Multi-Agent Distributed Constrained Optimization

Tutorial at AAMAS'22

Gauthier Picard   Filippo Bistaffa

Some contents adapted from previous tutorials (http://https//www2.isye.gatech.edu/~fferdinando3/cfp/AAMAS19/)

AAMAS 2022
MAY 9 - 13 AUCKLAND, NZ

ONERA
THE FRENCH AEROSPACE LAB

IIIA
Institut d'Investigació en
Intel·ligència Artificial

Introduction and Motivations

Who are we?

Gauthier Picard, PhD, Hab.
ONERA, the French Aerospace Lab
Expertises: DCOPs, self-organization, resource allocation

Filippo Bistaffa, PhD
IIIA-CSIC, Barcelona
Expertises: coalition formation, parallel computing, shared mobility

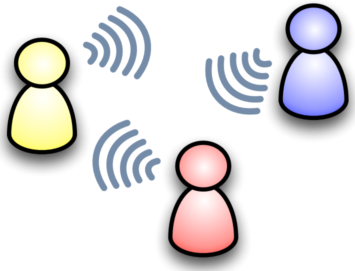
Introduction and Motivations

Multiagent Systems

- Agent: An entity that behaves autonomously in the pursuit of goals
- Multi-agent system: A system of multiple interacting agents

An agent is...

- Autonomous: Is of full control of itself
- Interactive: May communicate with other agents
- Reactive: Responds to changes in the environment or requests by other agents
- Proactive:Takes initiatives to achieve its goals



Introduction and Motivations

Research questions addressed during this tutorial

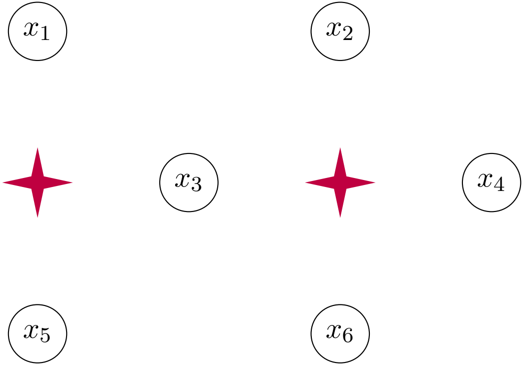
- How to make collective optimal decisions?
  - How to model the collective decision?
  - Which protocols to implement these decisions?
- How to form groups *wrt* to some utility criteria?
  - How to model the utility of each group?
  - How to express which groups are feasible or not?

Today's Menu

Today's Menu

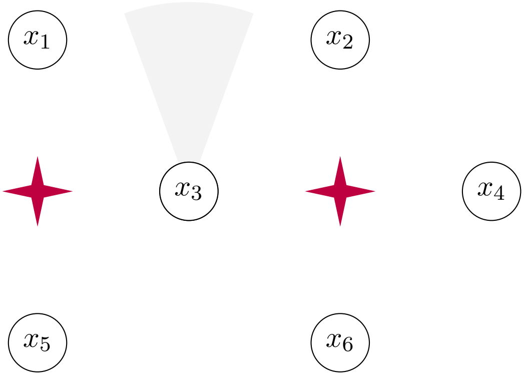
Motivating example

Sensor networks



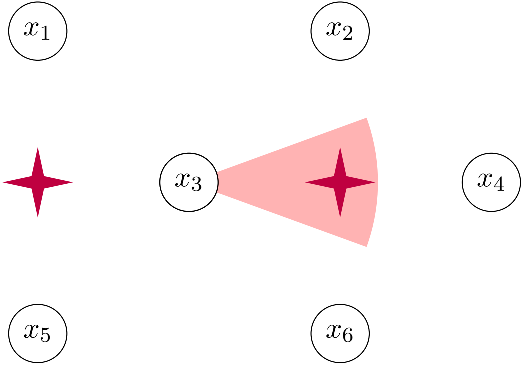
Motivating example

Sensor networks



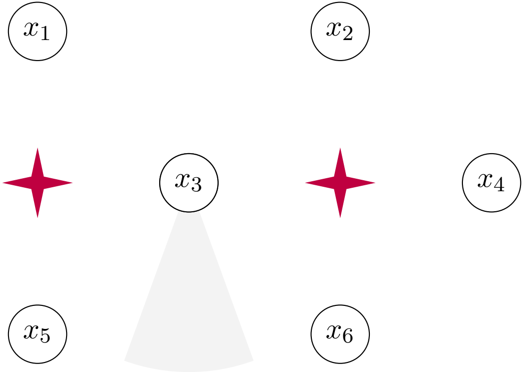
Motivating example

Sensor networks



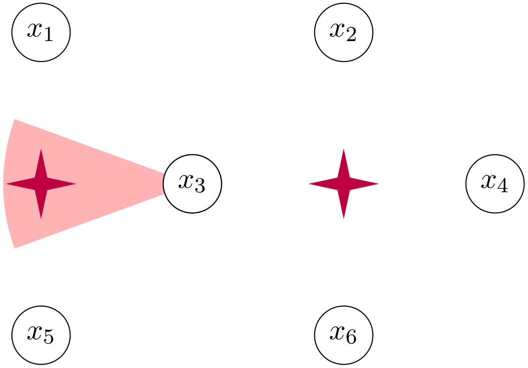
Motivating example

Sensor networks



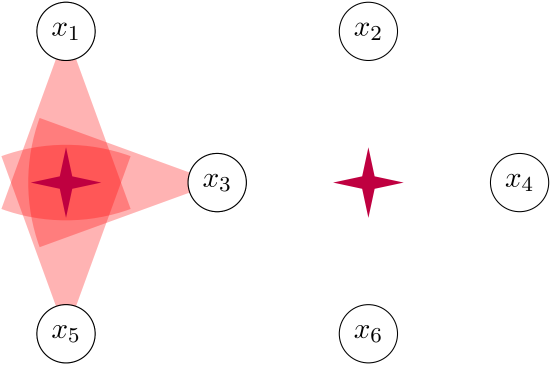
Motivating example

Sensor networks



Motivating example

Sensor networks



x_1	x_3	x_5	Sat?
N	N	N	✗
N	N	E	✗
	...		✗
S	W	N	✓
	...		✗
W	W	W	✗

Model the problem as a CSP!

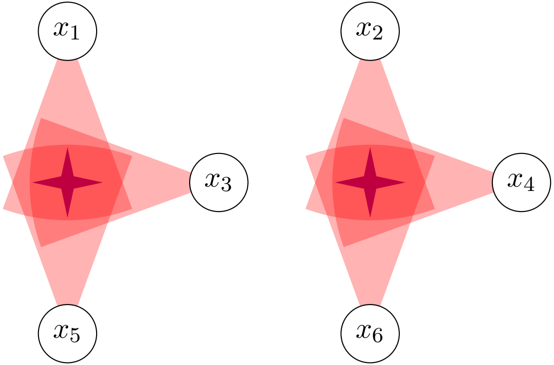
CSP

Constraint Satisfaction

- Variables $X = \{x_1, \dots, x_n\}$
- Domains $D = \{D_1, \dots, D_n\}$
- Constraints $C\{c_1, \dots, c_m\}$
  where a constraint $c_i \subseteq D_{i_1} \times D_{i_2} \times \dots \times D_{i_n}$ denotes the possible valid joint assignments for the variables $x_{i_1}, x_{i_1}, \dots, x_{i_n}$ it involves
- **Goal**: Find an assignment to all variables that satisfies all the constraints

CSP

Constraint Satisfaction

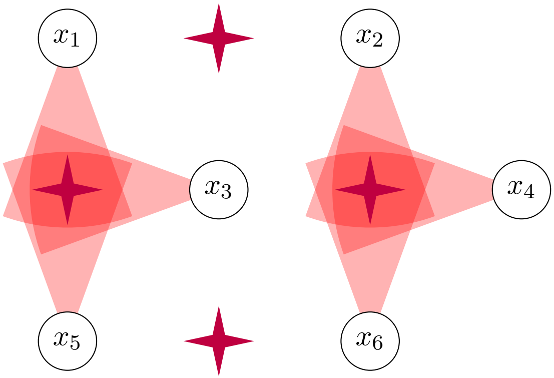


x_1	x_3	x_5	Sat?
N	N	N	✗
N	N	E	✗
	...		✗
S	W	N	✓
	...		✗
W	W	W	✗

Model the problem as a CSP!

Max-CSP

Max Constraint Satisfaction



x_1	x_3	x_5	Sat?
N	N	N	✗
N	N	E	✗
	...		✗
S	W	N	✓
	...		✗
W	W	W	✗

Model the problem
as a Max-CSP!

Max-CSP

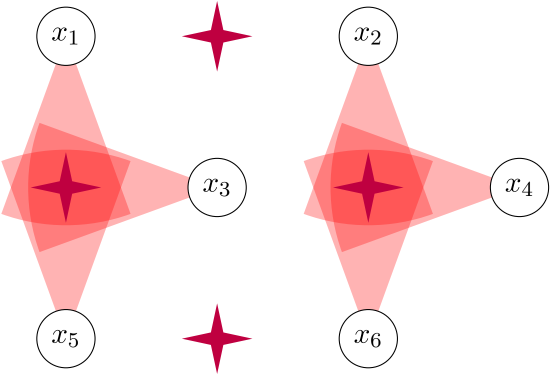
Max Constraint Satisfaction

- Variables $X = \{x_1, \ldots, x_n\}$
- Domains $D = \{D_1, \ldots, D_n\}$
- Constraints $C\{c_1, \ldots, c_m\}$
  where a constraint $c_i \subseteq D_{i_1} \times D_{i_2} \times \ldots \times D_{i_n}$ denotes the possible valid joint assignments for the variables $x_{i_1}, x_{i_1}, \ldots, x_{i_n}$ it involves

- **Goal**: Find an assignment to all variables that satisfies a maximum number of constraints

Max-CSP

Max Constraint Satisfaction

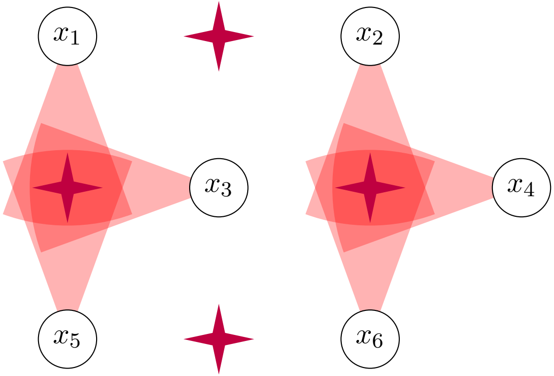


x_1	x_3	x_5	Sat?
N	N	N	✗
N	N	E	✗
	...		✗
S	W	N	✓
	...		✗
W	W	W	✗

Model the problem as a Max-CSP!

WCSP (or COP)

Constraint Optimization



x_1	x_3	x_5	Cost
N	N	N	∞
N	N	E	∞
	...		∞
S	W	N	10
	...		∞
W	W	W	∞

Model the problem
as a COP!

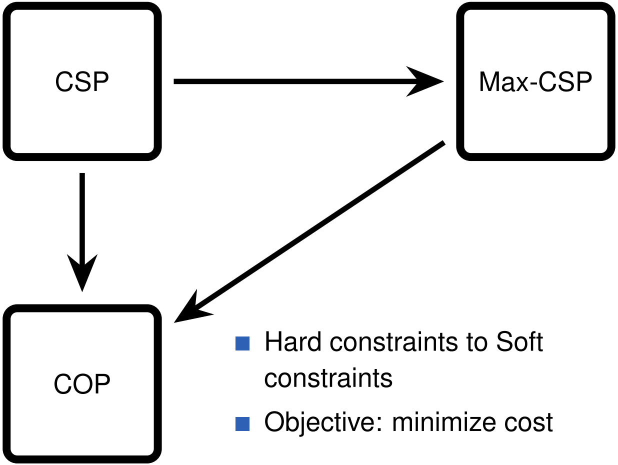
WCSP (or COP)

Constraint Optimization

- Variables $X = \{x_1, \ldots, x_n\}$
- Domains $D = \{D_1, \ldots, D_n\}$
- Constraints $C\{c_1, \ldots, c_m\}$
  where a constraint $c_i : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_n} \rightarrow \mathbb{R}_+ \cup \{\infty\}$ expresses the degree of constraint violation

- **Goal**: Find an assignment to all variables that minimizes the sum of all the constraints

Constraint Reasoning

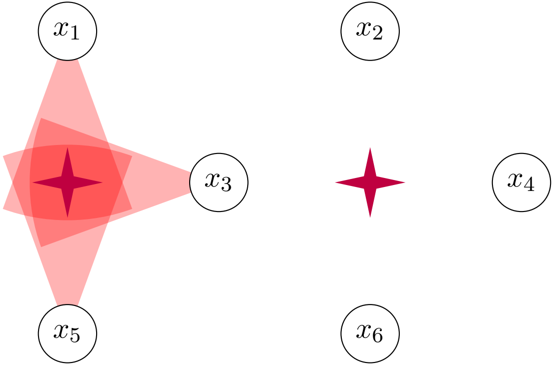


- Objective: maximize #constraints satisfied

- Hard constraints to Soft constraints
- Objective: minimize cost

WCSP (or COP)

Constraint Optimization

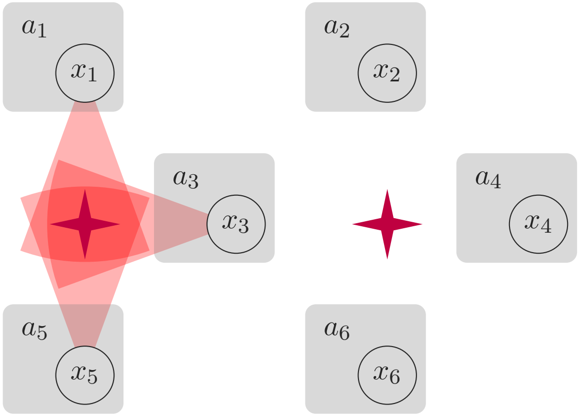


Imagine that each sensor is an autonomous agent

How should this problem be modeled and solved in a decentralized manner?

DCOP

Distributed Constraint Optimization [Modi et al., 2005]

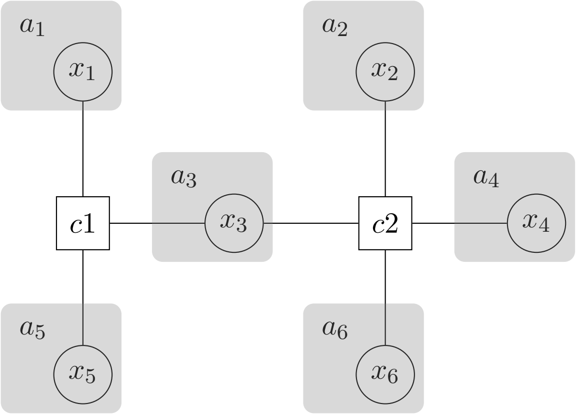


Imagine that each sensor is an autonomous agent

How should this problem be modeled and solved in a decentralized manner?

DCOP

Distributed Constraint Optimization [Modi et al., 2005]



DCOP

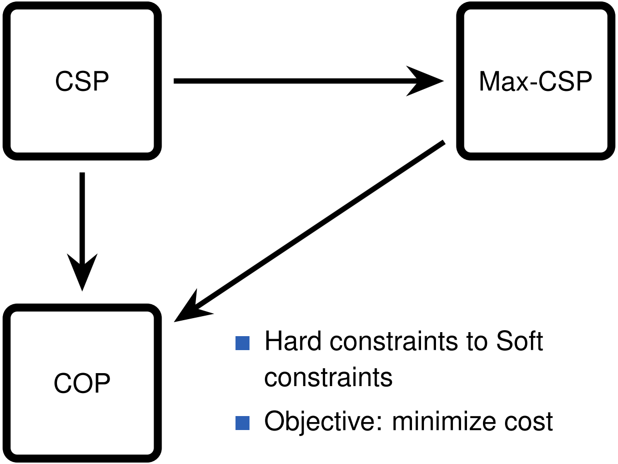
Distributed Constraint Optimization [Modi et al., 2005]

- Agents $X = \{a_1, \ldots, a_l\}$
- Variables $X = \{x_1, \ldots, x_n\}$
- Domains $D = \{D_1, \ldots, D_n\}$
- Constraints $C\{c_1, \ldots, c_m\}$
- Mapping of variables to agents

- **Goal**: Find an assignment to all variables that minimizes the sum of all the constraints

DCOP

Distributed Constraint Optimization [Modi et al., 2005]

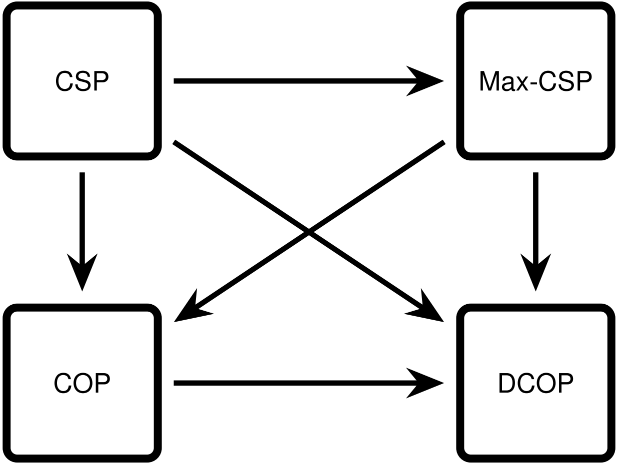


- Objective: maximize #constraints satisfied

- Hard constraints to Soft constraints
- Objective: minimize cost

DCOP

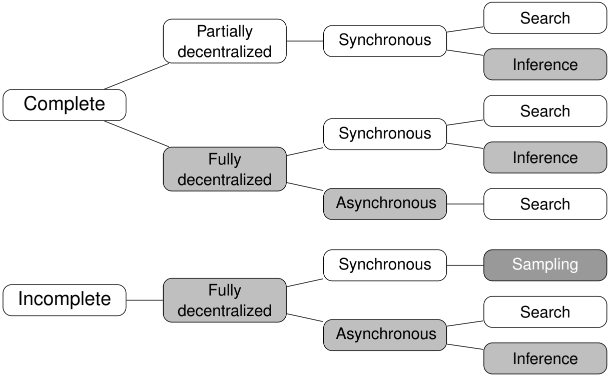
Distributed Constraint Optimization [Modi et al., 2005]



- Variables are controlled by agents
- Communication model
- Local knowledge

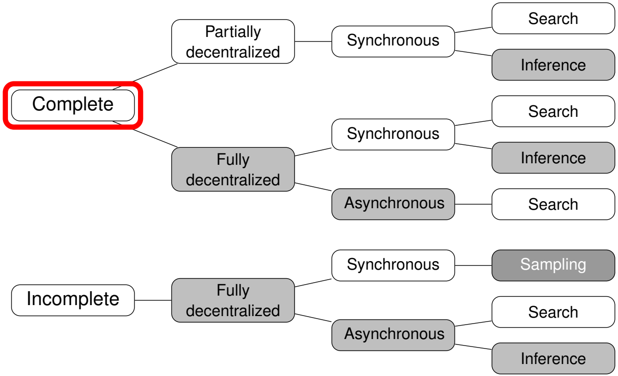
DCOP Algorithms

See [FIORETTO et al., 2018]



DCOP Algorithms

See [FIORETTO et al., 2018]

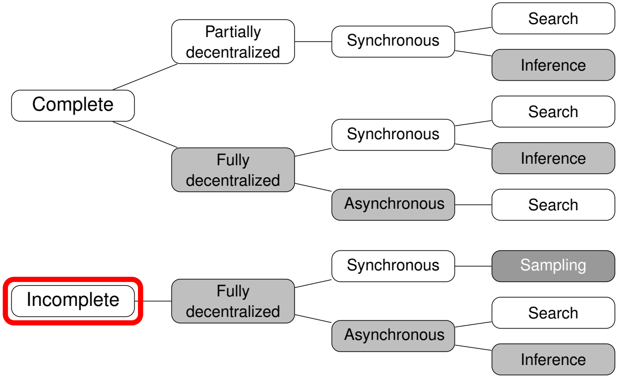


Important metrics

- Agent complexity
- Network loads
- Message size

DCOP Algorithms

See [FIORETTO et al., 2018]



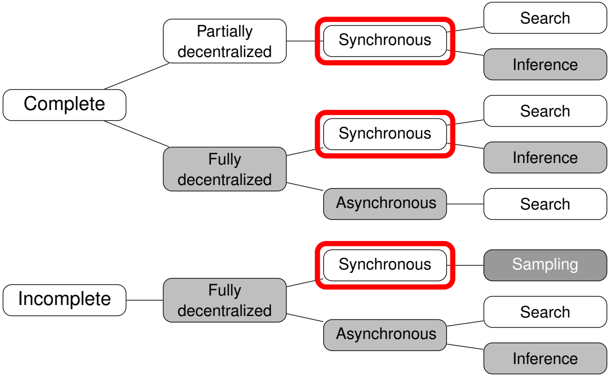
Important metrics

- Agent complexity
- Network loads
- Message size

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

DCOP Algorithms

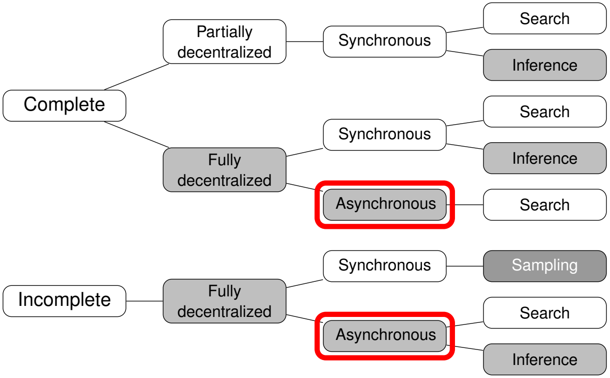
See [Fioretto et al., 2018]



- 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

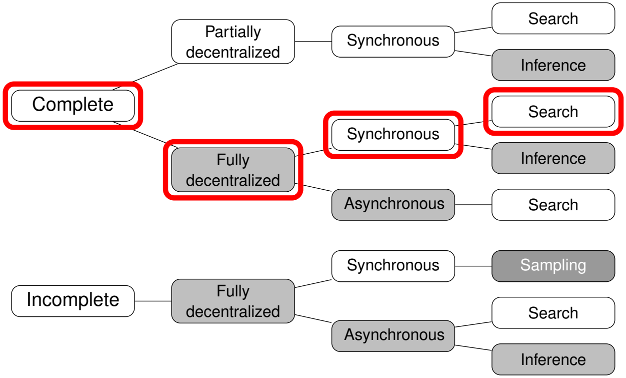
See [FIORETTO et al., 2018]



- 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]

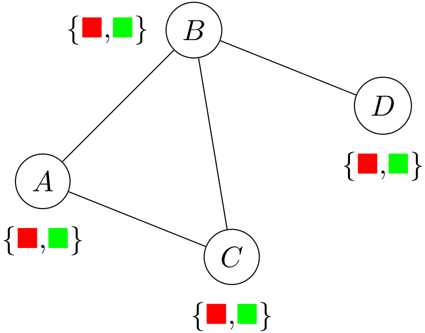


Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]

Synchronous Branch-and-Bound (SBB)

[Hirayama and Yokoo, 1997]

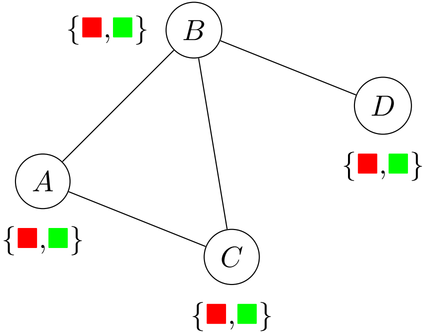


x_i	x_j	(A, B)	(A, C)	(B, C)	(B, C)
🟥	🟥	5	5	5	3
🟥	🟩	8	10	4	8
🟩	🟥	20	20	3	10
🟩	🟩	3	3	3	3

How do we solve this distributedly?

Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



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

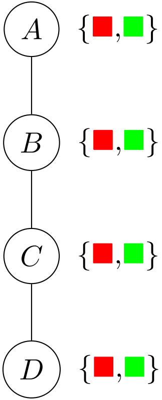
How do we solve this distributedly?

Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]

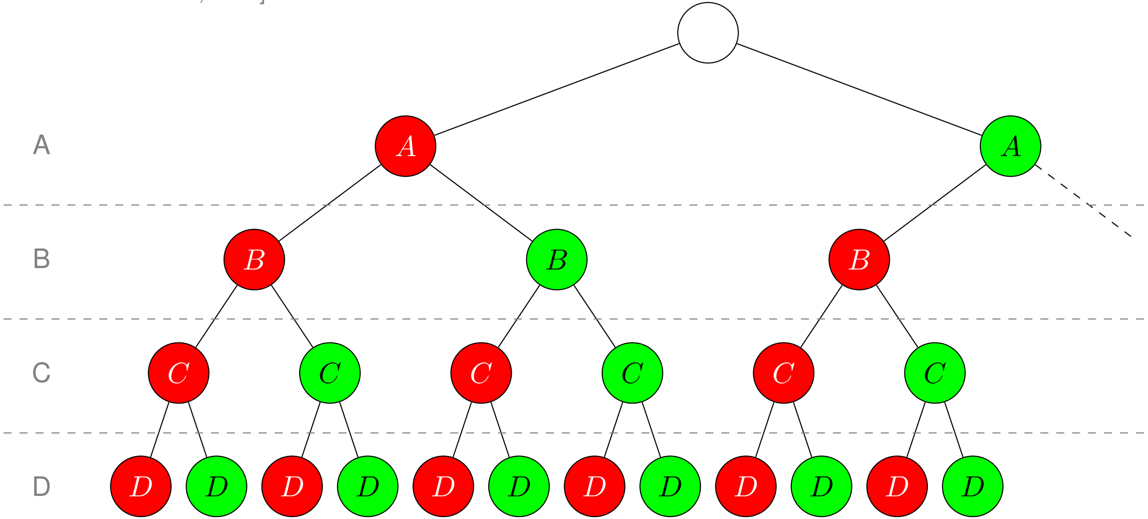
- Agents operate on a complete ordering
- Agents exchange CPA messages containing partial assignments
- When a solution is found, its solution cost as an UB is broadcasted to all agents
- The UB is used for branch pruning

Complete ordering



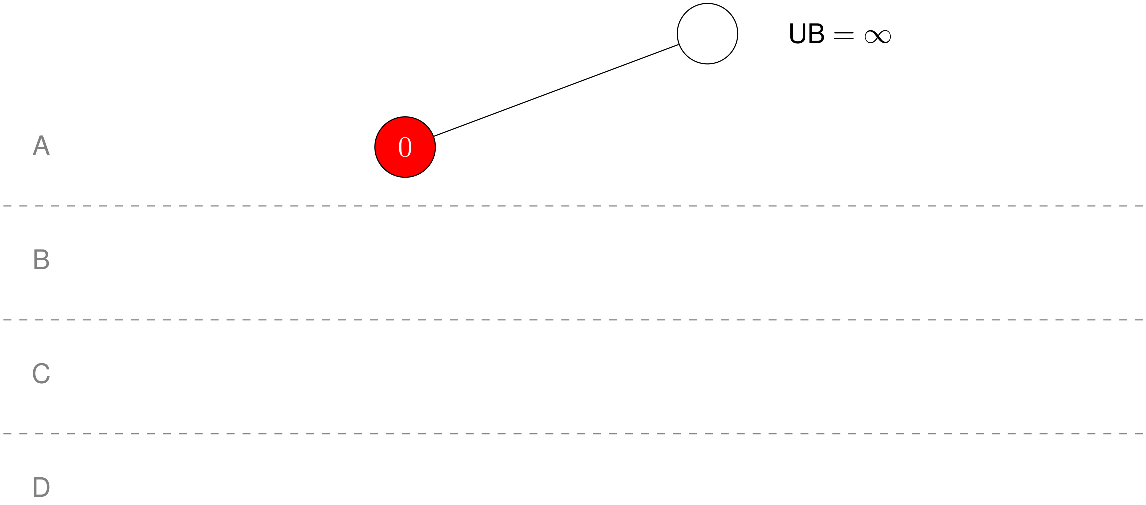
Synchronous Branch-and-Bound (SBB)

[Hirayama and Yokoo, 1997]



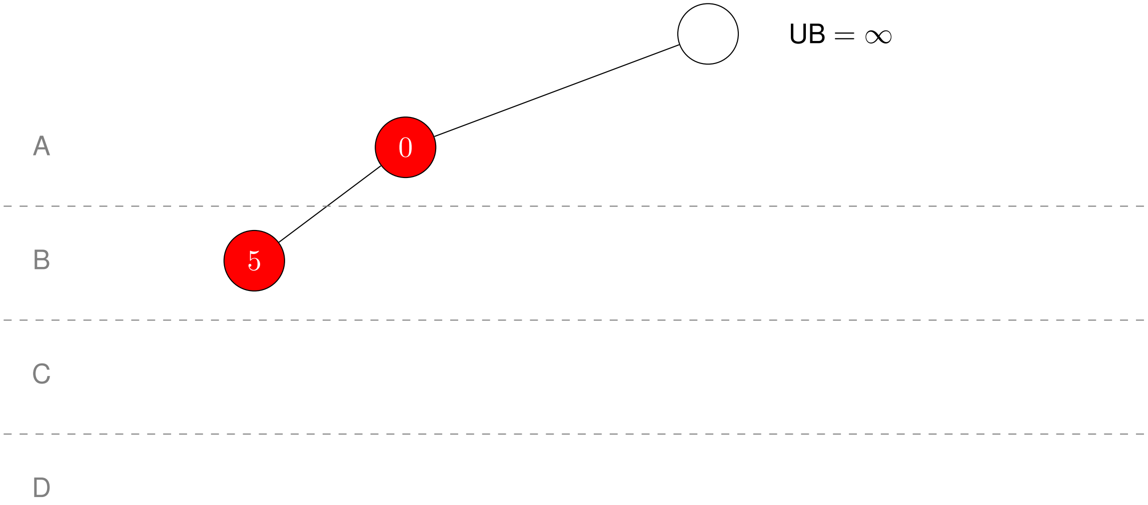
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



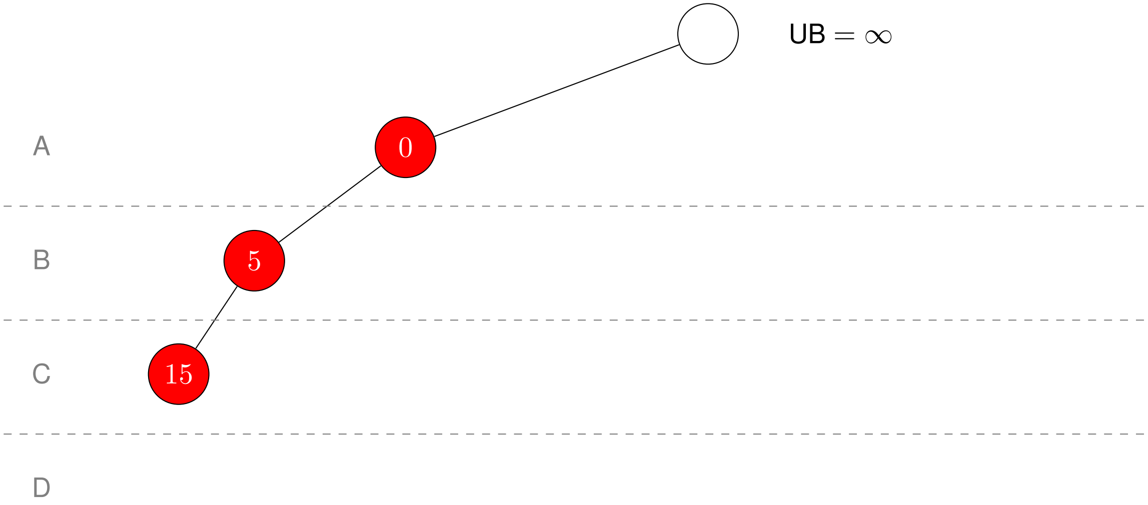
Synchronous Branch-and-Bound (SBB)

[Hirayama and Yokoo, 1997]



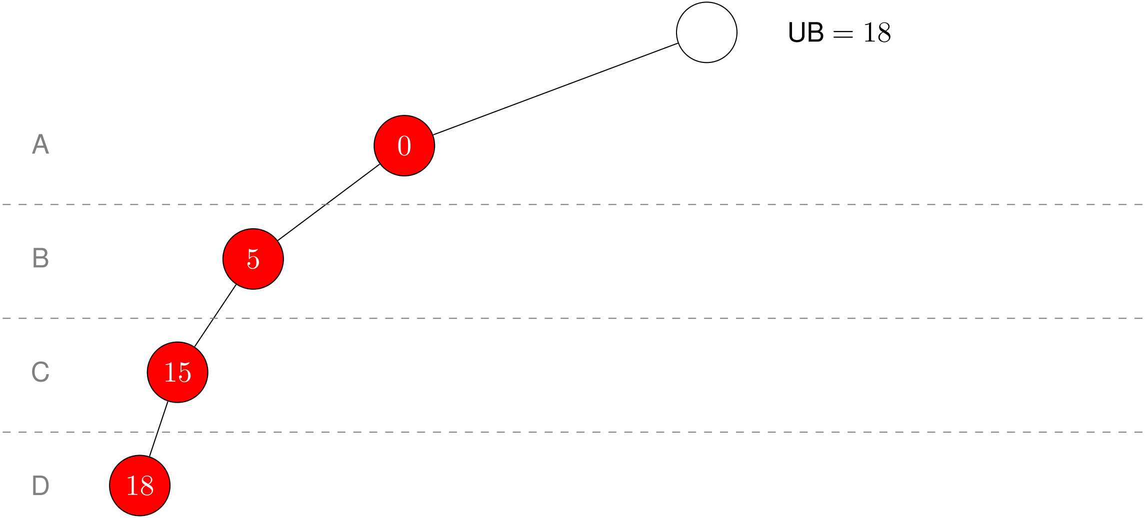
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



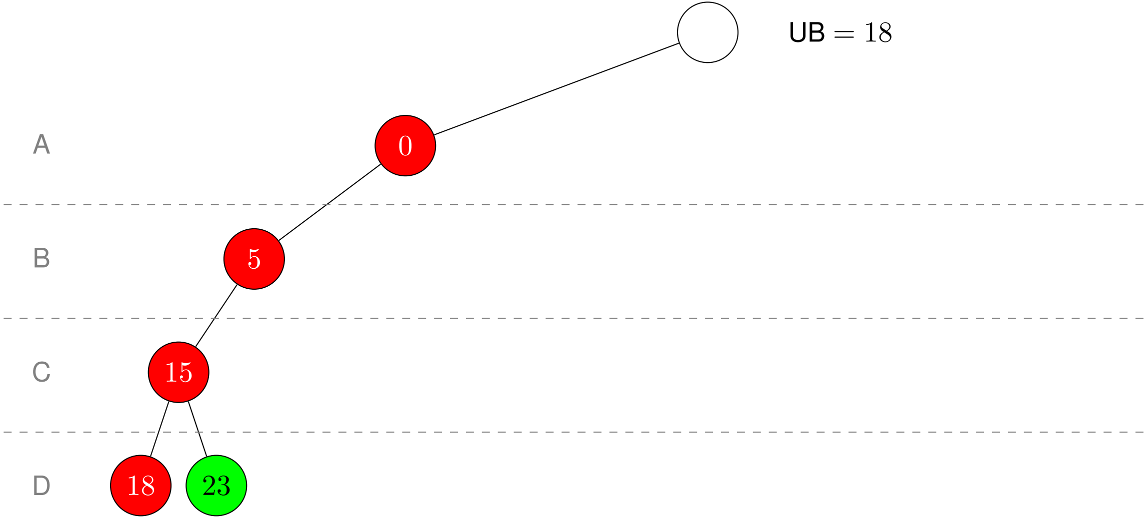
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



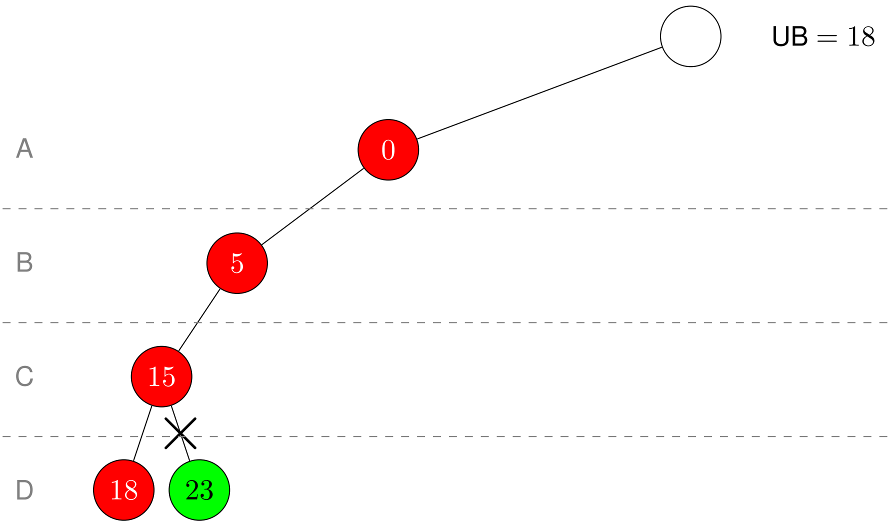
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



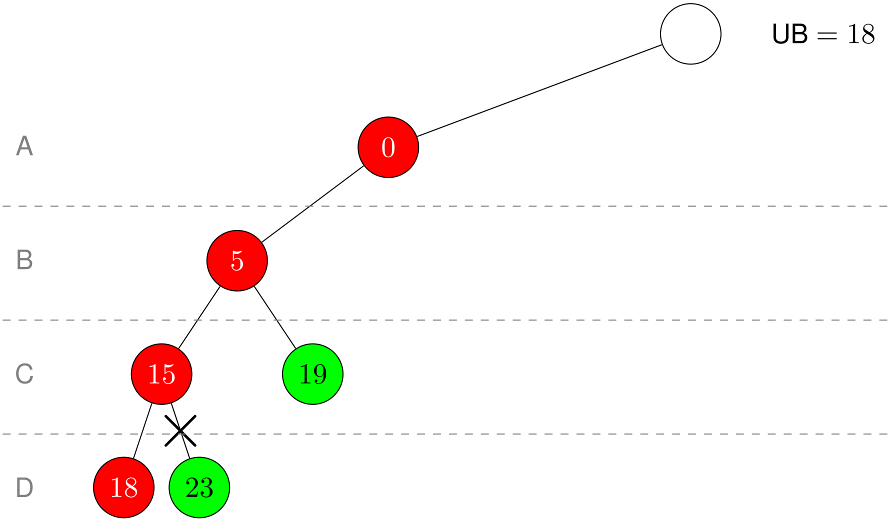
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



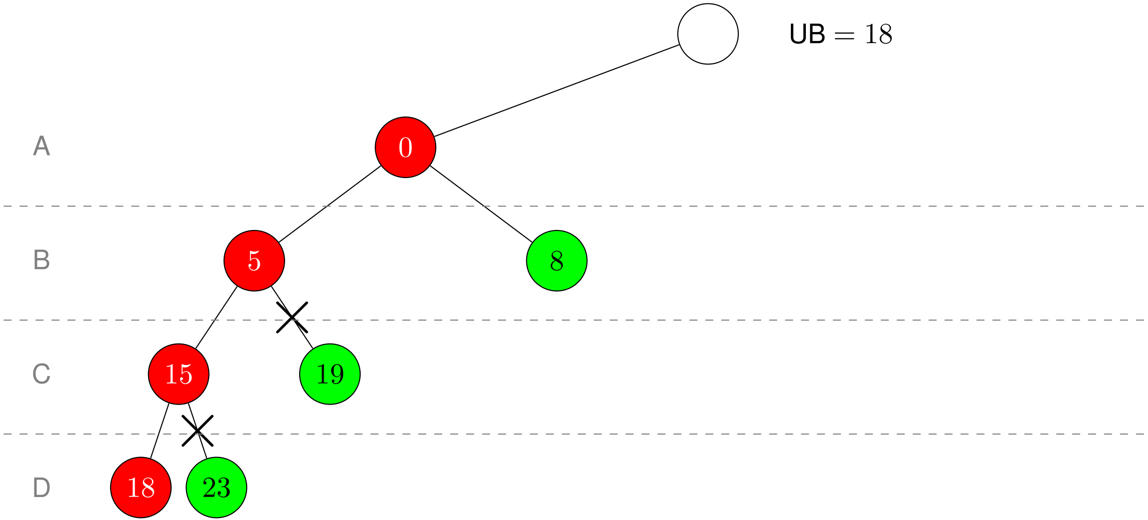
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



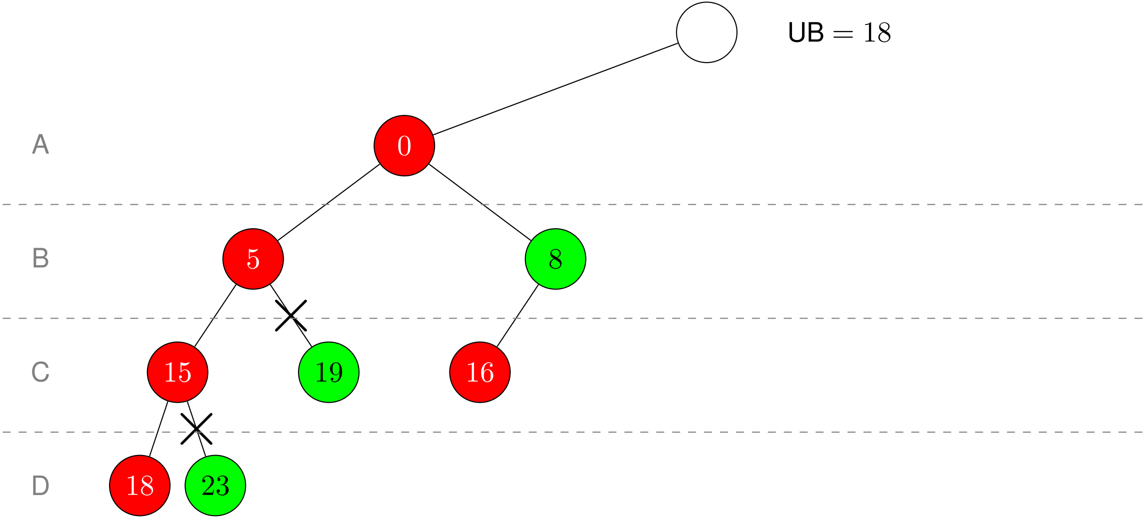
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



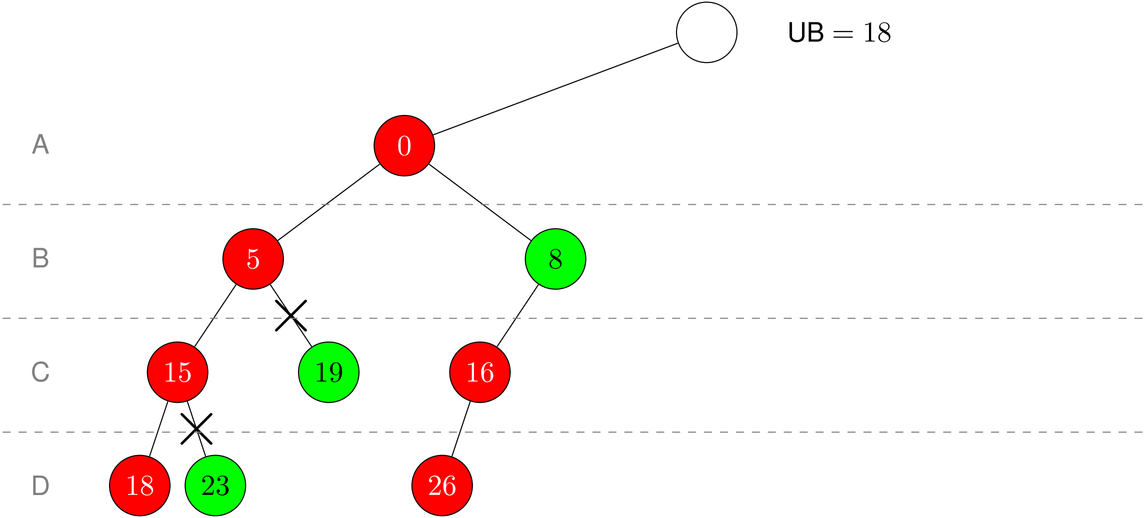
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



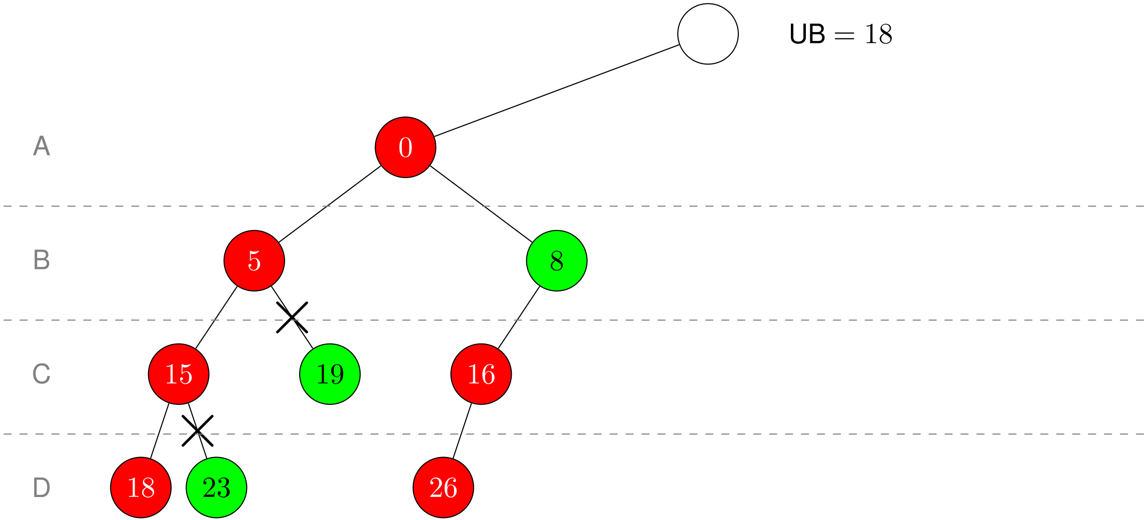
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



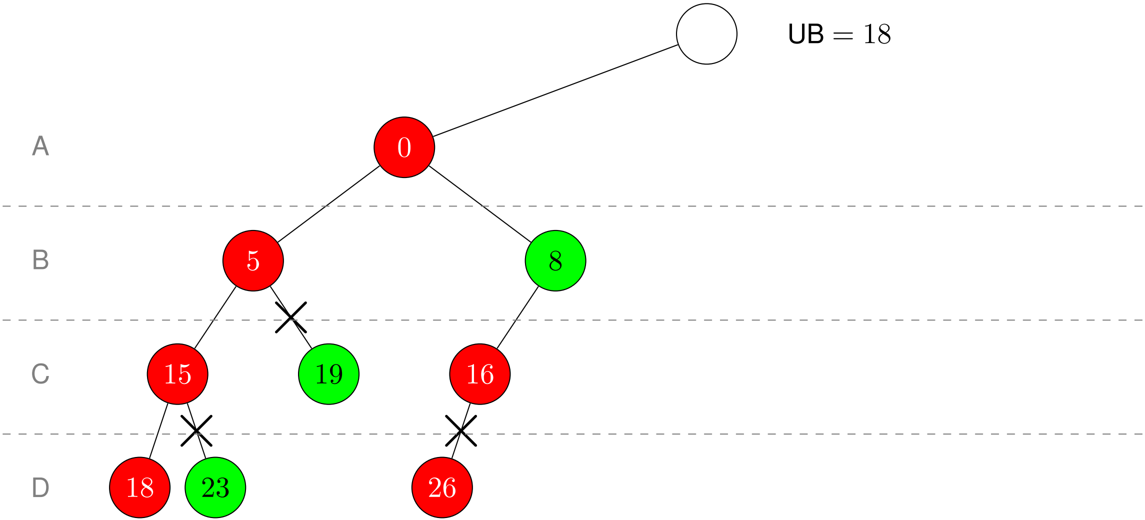
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



Synchronous Branch-and-Bound (SBB)

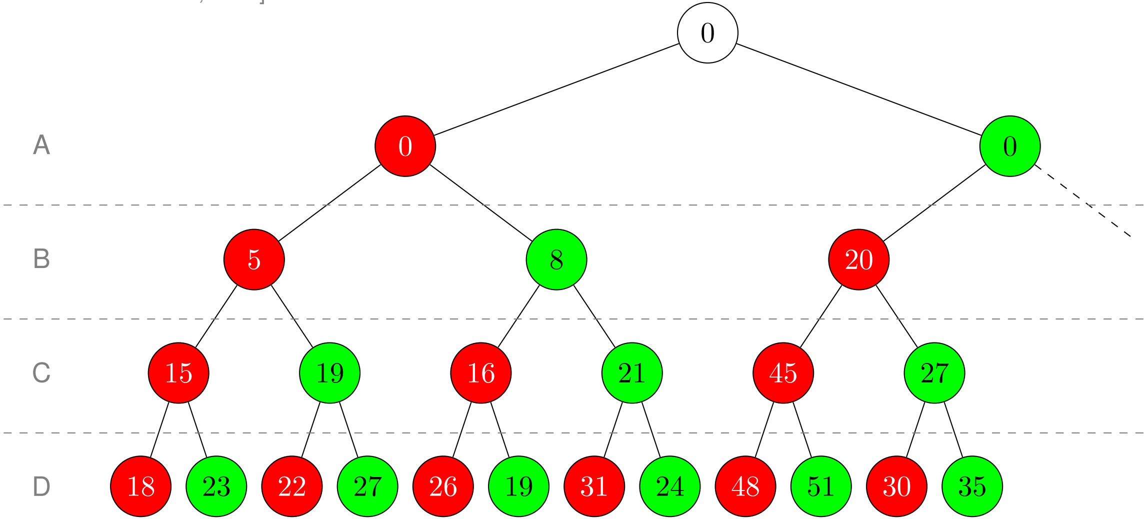
[HIRAYAMA and YOKOO, 1997]

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

branching factor = b
num variables = d

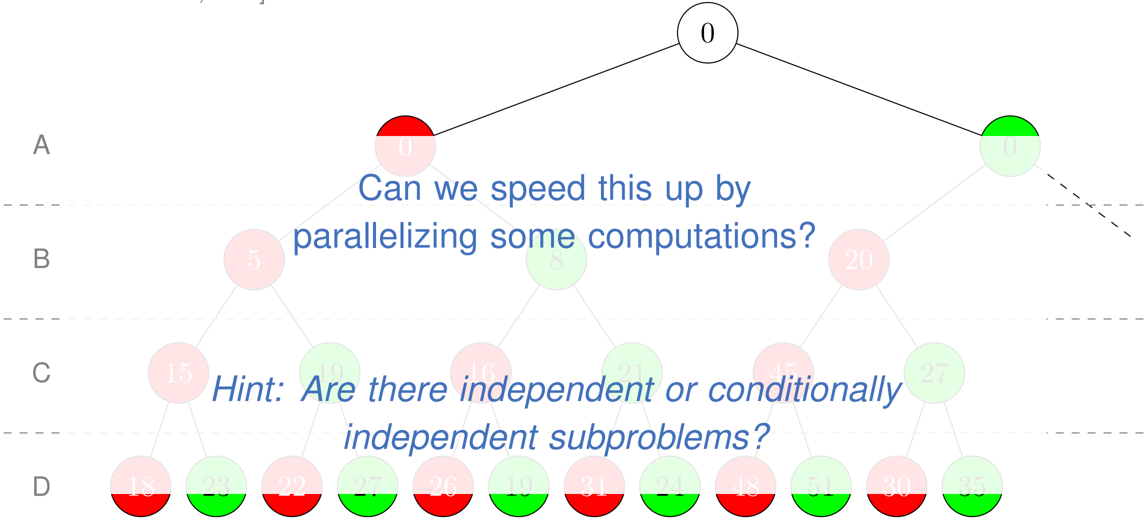
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]

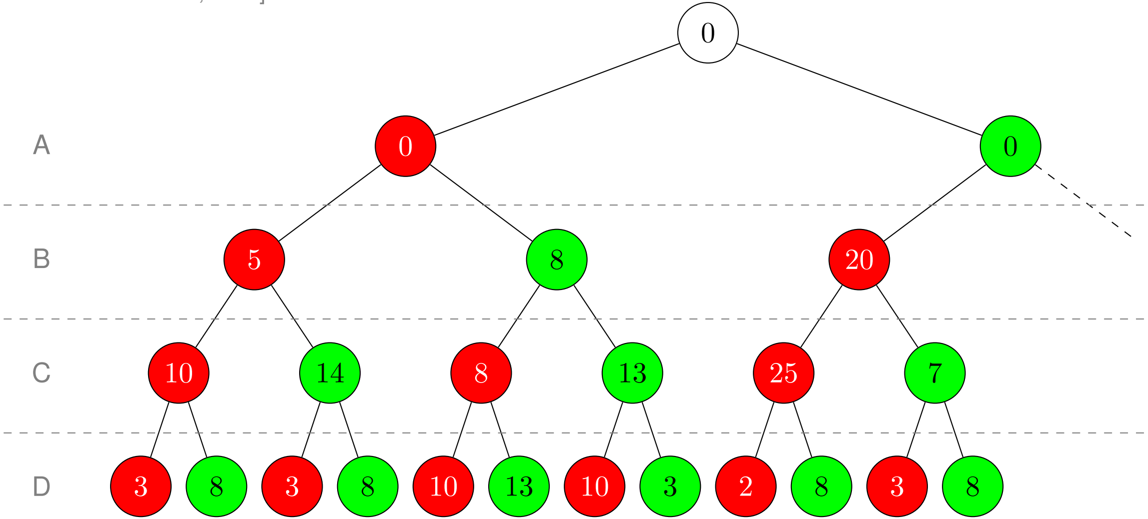


Can we speed this up by parallelizing some computations?

Hint: Are there independent or conditionally independent subproblems?

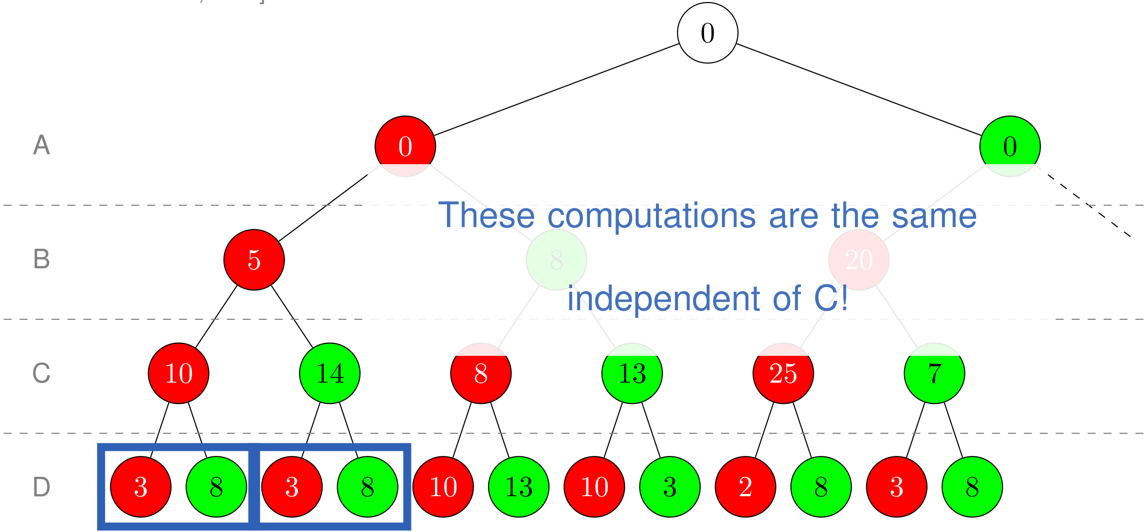
Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]

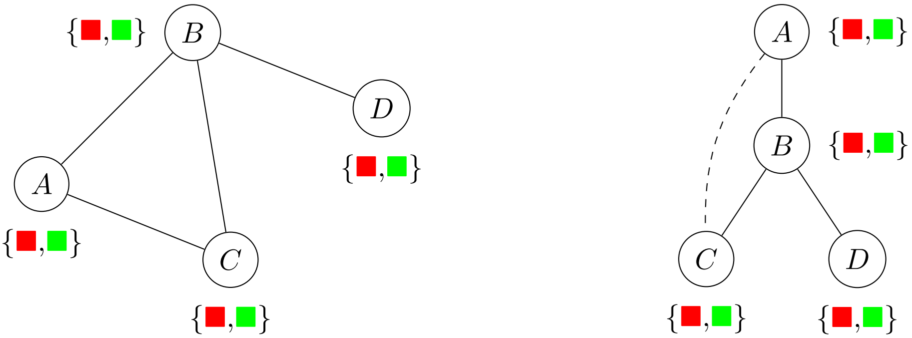


Synchronous Branch-and-Bound (SBB)

[HIRAYAMA and YOKOO, 1997]



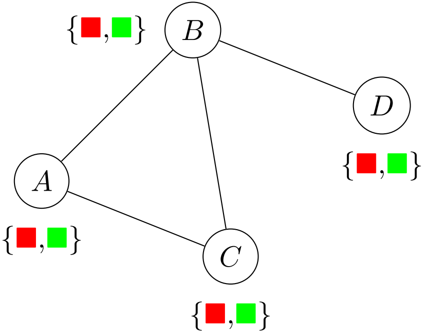
Pseudo-Tree

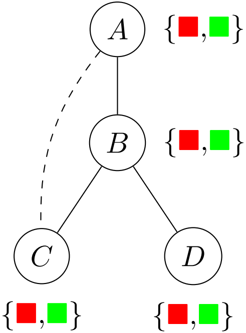


Definition (Pseudo-Tree)

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

Pseudo-Tree



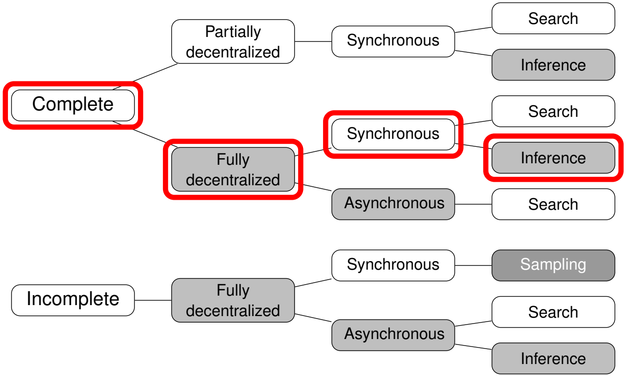


Definition (Pseudo-Tree)

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]



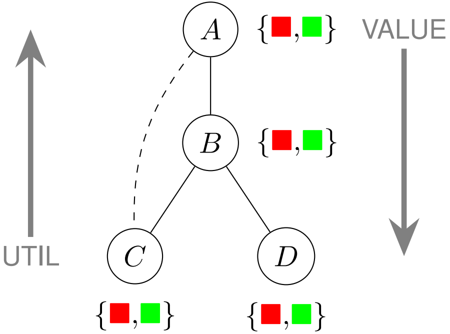
Distributed Pseudotree Optimization Procedure (DPOP)

[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]

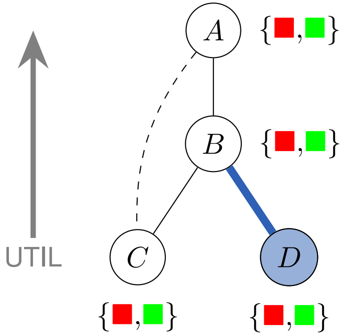
B	D	(B,D)
r	r	3
r	g	8
g	r	10
g	g	3

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

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

Message to B

B	cost
r	3
g	3



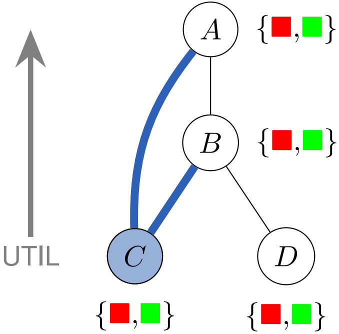
DPOP

[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



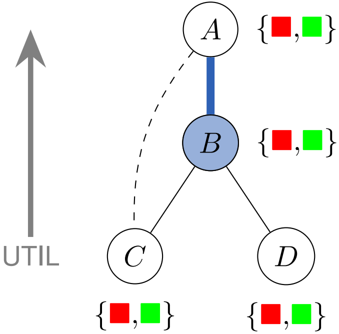
DPOP

[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

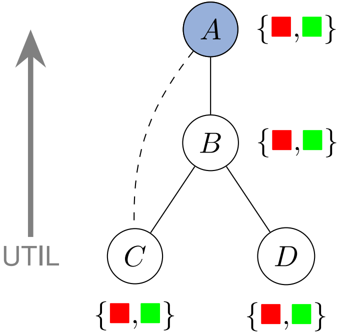


DPOP

[PETCU and FALTINGS, 2005b]

A	cost
r	18
g	12

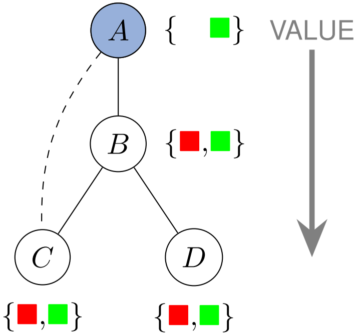
optimal cost $= 12$



DPOP

[PETCU and FALTINGS, 2005b]

A	cost
r	18
g	12

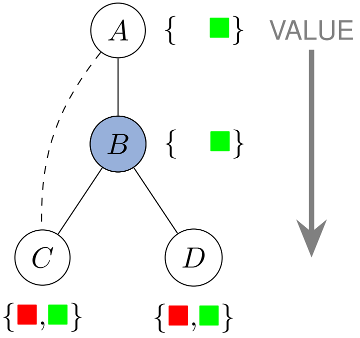


- Select value for $A = g$
- Send MSG "$A = g$" to agents B and C

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

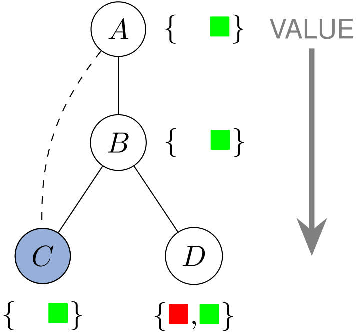


- Select value for $B = g$
- Send MSG "$B = g$" to agents C and D

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
$\mathbf{g}$	r	r	5	10	15
$\mathbf{g}$	r	g	4	3	7
$\mathbf{g}$	$\mathbf{g}$	r	3	10	13
$\mathbf{g}$	$\mathbf{g}$	g	3	3	6



- Select value for $C = g$

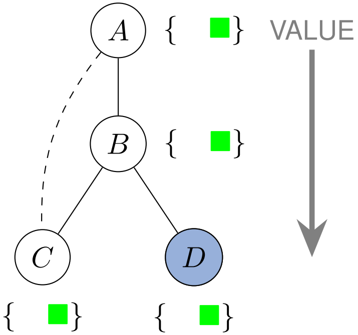
DPOP

[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

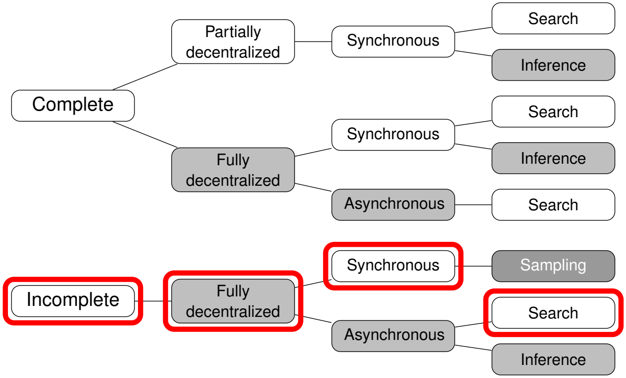
[PETCU and FALTINGS, 2005b]

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

branching factor = b
num variables = d

DCOP Algorithms

See [FIORETTO et al., 2018]



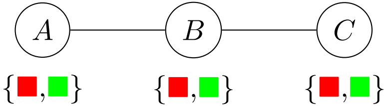
Distributed Local Search

[MAHESWARAN et al., 2004; ZHANG et al., 2003]

Local Search Algorithms

- DSA: Distributed Stochastic Search [ZHANG et al., 2005]
- MGM: Maximum Gain Messages Algorithm [MAHESWARAN et al., 2004]
- Note: we now maximize utilities
- Every agent individually decides whether to change its value or not
- Decision involves
  - knowing neighbors' values
  - calculation of utility gain by changing values
  - probabilities

x_i	x_j	(A, B)	(B, C)
		5	5
		5	0
		0	0
		8	8



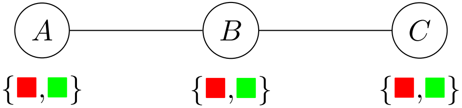
DSA Algorithm

[ZHANG et al., 2005]

- All agents execute the following
  - Randomly choose a value
  - while (termination is not met)
    - if (a new value is assigned): send the new value to neighbors
    - collect neighbors' new values if any
    - select and assign the next value based on assignment rule

DSA Algorithm

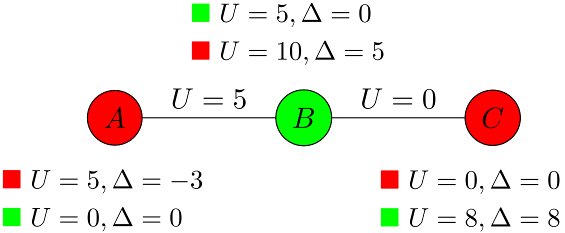
[ZHANG et al., 2005]



x_i	x_j	(A, B)	(B, C)
		5	5
		5	0
		0	0
		8	8

DSA Algorithm

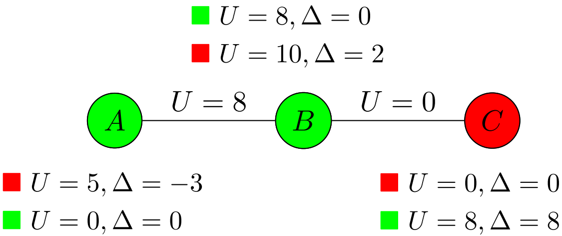
[ZHANG et al., 2005]



x_i	x_j	(A, B)	(B, C)
		5	5
		5	0
		0	0
		8	8

DSA Algorithm

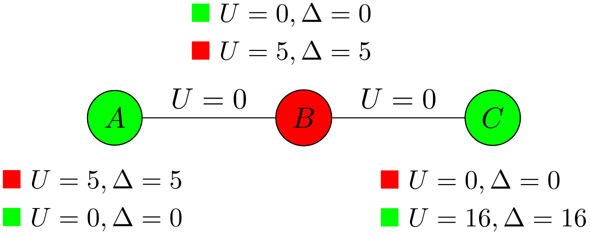
[ZHANG et al., 2005]



x_i	x_j	(A, B)	(B, C)
		5	5
		5	0
		0	0
		8	8

DSA Algorithm

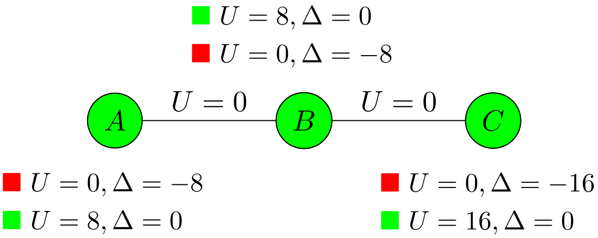
[ZHANG et al., 2005]



x_i	x_j	(A, B)	(B, C)
		5	5
		5	0
		0	0
		8	8

DSA Algorithm

[ZHANG et al., 2005]



x_i	x_j	(A, B)	(B, C)
		5	5
		5	0
		0	0
		8	8

MGM Algorithm

[MAHESWARAN et al., 2004]

- All agents execute the following
  - Randomly choose a value
  - while (termination is not met)
    - if (a new value is assigned): send the new value to neighbors
    - collect neighbors' new values if any
    - calculate gain and send it to neighbors
    - collect neighbors' gains
    - if (it has the highest gain among all neighbors): change value to the value that maximizes gain

Large Great if you need an anytime algorithm!

MGM Algorithm

[MAHESWARAN et al., 2004]

- All agents execute the following
  - Randomly choose a value
  - while (termination is not met)
    - if (a new value is assigned): send the new value to neighbors
    - collect neighbors' new values if any
    - calculate gain and send it to neighbors
    - collect neighbors' gains
    - if (it has the highest gain among all neighbors): change value to the value that maximizes gain

Large Great if you need an anytime algorithm!

MGM vs DSA

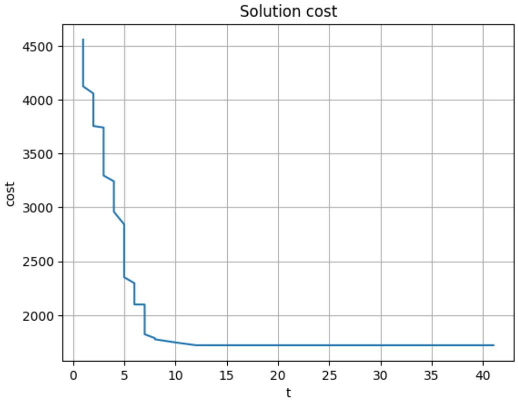


Figure: MGM

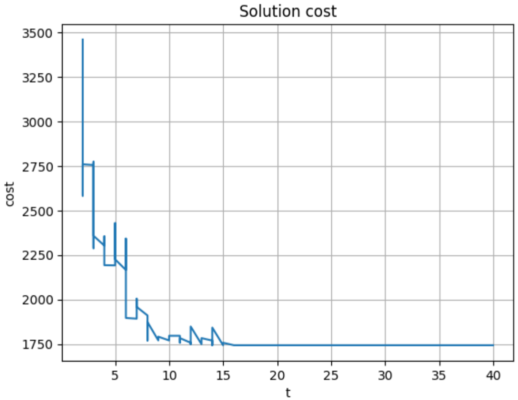


Figure: DSA

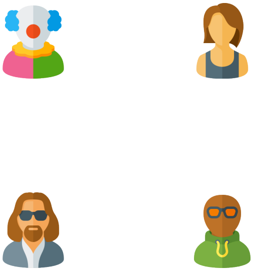
Extensions to the DCOP Framework

- 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
  - $\mathbb{E}[\text{DPOP}]$ 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]
- ...

Today’s Menu

Characteristic Function Games (CFGs)

[CHALKIADAKIS et al., 2011]



Set of Agents A

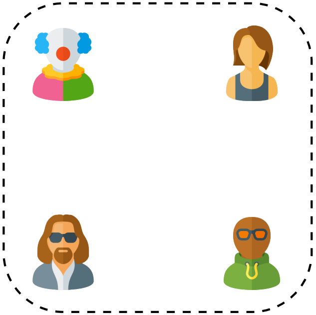
$A = \{$  $\}$

Characteristic Function $v(\cdot)$

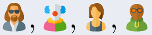
- $v(\{$  $\}) = 0$
- $v(\{$  $\}) = -7$
- $v(\{$ $\}) = 3$
- $\ldots$

Characteristic Function Games (CFGs)

[CHALKIADAKIS et al., 2011]



Set of Agents A

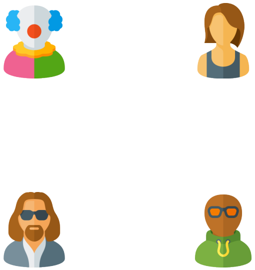
$A = \{$  $\}$

Characteristic Function $v(\cdot)$

- $v(\{$  $\}) = 0$
- $v(\{$  $\}) = -7$
- $v(\{$ $\}) = 3$
- $\ldots$

Characteristic Function Games (CFGs)

[Chalkiadakis et al., 2011]



Set of Agents A

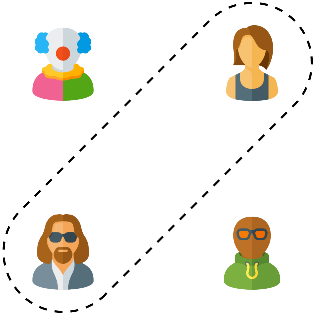
$A = \{\ ,\ ,\ ,\ \}$

Characteristic Function $v(\cdot)$

- $v(\{\ ,\ \}) = 0$
- $v(\{\ ,\ ,\ \}) = -7$
- $v(\{\ ,\ \}) = 3$
- $\dots$

Characteristic Function Games (CFGs)

[CHALKIADAKIS et al., 2011]



Set of Agents A

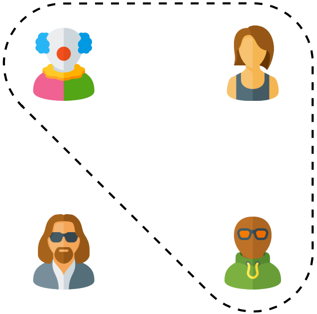
$A = \{ \quad , \quad , \quad , \quad \}$

Characteristic Function $v(\cdot)$

- $v(\{ \quad , \quad \}) = 0$
- $v(\{ \quad , \quad , \quad \}) = -7$
- $v(\{ \quad , \quad \}) = 3$
- $\ldots$

Characteristic Function Games (CFGs)

[CHALKIADAKIS et al., 2011]



Set of Agents A

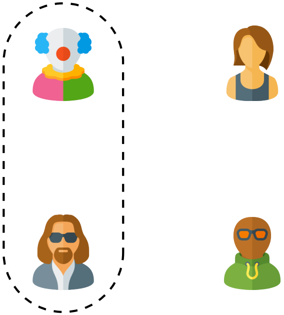
$A = \{$  $\}$

Characteristic Function $v(\cdot)$

- $v(\{$  $\}) = 0$
- $v(\{$  $\}) = -7$
- $v(\{$  $\}) = 3$
- $\ldots$

Characteristic Function Games (CFGs)

[CHALKIADAKIS et al., 2011]



Set of Agents A

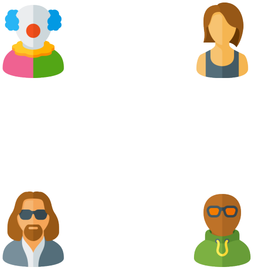
$A = \{ \ , \ , \ , \ \}$

Characteristic Function $v(\cdot)$

- $v(\{ \ , \ \}) = 0$
- $v(\{ \ , \ , \ \}) = -7$
- $v(\{ \ , \ \}) = 3$
- $\dots$

Characteristic Function Games (CFGs)

[CHALKIADAKIS et al., 2011]



Set of Agents A

$A = \{\ ,\ ,\ ,\ \}$

Characteristic Function $v(\cdot)$

- $v(\{\ ,\ \}) = 0$
- $v(\{\ ,\ ,\ \}) = -7$
- $v(\{\ ,\ \}) = 3$
- ...

Characteristic Function

[Chalkiadakis et al., 2011]

Characteristic Function

The function $v : \mathcal{P}(A) \rightarrow \mathbb{R}$ associates a value to *every coalition* (i.e., subset) of A

Exponential Complexity

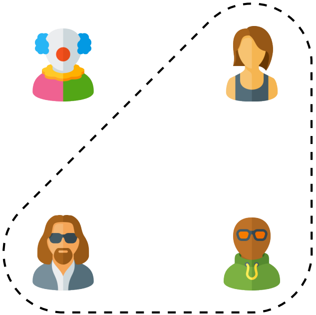
Representing $v(\cdot)$ as a *table* requires an *exponential* number of steps (i.e., $2^{|A|}$)

Mitigate this Complexity

(1) *Restrict* the set of coalitions or (2) consider $v(\cdot)$ with a specific *structure*

Cardinality-Restricted CFGs

[SHEHORY and KRAUS, 1998]



Maximum Cardinality k

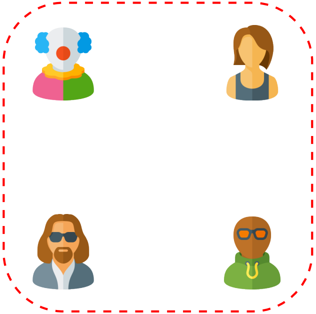
E.g., only coalitions of size ≤ 3 are feasible

Polynomial Number of Coalitions

Total number of coalitions is $\sum_{i=1}^{k} \binom{|A|}{i} = \mathcal{O}(|A|^k)$, i.e., *polynomial wrt* $|A|$

Cardinality-Restricted CFGs

[SHEHORY and KRAUS, 1998]



Maximum Cardinality k

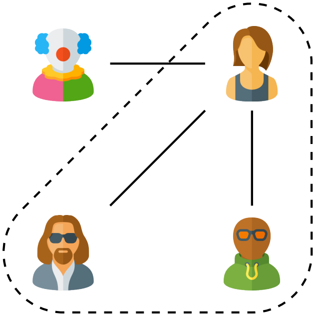
E.g., only coalitions of size ≤ 3 are feasible

Polynomial Number of Coalitions

Total number of coalitions is $\sum_{i=1}^{k} \binom{|A|}{i} = \mathcal{O}(|A|^k)$, i.e., *polynomial wrt* $|A|$

Graph-Restricted CFGs

[Myerson, 1977], [Demange, 2004]



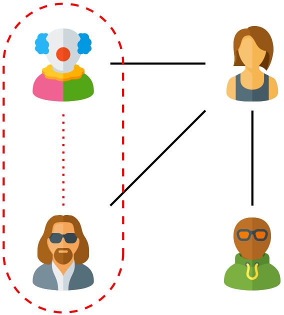
Graph G among Agents

Connected Subgraphs

A coalition is *feasible* only if it induces a *connected* subgraph of G

Graph-Restricted CFGs

[Myerson, 1977], [Demange, 2004]



Graph G among Agents

Connected Subgraphs

A coalition is *feasible* only if it induces a *connected* subgraph of G

Real-World Example: Social Ridesharing

[Bistaffa et al., 2017a]

Social Ridesharing

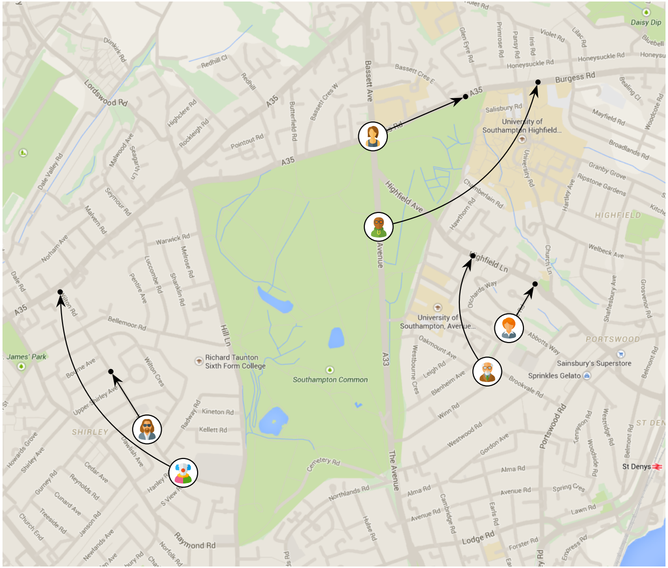
Arrange *cost-effective* shared cars among agents connected by a *social network*

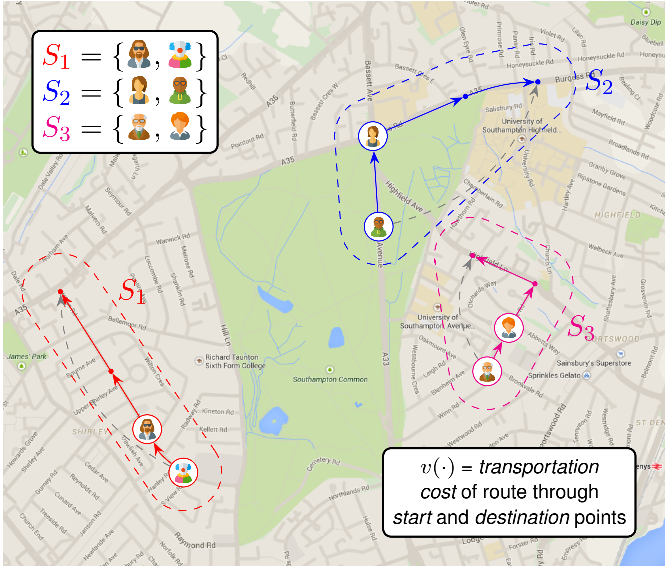
Cardinality-Based Constraints

Cars (i.e., coalitions) can contain *up to 5 passengers*

Graph-Based Constraints

We only form coalitions among “*friends of friends*” (connected subgraph)

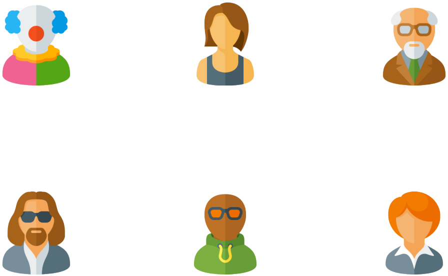

Lordswood Rd
Redhill Cl
A35
Bassett Ave
Burgess Rd
Salisbury Rd
University of Southampton Highfield...
Daisy Dip
Honeysuckle Rd
Highfield Ave
HIGHFIELD
Highfield Ln
University of Southampton, Avenue...
PORTSWOOD
Sainsbury's Superstore
Sprinkles Gelato
Richard Taunton Sixth Form College
Southampton Common
James' Park
SHIRLEY
Hill Ln
A33
The Avenue
Raymond Rd
Lodge Rd
Portswood Rd
St Denys
Cemetery Rd
Northlands Rd
Bellemoor Rd
Warwick Rd
Pointout Rd



$S_1 = \{\ ,\ \}$
$S_2 = \{\ ,\ \}$
$S_3 = \{\ ,\ \}$
S_1
S_2
S_3
$v(\cdot)$ = *transportation cost* of route through *start* and *destination* points
Southampton Common
University of Southampton Highfield...
University of Southampton, Avenue...
Richard Taunton Sixth Form College
Sainsbury's Superstore
Sprinkles Gelato
James' Park
HIGHFIELD
SHIRLEY
Highfield Ave
Bassett Ave
Burgess Rd
A35
A33
Hill Ln
Raymond Rd
Cemetery Rd
Northlands Rd
Daisy Dip
Honeysuckle Rd
Salisbury Rd
Broadlands Rd
Granby Grove
Ripstone Gardens
Welbeck Ave
Brookvale Rd
Abbotts Way
Orchards Way
Bellemoor Rd
Warwick Rd
Kineton Rd
Kellett Rd
Pointout Rd

Coalition Structure Generation (CSG)

[Rahwan et al., 2015]

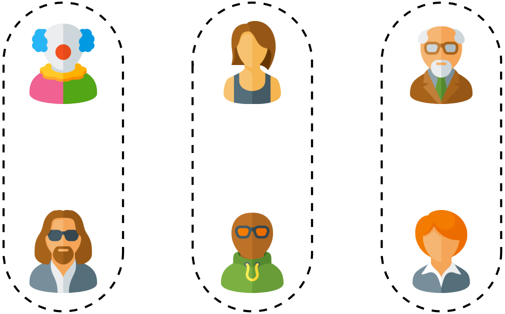


Solving the Coalition Structure Generation (CSG) Problem

Compute the partition $\mathcal{S}$ of A into *feasible* coalitions that *maximizes* the sum $\sum_{S \in \mathcal{S}} v(S)$

Coalition Structure Generation (CSG)

[RAHWAN et al., 2015]

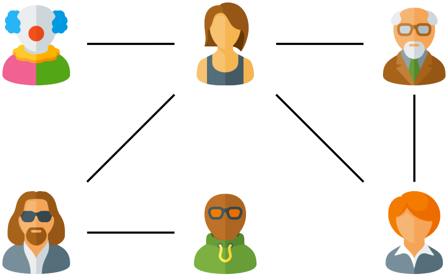


Solving the Coalition Structure Generation (CSG) Problem

Compute the partition $\mathcal{S}$ of A into *feasible* coalitions that *maximizes* the sum $\sum_{S \in \mathcal{S}} v(S)$

Graph-Restricted Coalition Structure Generation (CSG)

[RAHWAN et al., 2015]

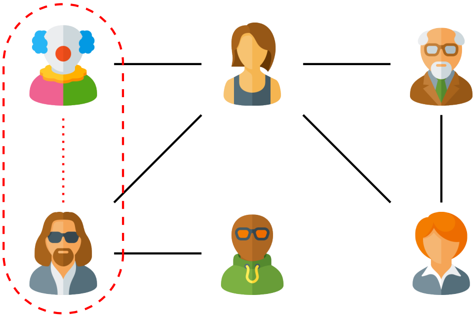


Solving the Coalition Structure Generation (CSG) Problem

Compute the partition $\mathcal{S}$ of A into *feasible* coalitions that *maximizes* the sum $\sum_{S \in \mathcal{S}} v(S)$

Graph-Restricted Coalition Structure Generation (CSG)

[RAHWAN et al., 2015]

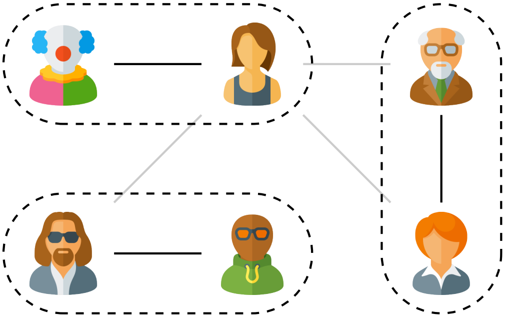


Solving the Coalition Structure Generation (CSG) Problem

Compute the partition $\mathcal{S}$ of A into *feasible* coalitions that *maximizes* the sum $\sum_{S \in \mathcal{S}} v(S)$

Graph-Restricted Coalition Structure Generation (CSG)

[RAHWAN et al., 2015]

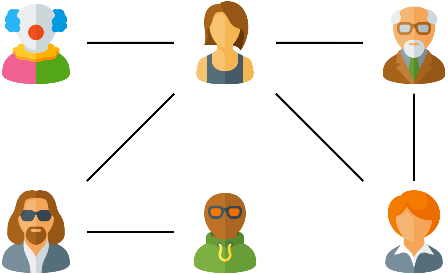


Solving the Coalition Structure Generation (CSG) Problem

Compute the partition $\mathcal{S}$ of A into *feasible* coalitions that *maximizes* the sum $\sum_{S \in \mathcal{S}} v(S)$

CSG Approaches based on Search

[BISTAFFA et al., 2017b]

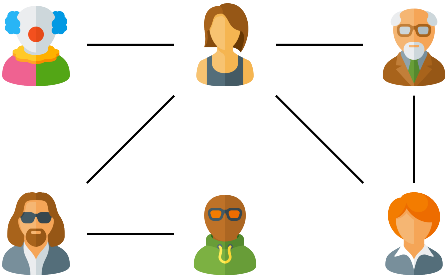


Edge Contraction Operation

Contraction of edge $(S_i, S_j) \rightarrow$ form coalition $S_i \cup S_j$

CSG Approaches based on Search

[BISTAFFA et al., 2017b]

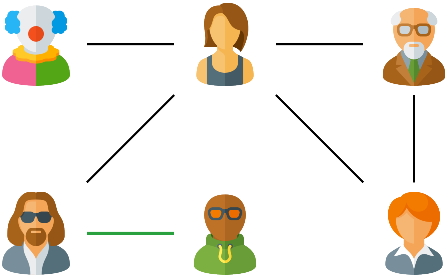


Edge Contraction Operation

Contraction of edge $(S_i, S_j) \to$ form coalition $S_i \cup S_j$

CSG Approaches based on Search

[BISTAFFA et al., 2017b]

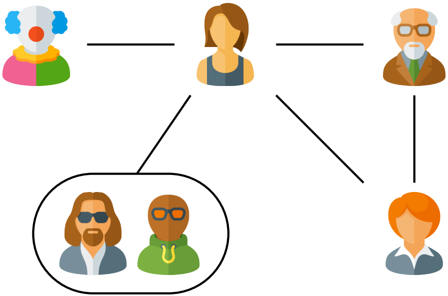


Edge Contraction Operation

Contraction of edge $(S_i, S_j) \rightarrow$ form coalition $S_i \cup S_j$

CSG Approaches based on Search

[BISTAFFA et al., 2017b]

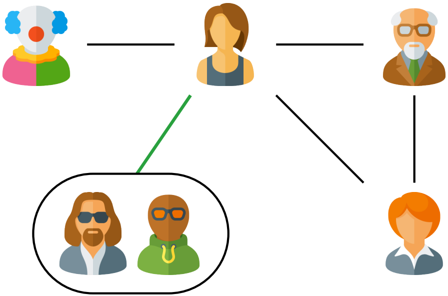


Edge Contraction Operation

Contraction of edge $(S_i, S_j) \rightarrow$ form coalition $S_i \cup S_j$

CSG Approaches based on Search

[Bistaffa et al., 2017b]

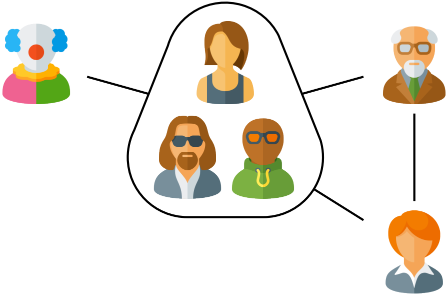


Edge Contraction Operation

Contraction of edge $(S_i, S_j) \rightarrow$ form coalition $S_i \cup S_j$

CSG Approaches based on Search

[BISTAFFA et al., 2017b]

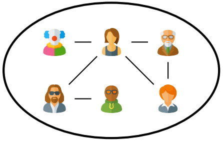


Edge Contraction Operation

Contraction of edge $(S_i, S_j) \rightarrow$ form coalition $S_i \cup S_j$

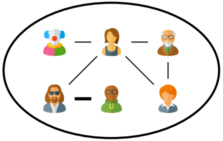
CSG Approaches based on Search

[Bistaffa et al., 2017b]



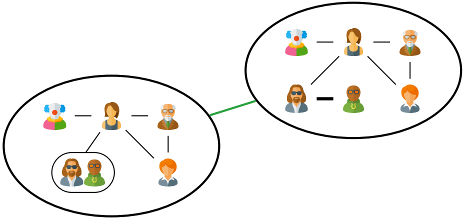
CSG Approaches based on Search

[BISTAFFA et al., 2017b]



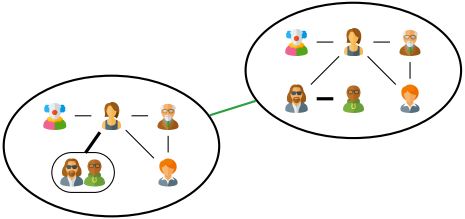
CSG Approaches based on Search

[Bistaffa et al., 2017b]



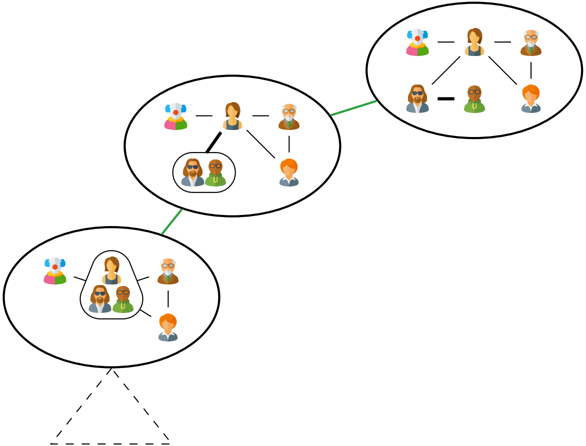
CSG Approaches based on Search

[BISTAFFA et al., 2017b]



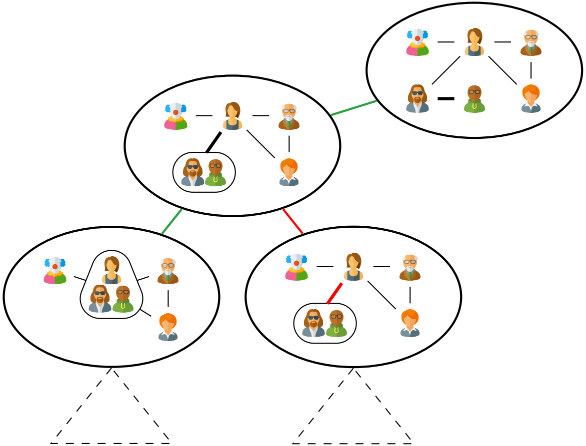
CSG Approaches based on Search

[BISTAFFA et al., 2017b]



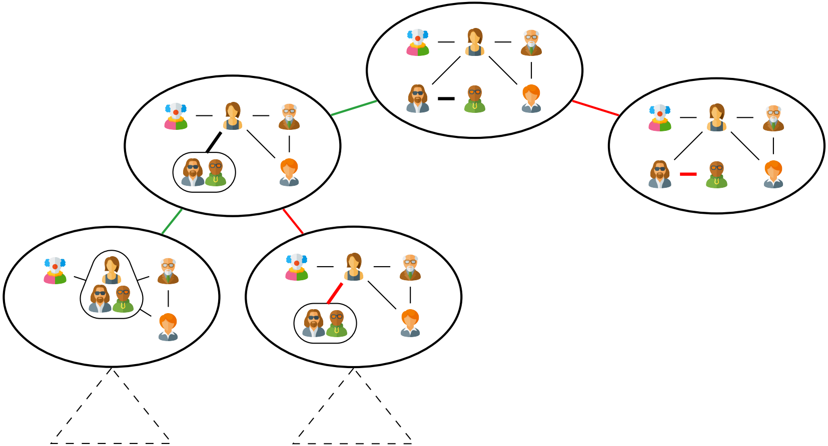
CSG Approaches based on Search

[BISTAFFA et al., 2017b]



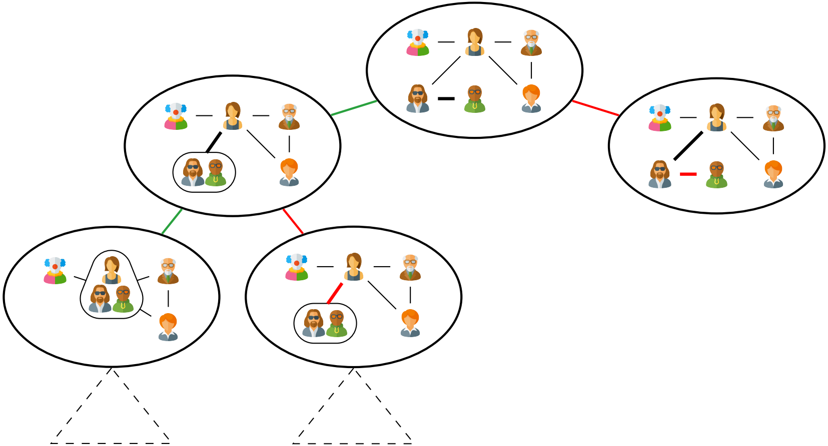
CSG Approaches based on Search

[BISTAFFA et al., 2017b]



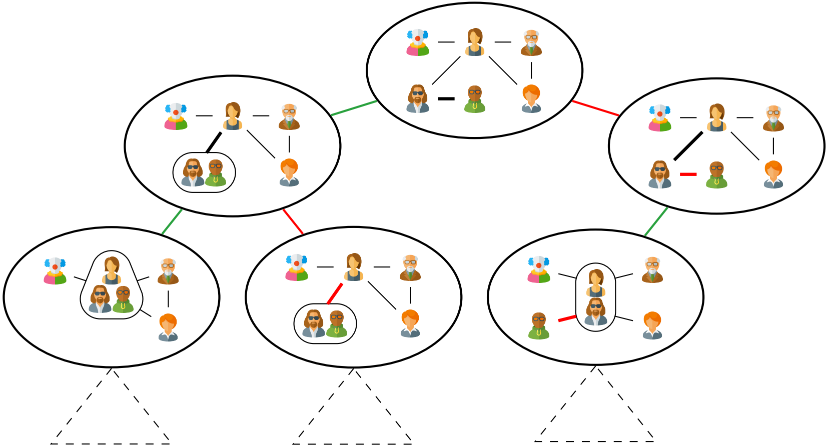
CSG Approaches based on Search

[BISTAFFA et al., 2017b]



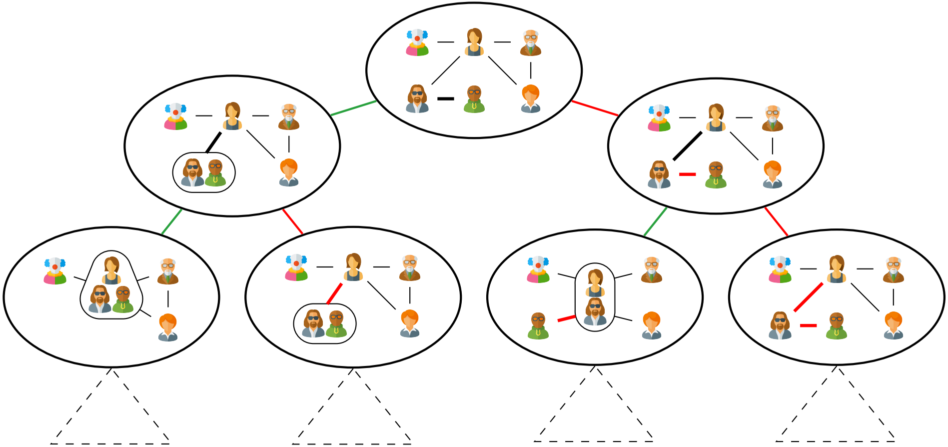
CSG Approaches based on Search

[BISTAFFA et al., 2017b]



CSG Approaches based on Search

[BISTAFFA et al., 2017b]



CSG Approaches based on Search

[Bistaffa et al., 2017b]

CFSS Algorithm

- Builds a *Binary Decision Diagram* (BDD) by contracting (or not) an edge at each step
- Each coalition structure (i.e., partition of A) is represented *only once* in the BDD
- The optimal coalition structure is computed by doing a *depth-first* traversal of the BDD

Pros

Approximate algorithm with quality guarantees if used in conjunction with *Branch-and-Bound*

Cons

Performance depends on the assumption that $v(\cdot)$ can be expressed in *closed-form*

CSG Approaches based on Search

[BISTAFFA et al., 2017b]

CFSS Algorithm

- Builds a *Binary Decision Diagram* (BDD) by contracting (or not) an edge at each step
- Each coalition structure (i.e., partition of A) is represented *only once* in the BDD
- The optimal coalition structure is computed by doing a *depth-first* traversal of the BDD

Pros

Approximate algorithm with quality guarantees if used in conjunction with *Branch-and-Bound*

Cons

Performance depends on the assumption that $v(\cdot)$ can be expressed in *closed-form*

CSG Approaches based on Search

[BISTAFFA et al., 2017b]

CFSS Algorithm

- Builds a *Binary Decision Diagram* (BDD) by contracting (or not) an edge at each step
- Each coalition structure (i.e., partition of A) is represented *only once* in the BDD
- The optimal coalition structure is computed by doing a *depth-first* traversal of the BDD

Pros

Approximate algorithm with quality guarantees if used in conjunction with *Branch-and-Bound*

Cons

Performance depends on the assumption that $v(\cdot)$ can be expressed in *closed-form*

CSG Approaches based on Integer Linear Programming

Background on Integer Linear Programming

Weighted Knapsack Problem

We want to fill our knapsack (capacity = c) with the goal of maximizing the total value

What is the Optimal Subset of Object for $c = 5$?

A	Pick 🍎 (weight = 1) $\rightarrow 1$	$w(\text{🍎}) = 1, v(\text{🍎}) = 1$
B	Pick 💎 (weight = 2) $\rightarrow 4$	$w(\text{💎}) = 2, v(\text{💎}) = 4$
C	Pick 💍 (weight = 4) $\rightarrow 3$	$w(\text{💍}) = 4, v(\text{💍}) = 3$
D	Pick 👑 (weight = 5) $\rightarrow 9$	$w(\text{👑}) = 6, v(\text{👑}) = 1$
E	Pick 📟 (weight = 3) $\rightarrow 6$	$w(\text{📟}) = 3, v(\text{📟}) = 6$

CSG Approaches based on Integer Linear Programming

Background on Integer Linear Programming

Our Ingredients

- Let x_A, x_B, x_C, x_D, x_E be binary decision variables (either pick the object or not)
- Objective function: maximize the value of selected objects
- Constraint: do not exceed the knapsack capacity

Integer *Linear* Programming (ILP) Formulation

$$\begin{aligned} \text{maximize} \quad & 1 \cdot x_A + 4 \cdot x_B + 3 \cdot x_C + 9 \cdot x_D + 6 \cdot x_E && \text{(Values of selected objects)} \\ \text{subject to} \quad & 1 \cdot x_A + 2 \cdot x_B + 4 \cdot x_C + 5 \cdot x_D + 3 \cdot x_E \leq 5 && \text{(Capacity constraint)} \\ & x_A, x_B, x_C, x_D, x_E \in \{0, 1\} && \text{(Binary decision variables)} \end{aligned}$$

CSG Approaches based on Integer Linear Programming

Background on Integer Linear Programming

Our Ingredients

- Let x_A, x_B, x_C, x_D, x_E be binary decision variables (either pick the object or not)
- Objective function: maximize the value of selected objects
- Constraint: do not exceed the knapsack capacity

Integer *Linear* Programming (ILP) Formulation

maximize	$1 \cdot x_A + 4 \cdot x_B + 3 \cdot x_C + 9 \cdot x_D + 6 \cdot x_E$	(Values of selected objects)
subject to	$1 \cdot x_A + 2 \cdot x_B + 4 \cdot x_C + 5 \cdot x_D + 3 \cdot x_E \leq 5$	(Capacity constraint)
	$x_A, x_B, x_C, x_D, x_E \in \{0, 1\}$	(Binary decision variables)

CSG Approaches based on Integer Linear Programming

Background on Integer Linear Programming

Our Ingredients

- Let x_A, x_B, x_C, x_D, x_E be binary decision variables (either pick the object or not)
- Objective function: maximize the value of selected objects
- Constraint: do not exceed the knapsack capacity

Integer *Linear* Programming (ILP) Formulation

$$\begin{array}{lll} \text{maximize} & 1 \cdot x_A + 4 \cdot x_B + 3 \cdot x_C + 9 \cdot x_D + 6 \cdot x_E & \text{(Values of selected objects)} \\ \text{subject to} & 1 \cdot x_A + 2 \cdot x_B + 4 \cdot x_C + 5 \cdot x_D + 3 \cdot x_E \leq 5 & \text{(Capacity constraint)} \\ & x_A, x_B, x_C, x_D, x_E \in \{0, 1\} & \text{(Binary decision variables)} \end{array}$$

CSG Approaches based on Integer Linear Programming

Background on Integer Linear Programming

Our Ingredients

- Let x_A, x_B, x_C, x_D, x_E be binary decision variables (either pick the object or not)
- Objective function: maximize the value of selected objects
- Constraint: do not exceed the knapsack capacity

Integer *Linear* Programming (ILP) Formulation

$$
\begin{array}{lll}
\text{maximize} & 1 \cdot x_A + 4 \cdot x_B + 3 \cdot x_C + 9 \cdot x_D + 6 \cdot x_E & \text{(Values of selected objects)} \\
\text{subject to} & 1 \cdot x_A + 2 \cdot x_B + 4 \cdot x_C + 5 \cdot x_D + 3 \cdot x_E \leq 5 & \text{(Capacity constraint)} \\
 & x_A, x_B, x_C, x_D, x_E \in \{0, 1\} & \text{(Binary decision variables)}
\end{array}
$$

CSG Approaches based on Integer Linear Programming

Background on Integer Linear Programming

Our Ingredients

- Let x_A, x_B, x_C, x_D, x_E be binary decision variables (either pick the object or not)
- Objective function: maximize the value of selected objects
- Constraint: do not exceed the knapsack capacity

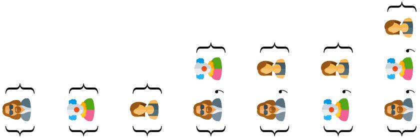
Integer *Linear* Programming (ILP) Formulation

$$\begin{aligned} \text{maximize} \quad & 1 \cdot x_A + 4 \cdot x_B + 3 \cdot x_C + 9 \cdot x_D + 6 \cdot x_E && \text{(Values of selected objects)} \\ \text{subject to} \quad & 1 \cdot x_A + 2 \cdot x_B + 4 \cdot x_C + 5 \cdot x_D + 3 \cdot x_E \leq 5 && \text{(Capacity constraint)} \\ & x_A, x_B, x_C, x_D, x_E \in \{0, 1\} && \text{(Binary decision variables)} \end{aligned}$$

CSG Approaches based on Integer Linear Programming

[Rahwan et al., 2015]

- Given A and a set $\mathcal{S}$ of *coalitions* (i.e., subsets) of A, let M be a $|A| \times |\mathcal{S}|$ matrix
- $M_{iS} = 1$ if and only if agent $a \in A$ is part of coalition $S \in \mathcal{S}$, $M_{iS} = 0$ otherwise



$$M = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

CSG Approaches based on Integer Linear Programming

[RAHWAN et al., 2015]

Objective of Coalition Structure Generation

Compute the *partition* of A that *maximizes* the sum of the corresponding values

ILP Formulation for Coalition Structure Generation

maximize	$\sum_{S \in \mathcal{S}} v(S) \cdot x_S$	(Value of each selected coalition)
subject to	$\sum_{S \in \mathcal{S}} M_{iS} \cdot x_S = 1 \quad \forall i \in A$	(Each agent exactly in *one* coalition)

CSG Approaches based on Integer Linear Programming

[Rahwan et al., 2015]

Solving Integer Linear Programs

ILPs can be solved with state-of-the-art solvers like CPLEX (very mature technology)

Pros

Does not require any assumption on $v(\cdot)$ (very general approach)

Cons

- Memory requirements can become unmanageable for more than 20–30 agents
- Difficult to directly exploit the structure of the problem (i.e., graph)

CSG Approaches based on Integer Linear Programming

[Rahwan et al., 2015]

Solving Integer Linear Programs

ILPs can be solved with state-of-the-art solvers like CPLEX (very mature technology)

Pros

Does not require any assumption on $v(\cdot)$ (very general approach)

Cons

- Memory requirements can become unmanageable for more than 20–30 agents
- Difficult to directly exploit the structure of the problem (i.e., graph)

CSG Approaches based on Integer Linear Programming

[Rahwan et al., 2015]

Solving Integer Linear Programs

ILPs can be solved with state-of-the-art solvers like CPLEX (very mature technology)

Pros

Does not require any assumption on $v(\cdot)$ (very general approach)

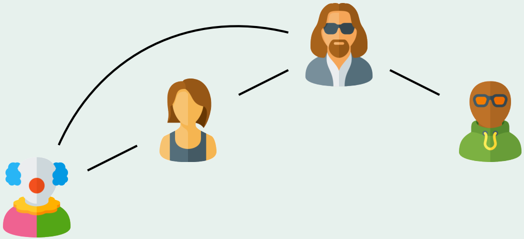
Cons

- Memory requirements can become unmanageable for more than 20–30 agents
- Difficult to directly exploit the structure of the problem (i.e., graph)

CSG as a COP

[BISTAFFA and FARINELLI, 2018]

Graph-Restricted CFG Example



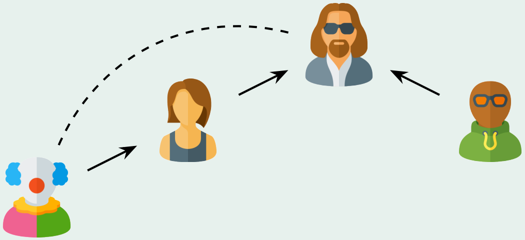
Pseudotree among Agents (Hierarchy)

Adjacent agents in the graph fall in the same branch of the tree (e.g., and)

CSG as a COP

[Bistaffa and Farinelli, 2018]

Graph-Restricted CFG Example



Pseudotree among Agents (Hierarchy)

Adjacent agents in the graph fall in the same branch of the tree (e.g.,  and )

CSG as a COP

[BISTAFFA and FARINELLI, 2018]

Challenge

How can we exploit the structure (i.e., hierarchy among agents)?

Main Idea

- Each coalition (i.e., decision variable) is “controlled” by the highest agent
- “Delegate” the formation of coalitions to descendants by means of *required* variables

CSG as a COP

[BISTAFFA and FARINELLI, 2018]

Challenge

How can we exploit the structure (i.e., hierarchy among agents)?

Main Idea

- Each coalition (i.e., decision variable) is “controlled” by the highest agent
- “Delegate” the formation of coalitions to descendants by means of *required* variables

CSG as a COP

[Bistaffa and Farinelli, 2018]

Challenge

How can we exploit the structure (i.e., hierarchy among agents)?

Main Idea

- Each coalition (i.e., decision variable) is "controlled" by the highest agent
- "Delegate" the formation of coalitions to descendants by means of *required* variables

CSG as a COP

[BISTAFFA and FARINELLI, 2018]

Challenge

How can we exploit the structure (i.e., hierarchy among agents)?

Main Idea

- Each coalition (i.e., decision variable) is “controlled” by the highest agent
- “Delegate” the formation of coalitions to descendants by means of *required* variables

CSG as a COP

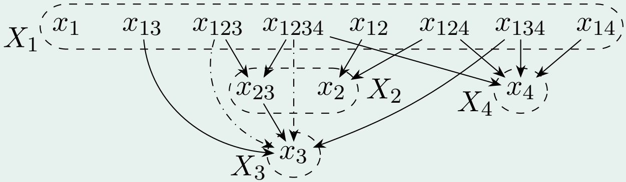
[BISTAFFA and FARINELLI, 2018]

Challenge

How can we exploit the structure (i.e., hierarchy among agents)?

Main Idea

- Each coalition (i.e., decision variable) is "controlled" by the highest agent
- "Delegate" the formation of coalitions to descendants by means of *required* variables



CSG as a COP

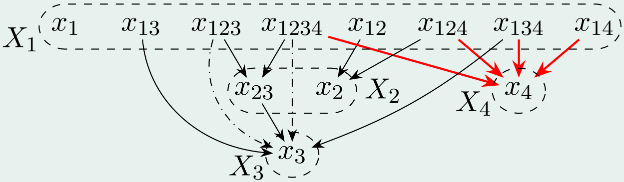
[Bistaffa and Farinelli, 2018]

Challenge

How can we exploit the structure (i.e., hierarchy among agents)?

Main Idea

- Each coalition (i.e., decision variable) is "controlled" by the highest agent
- "Delegate" the formation of coalitions to descendants by means of *required* variables



CSG as a COP

[Bistaffa and Farinelli, 2018]

Required Variables

- Any two variables that require the same variable *cannot* be enabled simultaneously
- As a result *no overlapping variables* are activated *at the same time*

Number of Constraints

- Naive COP: $\binom{\#\text{ coalitions}}{2}$
- This approach: linear *wrt* the number of agents

Open Question

Can we make this COP a Distributed COP (DCOP)?

CSG as a COP

[BISTAFFA and FARINELLI, 2018]

Required Variables

- Any two variables that require the same variable *cannot* be enabled simultaneously
- As a result *no overlapping variables* are activated *at the same time*

Number of Constraints

- Naive COP: $\binom{\text{\# coalitions}}{2}$
- This approach: linear *wrt* the number of agents

Open Question

Can we make this COP a Distributed COP (DCOP)?

CSG as a COP

[Bistaffa and Farinelli, 2018]

Required Variables

- Any two variables that require the same variable *cannot* be enabled simultaneously
- As a result *no overlapping variables* are activated *at the same time*

Number of Constraints

- Naive COP: $\binom{\#\text{ coalitions}}{2}$
- This approach: linear *wrt* the number of agents

Open Question

Can we make this COP a Distributed COP (DCOP)?

Today's Menu

Today's Menu

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	$\leftarrow$	60
AND	shutter_living_room	$\leftarrow$	0

SECP model

Example application to ambient intelligence scenario

- **Actuators**
  - *Decision* variable x_i, domain $\mathcal{D}_{x_i}$
  - Cost function $c_i : \mathcal{D}_{x_i} \to \mathbb{R}$
- **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 $\phi_j : \prod_{\varsigma \in \sigma(\phi_j)} \mathcal{D}_\varsigma \to \mathcal{D}_{y_j}$
- **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

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

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

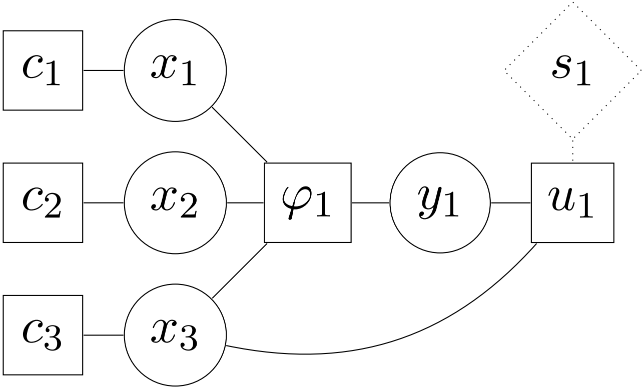
Mono-objective DCOP formulation

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

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

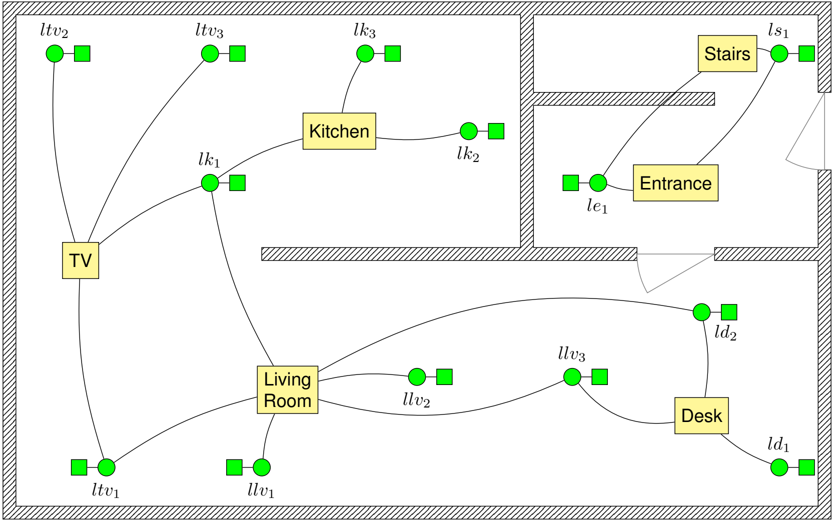
Formulating SECP as a DCOP

Representing a DCOP as a factor graph

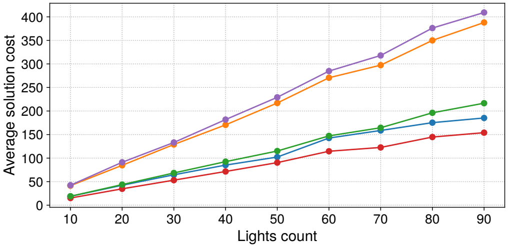


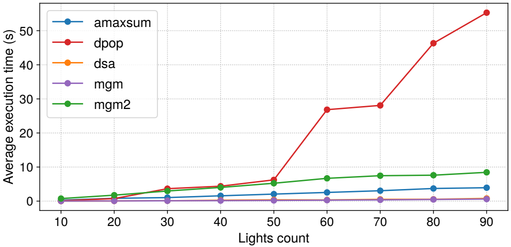
SECP Factor Graph

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]

Today's Menu

Observation Scheduling in Multi-Owner Constellations [Picard, 2022]

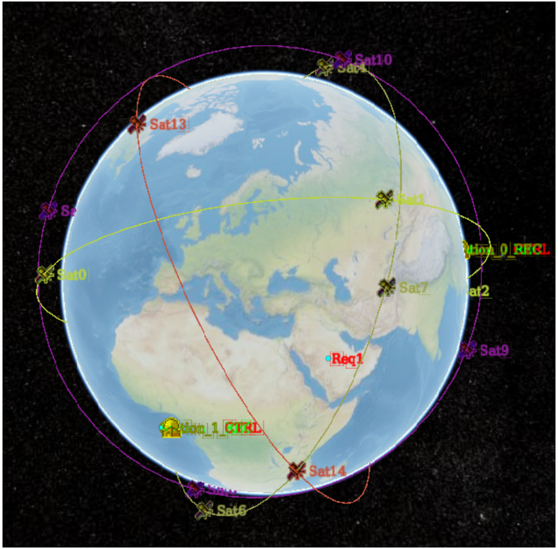
- Increasing size of deployed EOS constellations

⇒ 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)



Observation Scheduling in Multi-Owner Constellations [Picard, 2022]

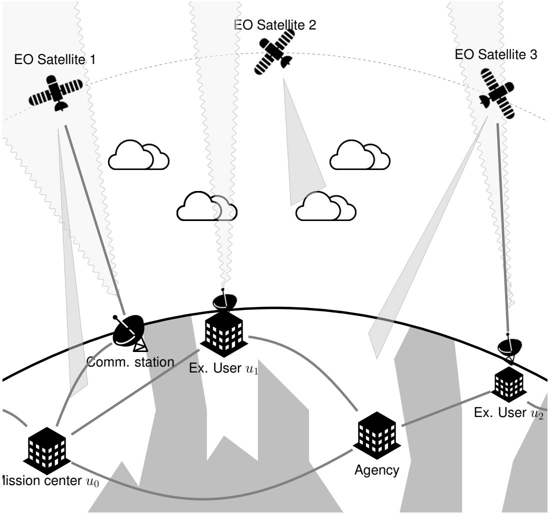
- Increasing size of deployed EOS constellations

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

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

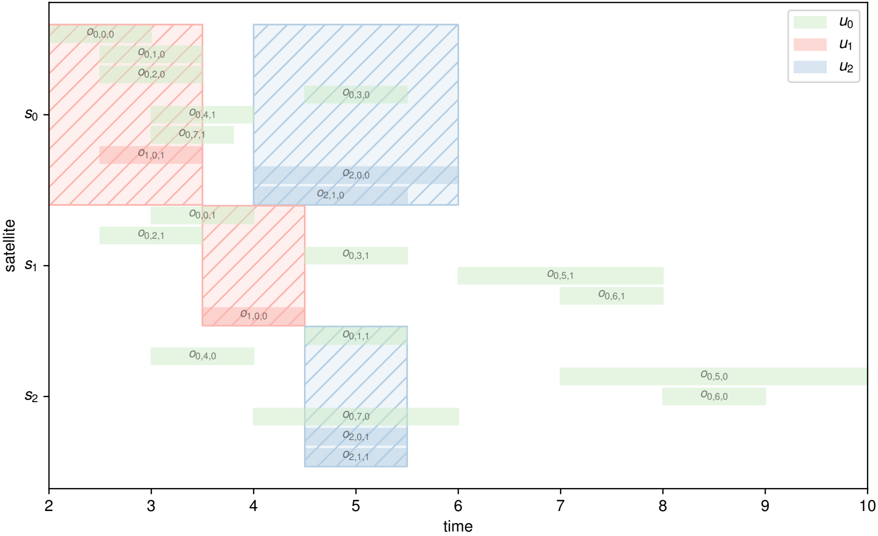
- We focus here on collective observation scheduling on a constellation where some users have exclusive access to some orbit portions

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



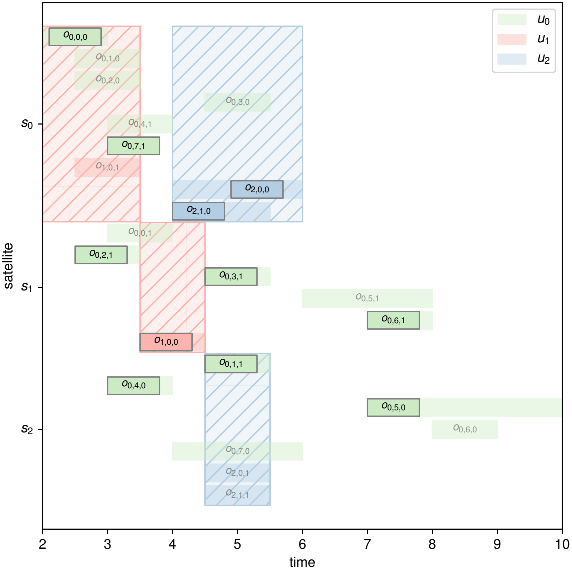
Scheduling Observations with Multiple Exclusive Orbit Portions

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

- 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

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:

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

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

with $\mathcal{O}[u]^r = \{o \in \theta r | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u,$ 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

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:

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

- 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:

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

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

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

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:

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

- 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 $\mathcal{O}[u]^r = \{o \in \theta r | \exists (s, (t_u^{\text{start}}, t_u^{\text{end}})) \in e_u, \text{ s.t. } s_o = s, [t_u^{\text{start}}, t_u^{\text{end}}] \cap [t_o^{\text{start}}, t_o^{\text{end}}] \neq \emptyset\}$ 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

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:

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

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

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)

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

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

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

- 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

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

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

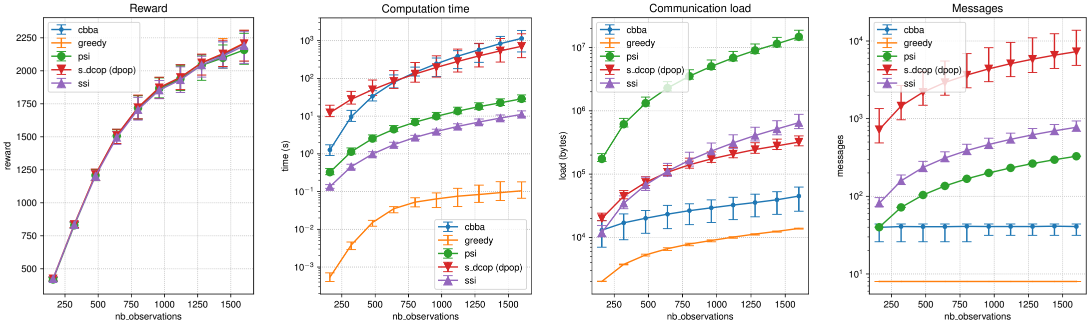
$$c(x_{e,o}) = \pi(o, \mathcal{M}_{u_o}), \quad \forall x_{e,o} \in \mathcal{X} \tag{7}$$

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

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

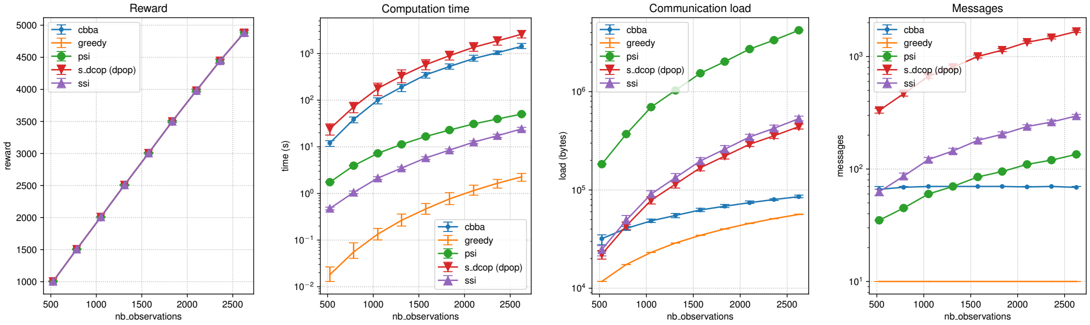
Highly conflicting randomly generated problems

5-min horizon with overlapping requests and limited capacity



Realistic randomly generated problems

6-hour horizon with numerous requests and large capacity



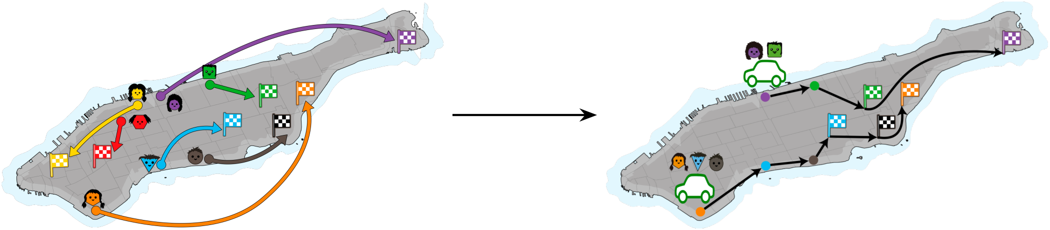
Today’s Menu

Shared Mobility as (Online) Coalition Structure Generation

[BISTAFFA et al., 2019]

What is Shared Mobility for Us?

Arrange *shared rides* (coalitions) among users that submit *real-time* requests, with the objective of *maximizing* a given *objective function*



Shared Mobility as (Online) Coalition Structure Generation

[ibid.]

Our Task

At each time step, arrange a (possibly empty) set of non-overlapping feasible cars among the requests currently active in the system

Our Objective Function

Maximize environmental benefits 🌳 and quality of service 🕑

Our Case Study [Bistaffa et al., 2019]

Densely populated areas (e.g., Manhattan) with request rate of 400 reqs/minute

Shared Mobility as (Online) Coalition Structure Generation

[ibid.]

Our Task

At each time step, arrange a (possibly empty) set of non-overlapping feasible cars among the requests currently active in the system

Our Objective Function

Maximize environmental benefits 🌳 and quality of service 🕑

Our Case Study [Bistaffa et al., 2019]

Densely populated areas (e.g., Manhattan) with request rate of 400 reqs/minute

Shared Mobility as (Online) Coalition Structure Generation

[ibid.]

Our Task

At each time step, arrange a (possibly empty) set of non-overlapping feasible cars among the requests currently active in the system

Our Objective Function

Maximize environmental benefits 🌳 and quality of service 🕑

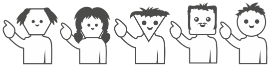
Our Case Study [Bistaffa et al., 2019]

Densely populated areas (e.g., Manhattan) with request rate of 400 reqs/minute

Input of the Online CSG Problem

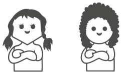
[BISTAFFA et al., 2019]

Incoming Requests

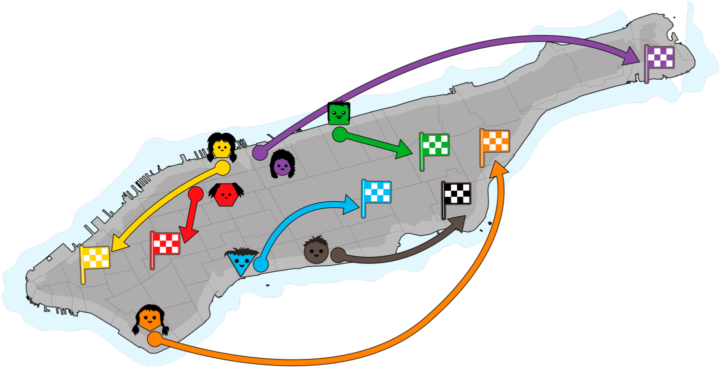


“I just issued a trip request”

Waiting Trip Requests



“I am waiting to share my ride”



Input of the Online CSG Problem

[BISTAFFA et al., 2019]

Example of a Shared Mobility Request

"I want to go from point i to point j, and I am willing to wait δ minutes to be picked up by somebody ($d =$ false) / before I leave with *my own car* ($d =$ true)"

- $r = \langle i, j, d, \delta \rangle$ (A request is a tuple r)
- $r \in R_t$ (The system receives a set R_t of requests at each time step t)
- $\langle R_1, \ldots, R_t, \ldots, R_h \rangle$ (Sequence of inputs over a time horizon h)
- The input sequence is *not known a priori* (Online optimization problem)

Input of the Online CSG Problem

[BISTAFFA et al., 2019]

Example of a Shared Mobility Request

"I want to go from point i to point j, and I am willing to wait δ minutes to be picked up by somebody ($d =$ false) / before I leave with *my own car* ($d =$ true)"

- $r = \langle i, j, d, \delta \rangle$ (A request is a tuple r)
- $r \in R_t$ (The system receives a set R_t of requests at each time step t)
- $\langle R_1, \ldots, R_t, \ldots, R_h \rangle$ (Sequence of inputs over a time horizon h)
- The input sequence is *not known a priori* (Online optimization problem)

Input of the Online CSG Problem

[Bistaffa et al., 2019]

Example of a Shared Mobility Request

"I want to go from point i to point j, and I am willing to wait δ minutes to be picked up by somebody ($d =$ false) / before I leave with *my own car* ($d =$ true)"

- $r = \langle i, j, d, \delta \rangle$ (A request is a tuple r)
- $r \in R_t$ (The system receives a set R_t of requests at each time step t)
- $\langle R_1, \ldots, R_t, \ldots, R_h \rangle$ (Sequence of inputs over a time horizon h)
- The input sequence is *not known a priori* (Online optimization problem)

Input of the Online CSG Problem

[BISTAFFA et al., 2019]

Example of a Shared Mobility Request

"I want to go from point i to point j, and I am willing to wait δ minutes to be picked up by somebody ($d =$ false) / before I leave with *my own car* ($d =$ true)"

- $r = \langle i, j, d, \delta \rangle$ (A request is a tuple r)
- $r \in R_t$ (The system receives a set R_t of requests at each time step t)
- $\langle R_1, \ldots, R_t, \ldots, R_h \rangle$ (Sequence of inputs over a time horizon h)
- The input sequence is *not known a priori* (Online optimization problem)

Input of the Online CSG Problem

[BISTAFFA et al., 2019]

Example of a Shared Mobility Request

"I want to go from point i to point j, and I am willing to wait δ minutes to be picked up by somebody ($d =$ false) / before I leave with *my own car* ($d =$ true)"

- $r = \langle i, j, d, \delta \rangle$ (A request is a tuple r)
- $r \in R_t$ (The system receives a set R_t of requests at each time step t)
- $\langle R_1, \ldots, R_t, \ldots, R_h \rangle$ (Sequence of inputs over a time horizon h)
- The input sequence is *not known a priori* (Online optimization problem)

Value $v(S)$ of a Coalition S

[BISTAFFA et al., 2019]

- The *value* (utility) of a coalition S is defined as:

$$v(S) = \overbrace{\rho_{\text{CO}_2} \cdot E_{\text{CO}_2}(S) + \rho_{\text{noise}} \cdot E_{\text{noise}}(S) + \rho_{\text{traffic}} \cdot E_{\text{traffic}}(S)}^{\text{environmental benefits}} + \overbrace{\rho_{\text{QoS}} \cdot Q(S)}^{\text{quality of service}}$$

- $|S| \leq k$ (Maximum cardinality constraint)

$$F(S) = |S| \leq k \, \wedge \, \ldots$$

- $\mathcal{F}(R) = \{S \in 2^R \mid F(S)\}$ (Set of feasible coalitions from a set R of requests)

Value $v(S)$ of a Coalition S

[BISTAFFA et al., 2019]

- The *value* (utility) of a coalition S is defined as:

$$v(S) = \overbrace{\rho_{\text{CO}_2} \cdot E_{\text{CO}_2}(S) + \rho_{\text{noise}} \cdot E_{\text{noise}}(S) + \rho_{\text{traffic}} \cdot E_{\text{traffic}}(S)}^{\text{environmental benefits}} + \overbrace{\rho_{\text{QoS}} \cdot Q(S)}^{\text{quality of service}}$$

- $|S| \leq k$ (Maximum cardinality constraint)

$$F(S) = |S| \leq k \;\wedge\; \ldots$$

- $\mathcal{F}(R) = \{S \in 2^R \mid F(S)\}$ (Set of feasible coalitions from a set R of requests)

Value $v(S)$ of a Coalition S

[BISTAFFA et al., 2019]

- The *value* (utility) of a coalition S is defined as:

$$v(S) = \overbrace{\rho_{\text{CO}_2} \cdot E_{\text{CO}_2}(S) + \rho_{\text{noise}} \cdot E_{\text{noise}}(S) + \rho_{\text{traffic}} \cdot E_{\text{traffic}}(S)}^{\text{environmental benefits}} + \overbrace{\rho_{\text{QoS}} \cdot Q(S)}^{\text{quality of service}}$$

- $|S| \leq k$ (Maximum cardinality constraint)

$$F(S) = |S| \leq k \;\wedge\; \ldots$$

- $\mathcal{F}(R) = \left\{S \in 2^R \;\middle|\; F(S)\right\}$ (Set of feasible coalitions from a set R of requests)

Curse of Dimensionality

[BISTAFFA et al., 2019]

- Recall that $\mathcal{F}(R) = \{S \in 2^R \mid F(S)\}$
- With $|S| \leq k$, $|\mathcal{F}(R)| \leq \sum_{i=1}^{k} \binom{|R|}{i}$, i.e., $\mathcal{O}(|R|^k)$ (Polynomial complexity)
- In practice, $|R_t|$ can be as high as 400 (Request rate in NY taxi dataset)

Scalability Problem

Enumerating all coalitions in $\mathcal{F}(R)$ is impractical, especially in realistic application scenarios with *very limited time budget* for the solution

Our Solution

Consider a restricted set $\hat{\mathcal{F}}(R)$ of *good candidate coalitions* instead of $\mathcal{F}(R)$

Curse of Dimensionality

[BISTAFFA et al., 2019]

- Recall that $\mathcal{F}(R) = \{S \in 2^R \mid F(S)\}$
- With $|S| \leq k$, $|\mathcal{F}(R)| \leq \sum_{i=1}^{k} \binom{|R|}{i}$, i.e., $\mathcal{O}(|R|^k)$ (Polynomial complexity)
- In practice, $|R_t|$ can be as high as 400 (Request rate in NY taxi dataset)

Scalability Problem

Enumerating all coalitions in $\mathcal{F}(R)$ is impractical, especially in realistic application scenarios with *very limited time budget* for the solution

Our Solution

Consider a restricted set $\hat{\mathcal{F}}(R)$ of *good candidate coalitions* instead of $\mathcal{F}(R)$

Curse of Dimensionality

[BISTAFFA et al., 2019]

- Recall that $\mathcal{F}(R) = \{S \in 2^R \mid F(S)\}$
- With $|S| \leq k$, $|\mathcal{F}(R)| \leq \sum_{i=1}^{k} \binom{|R|}{i}$, i.e., $\mathcal{O}(|R|^k)$ (Polynomial complexity)
- In practice, $|R_t|$ can be as high as 400 (Request rate in NY taxi dataset)

Scalability Problem

Enumerating all coalitions in $\mathcal{F}(R)$ is impractical, especially in realistic application scenarios with *very limited time budget* for the solution

Our Solution

Consider a restricted set $\hat{\mathcal{F}}(R)$ of *good candidate coalitions* instead of $\mathcal{F}(R)$

Curse of Dimensionality

[BISTAFFA et al., 2019]

- Recall that $\mathcal{F}(R) = \{S \in 2^R \mid F(S)\}$
- With $|S| \leq k$, $|\mathcal{F}(R)| \leq \sum_{i=1}^{k} \binom{|R|}{i}$, i.e., $\mathcal{O}(|R|^k)$ (Polynomial complexity)
- In practice, $|R_t|$ can be as high as 400 (Request rate in NY taxi dataset)

Scalability Problem

Enumerating all coalitions in $\mathcal{F}(R)$ is impractical, especially in realistic application scenarios with *very limited time budget* for the solution

Our Solution

Consider a restricted set $\hat{\mathcal{F}}(R)$ of *good candidate coalitions* instead of $\mathcal{F}(R)$

Curse of Dimensionality

[Bistaffa et al., 2019]

- Recall that $\mathcal{F}(R) = \{S \in 2^R \mid F(S)\}$
- With $|S| \leq k$, $|\mathcal{F}(R)| \leq \sum_{i=1}^{k} \binom{|R|}{i}$, i.e., $\mathcal{O}(|R|^k)$ (Polynomial complexity)
- In practice, $|R_t|$ can be as high as 400 (Request rate in NY taxi dataset)

Scalability Problem

Enumerating all coalitions in $\mathcal{F}(R)$ is impractical, especially in realistic application scenarios with *very limited time budget* for the solution

Our Solution

Consider a restricted set $\hat{\mathcal{F}}(R)$ of *good candidate coalitions* instead of $\mathcal{F}(R)$

Curse of Dimensionality

[BISTAFFA et al., 2019]

- Recall that $\mathcal{F}(R) = \left\{S \in 2^R \mid F(S)\right\}$
- With $|S| \leq k$, $|\mathcal{F}(R)| \leq \sum_{i=1}^{k} \binom{|R|}{i}$, i.e., $\mathcal{O}(|R|^k)$ (Polynomial complexity)
- In practice, $|R_t|$ can be as high as 400 (Request rate in NY taxi dataset)

Scalability Problem

Enumerating all coalitions in $\mathcal{F}(R)$ is impractical, especially in realistic application scenarios with *very limited time budget* for the solution

Our Solution

Consider a restricted set $\hat{\mathcal{F}}(R)$ of *good candidate coalitions* instead of $\mathcal{F}(R)$

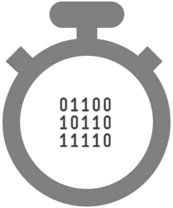
Generation of Good Candidate Coalitions (Step 1)

[BISTAFFA et al., 2019]

- CO_2 emissions
- Acoustic pollution
- Traffic congestion
- Quality of service

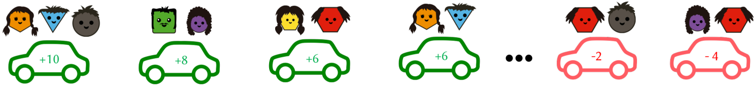
→

20 seconds



Probabilistic Greedy Algorithm

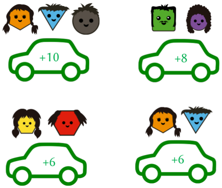
Candidate Cars



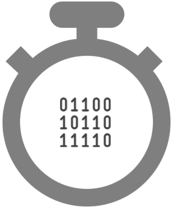
ILP Optimization (Step 2)

[BISTAFFA et al., 2019]

Good Candidates

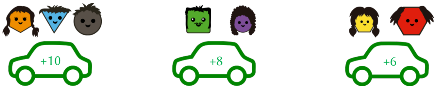


40 seconds



ILP
Solver

ILP Solution



Approximated ILP Formulation

[BISTAFFA et al., 2019]

$$\text{maximize} \quad \sum_{S \in \hat{\mathcal{F}}(\text{Pool})} v(S) \cdot x_S$$

(Only good candidates)

$$\text{such that} \quad x_S + x_{S'} \leq 1 \quad \forall\, \hat{\mathcal{F}}(\text{Pool}) : S \cap S' \neq \varnothing$$

Computational Advantage

Approximated ILP has a number of variables that is $< 0.01\%$ of the optimal ILP

Approximated ILP Formulation

[Bistaffa et al., 2019]

$$\text{maximize} \quad \sum_{S \in \hat{\mathcal{F}}(\text{Pool})} v(S) \cdot x_S \qquad \text{(Only good candidates)}$$

$$\text{such that} \quad x_S + x_{S'} \leq 1 \quad \forall\ \hat{\mathcal{F}}(\text{Pool}) : S \cap S' \neq \varnothing$$

Computational Advantage

Approximated ILP has a number of variables that is $< 0.01\%$ of the optimal ILP

Today's Menu

Collective Energy Purchasing

[FARINELLI et al., 2013]

Collective Energy Purchasing Scenario

- Each agent has an energy consumption profile
- Customers form coalitions to buy energy at reduced tariffs from two different markets:
  - *Spot market:* a short-term market intended for smaller amounts of energy
  - *Forward market:* a long-term market to buy more energy at cheaper prices

Collective Energy Purchasing

[FARINELLI et al., 2013]

Collective Energy Purchasing Scenario

- Each agent has an energy consumption profile
- Customers form coalitions to buy energy at reduced tariffs from two different markets:
  - *Spot market:* a short-term market intended for smaller amounts of energy
  - *Forward market:* a long-term market to buy more energy at cheaper prices

Collective Energy Purchasing

[FARINELLI et al., 2013]

Collective Energy Purchasing Scenario

- Each agent has an energy consumption profile
- Customers form coalitions to buy energy at reduced tariffs from two different markets:
  - *Spot market:* a short-term market intended for smaller amounts of energy
  - *Forward market:* a long-term market to buy more energy at cheaper prices

Collective Energy Purchasing

[FARINELLI et al., 2013]

Collective Energy Purchasing Scenario

- Each agent has an energy consumption profile
- Customers form coalitions to buy energy at reduced tariffs from two different markets:
  - *Spot market:* a short-term market intended for smaller amounts of energy
  - *Forward market:* a long-term market to buy more energy at cheaper prices

Collective Energy Purchasing

[FARINELLI et al., 2013]

Collective Energy Purchasing Scenario

- Each agent has an energy consumption profile
- Customers form coalitions to buy energy at reduced tariffs from two different markets:
  - *Spot market:* a short-term market intended for smaller amounts of energy
  - *Forward market:* a long-term market to buy more energy at cheaper prices

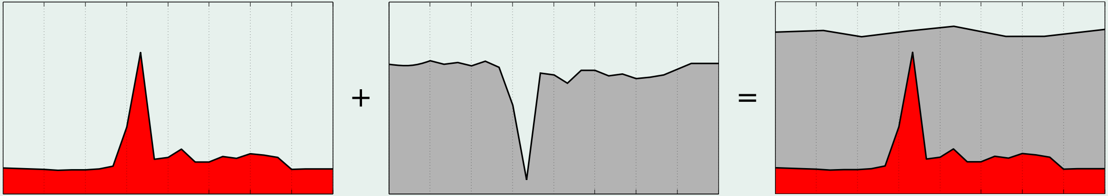
Collective Energy Purchasing

[FARINELLI et al., 2013]

Profile Merging

- Peaks in energy profiles require the use of *expensive*, *carbon-intensive*, peaking plant generators, resulting in higher consumers electricity bill
- A *flattened* profile results in a more efficient grid, with *lower carbon emissions* and *lower prices* for consumers

Example



Value $v(S)$ of a Coalition S

[BISTAFFA et al., 2017b]

- The *value* (utility) of a coalition S is defined as:

$$v(S) = \overbrace{\underbrace{\sum_{t=1}^{T} q_S^t(S) \cdot p_S}_{\text{Spot market}} + \underbrace{T \cdot q_F(S) \cdot p_F}_{\text{Forward market}}}^{\text{Purchased energy cost}} - \overbrace{\kappa(S)}^{\text{Coordination cost}}$$

- $q_S^t(S)$: energy purchased from spot market at time t
- $q_F(S)$: total energy purchased from forward market
- p_S: spot market energy price
- p_F: forward market energy price

Value $v(S)$ of a Coalition S

[BISTAFFA et al., 2017b]

- The *value* (utility) of a coalition S is defined as:

$$v(S) = \overbrace{\underbrace{\sum_{t=1}^{T} q_S^t(S) \cdot p_S}_{\text{Spot market}} + \underbrace{T \cdot q_F(S) \cdot p_F}_{\text{Forward market}}}^{\text{Superadditive } (v^+(S))} \overbrace{- \kappa(S)}^{\text{Subadditive } (v^-(S))}$$

- $q_S^t(S)$: energy purchased from spot market at time t
- $q_F(S)$: total energy purchased from forward market
- p_S: spot market energy price
- p_F: forward market energy price

$m + a$ Characteristic Functions

[BISTAFFA et al., 2017b]

$m + a$ Characteristic Function

- $m + a =$ Superadditive function $+$ subadditive function
  - Superadditive: $v(S_1 \cup S_2) > v(S_1) + (S_2)$
  - Subadditive: $v(S_1 \cup S_2) < v(S_1) + (S_2)$

Open Question

Is the characteristic function of shared mobility $m + a$?

$m + a$ Characteristic Functions

[Bistaffa et al., 2017b]

$m + a$ Characteristic Function

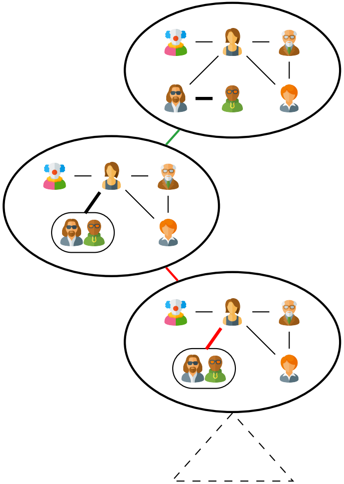
- $m + a =$ Superadditive function $+$ subadditive function
  - Superadditive: $v(S_1 \cup S_2) > v(S_1) + (S_2)$
  - Subadditive: $v(S_1 \cup S_2) < v(S_1) + (S_2)$

Open Question

Is the characteristic function of shared mobility $m + a$?

Branch-and-Bound for $m + a$ Characteristic Functions

[BISTAFFA et al., 2017b]



Can we Find an Upper Bound on $v(\cdot)$ in this Subtree?

Upper Bound M for $m + a$ Functions

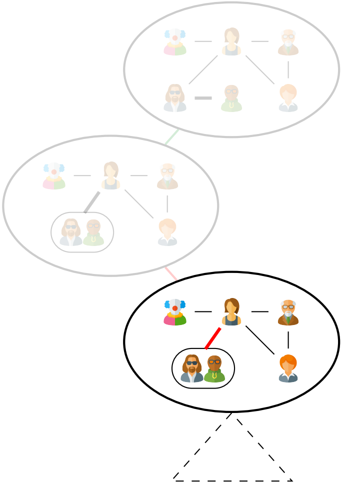
$$M = v^{+}(\textit{contract all edges}) + v^{-}(\textit{contract no edge})$$

Branch-and-Bound Algorithm

If M is $<$ than current best solution, do not visit this subtree

Branch-and-Bound for $m + a$ Characteristic Functions

[Bistaffa et al., 2017b]



Can we Find an Upper Bound on $v(\cdot)$ in this Subtree?

Upper Bound M for $m + a$ Functions

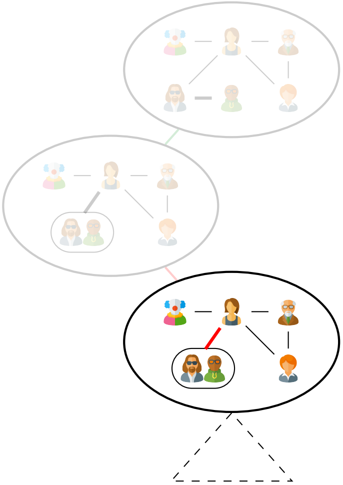
$M = v^+$(*contract all edges*) $+ v^-$(*contract no edge*)

Branch-and-Bound Algorithm

If M is $<$ than current best solution, do not visit this subtree

Branch-and-Bound for $m + a$ Characteristic Functions

[BISTAFFA et al., 2017b]



Can we Find an Upper Bound on $v(\cdot)$ in this Subtree?

Upper Bound M for $m + a$ Functions

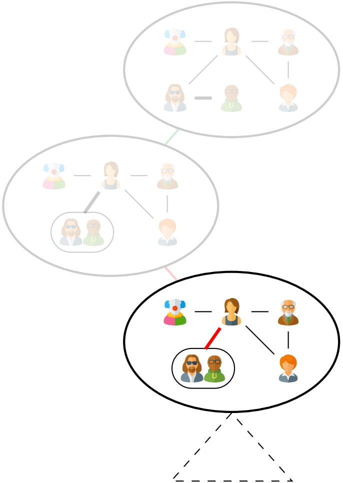
$$M = v^{+}(\textit{contract all edges}) + v^{-}(\textit{contract no edge})$$

Branch-and-Bound Algorithm

If M is $<$ than current best solution, do not visit this subtree

Branch-and-Bound for $m + a$ Characteristic Functions

[BISTAFFA et al., 2017b]



Can we Find an Upper Bound on $v(\cdot)$ in this Subtree?

Upper Bound M for $m + a$ Functions

$$M = v^{+}(\textit{contract all edges}) + v^{-}(\textit{contract no edge})$$

Branch-and-Bound Algorithm

If M is $<$ than current best solution, do not visit this subtree

Today's Menu

Programming and Evaluating DCOP Algorithms

Several libraries currently exist for the study of DCOP

- **AgentZero** is a Java-based library [LUTATI et al., 2014]
- **Frodo2** is actively developed[1] at École Polytechnique Fédérale de Lausanne (EPFL) [LÉAUTÉ et al., 2009]
- **DisChoco** is also Java-based and supports real distributed settings WAHBI et al., 2011, but discontinued since 2014
- **pyDCOP** is a python library developed by Orange Labs, focusing on Internet-of-Things and dynamic deployment of DCOPs [RUST et al., 2019]

[1] https://frodo-ai.tech/

Programming and Evaluating DCOP Algorithms

Several libraries currently exist for the study of DCOP

- **AgentZero** is a Java-based library [Lutati et al., 2014]
- **Frodo2** is actively developed[1] at École Polytechnique Fédérale de Lausanne (EPFL) [Léauté et al., 2009]
- **DisChoco** is also Java-based and supports real distributed settings Wahbi et al., 2011, but discontinued since 2014
- **pyDCOP** is a python library developed by Orange Labs, focusing on Internet-of-Things and dynamic deployment of DCOPs [Rust et al., 2019]

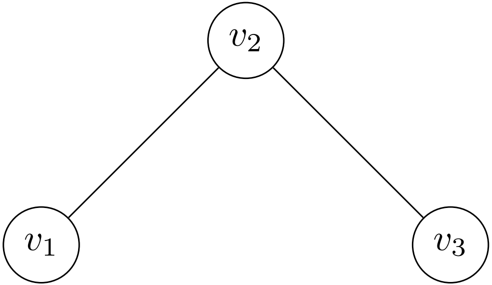
[1] https://frodo-ai.tech/

Hands on PyDCOP

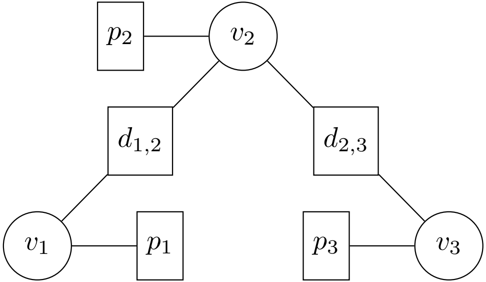
- Install VirtualBox
- Import the pyDCOP Virtual Machine (`http://bit.ly/pyDCOP`)
  - It's a Debian image with everything preinstalled:
  - python3, pyDCOP, matplotlib, glpk, etc.
- Alternatively, follow
  `https://pydcop.readthedocs.io/en/latest/installation.html`

1. `https://pydcop.readthedocs.io/en/latest/tutorials/getting_started.html`
2. `https://pydcop.readthedocs.io/en/latest/tutorials/analysing_results.html`

Graph Coloring



(a) constraints graph



(b) factor graph

- **Objective:** minimize
- **Domain:** 2 colors R and B
- **Variables:** V_1, V_2, V_3
- **Constraints:** neighbors must have different colors + preferences
- **Agents:** 3 agents

pyDCOP yaml format

graph_coloring.yaml

```yaml
name: graph coloring
objective: min

domains:
  colors:
    values: [R, G]

variables:
  v1:
    domain: colors
  v2:
    domain: colors
  v3:
    domain: colors
```

```yaml
constraints:
    pref_1:
      type: extensional
      variables: v1
      values:
        -0.1: R
        0.1: G
    pref_2:
      type: extensional
      variables: v2
      values:
        -0.1: G
        0.1: R
    pref_3:
      type: extensional
      variables: v3
      values:
        -0.1: G
        0.1: R
    diff_1_2:
      type: intention
      function: 10 if v1 == v2 else 0
    diff_2_3:
      type: intention
      function: 10 if v3 == v2 else 0

agents: [a1, a2, a3, a4, a5]
```

Solving the Graph Coloring DCOP

Command:

```
$ pydcop solve --algo dpop graph_coloring.yaml
```

Output:

```
 ...
"assignment": {
   "v1": "R",
   "v2": "G",
   "v3": "R"
 },
 "cost": -0.1,
 ...
```

With other algorithms:

```
$ pydcop --timeout 2 solve --algo dsa graph_coloring.yaml
$ pydcop solve --algo mgm --algo_params stop_cycle:20 \
        graph_coloring.yaml
```

Results

Full results :


```json
{
  "agt_metrics": {
    ...
  },
  "assignment": {
    "v1": "R",
    "v2": "G",
    "v3": "R"
  },
  "cost": -0.1,
  "cycle": 20,
  "msg_count": 158,
  "msg_size": 158,
  "status": "FINISHED",
  "time": 0.03201029699994251,
  "violation": 0
}
```

Logs

Simple:

use `-v 0..3`

```
$ pydcop -v 3 solve --algo dsa --algo_params stop_cycle:20 graph_coloring.
    yaml
```

Precise :

use `-log <log.conf>`

```
$ pydcop --log log.conf solve --algo dsa --algo_params stop_cycle:10
    graph_coloring.yaml
```

Now, look at `algo.log`

Run-time metrics

periodic: `"--collect_on period --period <p>"`

```
$ pydcop --log log.conf -t 10 solve \
    --collect_on period --period 1 --run_metric ./metrics.csv \
    --algo dsa graph_coloring.yaml
```

cycle: `"--collect_on cycle_change"`

Only supported with synchronous algorithms !

```
$ pydcop solve --algo mgm --algo_params stop_cycle:20 \
     --collect_on cycle_change --run_metric ./metrics.csv \
     graph_coloring_50.yaml
```

value: `"--collect_on value_change"`

```
$ pydcop -t 5 solve --algo mgm --collect_on value_change \
     --run_metric ./metrics_on_value.csv \
     graph_coloring_50.yaml
```

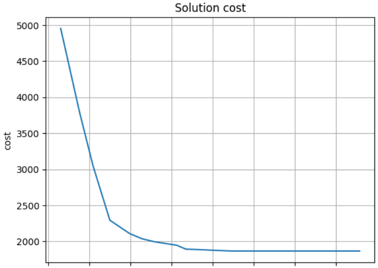
Run-time metrics

With a bigger graph coloring problem

```
$ pydcop solve --algo mgm --algo_params stop_cycle:20 \
               --collect_on cycle_change \
               --run_metric ./metrics.csv \
               graph_coloring_50.yaml
```

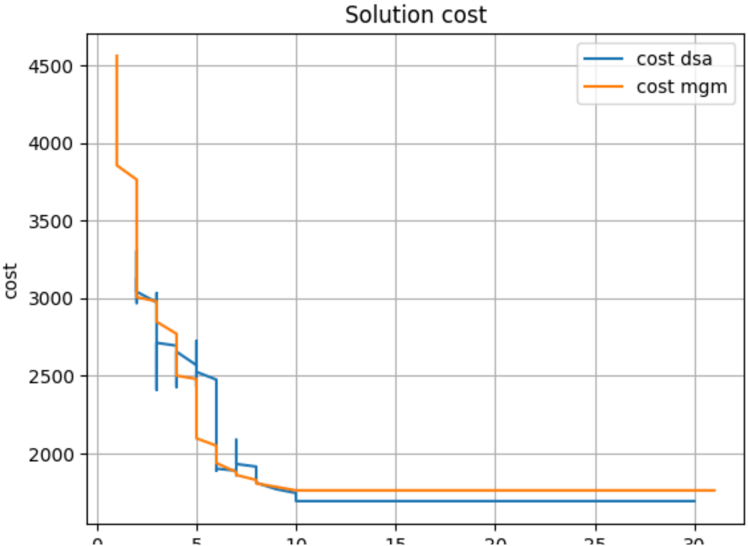
Plotting with matplotlib

```
$ python3 plot_cost.py ./metrics.csv
```



Run-time metrics

MGM (1720) and DSA (1647) , both with 30 cycles



Today's Menu

Conclusion and Wrap-up

What We've Seen Today

- 2 major multi-agent constraint optimization frameworks: **DCOP**, **CF**
  - DCOP: how to **collectively solve** constraint optimization problems
  - CF: how to **form coalitions/groups** with respect to some criteria and constraints
- Various **techniques and algorithms** to attach these problems
- Examples of **applications** in the transportation, IoT, space and energy domain

Conclusion and Wrap-up

Open questions

Distributed constraint optimization

- How to **decompose or regroup** as to reduce **interactions**?
- How to **structure** the system as to improve **parallelism**?
- How to deploy and make systems robust and resilient in **dynamic environments**?

Coalition formation

- Which other **realistic** scenarios can we model as $m + a$?
- Can we exploit some other **properties** for scenarios that are not $m + a$ (e.g., shared mobility)?
- More in general, how can we **improve** the **scalability** of CF approaches?

Common questions

- How to use DCOPs in CF and vice versa?
- Maintaining libraries and data sets

Special Thanks

Special thanks to all previous contributors to tutorials on multi-agent optimization and related topics, notably

Ferdinando Fioretto, Long Tran-Thanh, Pierre Rust, Enrico Pontelli, William Yeoh, Jesus Cerquides, Juan Antonio Rodriguez Aguilar, Alessandro Farinelli, Pedro Meseguer, Sarvapali Ramchurn, Amnon Meisels, Roie Zivan

References

BACCHUS, Fahiem and Peter VAN BEEK (1998). "On the Conversion between Non-Binary and Binary Constraint Satisfaction Problems". In: *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*. AAAI Press, pp. 311–318.

BISTAFFA, Filippo and Alessandro FARINELLI (2018). "A COP model for graph-constrained coalition formation". In: *Journal of Artificial Intelligence Research* 62, pp. 133–153.

BISTAFFA, Filippo, Alessandro FARINELLI, Georgios CHALKIADAKIS, and Sarvapali D. RAMCHURN (2017a). "A cooperative game-theoretic approach to the social ridesharing problem". In: *Artificial Intelligence* 246, pp. 86–117.

BISTAFFA, Filippo, Alessandro FARINELLI, Jesús CERQUIDES, Juan RODRÍGUEZ-AGUILAR, and Sarvapali D RAMCHURN (2017b). "Algorithms for graph-constrained coalition formation in the real world". In: *ACM Transactions on Intelligent Systems and Technology* 8.4, pp. 1–24.

BISTAFFA, Filippo, Christian BLUM, Jesús CERQUIDES, Alessandro FARINELLI, and Juan A RODRÍGUEZ-AGUILAR (2019). "A computational approach to quantify the benefits of ridesharing for policy makers and travellers". In: *IEEE Transactions on Intelligent Transportation Systems* 22.1, pp. 119–130.

CHALKIADAKIS, Georgios, Edith ELKIND, and Michael WOOLDRIDGE (2011). *Computational aspects of cooperative game theory*. Morgan & Claypool Publishers.

References (cont.)

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.

DECHTER, Rina (2003). *Constraint Processing*. Morgan Kaufmann.

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.

DEMANGE, Gabrielle (2004). "On group stability in hierarchies and networks". In: *Journal of Political Economy* 112.4, pp. 754–778.

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.

FARINELLI, Alessandro, Manuele BICEGO, Sarvapali RAMCHURN, and Mauro ZUCCHELLI (2013). "C-link: A hierarchical clustering approach to large-scale near-optimal coalition formation". In: *Twenty-third international joint conference on artificial intelligence*.

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.

References (cont.)

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.

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.

— (2005). "The distributed breakout algorithms". In: *Artificial Intelligence* 161.1-2, pp. 89–115. ISSN: 0004-3702.

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.

LÉAUTÉ, Thomas, Brammert OTTENS, and Radoslaw SZYMANEK (2009). "FRODO 2.0: An Open-Source Framework for Distributed Constraint Optimization". In: *Proceedings of the IJCAI'09 Distributed Constraint Reasoning Workshop (DCR'09)*. https://frodo-ai.tech. Pasadena, California, USA, pp. 160–164.

References (cont.)

LUTATI, Benny, Inna GONTMAKHER, Michael LANDO, Arnon NETZER, Amnon MEISELS, and Alon GRUBSHTEIN (2014). "AgentZero: A framework for simulating and evaluating multi-agent algorithms". In: *Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks*. Vol. 9783642544, pp. 309–327. ISBN: 9783642544323. DOI: 10.1007/978-3-642-54432-3_16.

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.

MAILLER, R. (2005). "Comparing two approaches to dynamic, distributed constraint satisfaction". In: *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*. ACM Press, pp. 1049–1056.

MAILLER, R. and V. LESSER (2004a). "Solving Distributed Constraint Optimization Problems Using Cooperative Mediation". In: *Proceedings of Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*. IEEE Computer Society, pp. 438–445.

— (2004b). "Using Cooperative Mediation to Solve Distributed Constraint Satisfaction Problems". In: *Proceedings of Third International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS'04)*. Vol. 1. IEEE Computer Society, pp. 446–453.

MAILLER, R. and V. R. LESSER (2006). "Asynchronous Partial Overlay: A New Algorithm for Solving Distributed Constraint Satisfaction Problems". In: *Journal of Artificial Intelligence Research* 25, pp. 529–576.

References (cont.)

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.

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.

MYERSON, Roger B (1977). "Graphs and cooperation in games". In: *Mathematics of operations research* 2.3, pp. 225–229.

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.

References (cont.)

- 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, G. and P. GLIZE (2006). "Model and Analysis of Local Decision Based on Cooperative Self-Organization for Problem Solving". In: *Multiagent and Grid Systems (MAGS)* 2.3, pp. 253–265.
- 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.
- RAHWAN, Talal, Tomasz P MICHALAK, Michael WOOLDRIDGE, and Nicholas R JENNINGS (2015). "Coalition structure generation: A survey". In: *Artificial Intelligence* 229, pp. 139–174.
- 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.
- RUSSEL, S. and P. NORVIG (1995). *Artificial Intelligence: a Modern Approach*. Prentice-Hall.
- 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/.

References (cont.)

RUST, Pierre, Gauthier PICARD, and Fano RAMPARANY (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.

— (2019). "pyDCOP, a DCOP library for IoT and dynamic systems". In: *International Workshop on Optimisation in Multi-Agent Systems (OptMAS@AAMAS 2019)*.

— (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.

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.

References (cont.)

Shehory, Onn and Sarit Kraus (May 1998). "Methods for Task Allocation via Agent Coalition Formation". In: *Artificial Intelligence* 101.1-2, pp. 165–200.

Shoham, Y. and K. Leyton-Brown (2008). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

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.

Stranders, R., F.M. Delle Fave, A. Rogers, and N.R. Jennings (2011). "U-GDL: A decentralised algorithm for DCOPs with Uncertainty". URL: `https://eprints.soton.ac.uk/273037/`.

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.

Wahbi, Mohamed, Redouane Ezzahir, Christian Bessiere, and El Houssine Bouyakhf (2011). "DisChoco 2: A Platform for Distributed Constraint Reasoning". In: *Proceedings of the IJCAI'11 workshop on Distributed Constraint Reasoning*. DCR'11. Barcelona, Catalonia, Spain, pp. 112–121. URL: `http://dischoco.sourceforge.net/`.

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. (1994). "Weak-Commitment Search for Solving Constraint Satisfaction Problems". In: *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*. AAAI Press, pp. 313–318.

— (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.

Hands on PyDCOP I

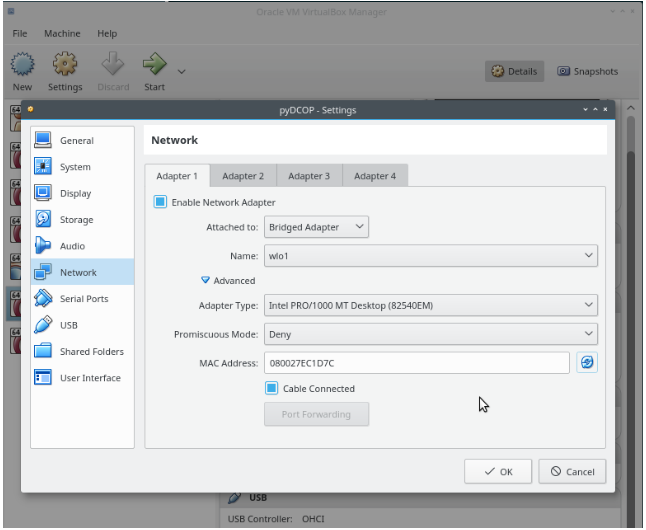
Virtual machine Setup

Before starting the VM:

- "Bridged adapter" mode
- Select wifi network adapter
- Reset MAC Address

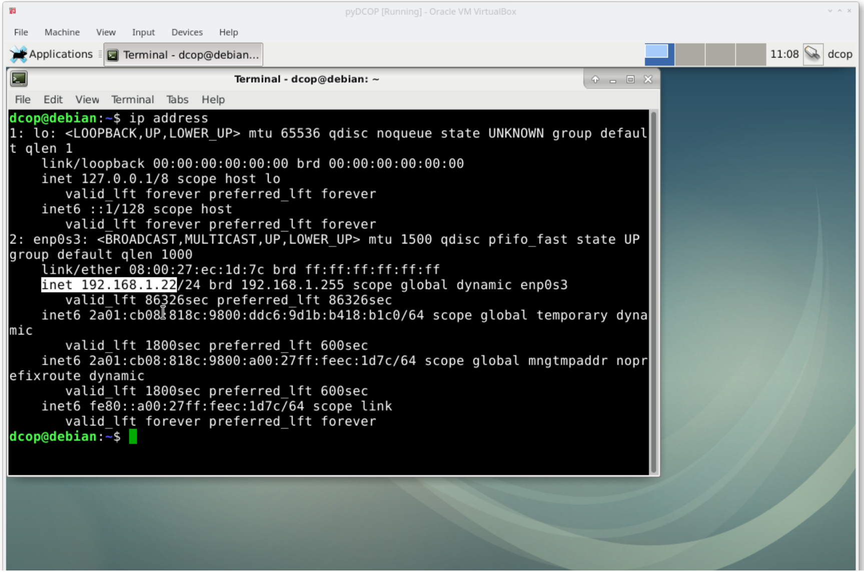
Then

- Start the VM
- login: `dcop / pyDCOP`
- Launch a terminal
- Note down the IP with `ip address`



Hands on PyDCOP I

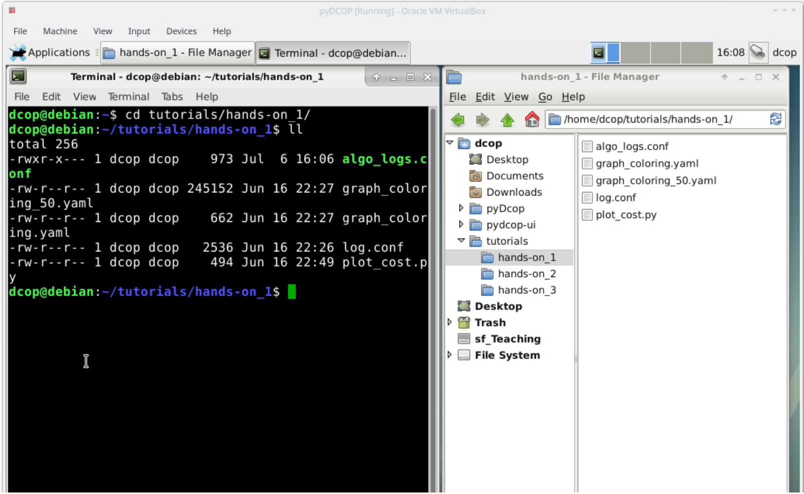
Virtual machine Setup



Hands on PyDCOP I

Files for the tutorials are in `/home/dcop/tutorials`.

```
$ cd /home/dcop/tutorials/hands-on_1
```



Hands on PyDCOP I

Web-ui

Web-base agent graphical interface:

- Run the web application

```
$ cd ~/pydcop-ui
$ python3 -m http.server
```

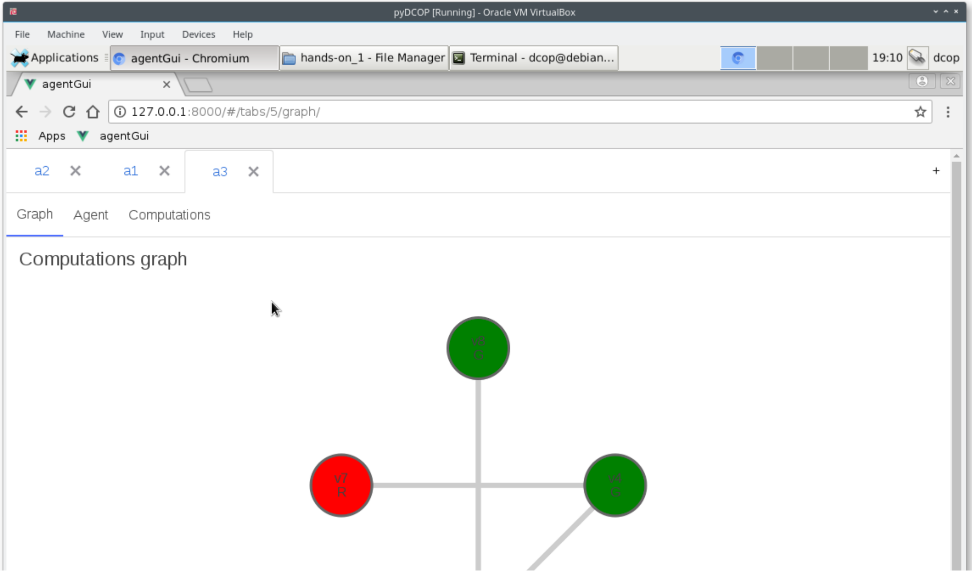
- Launch a browser on `http://127.0.0.1:8000`
- Solve the dcop with the option `--uiport <port>` (also, use `--delay <delay>`)

```
$ pydcop -v 3 solve -a mgm -d adhoc --delay 2 --uiport 10000
./graph_coloring_3agts_10vars.yaml
```

- Each agent exposes a web-socket, the web application connects to these websockets and display the agents' state.

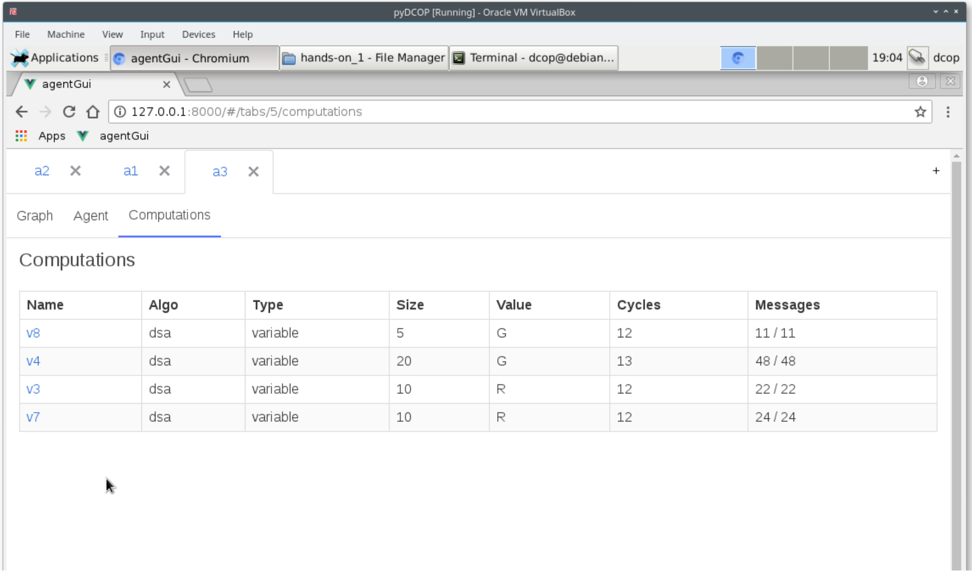
Hands on PyDCOP I

Web-ui



Hands on PyDCOP I

Web-ui



Name	Algo	Type	Size	Value	Cycles	Messages
v8	dsa	variable	5	G	12	11 / 11
v4	dsa	variable	20	G	13	48 / 48
v3	dsa	variable	10	R	12	22 / 22
v7	dsa	variable	10	R	12	24 / 24