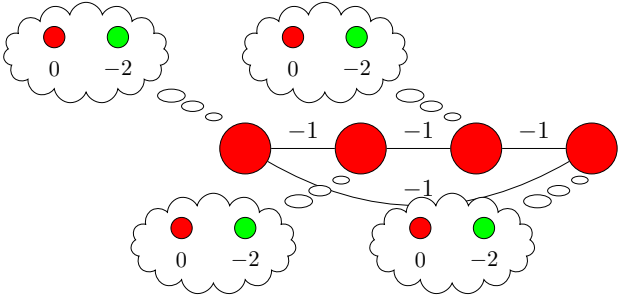
- Local knowledge
- Parallel execution
  - A greedy local move might be harmful/useless
  - Need coordination



-4

Distributed Local Greedy approaches

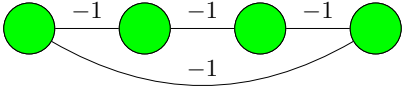
- Local knowledge
- Parallel execution
  - A greedy local move might be harmful/useless
  - Need coordination



-4

Distributed Local Greedy approaches

- Local knowledge
- Parallel execution
  - A greedy local move might be harmful/useless
  - Need coordination

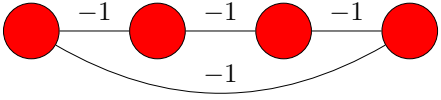


-4

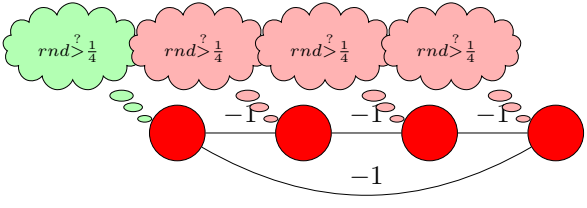
Distributed Stochastic Search Algorithm (DSA) (ZHANG et al., 2005)

- Greedy local search with activation probability to mitigate issues with parallel executions
- DSA-1: change value of one variable at time
- Initialize agents with a random assignment and communicate values to neighbors
- Each agent:
  - Generates a random number and execute only if rnd less than activation probability
  - When executing changes value maximizing local gain
  - Communicate possible variable change to neighbors

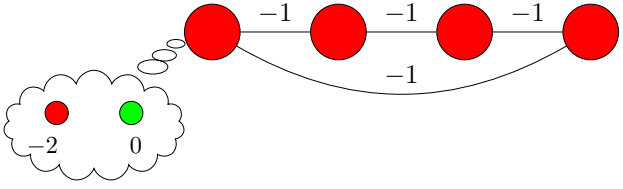
DSA-1: Execution Example



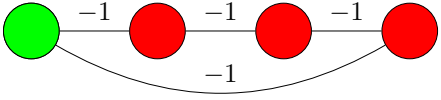
DSA-1: Execution Example



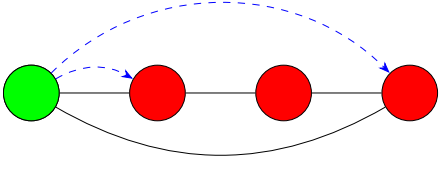
DSA-1: Execution Example



DSA-1: Execution Example



DSA-1: Execution Example



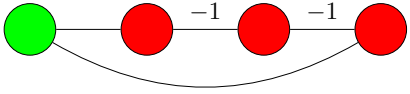
DSA-1: Discussion

- Extremely “cheap” (computation/communication)
- Good performance in various domains
  - e.g. target tracking (FITZPATRICK and MEERTENS, 2003; ZHANG et al., 2003)
  - Shows an anytime property (not guaranteed)
  - Benchmarking technique for coordination
- Problems
  - Activation probablity must be tuned (ZHANG et al., 2003)
  - No general rule, hard to characterise results across domains

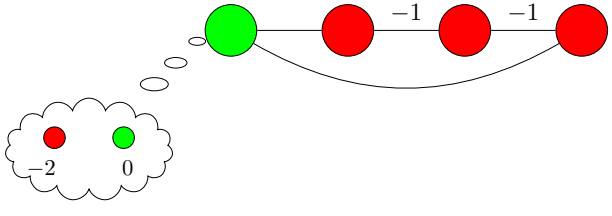
Maximum Gain Message (MGM-1) (MAHESWARAN et al., 2004)

- Coordinate to decide who is going to move
  - Compute and exchange possible gains
  - Agent with maximum (positive) gain executes
- Analysis
  - Empirically, similar to DSA
  - More communication (but still linear)
  - No Threshold to set
  - Guaranteed to be monotonic (Anytime behavior)

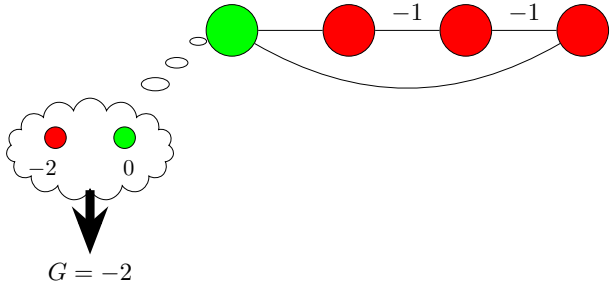
MGM-1: Example



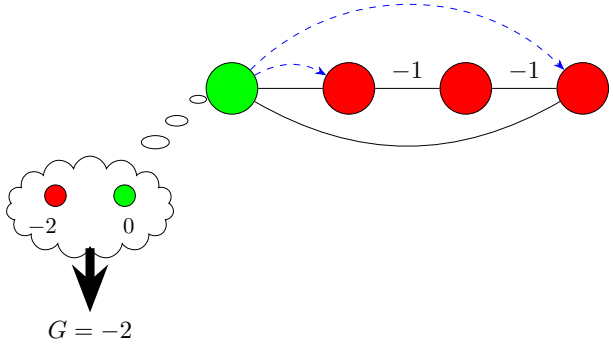
MGM-1: Example



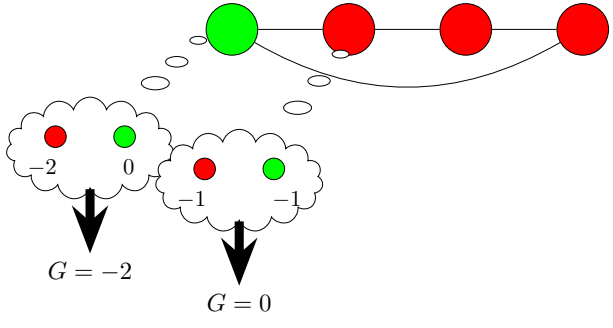
MGM-1: Example



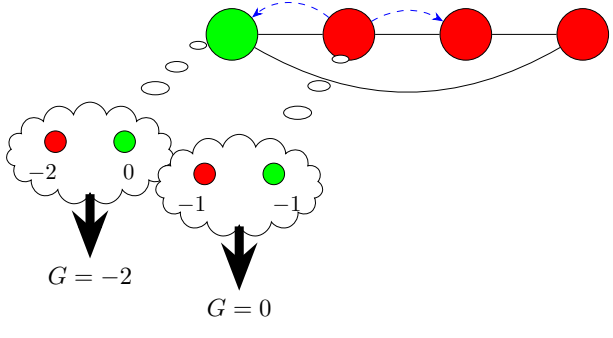
MGM-1: Example



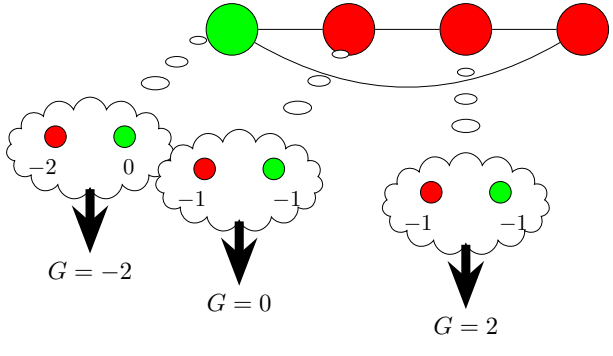
MGM-1: Example



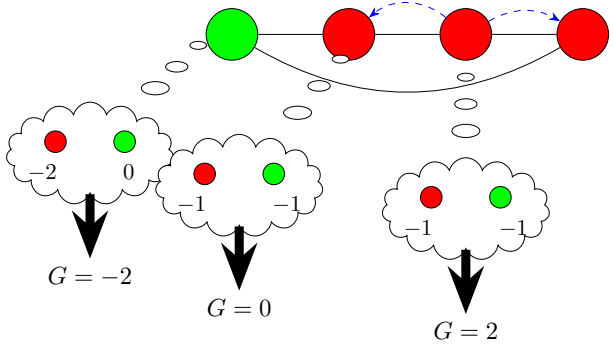
MGM-1: Example



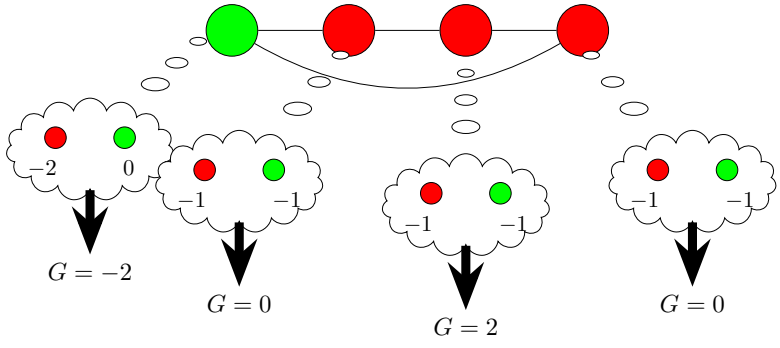
MGM-1: Example



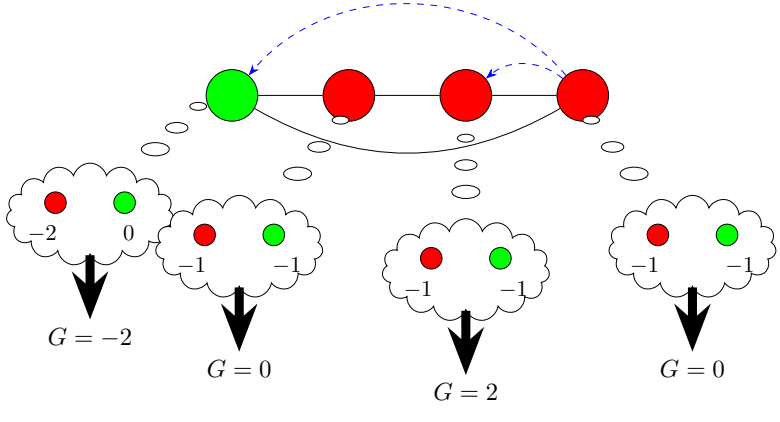
MGM-1: Example



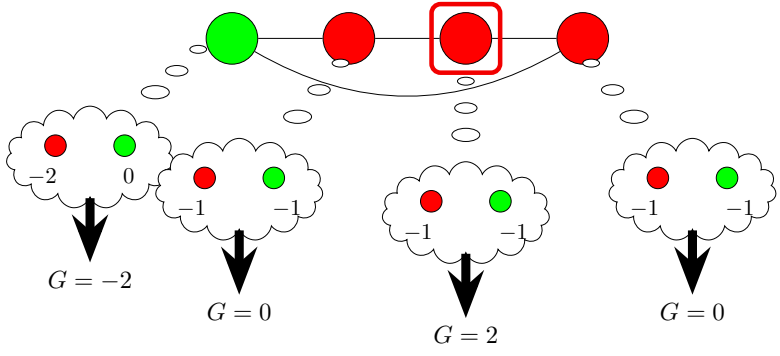
MGM-1: Example



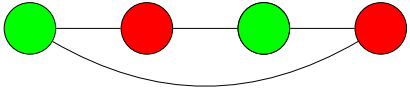
MGM-1: Example



MGM-1: Example



MGM-1: Example



MGM vs DSA

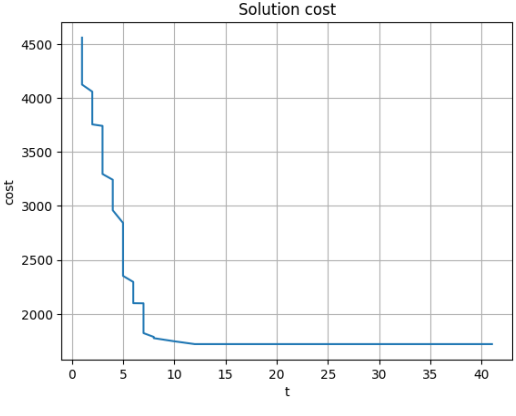


Figure: MGM

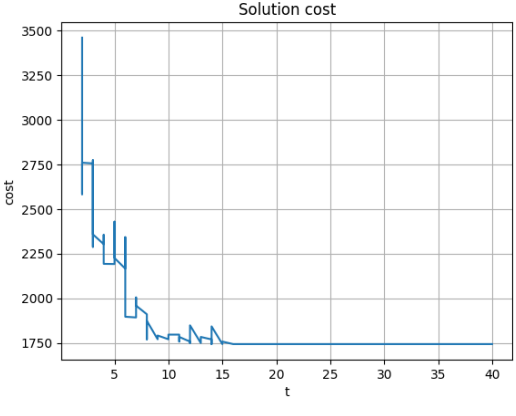


Figure: DSA

To sum up on local greedy approaches

- Exchange local values for variables
  - Similar to search based methods (e.g. ADOPT)
- Consider only local information when maximizing
  - Values of neighbors
- Anytime behaviors
- Could result in very bad solutions

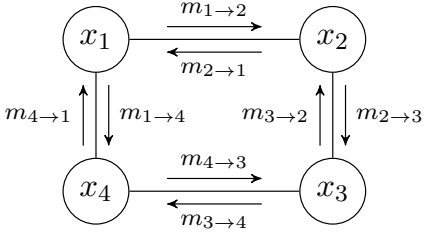
GDL-based approaches

- Generalized Distributive Law (AJI and MCELIECE, 2000)
  - Unifying framework for inference in Graphical models
  - Builds on basic mathematical properties of semi-rings
  - Widely used in Info theory, Statistical physics, Probabilistic models
- Max-sum
  - DCOP settings: maximise social welfare

	K	"$(+,0)$"	"$(\cdot,1)$"	short name
1.	A	$(+,0)$	$(\cdot,1)$	
2.	$A[x]$	$(+,0)$	$(\cdot,1)$	
3.	$A[x,y,\ldots]$	$(+,0)$	$(\cdot,1)$	
4.	$[0,\infty)$	$(+,0)$	$(\cdot,1)$	sum-product
5.	$(0,\infty]$	$(\min,\infty)$	$(\cdot,1)$	min-product
6.	$[0,\infty)$	$(\max,0)$	$(\cdot,1)$	max-product
7.	$(-\infty,\infty]$	$(\min,\infty)$	$(+,0)$	min-sum
8.	$[-\infty,\infty)$	$(\max,-\infty)$	$(+,0)$	max-sum
9.	$\{0,1\}$	$(\mathtt{OR},0)$	$(\mathtt{AND},1)$	Boolean
10.	2^S	$(\cup,\emptyset)$	$(\cap,S)$	
11.	Λ	$(\vee,0)$	$(\wedge,1)$	
12.	Λ	$(\wedge,1)$	$(\vee,0)$.	

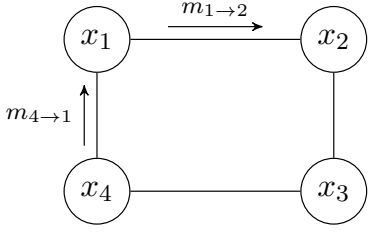
Max-Sum

Agents iteratively computes local functions that depend only on the variable they control



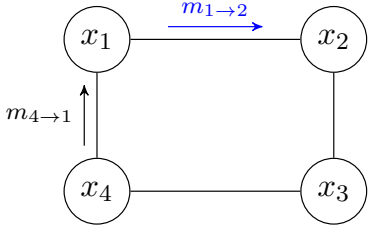
Max-Sum

Agents iteratively computes local functions that depend only on the variable they control



Max-Sum

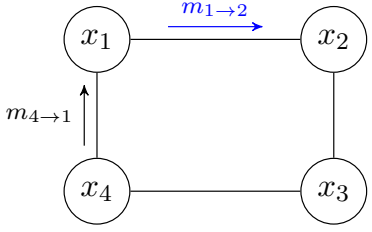
Agents iteratively computes local functions that depend only on the variable they control



$$m_{1\to 2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to 1}(x_1))$$

Max-Sum

Agents iteratively computes local functions that depend only on the variable they control

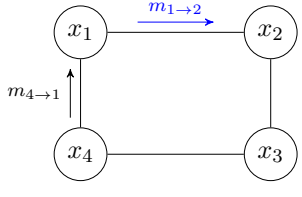


Shared constraint

$$m_{1\to2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to1}(x_1))$$

Max-Sum

Agents iteratively computes local functions that depend only on the variable they control



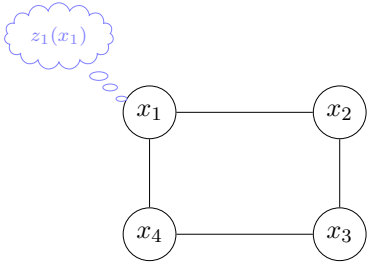
Shared constraint

$$m_{1\to 2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to 1}(x_1))$$

All incoming messages except x_2

Max-Sum

Agents iteratively computes local functions that depend only on the variable they control



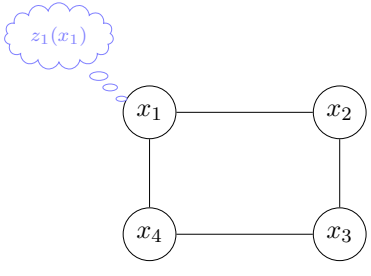
Shared constraint

$$m_{1\to 2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to 1}(x_1))$$

All incoming messages except x_2

Max-Sum

Agents iteratively computes local functions that depend only on the variable they control



Shared constraint

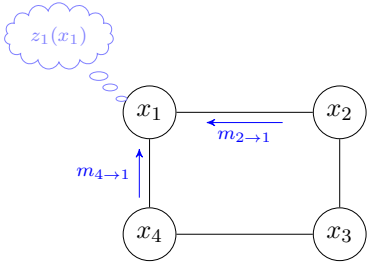
$$m_{1\to 2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to 1}(x_1))$$

All incoming messages except x_2

$$z_1(x_1) = m_{4\to 1}(x_1) + m_{2\to 1}(x_1)$$

Max-Sum

Agents iteratively computes local functions that depend only on the variable they control



Shared constraint

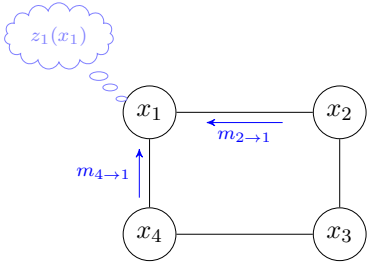
$$m_{1\to2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to1}(x_1))$$

$$z_1(x_1) = m_{4\to1}(x_1) + m_{2\to1}(x_1)$$

All incoming messages except x_2

Max-Sum

Agents iteratively computes local functions that depend only on the variable they control



Shared constraint

$$m_{1\to2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to1}(x_1))$$

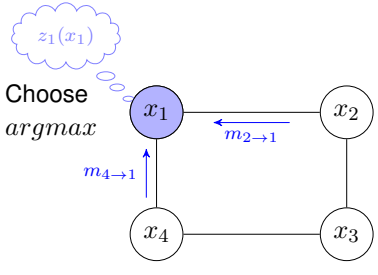
All incoming messages except x_2

$$z_1(x_1) = m_{4\to1}(x_1) + m_{2\to1}(x_1)$$

All incoming messages

Max-Sum

Agents iteratively computes local functions that depend only on the variable they control



Shared constraint

$$m_{1\to2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to1}(x_1))$$

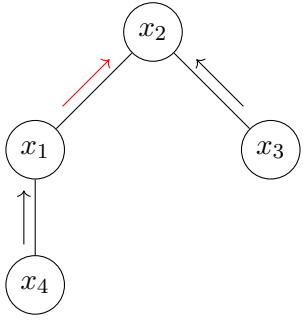
All incoming messages except x_2

$$z_1(x_1) = m_{4\to1}(x_1) + m_{2\to1}(x_1)$$

All incoming messages

Max-Sum on acyclic graphs

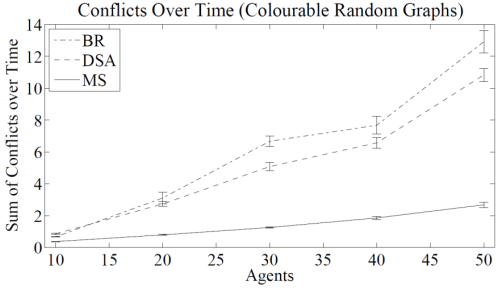
- Max-sum Optimal on acyclic graphs
  - Different branches are independent
  - Each agent can build a correct estimation of its contribution to the global problem (z functions)
- Message equations very similar to Util messages in DPOP
  - Sum messages from children and shared constraint
  - Maximize out agent variable
  - GDL generalizes DPOP (VINYALS et al., 2011)

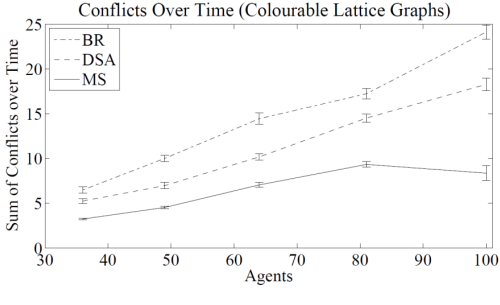


$$m_{1\to 2}(x_2) = max_{x_1}(F_{12}(x_1, x_2) + m_{4\to 1}(x_1))$$

Max-sum Performance

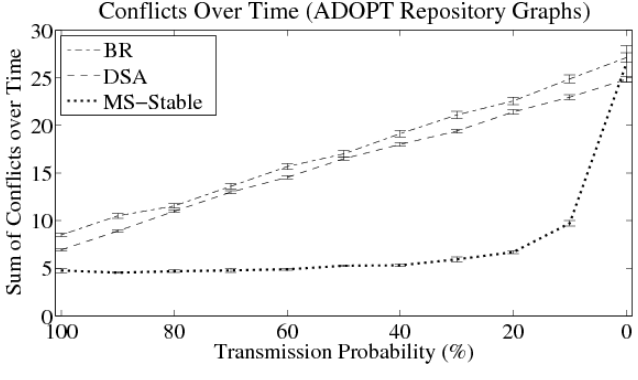
- Good performance on loopy networks (FARINELLI et al., 2008)
  - When it converges very good results
    - Interesting results when only one cycle (WEISS, 2000)
  - We could remove cycle but pay an exponential price (see DPOP)





Max-Sum for low power devices

- Low overhead
  - Msgs number/size
- Asynchronous computation
  - Agents take decisions whenever new messages arrive
- Robust to message loss



Contents

Panorama

Algorithm	Type	Memory	Messages	Remarks
SBB	COP	Polynomial	Exponential	Complete
ADOPT	COP	Polynomial	Exponential	Complete
DPOP	COP	Exponential	Linear	Complete
DSA	COP	Linear	?	Not complete
MGM	COP	Linear	?	Not complete
Max-Sum	COP	Exponential	Linear on acyclic	Complete on trees

Table: DCOP algorithms

Extensions to the DCOP Framework

- Dynamic DCOPs
  - SDPOP (Petcu and Faltings, 2005a), I-ADOPT and I-BnB-ADOPT (Yeoh et al., 2011), FMS (Ramchurn et al., 2010)
- Multi-Objective DCOPs
  - MO-SBB (Medi et al., 2014), Pseudo-tree Based Algorithm (Matsui et al., 2012), B-MOMS (Delle Fave et al., 2011), DP-AOF (Okimoto et al., 2013)
- Asymetric DCOPs
  - SyncABB-2ph, SyncABB-1ph, ACLS, MCS-MGM (Grinshpoun et al., 2013)
- Probabilistic DCOPs
  - $\mathbb{E}$[DPOP] and SD-DPOP (Léauté and Faltings, 2011; Nguyen et al., 2012), U-GDL (Stranders et al., 2011)
- Continuous DCOPs
  - CMS (Stranders et al., 2009), HCMS (Voice et al., 2010), PFD (Choudhury et al., 2020), EC-DPOP, AC-DPOP, CAC-DPOP, C-DSA (Hoang et al., 2020), C-CoCoA (Sarker et al., 2021)
- ...

Contents

Using Distributed Problem Solving

Problem and Environment Characteristics

- Geographic distribution
  - ex: agents are physically distributed, and solving the whole problem is not possible in a centralised manner
- Constraint network topology
  - ex: bounded vertex degrees or large constraint graph diameter
- Knowledge encapsulation
  - ex: *privacy* preserving, limited knowledge
- Dynamics
  - ex: rather than solving the whole problem again, only repair sub-problems

Some Applications

- Frequency assignment
- Scheduling
- Resource allocation, Manufacturing control
- Supply chain

SECP model

Smart Environment Configuration Problem (RUST et al., 2016)

- Example of applying DCOPs to a "real" problem
- Coordinate objects in the building
- Model
  - objects
  - relations between objects and environment
  - user objectives and requirements
- Formulate the problem as an optimization problem

SECP model

Smart Environment Configuration Problem (RUST et al., 2016)

Focus on smart lighting use cases

- **Objects:** anything that can produce light: light bulbs, windows with rolling shutter, etc.
- **User preferences:** having a predefined luminosity level in a room, under some conditions
- **Energy efficiency**

Linking objects and user preferences:

- How to model the luminosity in a room ? **variable**
- How to model the dependency between the light sources and the luminosity ? **function** / **constraint**

SECP model

Example application to ambient intelligence scenario

- **Actuators**
  - Connected light bulbs, TV, Rolling shutters, ...
- **Sensors**
  - Presence detector, Luminosity Sensor, etc.
- **Physical Dependency Models**
  - E.g. Living-room light model
- **User Preferences**
  - Expressed as rules :

```
IF    presence_living_room       =   1
AND   light_sensor_living_room   <   60
THEN  light_level_living_room    ←   60
AND   shutter_living_room        ←   0
```

SECP model

Example application to ambient intelligence scenario

- **Actuators**
  - *Decision* variable x_i, domain $\mathcal{D}_{x_i}$
  - Cost function $c_i : \mathcal{D}_{x_i} \rightarrow \mathbb{R}$
- **Sensors**
  - *Read-only* variable s_l, domain $\mathcal{D}_{s_l}$
- **Physical Dependency Models** $\langle y_j, \phi_j \rangle$
  - Give the expected state of the environment from a set of actuator-variables influencing this model
  - Variable y_j representing the *expected* state of the environment
  - Function $\phi_j : \prod_{\varsigma \in \sigma(\phi_j)} \mathcal{D}_\varsigma \rightarrow \mathcal{D}_{y_j}$
- **User Preferences**
  - Utility function u_k
  - Distance from the current expected state to the target state of the environment

Formulating SECP as a DCOP

Multi-objective optimization problem

$$\min_{x_i \in \nu(\mathfrak{A})} \sum_{i \in \mathfrak{A}} c_i \quad \text{and} \quad \max_{\substack{x_i \in \nu(\mathfrak{A}) \\ y_j \in \nu(\Phi)}} \sum_{k \in \mathfrak{R}} u_k$$

$$\textbf{s.t.} \;\; \phi_j(x_j^1, \ldots, x_j^{\overline{\phi_j}}) = y_j \quad \forall y_j \in \nu(\Phi)$$

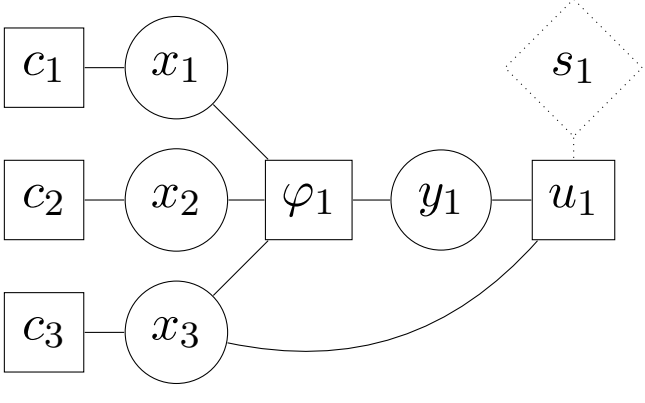
Mono-objective DCOP formulation

$$\max_{\substack{x_i \in \nu(\mathfrak{A}) \\ y_j \in \nu(\Phi)}} \omega_u \sum_{k \in \mathfrak{R}} u_k - \omega_c \sum_{i \in \mathfrak{A}} c_i + \sum_{\varphi_j \in \mathbf{\Phi}} \varphi_j$$

$$\varphi_j(x_j^1, \ldots, x_j^{|\sigma(\phi_j)|}, y_j) = \begin{cases} 0 & \text{if } \phi_j(x_j^1, \ldots, x_j^{|\sigma(\phi_j)|}) = y_j \\ -\infty & \text{otherwise} \end{cases}$$

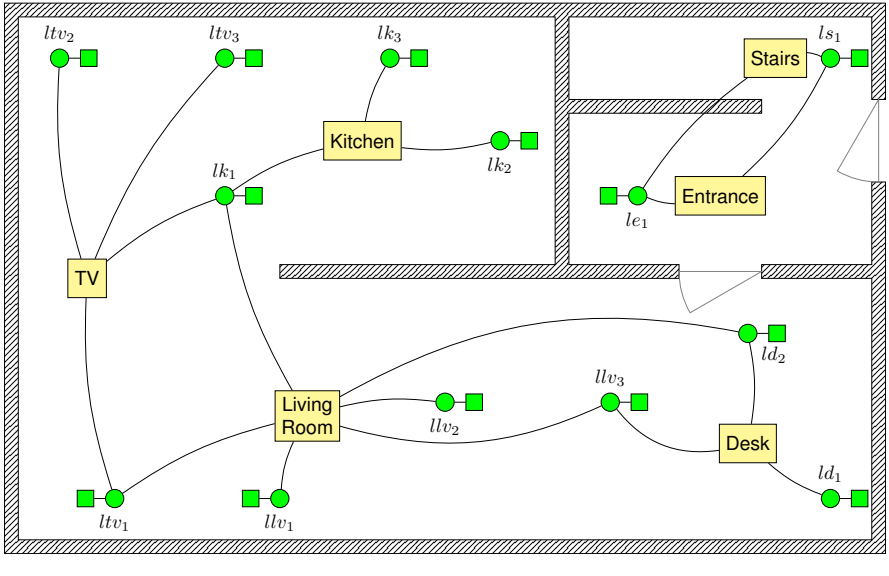
Formulating SECP as a DCOP

Representing a DCOP as a factor graph

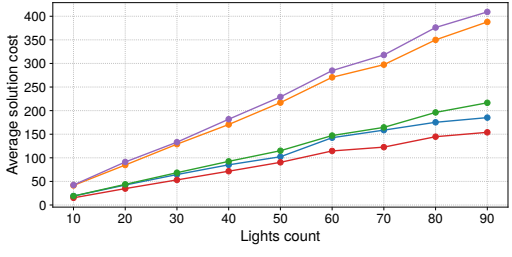


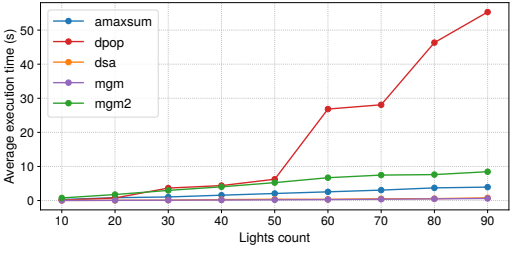
SECP Factor Graph

in a house (without rules)



Algorithms' performances





- Best solutions: DPOP, MaxSum, MGM2
- Worst runtime: DPOP
- Best compromise: MaxSum, MGM2

SECP: further readings

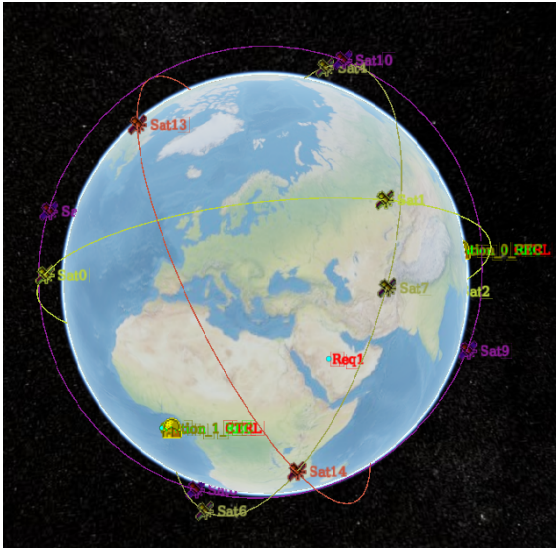
- Experiments with various algorithms (RUST et al., 2016, 2022)
- How to deploy DCOPs (RUST et al., 2017, 2022)
- How to adapt deployment at runtime (RUST et al., 2018, 2020, 2022)

Observation Scheduling in Multi-Owner Constellations (PICARD, 2022)

- Increasing size of deployed EOS constellations

⇒ Observe any point on Earth at higher frequency, e.g. Planet constellation

- **but**, requires to improve coordination and cooperation between assets and stakeholders



Observation Scheduling in Multi-Owner Constellations (PICARD, 2022)

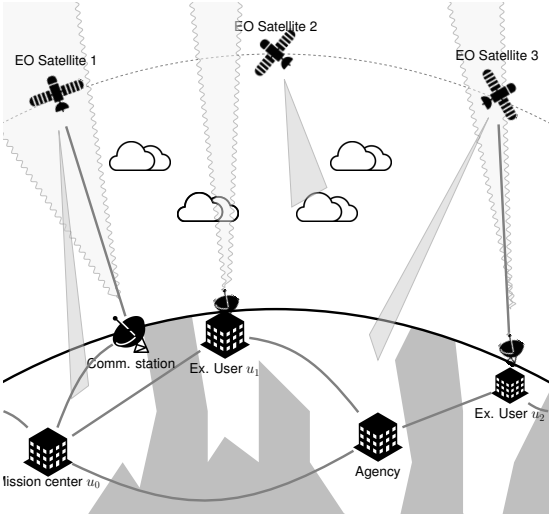
- Increasing size of deployed EOS constellations

⇒ Observe any point on Earth at higher frequency, e.g. Planet constellation

- **but**, requires to improve coordination and cooperation between assets and stakeholders

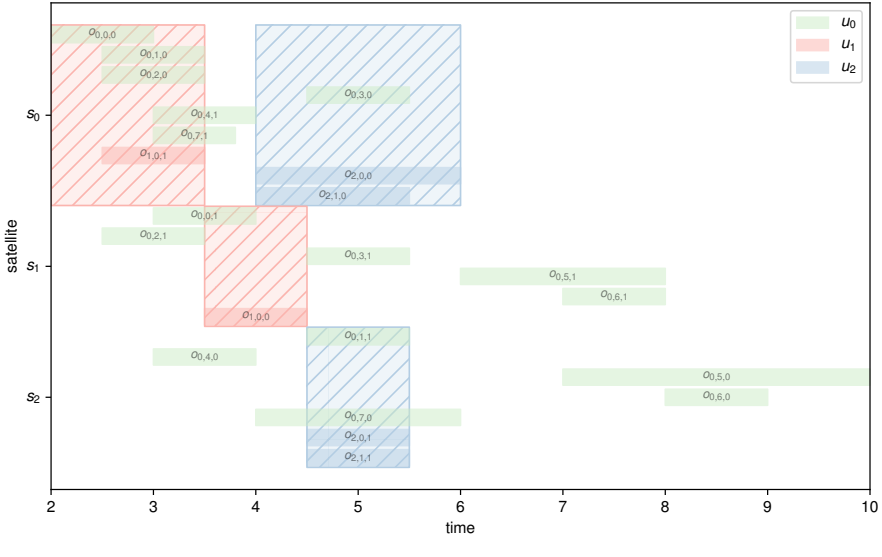
- We focus here on collective observation scheduling on a constellation where some users have exclusive access to some orbit portions

⇒ Answer to strong user expectations to benefit both from a shared system (to reduce costs) and a proprietary system (total control and confidentiality)



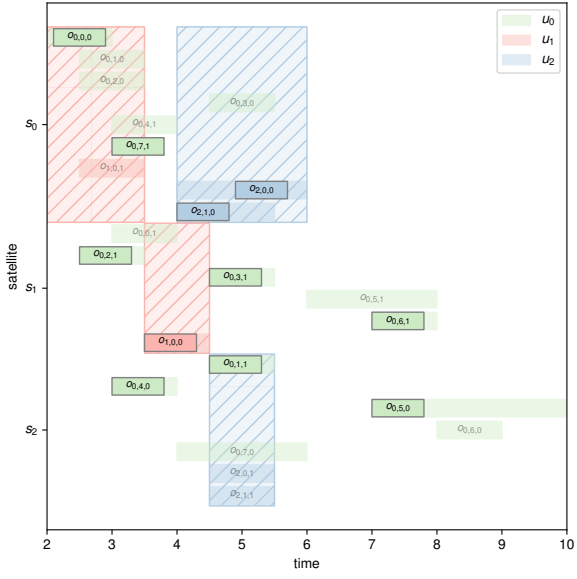
Scheduling Observations with Multiple Exclusive Orbit Portions

Illustrative Example



Scheduling Observations with Multiple Exclusive Orbit Portions

Illustrative Example



DCOP Model

A DCOP $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$ is defined for a given request r, and a current scheduling

DCOP Model

A DCOP $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$ is defined for a given request r, and a current scheduling

- The agents are the exclusive users which can potentially schedule r:

$$\mathcal{A} = \{u \in \mathcal{U}^{\text{ex}} | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \exists o \in \theta_r \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \varnothing\} \tag{1}$$

DCOP Model

A DCOP $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$ is defined for a given request r, and a current scheduling

- The agents are the exclusive users which can potentially schedule r:

$$\mathcal{A} = \{u \in \mathcal{U}^{\text{ex}} | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \exists o \in \theta_r \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \varnothing\} \tag{1}$$

- Each agent u owns binary decision variables, one for each observation $o \in \mathcal{O}[u]^r$ and exclusive e in its exclusives e_u, stating whether it schedules o in e or not:

$$\mathcal{X} = \{x_{e,o} | e \in \bigcup_{u \in \mathcal{A}} e_u, o \in \mathcal{O}[u]^r\} \tag{2}$$

$$\mathcal{D} = \{\mathcal{D}_{x_{e,o}} = \{0, 1\} | x_{e,o} \in \mathcal{X}\} \tag{3}$$

DCOP Model

A DCOP $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$ is defined for a given request r, and a current scheduling

- The agents are the exclusive users which can potentially schedule r:

$$\mathcal{A} = \{u \in \mathcal{U}^{\text{ex}} | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \exists o \in \theta_r \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \varnothing\} \tag{1}$$

- Each agent u owns binary decision variables, one for each observation $o \in \mathcal{O}[u]^r$ and exclusive e in its exclusives e_u, stating whether it schedules o in e or not:

$$\mathcal{X} = \{x_{e,o} | e \in \bigcup_{u \in \mathcal{A}} e_u, o \in \mathcal{O}[u]^r\} \tag{2}$$

$$\mathcal{D} = \{\mathcal{D}_{x_{e,o}} = \{0, 1\} | x_{e,o} \in \mathcal{X}\} \tag{3}$$

with $\mathcal{O}[u]^r = \{o \in \theta r | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \varnothing\}$ are observations related to request r that can be scheduled on u's exclusives

DCOP Model

A DCOP $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$ is defined for a given request r, and a current scheduling

- The agents are the exclusive users which can potentially schedule r:

$$\mathcal{A} = \{u \in \mathcal{U}^{\text{ex}} | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \exists o \in \theta_r \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \varnothing\} \tag{1}$$

- Each agent u owns binary decision variables, one for each observation $o \in \mathcal{O}[u]^r$ and exclusive e in its exclusives e_u, stating whether it schedules o in e or not:

$$\mathcal{X} = \{x_{e,o} | e \in \bigcup_{u \in \mathcal{A}} e_u, o \in \mathcal{O}[u]^r\} \tag{2}$$

$$\mathcal{D} = \{\mathcal{D}_{x_{e,o}} = \{0, 1\} | x_{e,o} \in \mathcal{X}\} \tag{3}$$

  with $\mathcal{O}[u]^r = \{o \in \theta r | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \varnothing\}$ are observations related to request r that can be scheduled on u's exclusives

- μ associates each variable $x_{e,o}$ to e's owner

DCOP Model (cont.)

- Constraints should check that at most one observation is scheduled per request (4), that satellites are not overloaded (5), that at most one agent serves the same observation (6)

$$\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq 1, \quad \forall u \in \mathcal{X}, \forall o \in \mathcal{O}[u]^r \tag{4}$$

$$\sum_{o \in \{o \in \mathcal{O}[u]^r | u \in \mathcal{A}, s_o = s\}, e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq \kappa_s^*, \ \forall s \in \mathcal{S} \tag{5}$$

$$\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq 1, \quad \forall o \in \mathcal{O} \tag{6}$$

DCOP Model (cont.)

- Constraints should check that at most one observation is scheduled per request (4), that satellites are not overloaded (5), that at most one agent serves the same observation (6)

$$\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq 1, \quad \forall u \in \mathcal{X}, \forall o \in \mathcal{O}[u]^r \tag{4}$$

$$\sum_{o \in \{o \in \mathcal{O}[u]^r | u \in \mathcal{A}, s_o = s\}, e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq \kappa_s^*, \ \forall s \in \mathcal{S} \tag{5}$$

$$\sum_{e \in \bigcup_{u \in \mathcal{A}} e_u} x_{e,o} \leq 1, \quad \forall o \in \mathcal{O} \tag{6}$$

- The cost to integrate an observation in the current user's schedule should be assessed to guide the optimization process

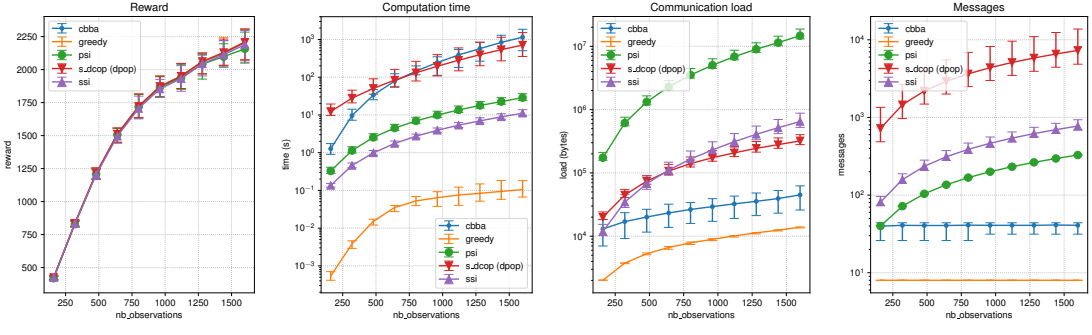
$$c(x_{e,o}) = \pi(o, \mathcal{M}_{u_o}), \quad \forall x_{e,o} \in \mathcal{X} \tag{7}$$

where π evaluates the best cost obtained when scheduling o and any combination of observations from $\mathcal{M}_{u_o}$, as to consider all possible revisions of u_o's current schedule

$$\mathcal{C} = \{(4), (5), (6), (7)\} \tag{8}$$

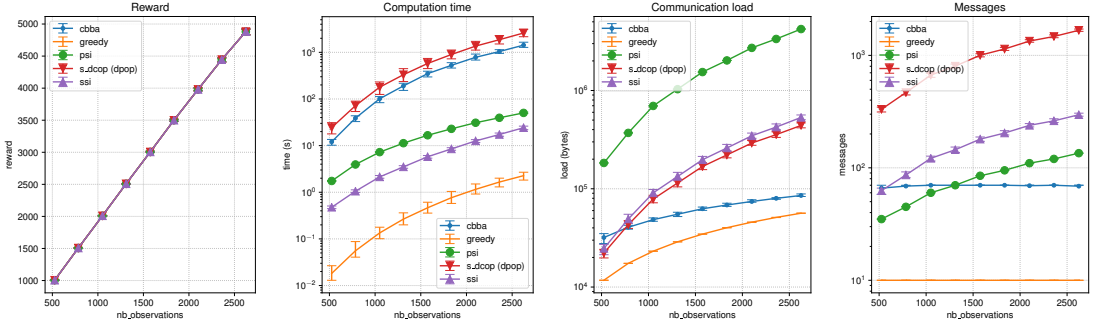
Highly conflicting randomly generated problems

5-min horizon with overlapping requests and limited capacity



Realistic randomly generated problems

6-hour horizon with numerous requests and large capacity



References

AJI, S.M. and R.J. MCELIECE (2000). “The generalized distributive law”. In: *Information Theory, IEEE Transactions on* 46.2, pp. 325–343. ISSN: 0018-9448. DOI: 10.1109/18.825794.

CHOUDHURY, Moumita, Saaduddin MAHMUD, and Md. Mosaddek KHAN (2020). “A Particle Swarm Based Algorithm for Functional Distributed Constraint Optimization Problems”. In: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, pp. 7111–7118. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6198.

DELLE FAVE, F.M., R. STRANDERS, A. ROGERS, and N.R. JENNINGS (2011). “Bounded Decentralised Coordination over Multiple Objectives”. In: *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '11)*, pp. 371–378.

FARINELLI, A., A. ROGERS, A. PETCU, and N. R. JENNINGS (2008). “Decentralised Coordination of Low-power Embedded Devices Using the Max-sum Algorithm”. In: *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '08. International Foundation for Autonomous Agents and Multiagent Systems, pp. 639–646. ISBN: 978-0-9817381-1-6.

FIORETTO, F., E. PONTELLI, and W. YEOH (2018). “Distributed Constraint Optimization Problems and Applications: A Survey”. In: *Journal of Artificial Intelligence Research* 61, pp. 623–698.

FITZPATRICK, Stephen and Lambert MEERTENS (2003). “Distributed Coordination through Anarchic Optimization”. In: *Distributed Sensor Networks: A Multiagent Perspective*. Ed. by Victor LESSER, Charles L. ORTIZ, and Milind TAMBE. Boston, MA: Springer US, pp. 257–295. ISBN: 978-1-4615-0363-7.

References (cont.)


Grinshpoun, T., A. Grubshtein, R. Zivan, A. Netzer, and A. Meisels (2013). “Asymmetric Distributed Constraint Optimization Problems”. In: *J. Artif. Int. Res.* 47.1, pp. 613–647. ISSN: 1076-9757. URL: http://dl.acm.org/citation.cfm?id=2566972.2566988.


Hirayama, K. and M. Yokoo (1997). “Distributed partial constraint satisfaction problem”. In: *Principles and Practice of Constraint Programming-CP97*. Springer, pp. 222–236. ISBN: 978-3-540-69642-1.


Hoang, Khoi D., William Yeoh, Makoto Yokoo, and Zinovi Rabinovich (2020). “New Algorithms for Continuous Distributed Constraint Optimization Problems”. In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ’20. Auckland, New Zealand: International Foundation for Autonomous Agents and Multiagent Systems, p. 502510. ISBN: 9781450375184.


Léauté, T. and B. Faltings (2011). “Distributed Constraint Optimization Under Stochastic Uncertainty”. In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI’11)*. AAAI Press, pp. 68–73. URL: http://dl.acm.org/citation.cfm?id=2900423.2900434.


Maheswaran, R.T., J.P. Pearce, and M. Tambe (2004). “Distributed Algorithms for DCOP: A Graphical-Game-Based Approach”. In: *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems (PDCS), San Francisco, CA*, pp. 432–439.


Matsui, T., M. Silaghi, K. Hirayama, M. Yokoo, and H. Matsuo (2012). “Distributed Search Method with Bounded Cost Vectors on Multiple Objective DCOPs”. In: *PRIMA 2012: Principles and Practice of Multi-Agent Systems*. Springer, pp. 137–152. ISBN: 978-3-642-32729-2.


Medi, A., T. Okimoto, and K. Inoue (July 2014). “A two-phase complete algorithm for multi-objective distributed constraint optimization”. In: 18, pp. 573–580.

References (cont.)

MODI, P. J., W. SHEN, M. TAMBE, and M. YOKOO (2005). "ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees". In: *Artificial Intelligence* 161.2, pp. 149–180.

NGUYEN, D.T., W. YEOH, and H.C. LAU (2012). "Stochastic Dominance in Stochastic DCOPs for Risk-sensitive Applications". In: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*. Valencia, Spain, pp. 257–264. ISBN: 0-9817381-1-7, 978-0-9817381-1-6. URL: http://dl.acm.org/citation.cfm?id=2343576.2343613.

OKIMOTO, T., M. CLEMENT, and K. INOUE (2013). "AOF-Based Algorithm for Dynamic Multi-Objective Distributed Constraint Optimization". In: *Proceedings of the 7th International Workshop on Multi-disciplinary Trends in Artificial Intelligence (MIWAI'13)*. Springer, pp. 175–186. ISBN: 978-3-642-44948-2. DOI: 10.1007/978-3-642-44949-9_17. URL: http://dx.doi.org/10.1007/978-3-642-44949-9_17.

PETCU, A. and B. FALTINGS (2005a). "Superstabilizing, Fault-containing Distributed Combinatorial Optimization". In: *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI'05)*. AAAI Press, pp. 449–454. ISBN: 1-57735-236-x. URL: http://dl.acm.org/citation.cfm?id=1619332.1619405.

PETCU, Adrian and Boi FALTINGS (2005b). "A scalable method for multiagent constraint optimization". In: *IJCAI International Joint Conference on Artificial Intelligence*, pp. 266–271. ISBN: 1045-0823.

PICARD, Gauthier (2022). "Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions". In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS-22)*. IFAAMAS.

References (cont.)

Ramchurn, S. D., A. Farinelli, K. S. Macarthur, and N. R. Jennings (2010). "Decentralized Coordination in RoboCup Rescue". In: *Comput. J.* 53.9, pp. 1447–1461. ISSN: 0010-4620. DOI: 10.1093/comjnl/bxq022. URL: http://dx.doi.org/10.1093/comjnl/bxq022.

Rust, Pierre, Gauthier Picard, and Fano Ramparany (2016). "Using Message-passing DCOP Algorithms to Solve Energy-efficient Smart Environment Configuration Problems". In: *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*. Ed. by S. Kambhampati. AAAI Press, pp. 468–474. URL: http://www.ijcai.org/Proceedings/2016/.

— (2017). "On the Deployment of Factor Graph Elements to Operate Max-Sum in Dynamic Ambient Environments". In: *Autonomous Agents and Multiagent Systems – AAMAS 2017 Workshops, Best Papers, Sao Paulo, Brazil, May 8-12, 2017, Revised Selected Papers*. Ed. by G. Sukthankar and J.A. Rodriguez-Aguilar. Vol. 10642. Lecture Notes in Artificial Intelligence (LNAI). Extended Version. Springer, pp. 116–137. DOI: 10.1007/978-3-319-71682-4_8.

— (2018). "Self-Organized and Resilient Distribution of Decisions over Dynamic Multi-Agent Systems". In: *International Workshop on Optimisation in Multi-Agent Systems (OptMAS@AAMAS 2018)*. URL: http://www-personal.umich.edu/~fioretto/cfp/OPTMAS18/papers/paper_13.pdf.

— (2020). "Resilient Distributed Constraint Optimization in Physical Multi-Agent Systems". In: *European Conference on Artificial Intelligence (ECAI)*. Vol. 325. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 195–202. DOI: 10.3233/FAIA200093. URL: http://ecai2020.eu/papers/108_paper.pdf.

— (2022). "Resilient Distributed Constraint Reasoning to Autonomously Configure and Adapt IoT Environments". In: *ACM Transactions on Internet of Things*. in press. DOI: http://dx.doi.org/10.1145/3507907.

References (cont.)

SARKER, Amit, Moumita CHOUDHURY, and Md. Mosaddek KHAN (2021). "A Local Search Based Approach to Solve Continuous DCOPs". In: *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '21. Virtual Event, United Kingdom: International Foundation for Autonomous Agents and Multiagent Systems, p. 11271135. ISBN: 9781450383073.

STRANDERS, R., F.M. DELLE FAVE, A. ROGERS, and N.R. JENNINGS (2011). "U-GDL: A decentralised algorithm for DCOPs with Uncertainty". Project Report. URL: https://eprints.soton.ac.uk/273037/.

STRANDERS, R., A. FARINELLI, A. ROGERS, and N. R. JENNINGS (2009). "Decentralised Coordination of Continuously Valued Control Parameters Using the Max-Sum Algorithm". In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS '09. Budapest, Hungary: International Foundation for Autonomous Agents and Multiagent Systems, p. 601608. ISBN: 9780981738161.

VINYALS, Meritxell, Juan A. RODRÍGUEZ-AGUILAR, and Jesus CERQUIDES (2011). "Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law". In: *Autonomous Agents and Multi-Agent Systems* 3.22, pp. 439–464. ISSN: 1387-2532. DOI: 10.1007/s10458-010-9132-7.

VOICE, Thomas, Ruben STRANDERS, Alex ROGERS, and Nicholas R. JENNINGS (2010). "A Hybrid Continuous Max-Sum Algorithm for Decentralised Coordination". In: *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. NLD: IOS Press, p. 6166. ISBN: 9781607506058.

WEISS, Yair (Jan. 2000). "Correctness of Local Probability Propagation in Graphical Models with Loops". In: *Neural Comput.* 12.1, pp. 1–41. ISSN: 0899-7667. DOI: 10.1162/089976600300015880. URL: http://dx.doi.org/10.1162/089976600300015880.

References (cont.)

YEOH, W., P. VARAKANTHAM, X. SUN, and S. KOENIG (2011). "Incremental DCOP Search Algorithms for Solving Dynamic DCOPs". In: *The 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '11)*, pp. 1069–1070. ISBN: 0-9826571-7-X, 978-0-9826571-7-1.

YOKOO, M. (2001). *Distributed Constraint Satisfaction: Foundations of Cooperation in Multi-Agent Systems*. Springer.

ZHANG, W., G. WANG, Z. XING, and L. WITTENBURG (2005). "Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks.". In: *Journal of Artificial Intelligence Research (JAIR)* 161.1-2, pp. 55–87.

ZHANG, Weixiong, Guandong WANG, Zhao XING, and Lars WITTENBURG (2003). "A Comparative Study of Distributed Constraint Algorithms". In: *Distributed Sensor Networks: A Multiagent Perspective*. Ed. by Victor LESSER, Charles L. ORTIZ, and Milind TAMBE. Boston, MA: Springer US, pp. 319–338.