DISTRIBUTED CONSTRAINTS SATISFACTION

BE4M36MAS - Multiagent systems

CENTRALIZED CASE

Constraint satisfaction problem

Find an assignment for variables that satisfy given constraints.

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ — set of *variables* to assign
- $\mathcal{D} = \{D_1, \ldots, D_n\}$ — set of *domains* ($x_i \in D_i$)
- $\mathcal{C} = \{C_1, \ldots, C_m\}$ — set of *constraints*

$C_i \subseteq D_{i_1} \times \cdots \times D_{i_r}$ denotes a r-ary constraint over variables $x_{i_1}, \ldots, x_{i_r}$

Constraint satisfaction problem

Solution: n-tuple $(d_1, \cdots, d_n)$, such that:

- $d_i \in D_i$, for $1 \leq i \leq n$
- $(d_{i_1}, \ldots, d_{i_r}) \in C_k$ for every constraint $C_k \subseteq D_{i_1} \times \cdots \times D_{i_r}$

Centralized algorithm

Synchronized backtracking

$v_i \leftarrow$ value from D_i consistent with $(v_1, \ldots, v_{i-1})$;
if *No such* v_i *exists* **then**
    backtrack ;
else if $i = n$ **then**
    stop ;
else
    `ChooseValue`$(x_{i+1}, (v_1, \ldots, v_i))$;
end

Algorithm 1: `ChooseValue`$(x_i, (v_1, \ldots, v_{i-1}))$

Centralized algorithm

Enhancements?

- AC-3 algorithm? (arc consistency - e.g. combinatorial optimization)

DISTRIBUTED CASE

Distributed case

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ — set of *variables* to assign
- $\mathcal{D} = \{D_1, \ldots, D_n\}$ — set of *domains* ($x_i \in D_i$)
- $\mathcal{C} = \{C_1, \ldots, C_m\}$ — set of *constraints*
- $\mathcal{A} = \{A_1, \ldots, A_k\}$ — set of *agents*

Every variable **must be** assigned to one of the agents.

$\rightarrow$ otherwise the DCSP problem is **not fully defined**

Distributed case

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ — set of *variables* to assign
- $\mathcal{D} = \{D_1, \ldots, D_n\}$ — set of *domains* ($x_i \in D_i$)
- $\mathcal{C} = \{C_1, \ldots, C_m\}$ — set of *constraints*
- $\mathcal{A} = \{A_1, \ldots, A_k\}$ — set of *agents*

Every variable **must be** assigned to one of the agents.

$\rightarrow$ otherwise the DCSP problem is **not fully defined**

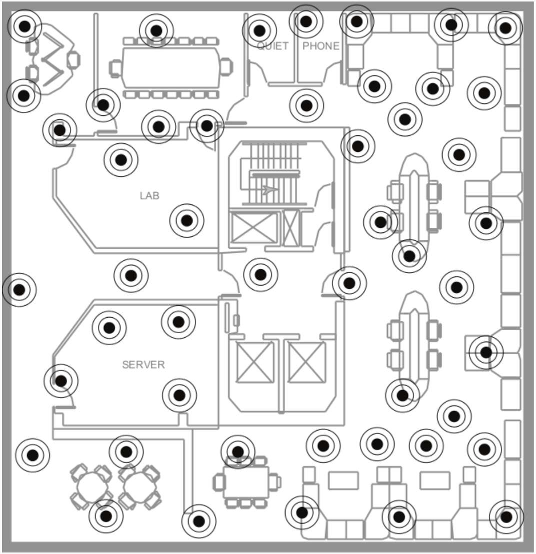
Distributed case

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ — set of *variables* to assign
- $\mathcal{D} = \{D_1, \ldots, D_n\}$ — set of *domains* ($x_i \in D_i$)
- $\mathcal{C} = \{C_1, \ldots, C_m\}$ — set of *constraints*
- $\mathcal{A} = \{A_1, \ldots, A_k\}$ — set of *agents*

Every variable **must be** assigned to one of the agents.

$\rightarrow$ otherwise the DCSP problem is **not fully defined**

Example



ASYNCHRONOUS BACKTRACKING

What do we assume?

- Every agent controls a single variable
- Constraints are binary

- Messages are delivered in a finite time (but this time may vary randomly)
- Messages from a single agent are delivered in the order they were sent

  $\rightarrow$ imagine an unreliable TCP/IP network

What does an agent initially knows?

- Total ordering of agents (priorities)
- Constraints he is involved in
- Domain of a variable controlled by himself

Data structures

- Current assignment
- Set of outgoing links ($\sim$ who needs to know my assignment)
- Set of incoming links ($\sim$ who will notify me about his assignment)
- **Agent view** — agent's idea about current assignment of other agents

  $\rightarrow$ May be out of sync!
- **Nogood store** — justification of forbidden values in the domain

Data structures

- Current assignment
- Set of outgoing links ($\sim$ who needs to know my assignment)
- Set of incoming links ($\sim$ who will notify me about his assignment)
- **Agent view** — agent's idea about current assignment of other agents

  $\rightarrow$ May be out of sync!
- **Nogood store** — justification of forbidden values in the domain

Data structures

- Current assignment
- Set of outgoing links ($\sim$ who needs to know my assignment)
- Set of incoming links ($\sim$ who will notify me about his assignment)
- **Agent view** — agent's idea about current assignment of other agents

  $\rightarrow$ May be out of sync!
- **Nogood store** — justification of forbidden values in the domain

Data structures

- Current assignment
- Set of outgoing links ($\sim$ who needs to know my assignment)
- Set of incoming links ($\sim$ who will notify me about his assignment)
- **Agent view** — agent's idea about current assignment of other agents

  $\rightarrow$ May be out of sync!
- **Nogood store** — justification of forbidden values in the domain

Data structures

- Current assignment
- Set of outgoing links ($\sim$ who needs to know my assignment)
- Set of incoming links ($\sim$ who will notify me about his assignment)
- **Agent view** — agent's idea about current assignment of other agents

  $\rightarrow$ May be out of sync!
- **Nogood store** — justification of forbidden values in the domain

Minimalistic example — meeting scheduling

- John needs to arrange a meeting with Bob and Alice
- As all agents, he is a busy guy — both meetings must happen in a single day
- Bob doesn't know about Alice's meeting and vice versa

Minimalistic example — meeting scheduling

$\mathcal{A} = \{\text{Alice}, \text{Bob}, \text{John}\}$

$\mathcal{X} = \{x_{\text{Alice}}, x_{\text{Bob}}, x_{\text{John}}\}$
Agent i controls variable x_i.

$\mathcal{D} = \{D_{\text{Alice}}, D_{\text{Bob}}, D_{\text{John}}\}$
$D_{\text{Alice}} = \{\text{Mon}, \text{Thu}\}$
$D_{\text{Bob}} = \{\text{Tue}, \text{Thu}\}$
$D_{\text{John}} = \{\text{Mon}, \text{Tue}, \text{Thu}\}$

$\mathcal{C} = \{x_{\text{Bob}} = x_{\text{John}}, x_{\text{Alice}} = x_{\text{John}}\}$

Minimalistic example — meeting scheduling

$\mathcal{A} = \{\text{Alice}, \text{Bob}, \text{John}\}$

$\mathcal{X} = \{x_{\text{Alice}}, x_{\text{Bob}}, x_{\text{John}}\}$
Agent i controls variable x_i.

$\mathcal{D} = \{D_{\text{Alice}}, D_{\text{Bob}}, D_{\text{John}}\}$
$D_{\text{Alice}} = \{\text{Mon}, \text{Thu}\}$
$D_{\text{Bob}} = \{\text{Tue}, \text{Thu}\}$
$D_{\text{John}} = \{\text{Mon}, \text{Tue}, \text{Thu}\}$

$\mathcal{C} = \{x_{\text{Bob}} = x_{\text{John}}, x_{\text{Alice}} = x_{\text{John}}\}$

Minimalistic example — meeting scheduling

$\mathcal{A} = \{\text{Alice}, \text{Bob}, \text{John}\}$

$\mathcal{X} = \{x_{\text{Alice}}, x_{\text{Bob}}, x_{\text{John}}\}$
Agent i controls variable x_i.

$\mathcal{D} = \{D_{\text{Alice}}, D_{\text{Bob}}, D_{\text{John}}\}$
$D_{\text{Alice}} = \{\text{Mon}, \text{Thu}\}$
$D_{\text{Bob}} = \{\text{Tue}, \text{Thu}\}$
$D_{\text{John}} = \{\text{Mon}, \text{Tue}, \text{Thu}\}$

$\mathcal{C} = \{x_{\text{Bob}} = x_{\text{John}}, x_{\text{Alice}} = x_{\text{John}}\}$

Minimalistic example — meeting scheduling

$\mathcal{A} = \{\text{Alice}, \text{Bob}, \text{John}\}$

$\mathcal{X} = \{x_{\text{Alice}}, x_{\text{Bob}}, x_{\text{John}}\}$
Agent i controls variable x_i.

$\mathcal{D} = \{D_{\text{Alice}}, D_{\text{Bob}}, D_{\text{John}}\}$
$D_{\text{Alice}} = \{\text{Mon}, \text{Thu}\}$
$D_{\text{Bob}} = \{\text{Tue}, \text{Thu}\}$
$D_{\text{John}} = \{\text{Mon}, \text{Tue}, \text{Thu}\}$

$\mathcal{C} = \{x_{\text{Bob}} = x_{\text{John}}, x_{\text{Alice}} = x_{\text{John}}\}$

Minimalistic example — meeting scheduling

$\mathcal{A} = \{\text{Alice}, \text{Bob}, \text{John}\}$

$\mathcal{X} = \{x_{\text{Alice}}, x_{\text{Bob}}, x_{\text{John}}\}$
Agent i controls variable x_i.

$\mathcal{D} = \{D_{\text{Alice}}, D_{\text{Bob}}, D_{\text{John}}\}$
$D_{\text{Alice}} = \{\text{Mon}, \text{Thu}\}$
$D_{\text{Bob}} = \{\text{Tue}, \text{Thu}\}$
$D_{\text{John}} = \{\text{Mon}, \text{Tue}, \text{Thu}\}$

$\mathcal{C} = \{x_{\text{Bob}} = x_{\text{John}}, x_{\text{Alice}} = x_{\text{John}}\}$

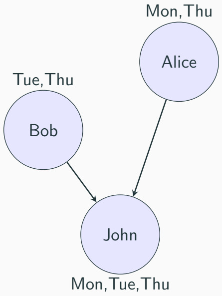
Minimalistic example — meeting scheduling

$\mathcal{A} = \{\text{Alice}, \text{Bob}, \text{John}\}$

$\mathcal{X} = \{x_{\text{Alice}}, x_{\text{Bob}}, x_{\text{John}}\}$
Agent i controls variable x_i.

$\mathcal{D} = \{D_{\text{Alice}}, D_{\text{Bob}}, D_{\text{John}}\}$
$D_{\text{Alice}} = \{\text{Mon}, \text{Thu}\}$
$D_{\text{Bob}} = \{\text{Tue}, \text{Thu}\}$
$D_{\text{John}} = \{\text{Mon}, \text{Tue}, \text{Thu}\}$

$\mathcal{C} = \{x_{\text{Bob}} = x_{\text{John}}, x_{\text{Alice}} = x_{\text{John}}\}$



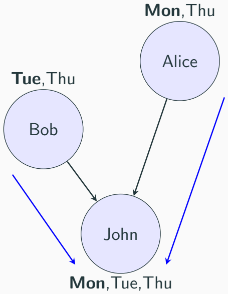
Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\varnothing$ John: $\varnothing$

Let's all propose a date and see what happens!

Bob $\rightarrow$ John:
Ok?(Bob $\rightarrow$ Tue)

Alice $\rightarrow$ John:
Ok?(Alice $\rightarrow$ Mon)



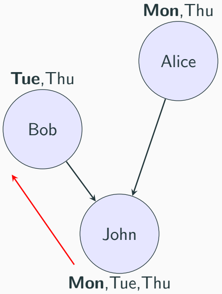
Alice: $\varnothing$ Bob: $\varnothing$ John: $\{\text{Alice} \rightarrow \text{Mon}, \text{Bob} \rightarrow \text{Tue}\}$

Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\varnothing$ John: $\{\text{Alice} \to \text{Mon}, \text{Bob} \to \text{Tue}\}$

John: Argh, I wanted to have both meetings in one day :-(Let's make them change their minds...

John $\to$ Bob:
$\text{Nogood}(\{\text{Bob} \to \text{Tue}, \text{Alice} \to \text{Mon}\})$



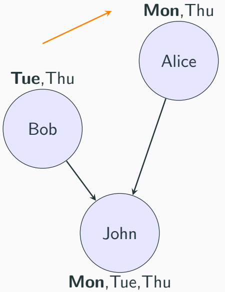
Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}\}$

Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}\}$

Bob: Who is that Alice? I've never heard of her.

Bob $\to$ Alice:
AddLink(Alice $\to$ Bob)



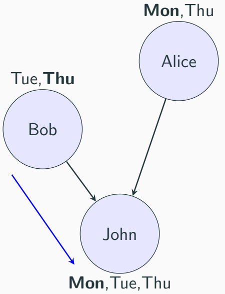
Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}\}$

Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}\}$

Bob: John told me that the meeting cannot happen on Tuesday if Alice opts for Monday. Let's try Thursday then...

Bob $\to$ John:
Ok?(Bob $\to$ Thu)



Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}, \text{Bob} \to \text{Thu}\}$

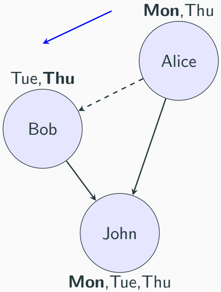
Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}, \text{Bob} \to \text{Thu}\}$

Alice: Bob, why are you so curious?

Alice $\to$ Bob:

Ok?(Alice $\to$ Mon)



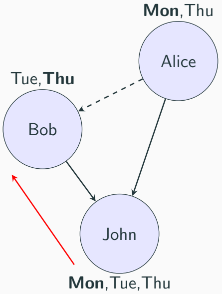
Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}, \text{Bob} \to \text{Thu}\}$

Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}, \text{Bob} \to \text{Thu}\}$

John: They tried it again. Alright, one more try...

John $\to$ Bob:
Nogood($\{\text{Bob} \to \text{Thu}, \text{Alice} \to \text{Mon}\}$)



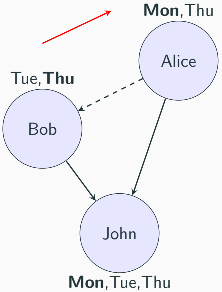
Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}\}$

Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Mon}\}$ John: $\{\text{Alice} \to \text{Mon}\}$

Bob: I have run out of options. It's up to Alice now...

Bob $\to$ Alice:
Nogood($\{\text{Alice} \to \text{Mon}\}$)



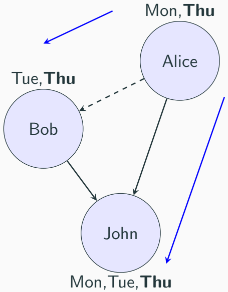
Alice: $\varnothing$ Bob: $\varnothing$ John: $\{\text{Alice} \to \text{Mon}\}$

Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\varnothing$ John: $\{\text{Alice} \to \text{Mon}\}$

Alice: I have one more option, let's try Thursday.

Alice $\to$ Bob, John:
$\color{blue}{\text{Ok?}(\{\text{Alice} \to \text{Thu}\})}$

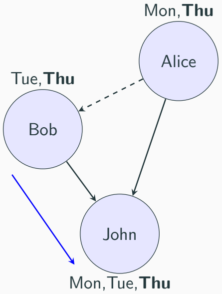


Alice: $\varnothing$ Bob: $\{\text{Alice} \to \text{Thu}\}$ John: $\{\text{Alice} \to \text{Thu}\}$

Minimalistic example — meeting scheduling

Alice: $\varnothing$ Bob: $\{\text{Alice} \rightarrow \text{Thu}\}$ John: $\{\text{Alice} \rightarrow \text{Thu}\}$

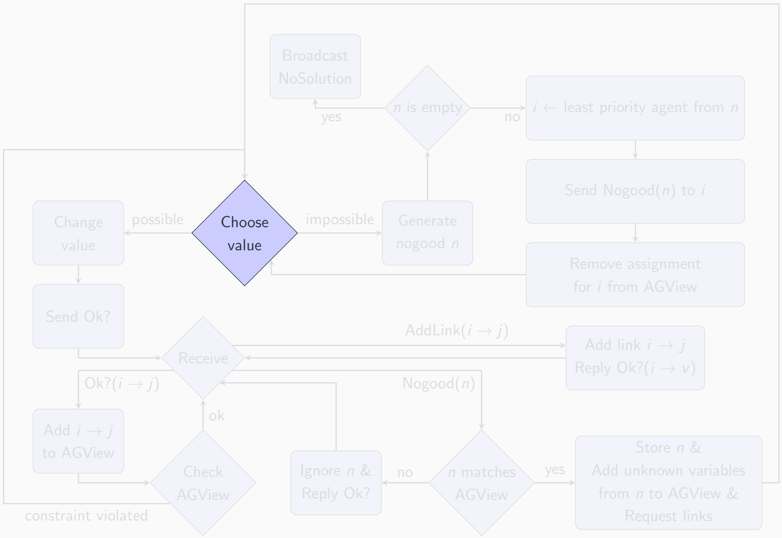
John: Finally. Thursday seems like a viable option.



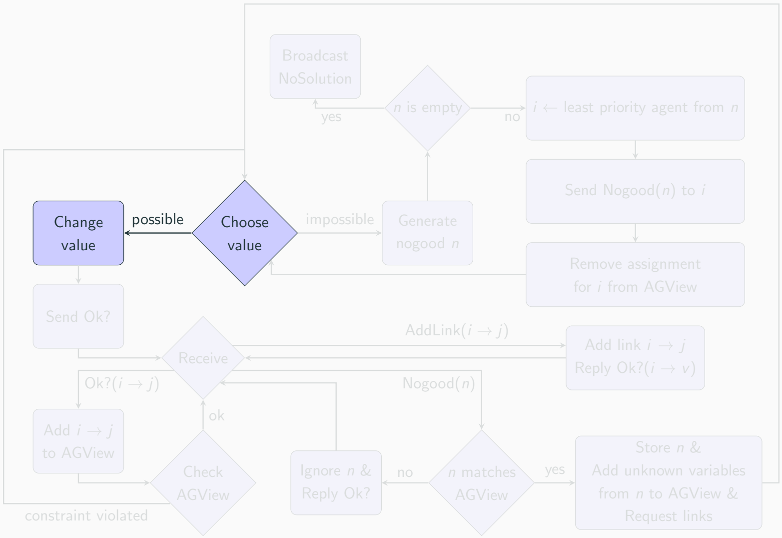
Alice: $\varnothing$ Bob: $\{\text{Alice} \rightarrow \text{Thu}\}$ John: $\{\text{Alice} \rightarrow \text{Thu}, \text{Bob} \rightarrow \text{Thu}\}$

Asynchronous backtracking

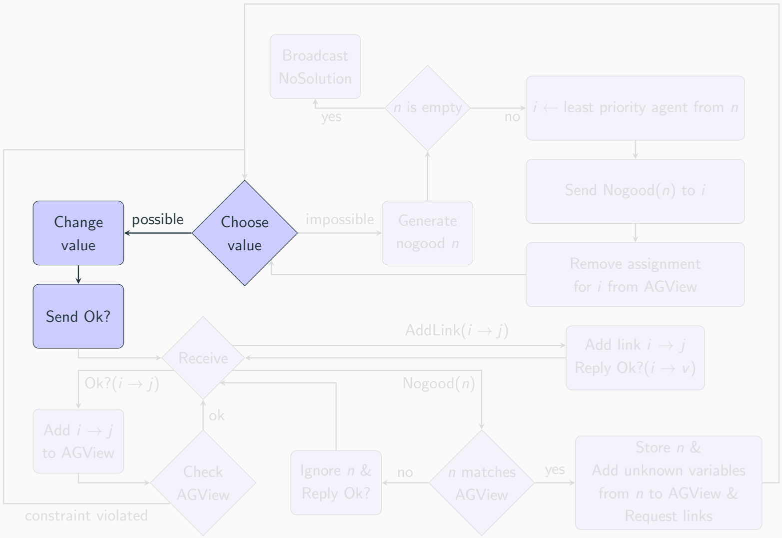
Asynchronous backtracking



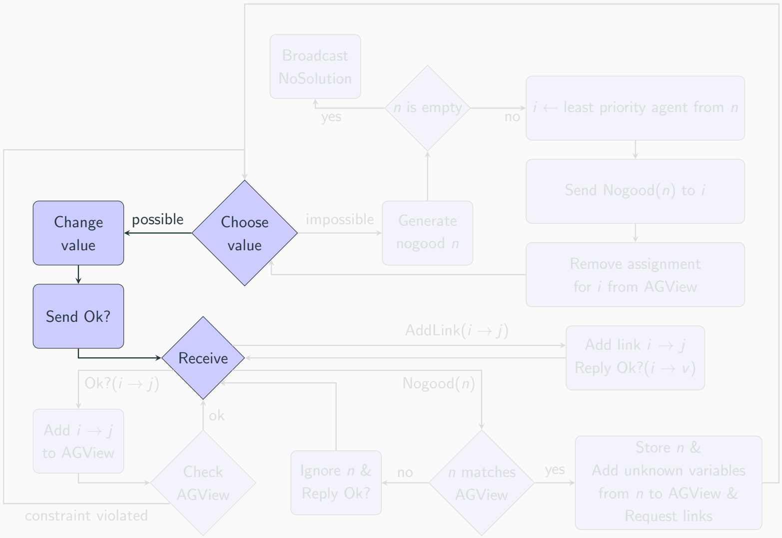
Asynchronous backtracking



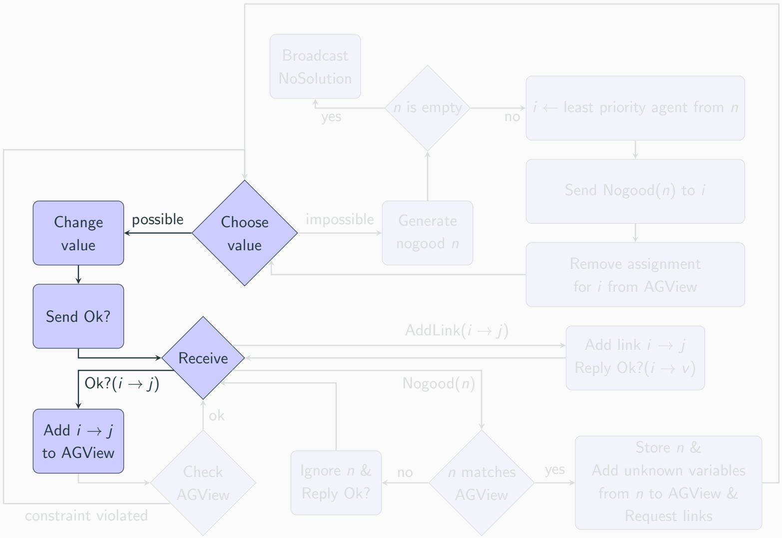
Asynchronous backtracking



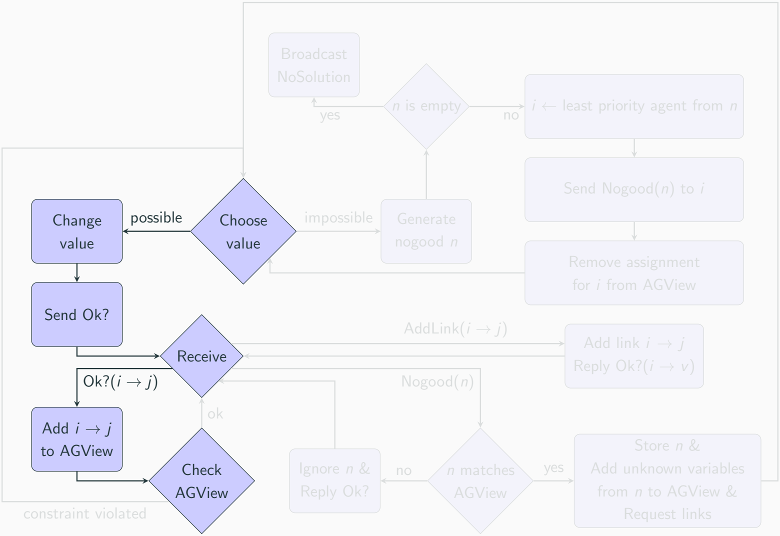
Asynchronous backtracking



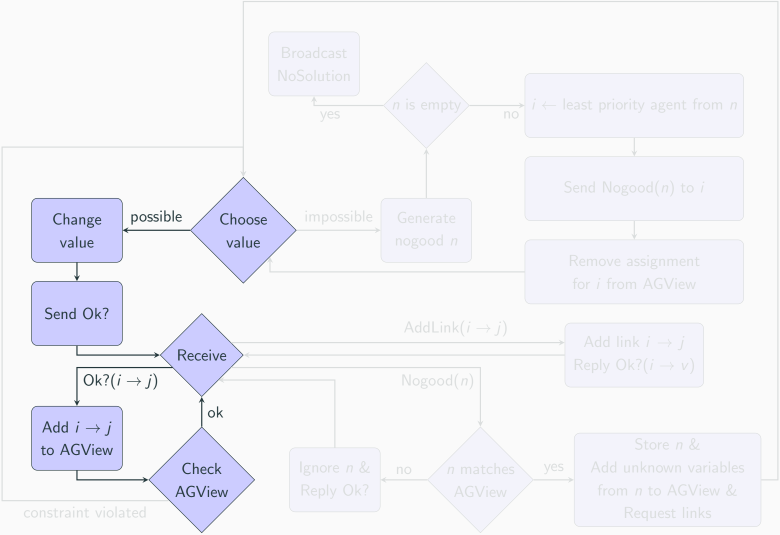
Asynchronous backtracking



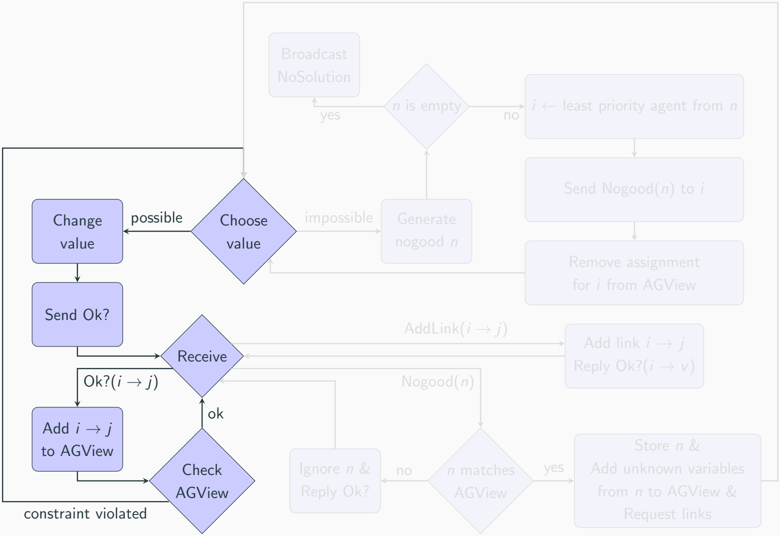
Asynchronous backtracking



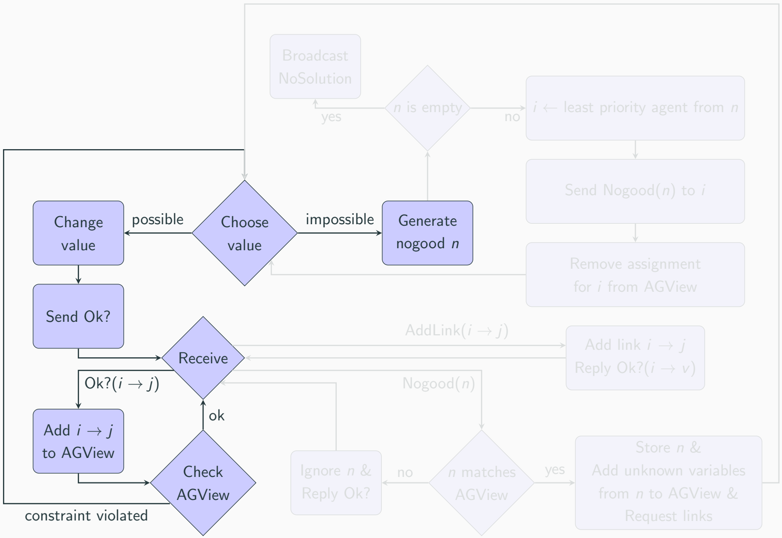
Asynchronous backtracking



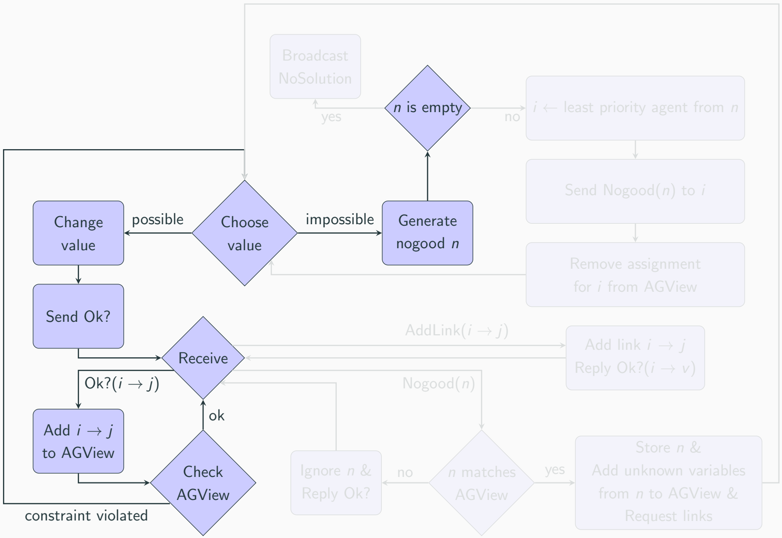
Asynchronous backtracking



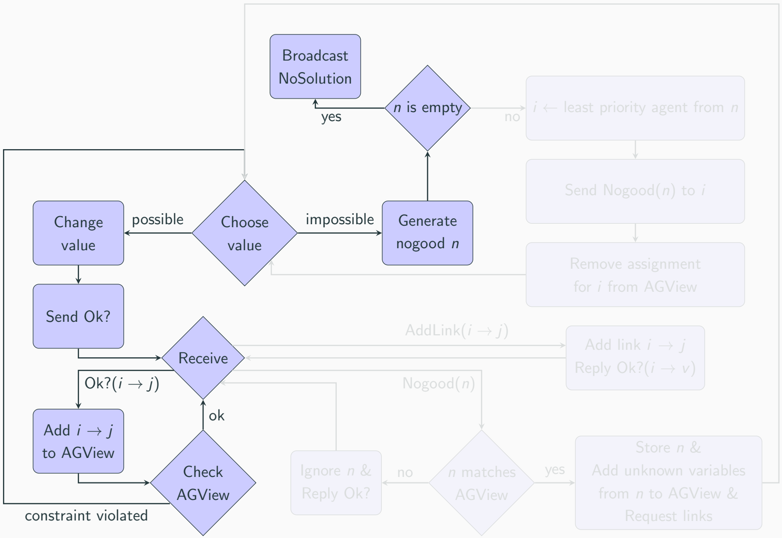
Asynchronous backtracking



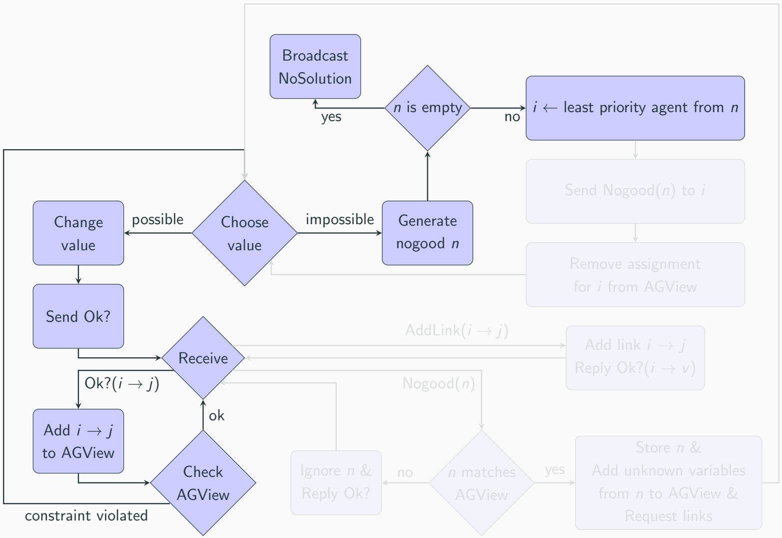
Asynchronous backtracking



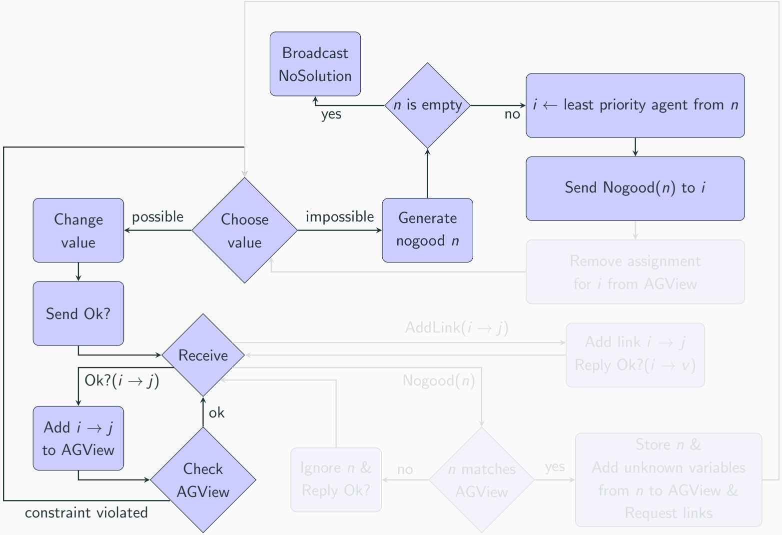
Asynchronous backtracking



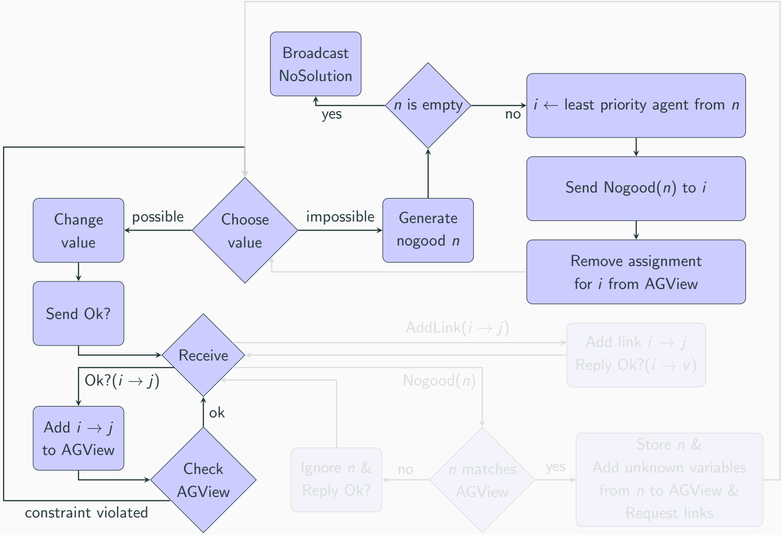
Asynchronous backtracking



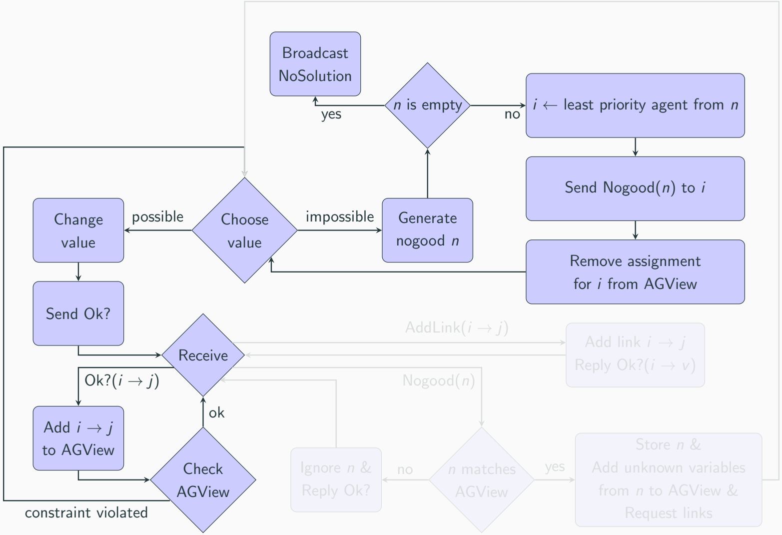
Asynchronous backtracking



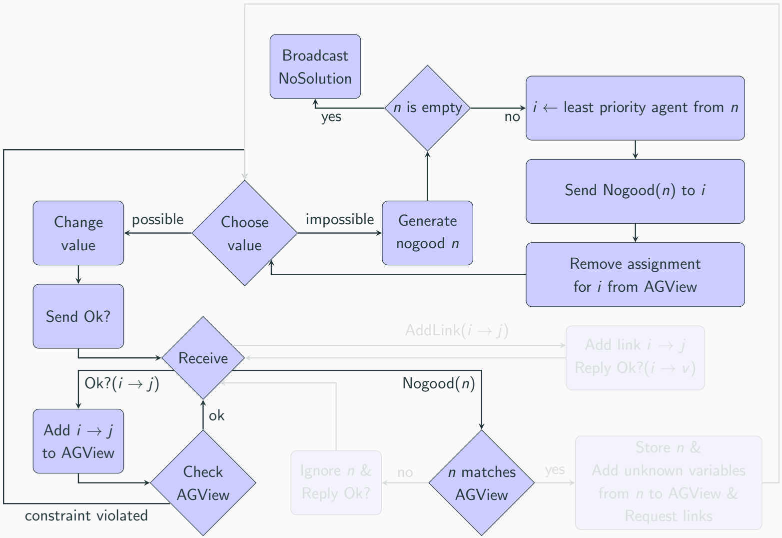
Asynchronous backtracking



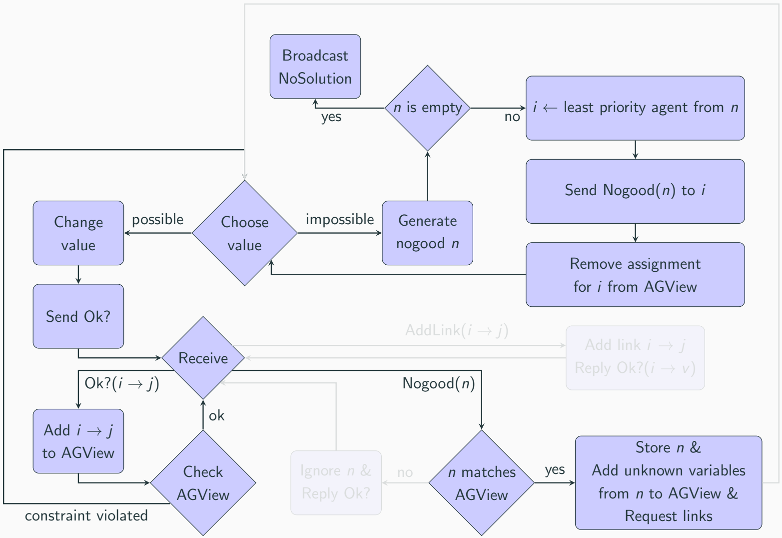
Asynchronous backtracking

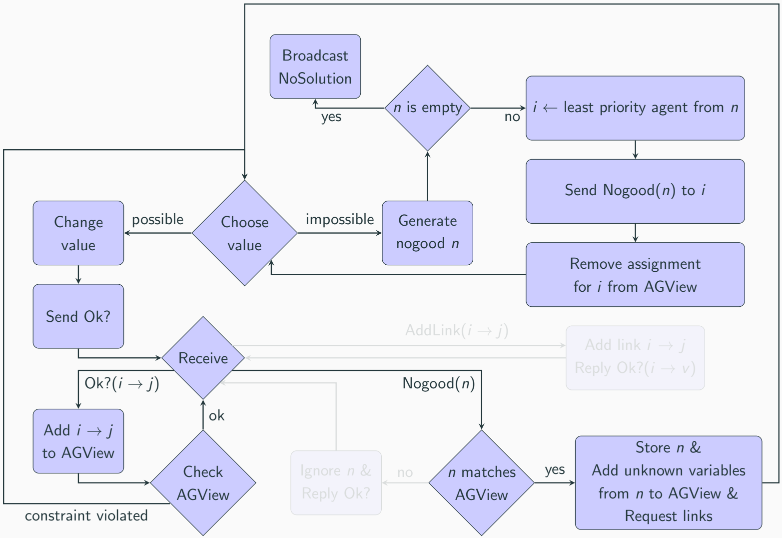


Asynchronous backtracking

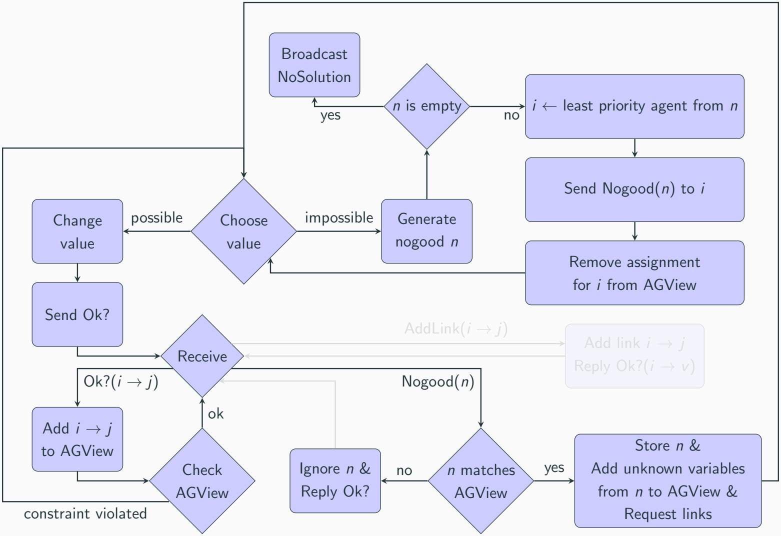


Asynchronous backtracking

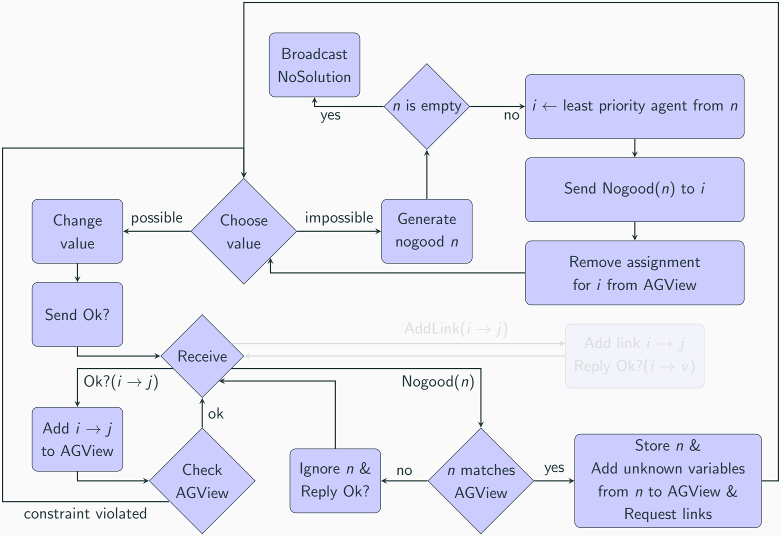



Broadcast NoSolution
yes
n is empty
no
i ← least priority agent from n
Send Nogood(n) to i
Remove assignment for i from AGView
Change value
possible
Choose value
impossible
Generate nogood n
Send Ok?
Receive
AddLink(i → j)
Add link i → j
Reply Ok?(i → v)
Ok?(i → j)
Nogood(n)
ok
Add i → j to AGView
Check AGView
Ignore n &
Reply Ok?
no
n matches AGView
yes
Store n &
Add unknown variables from n to AGView &
Request links
constraint violated

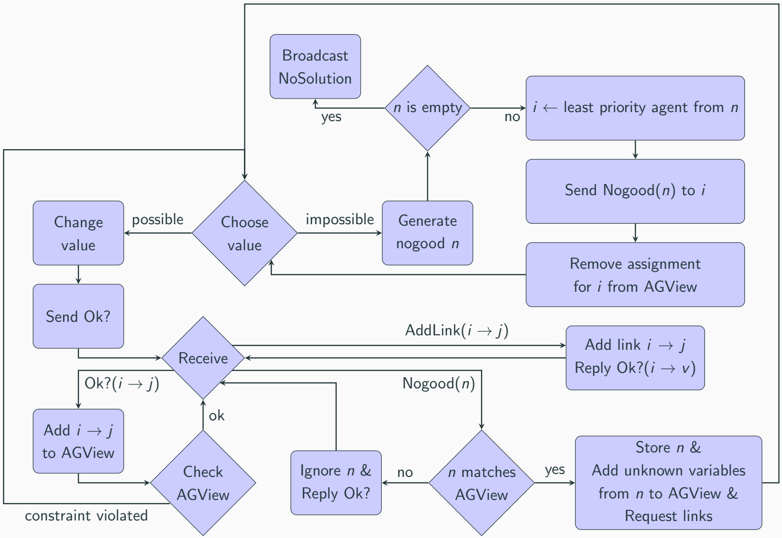
Asynchronous backtracking



Asynchronous backtracking



Asynchronous backtracking



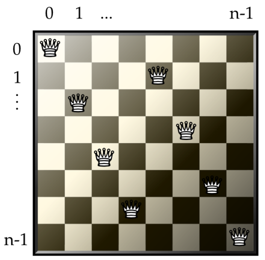
ASSIGNMENT

n-Queens problem in a distributed way

n queens from a $n \times n$ world had a serious dispute:

- They don't want to know of each other (i.e. no queen wants to have any other in her line of sight)
- They don't talk to each other except for few formal messages Ok? Nogood AddLink

Help them to find their place in the world!

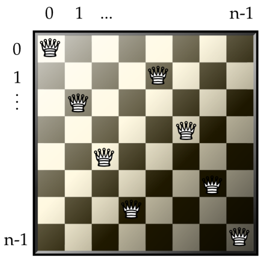


n-Queens problem in a distributed way

n queens from a $n \times n$ world had a serious dispute:

- They don't want to know of each other (i.e. no queen wants to have any other in her line of sight)
- They don't talk to each other except for few formal messages
  Ok? Nogood AddLink

Help them to find their place in the world!

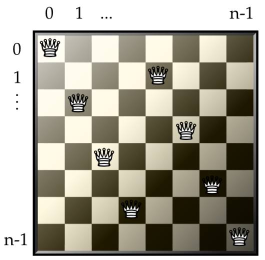


n-Queens problem in a distributed way

n queens from a $n \times n$ world had a serious dispute:

- They don't want to know of each other (i.e. no queen wants to have any other in her line of sight)
- They don't talk to each other except for few formal messages Ok? Nogood AddLink

Help them to find their place in the world!

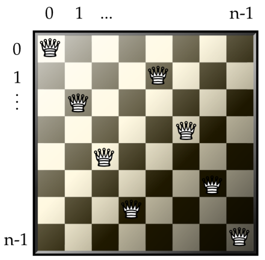


n-Queens problem in a distributed way

n queens from a $n \times n$ world had a serious dispute:

- They don't want to know of each other (i.e. no queen wants to have any other in her line of sight)
- They don't talk to each other except for few formal messages
  Ok? Nogood AddLink

Help them to find their place in the world!



n-Queens problem in a distributed way

Every agent controls **one queen** and decides about her position within its row.

In the end, one of the following has to happen:

- One of the agents reports that no solutions exists
- Each queen reports her position in her row (i.e. a column in which it is located)

  $\uparrow$ of course correctly ;-)

n-Queens problem in a distributed way

Any **asynchronous** and **distributed** solution is acceptable (e.g. ABT).

$\rightarrow$ No centralized knowledge allowed!

$\rightarrow$ No synchonization!

$\rightarrow$ No hardcoded solutions!

n-Queens problem in a distributed way

Total: **12 points**

- Solve 3×3 chessboard problem with 3 queens (3 points)
- Solve 4×4 chessboard problem with 4 queens (2 points)
- Solve 8×8 chessboard problem with 8 queens (2 points)
- Solve 12×12 chessboard problem with 12 queens (3 points)

n-Queens problem in a distributed way

Guaranteed termination detection (**1 point**)

- How to detect *quiescence* in an algorithmic way?
- You may want to get inspired by other DCSP/DCOP algorithms.

Quiescence should be discovered using **local knowledge** only.

$\rightarrow$ Sending whole solution to a single agent for verification is not an option!

n-Queens problem in a distributed way

Report (**1 point**)

- How is the n-queens problem modeled as a DCSP? (variables, domains, constraints, agents)
- How is the ABT algorithm customized for the n-queens problem?
- How do you determine priorities between agents?
- How do you detect that the search has terminated?

ABT EXAMPLE