Distributed Constraint Satisfaction Problems: 2 Asynchronous Algorithms

Thomas Léauté

16.412J - Cognitive Robotics

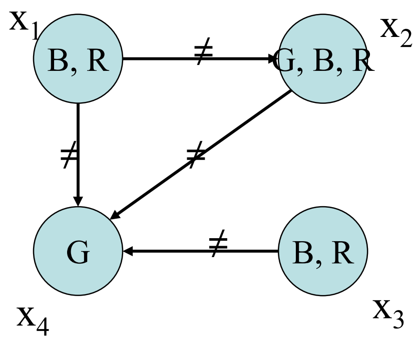
April 7, 2004

Presentation Outline

1. Introduction to CSPs and DCSPs
2. The Asynchronous Backtracking Algorithm
3. The Asynchronous Weak-Commitment Search Algorithm
4. Conclusion and Introduction to the Task Allocation Problem

M. Yokoo, E. Durfee, T. Ishida and K. Kuwabara, *Distributed Constraint Satisfaction Problem: Formalization and Algorithms*, IEEE Transactions on Knowledge and Data Engineering, VOL. 10, NO. 5, Sept/Oct 1998

1. Introduction to CSPs and DCSPs



1. Introduction to CSPs and DCSPs

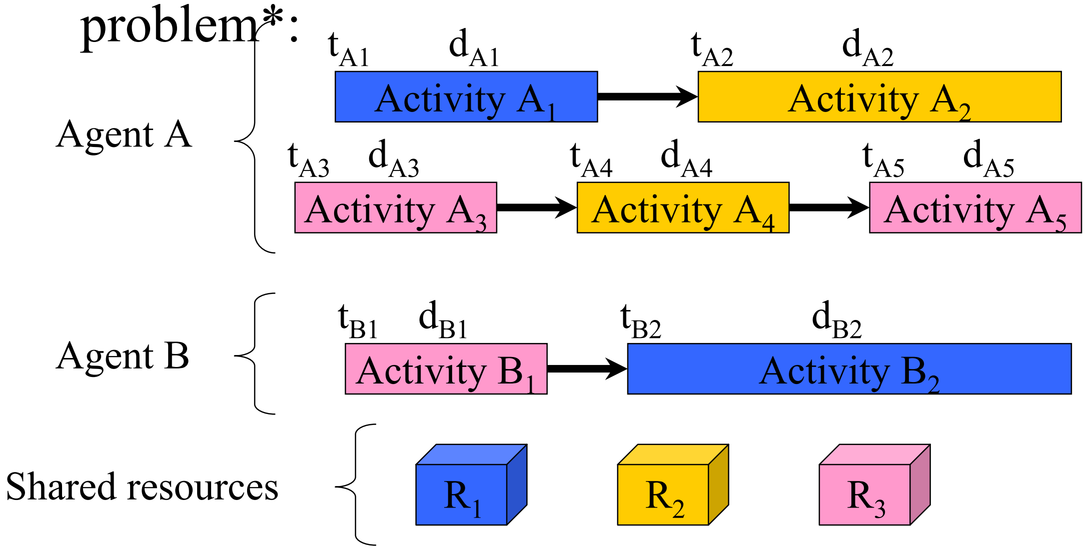
- Formal definition: a Constraint Satisfaction Problem (CSP) is a triple ($\mathcal{X}$, $\mathcal{D}$, $\mathcal{C}$), where
  - $\mathcal{X}$ is a list of variables $x_1, x_2, \ldots, x_n$
  - $\mathcal{D}$ is a list of finite, discrete value domains $D_1, D_2, \ldots, D_n$ assigned to the variables
  - $\mathcal{C}$ is a set of constraints $C_1, C_2, \ldots, C_n$ on the variables, a constraint being a predicate:

$$C_k : \quad D_{k_1} \times D_{k_2} \times \ldots \times D_{k_j} \rightarrow \{true, false\}$$

- A solution to the problem is an assignment to the variables that satisfies all the constraints

1. Introduction to CSPs and DCSPs

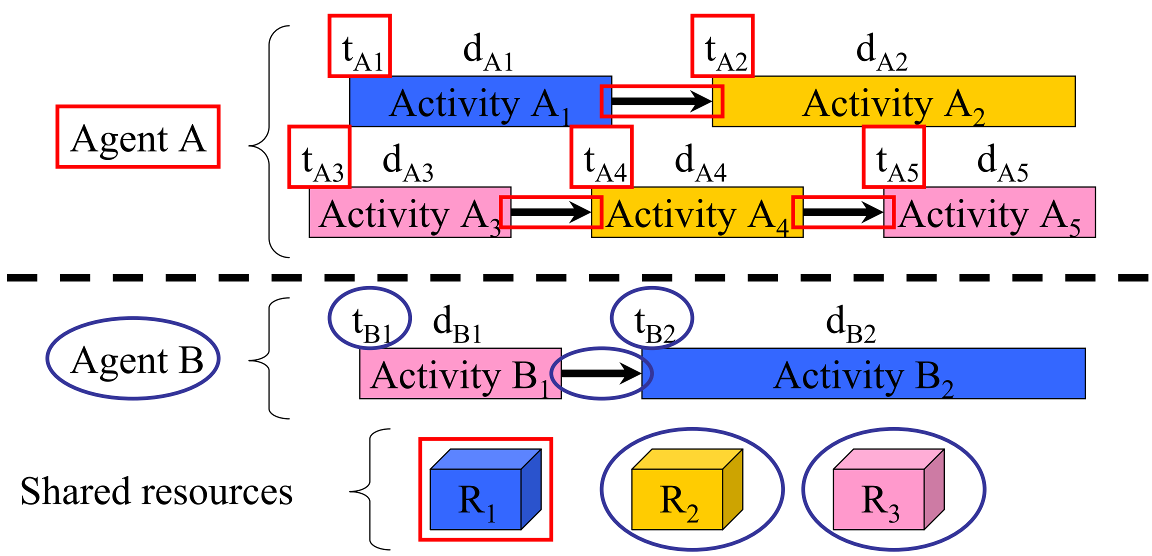
- Many AI problems can be formulated as CSPs
- Example of a multi-agent scheduling problem*:



* K. Sycara, S. Roth, N. Nadeh and M. Fox, *Distributed Constrained Heuristic Search*, IEEE Transactions on Systems, Man, and Cybernetics, VOL. 21, NO. 6, Nov/Dec 1991

1. Introduction to CSPs and DCSPs

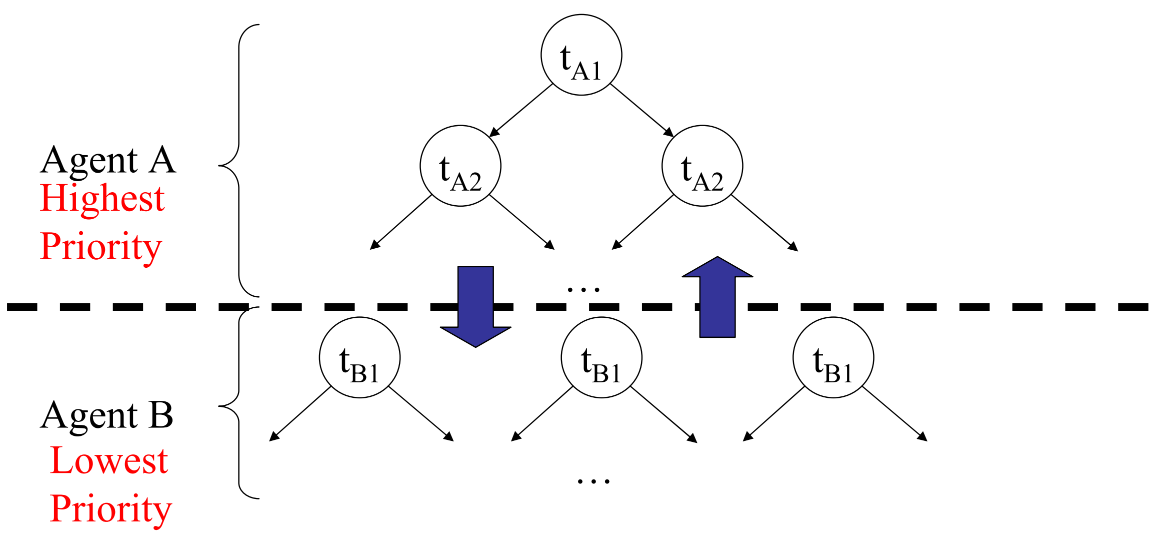
- Split the problem in coupled sub-problems: distribute the variables AND the constraints among the agents



* K. Sycara, S. Roth, N. Nadeh and M. Fox, *Distributed Constrained Heuristic Search*, IEEE Transactions on Systems, Man, and Cybernetics, VOL. 21, NO. 6, Nov/Dec 1991

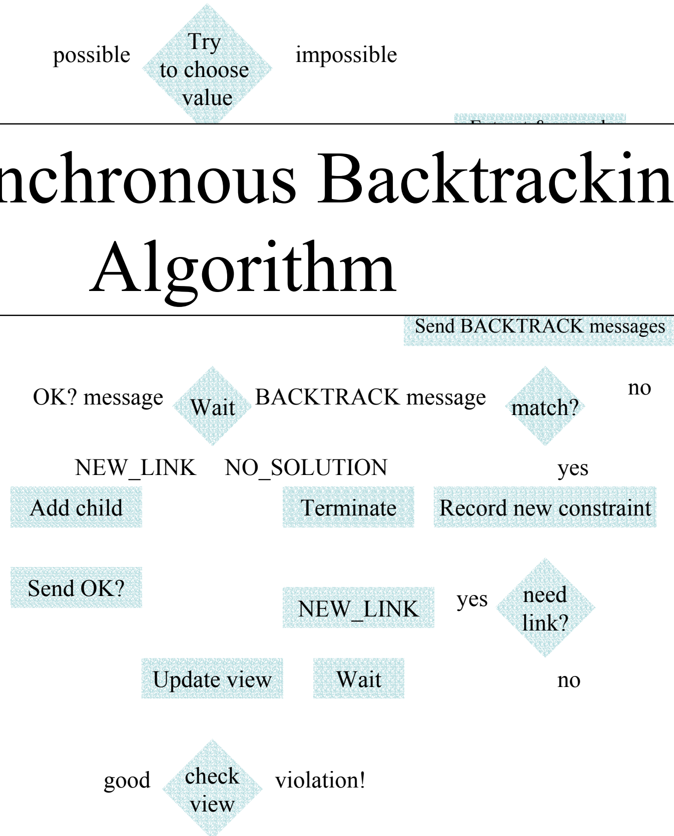
- Centralized method: one leader agent gathers all the information from other agents and solves the problem
  - Prohibitive cost of collecting information
  - Security/Privacy reasons
  - Not computationally efficient

- Synchronous Backtracking method:



  - Sequential => not computationally efficient

2. The Asynchronous Backtracking Algorithm

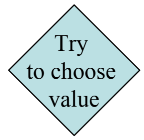


2. Asynchronous Backtracking

- Assumptions:
  - Given priority order on the agents
  - An agent must be able to send messages to any other agent
  - Each agent has exactly ONE single variable

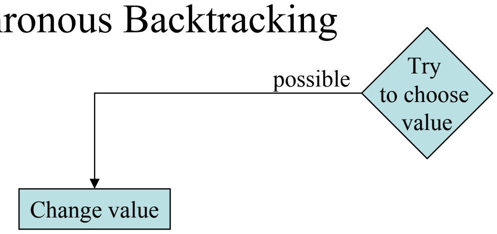
- Key ideas:
  - Agents work concurrently (= “asynchronously”), exchanging messages to collect required information
  - Conflict-directed search

2. Asynchronous Backtracking



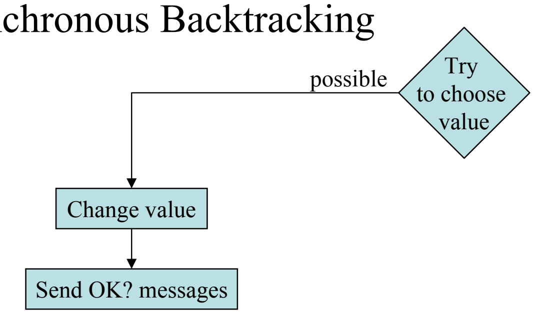
The agent selects a value for its variable
satisfying the constraints whose enforcement
it is responsible for

2. Asynchronous Backtracking



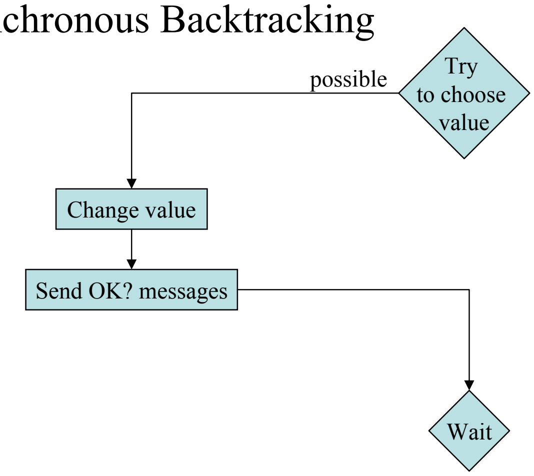
If there is at least one value satisfying the constraints,
the agent picks one and changes the assignment
to its variable

2. Asynchronous Backtracking



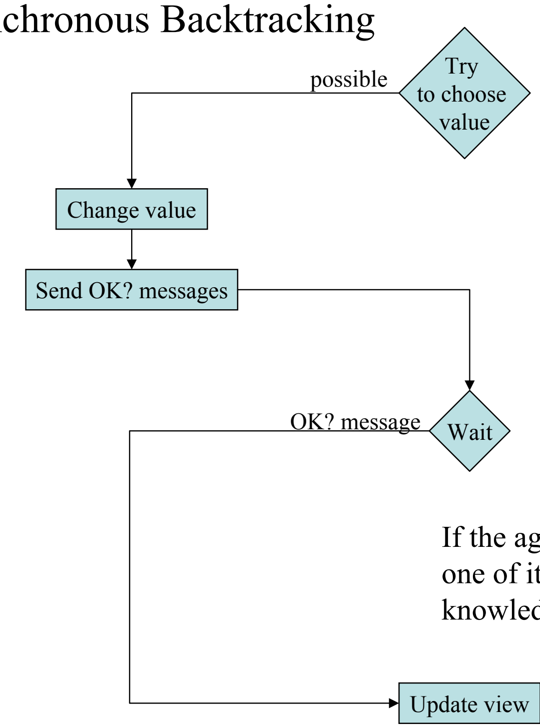
The agent communicates its new assignment to its children through "OK?" messages

2. Asynchronous Backtracking



The agent then waits for answers to his OK? messages from its children (and for other OK? messages from its parents)

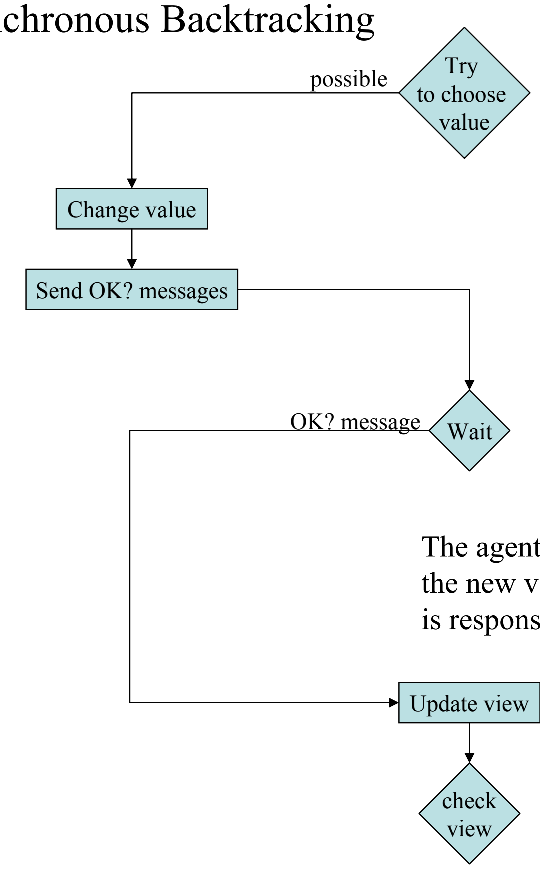
2. Asynchronous Backtracking



A's view	
B	1
C	?
D	3
E	4
F	?
G	?
H	?

If the agent receives an OK? message from one of its parents, it updates its "view", i.e. its knowledge of the values of the other variables

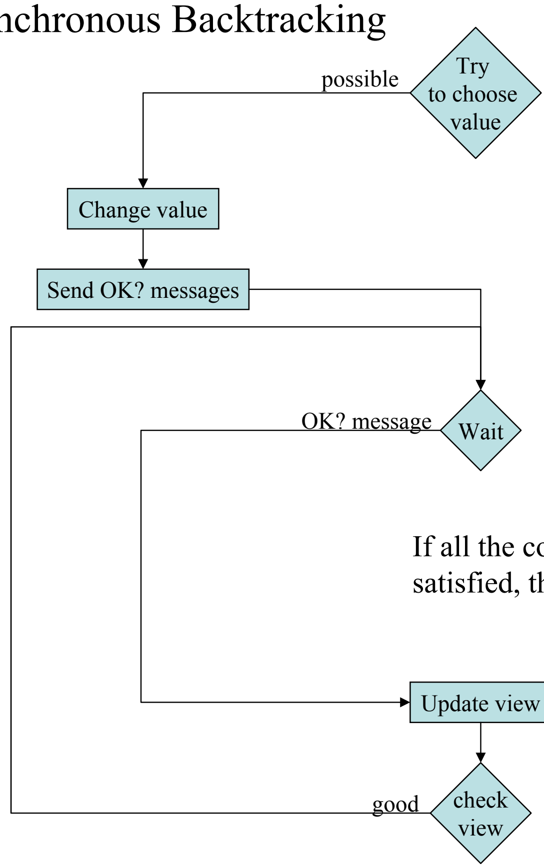
2. Asynchronous Backtracking



A's view	
B	1
C	?
D	3
E	4
F	?
G	?
H	?

The agent then checks its view, making sure that the new values do not violate any constraint it is responsible for

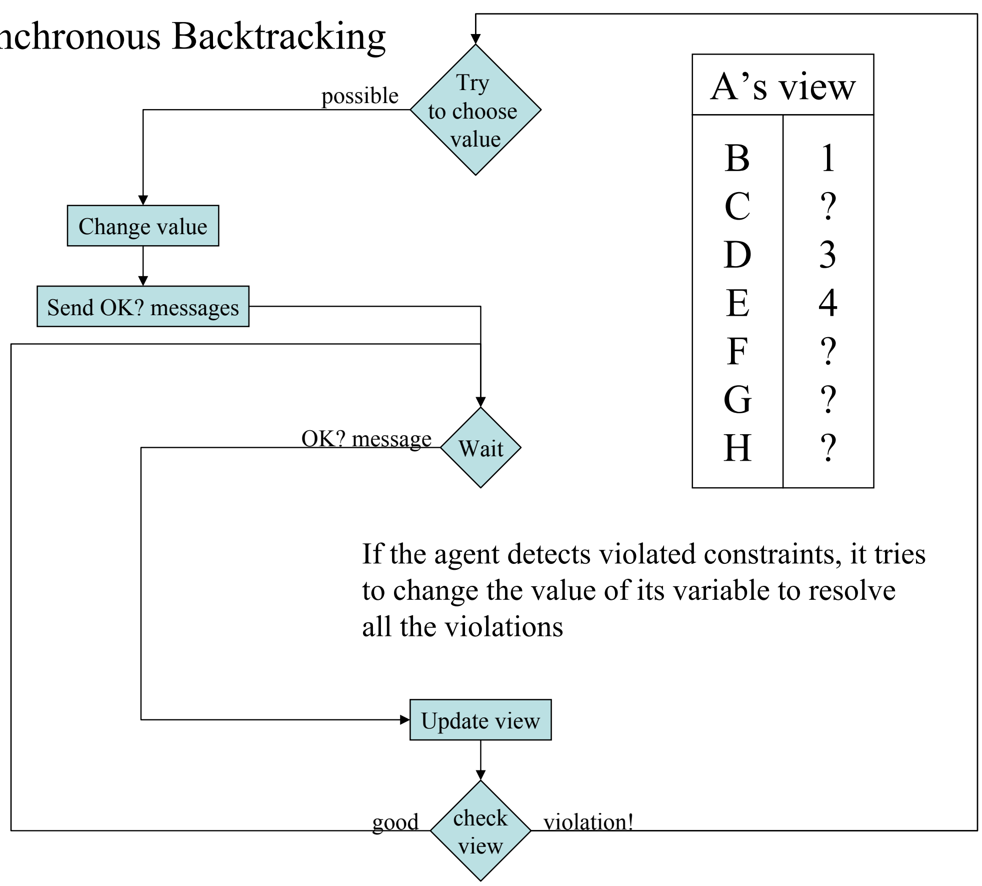
2. Asynchronous Backtracking



A's view	
B	1
C	?
D	3
E	4
F	?
G	?
H	?

If all the constraints it is responsible for are still satisfied, the agent keeps waiting for messages

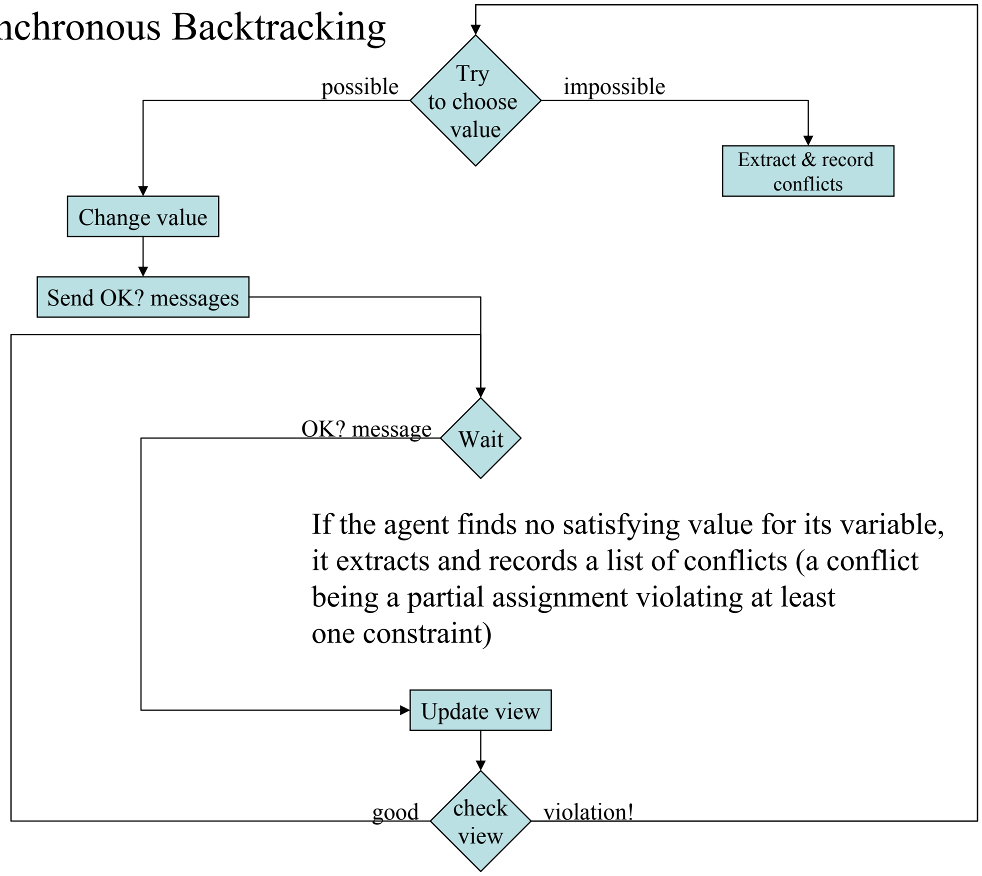
2. Asynchronous Backtracking



A's view	
B	1
C	?
D	3
E	4
F	?
G	?
H	?

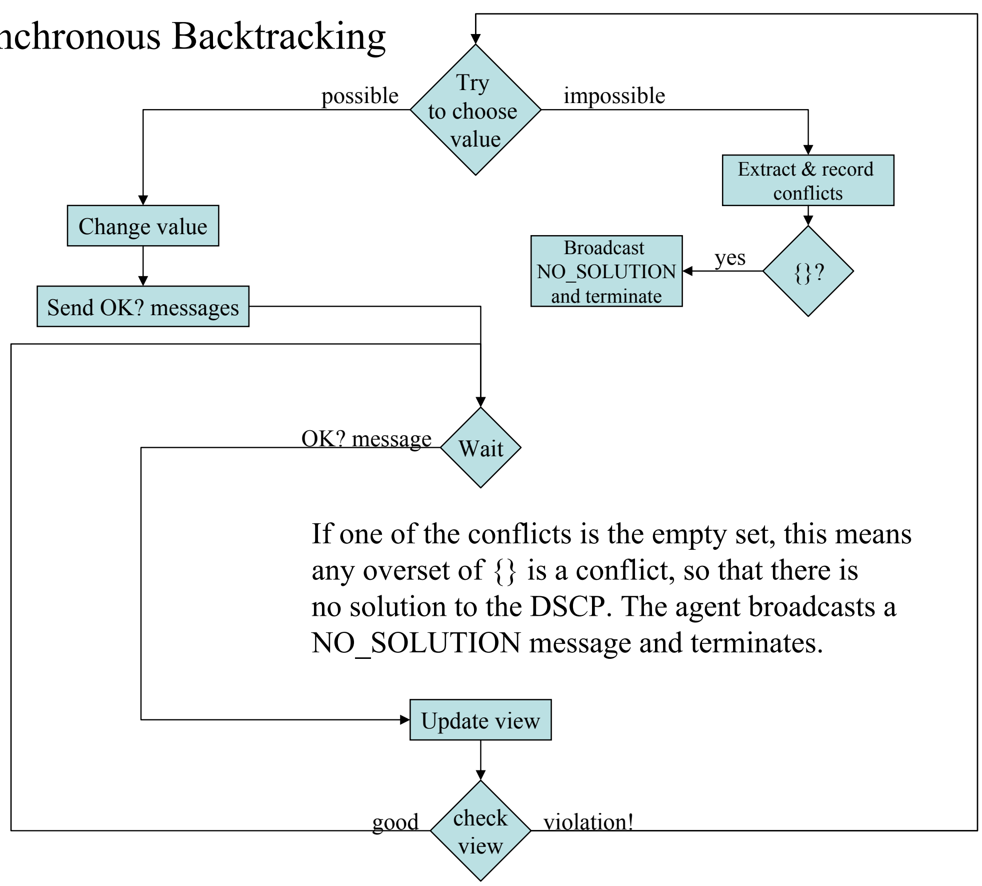
If the agent detects violated constraints, it tries to change the value of its variable to resolve all the violations

2. Asynchronous Backtracking



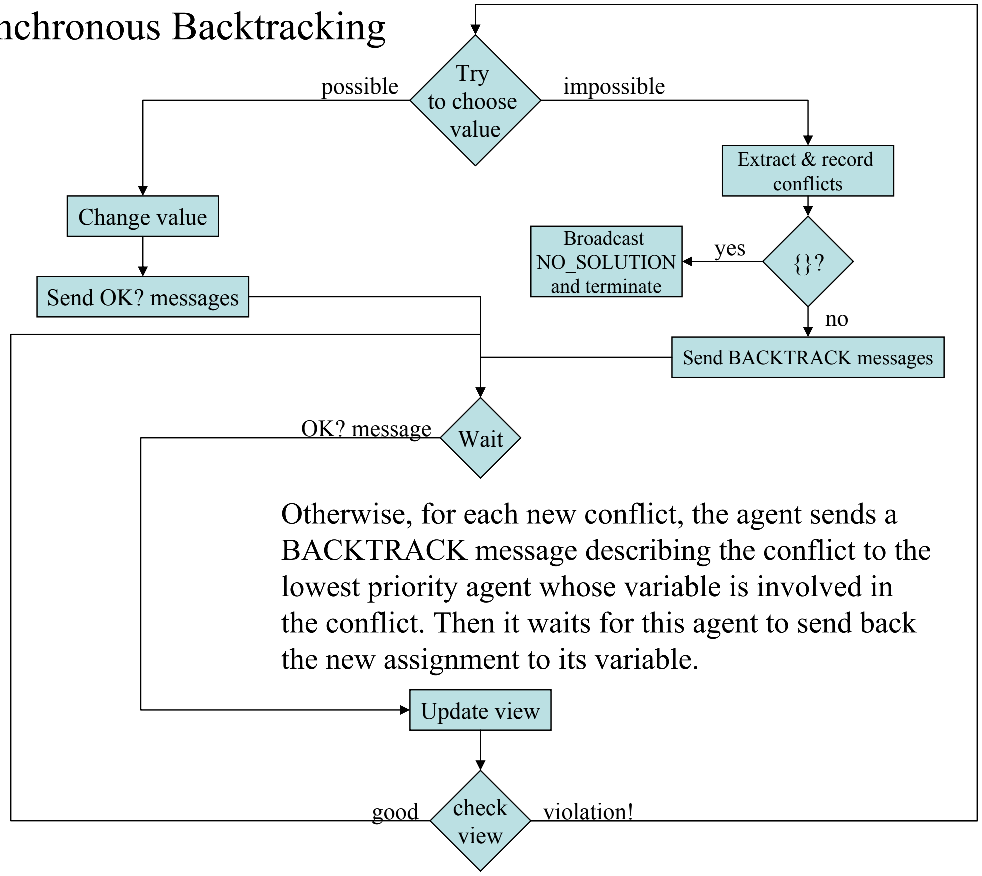
If the agent finds no satisfying value for its variable, it extracts and records a list of conflicts (a conflict being a partial assignment violating at least one constraint)

2. Asynchronous Backtracking



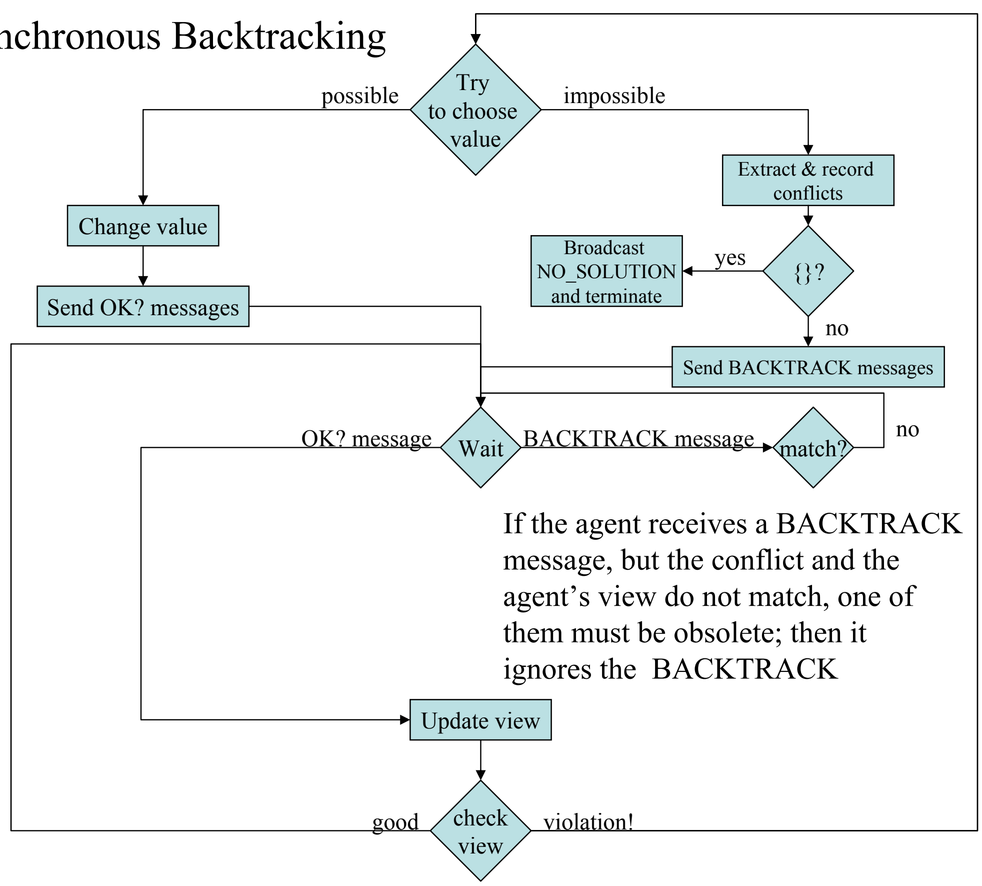
If one of the conflicts is the empty set, this means any overset of {} is a conflict, so that there is no solution to the DSCP. The agent broadcasts a NO_SOLUTION message and terminates.

2. Asynchronous Backtracking



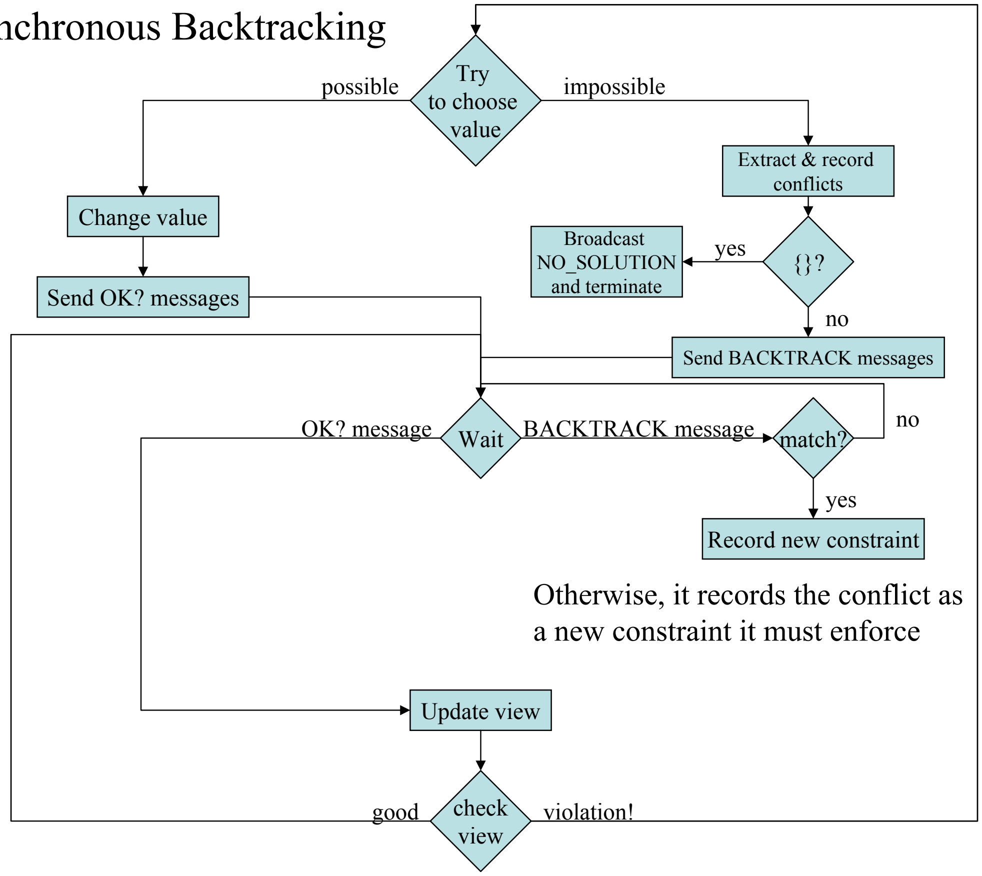
Otherwise, for each new conflict, the agent sends a BACKTRACK message describing the conflict to the lowest priority agent whose variable is involved in the conflict. Then it waits for this agent to send back the new assignment to its variable.

2. Asynchronous Backtracking



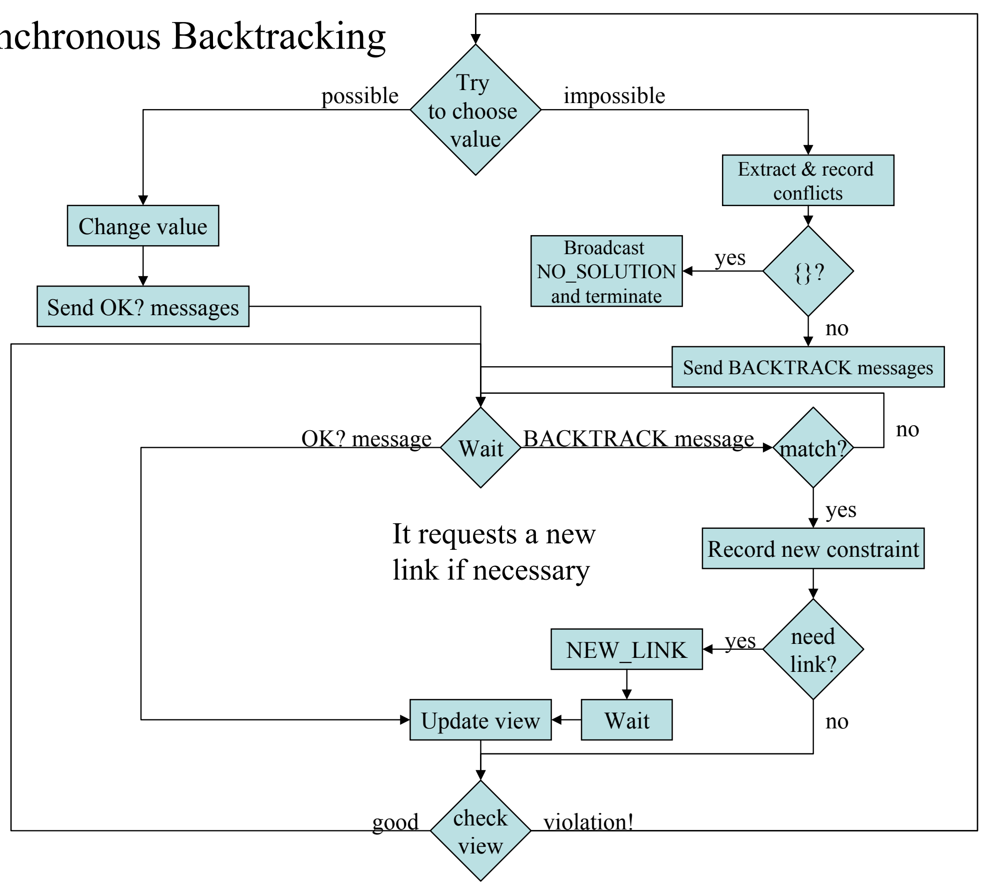
If the agent receives a BACKTRACK message, but the conflict and the agent's view do not match, one of them must be obsolete; then it ignores the BACKTRACK

2. Asynchronous Backtracking



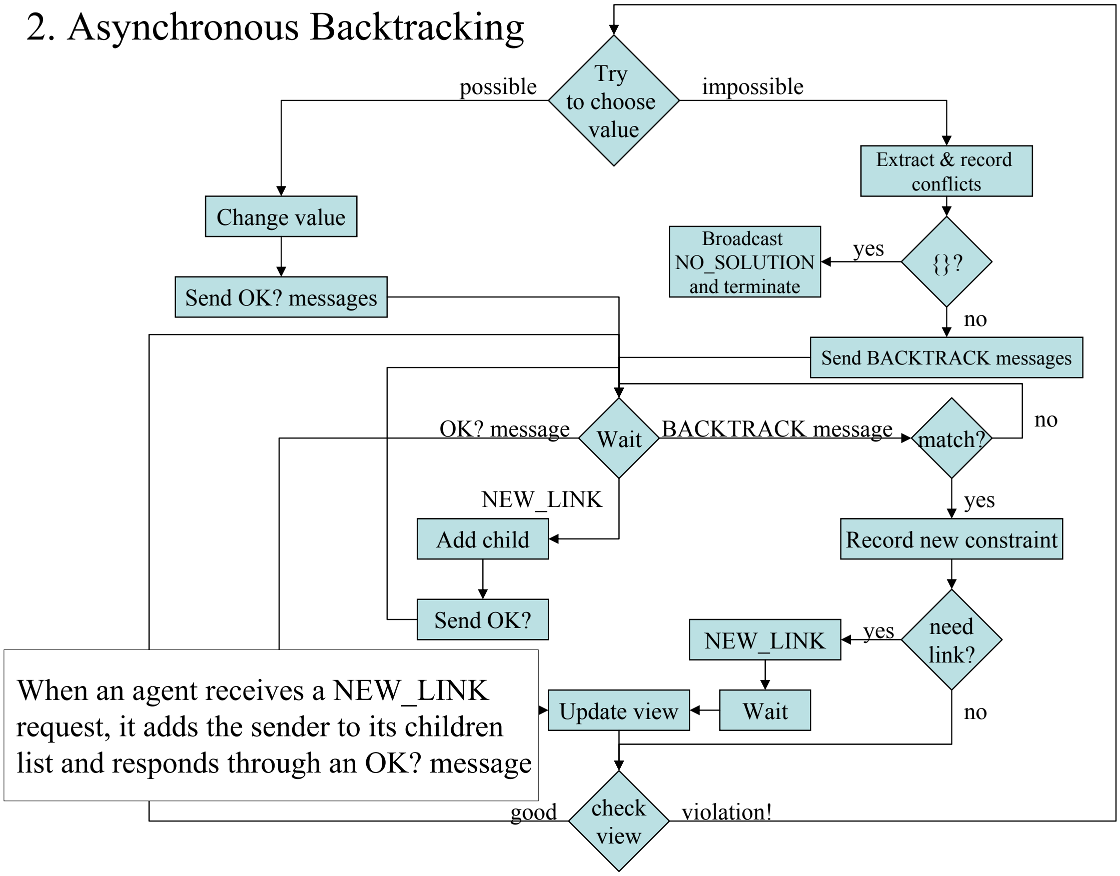
Otherwise, it records the conflict as a new constraint it must enforce

2. Asynchronous Backtracking



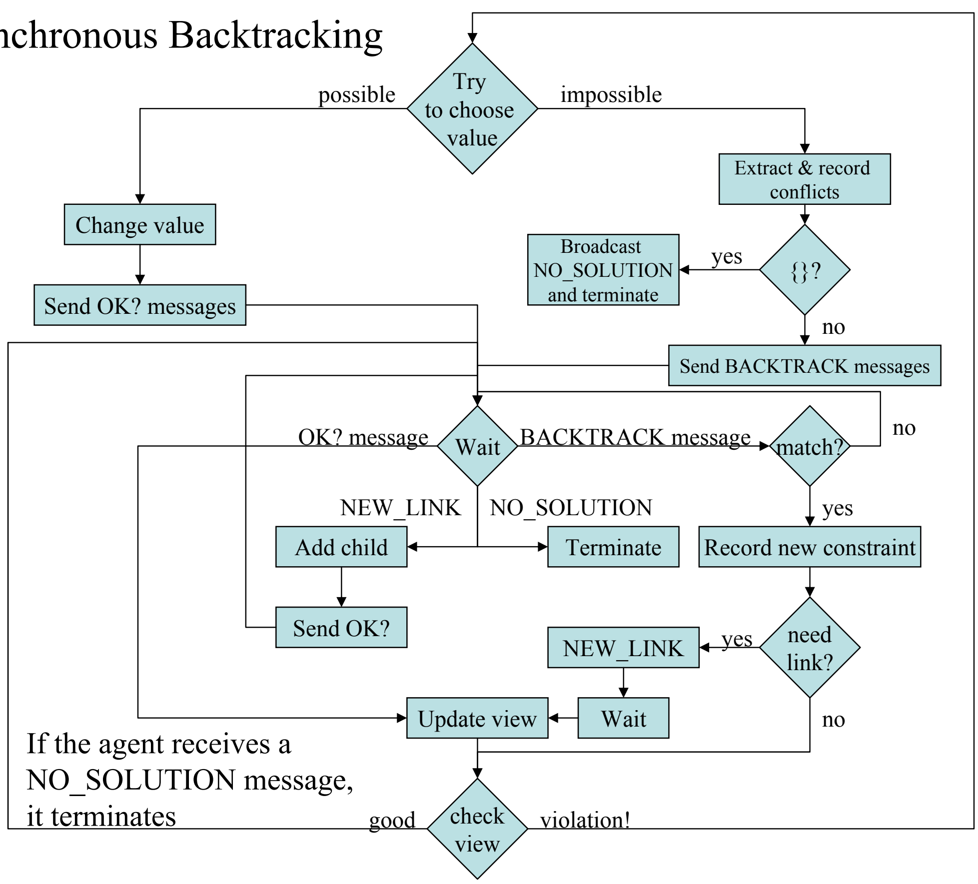
It requests a new link if necessary

2. Asynchronous Backtracking



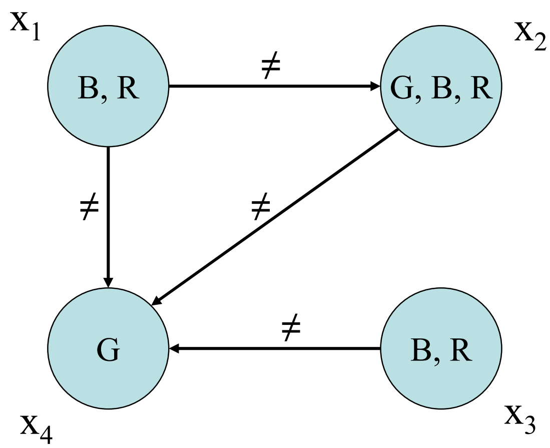
When an agent receives a NEW_LINK request, it adds the sender to its children list and responds through an OK? message

2. Asynchronous Backtracking

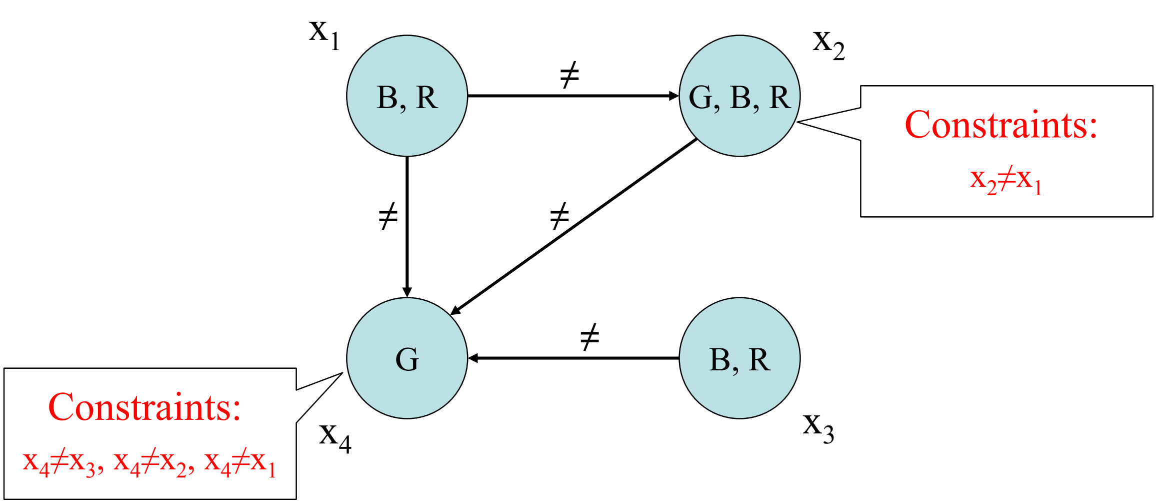


If the agent receives a NO_SOLUTION message, it terminates

2. Asynchronous Backtracking: The Graph Coloring Example



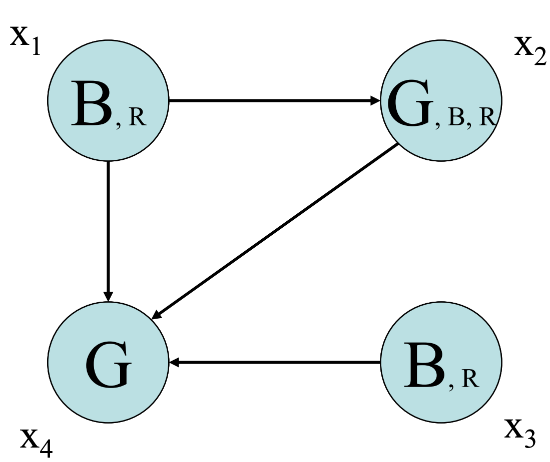
2. Asynchronous Backtracking: The Graph Coloring Example

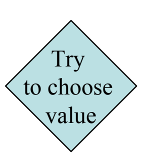


2. Asynchronous Backtracking: The Graph Coloring Example

<table>
<tr><td>NAME</td><td colspan="2">VALUE</td><td>DOMAIN</td></tr>
<tr><td rowspan="3">VIEW</td><td>CHILDREN</td><td colspan="2">PARENTS</td></tr>
<tr><td colspan="3">KNOWN CONFLICTS</td></tr>
<tr><td colspan="3">CONSTRAINTS TO ENFORCE</td></tr>
</table>

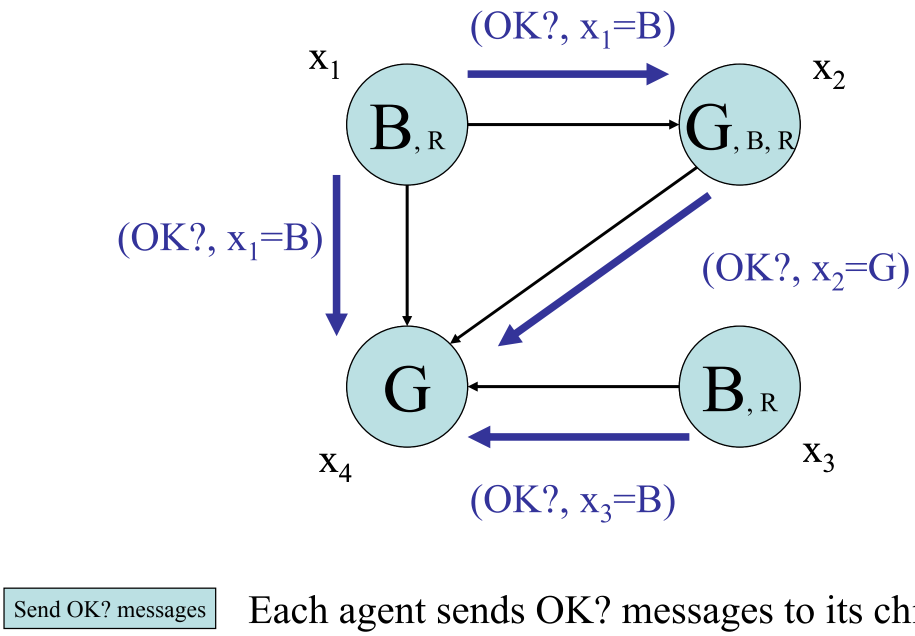
2. Asynchronous Backtracking: The Graph Coloring Example





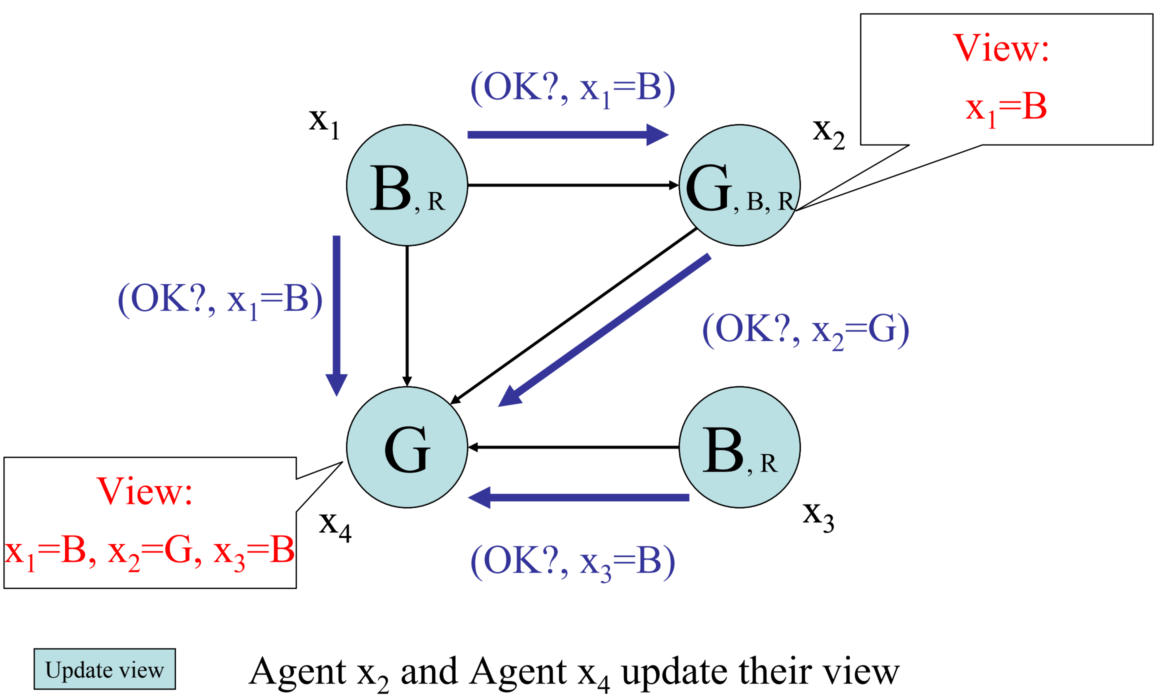
Each agent chooses an assignment to its variable

2. Asynchronous Backtracking: The Graph Coloring Example



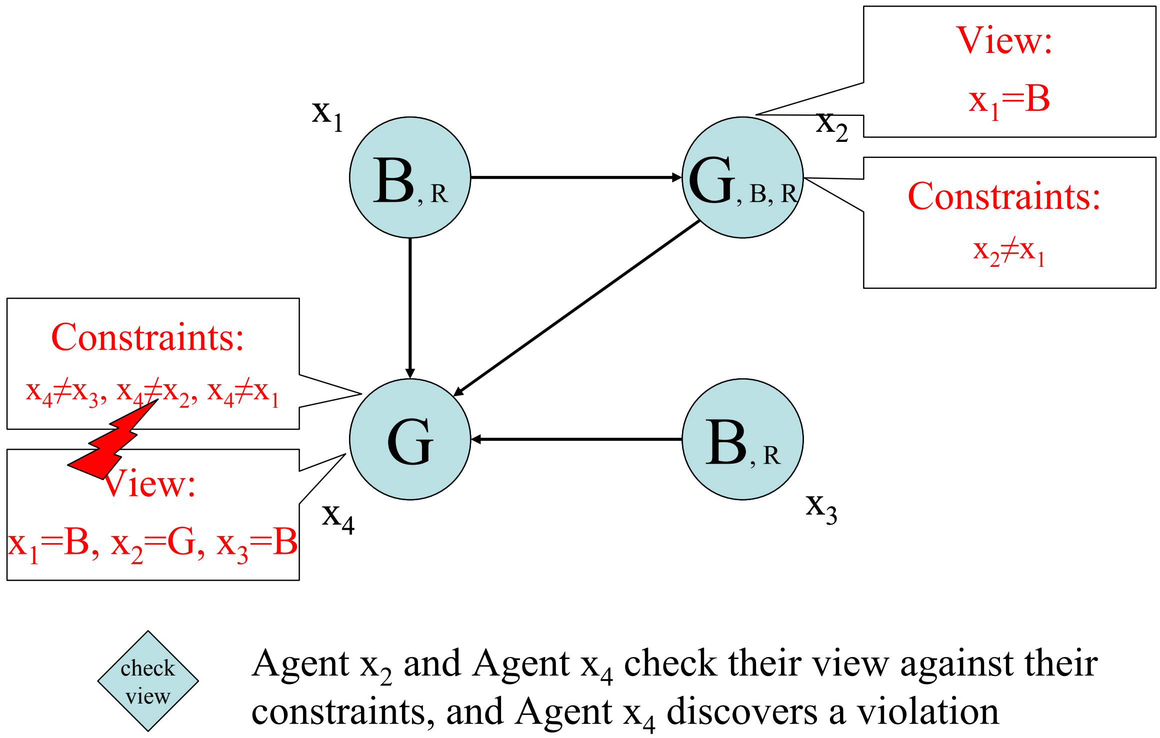
Each agent sends OK? messages to its children

2. Asynchronous Backtracking: The Graph Coloring Example



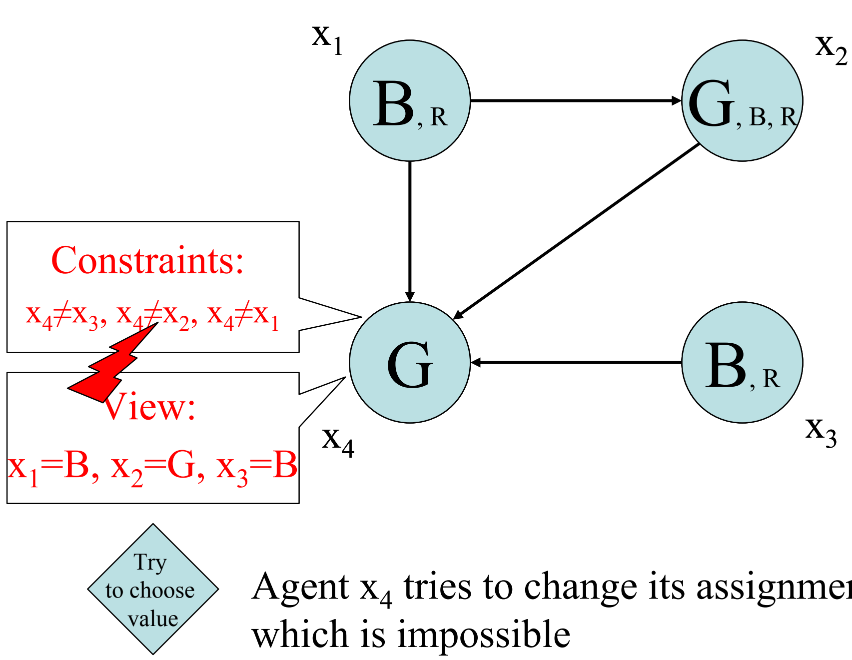
Agent x_2 and Agent x_4 update their view

2. Asynchronous Backtracking: The Graph Coloring Example



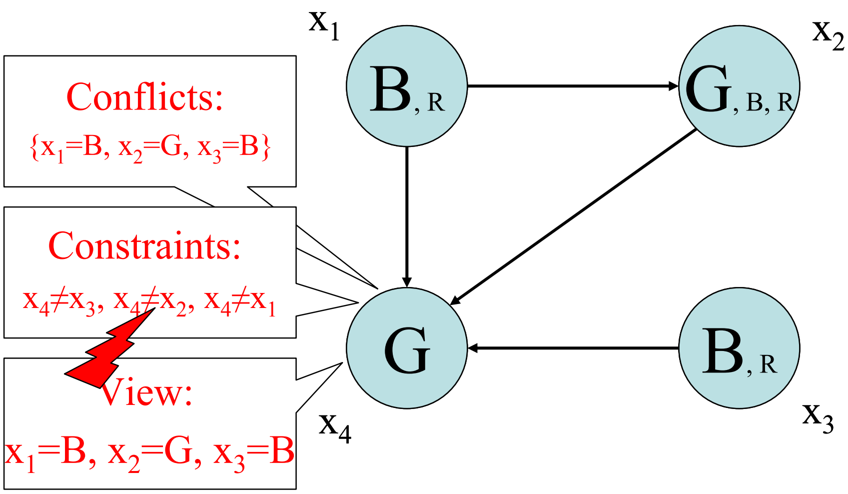
Agent x_2 and Agent x_4 check their view against their constraints, and Agent x_4 discovers a violation

2. Asynchronous Backtracking: The Graph Coloring Example



Agent x_4 tries to change its assignment, which is impossible

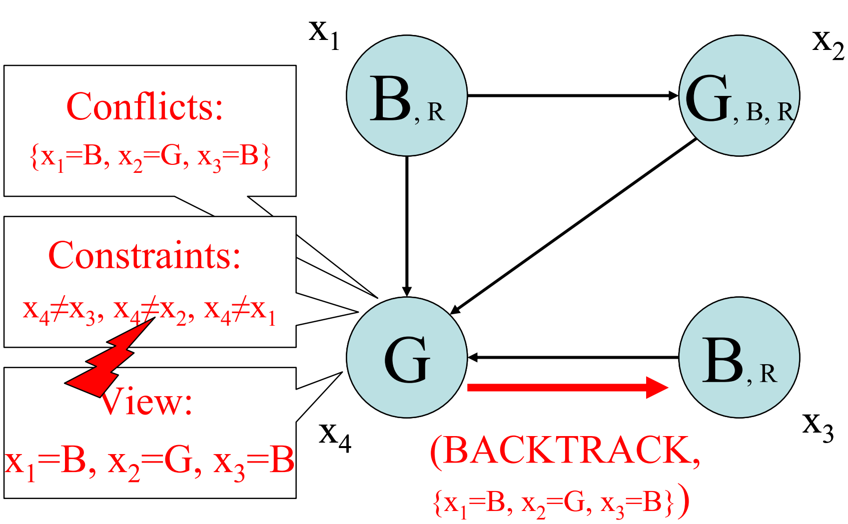
2. Asynchronous Backtracking: The Graph Coloring Example



Extract & record conflicts

Agent x_4 extracts and records the conflicts

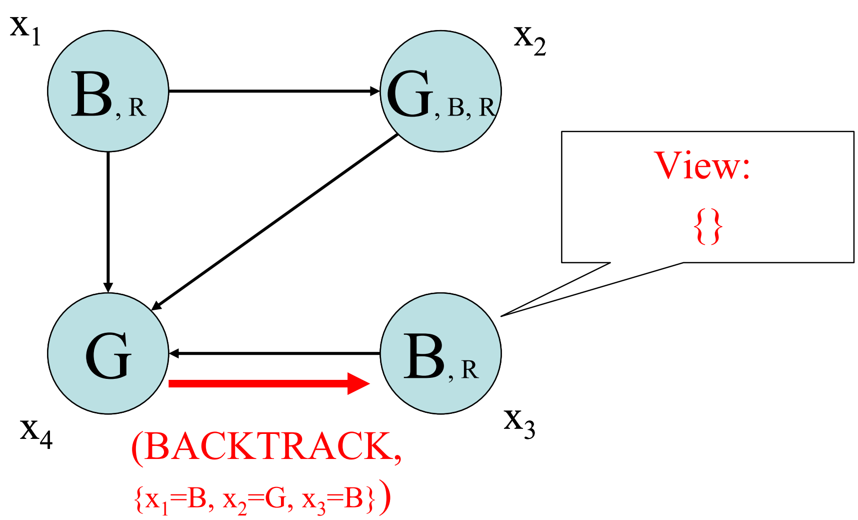
2. Asynchronous Backtracking: The Graph Coloring Example



Send BACKTRACK messages

{} is not among the new conflicts, so
Agent x_4 sends BACKTRACK messages

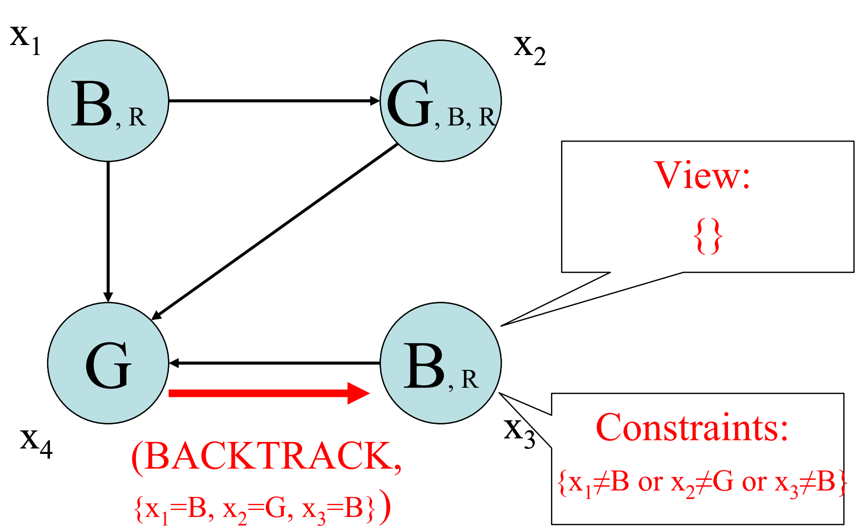
2. Asynchronous Backtracking: The Graph Coloring Example



match?

Agent x_3 receives the message and checks the conflict against its view

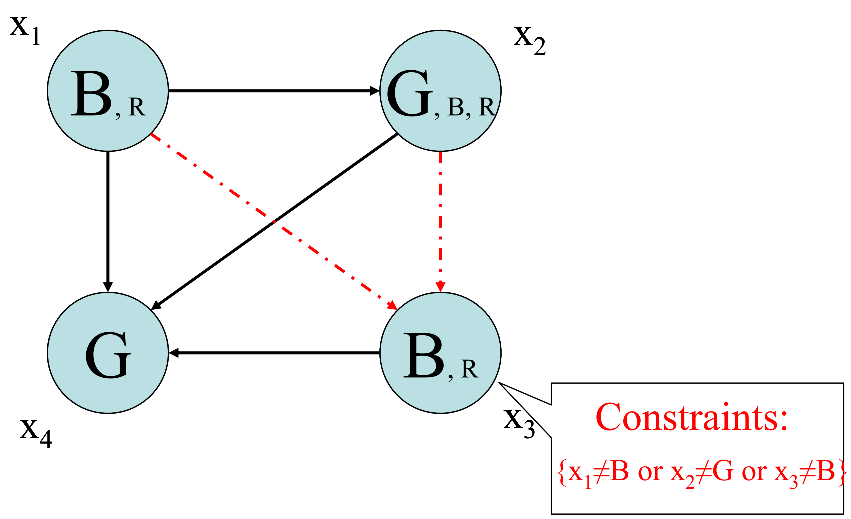
2. Asynchronous Backtracking: The Graph Coloring Example



Record new constraint

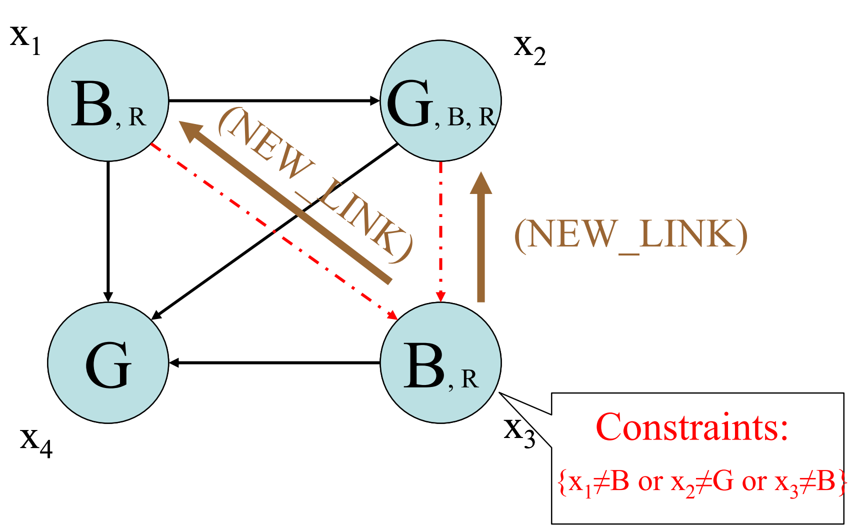
Agent x_3 records the conflict as a new constraint

2. Asynchronous Backtracking: The Graph Coloring Example



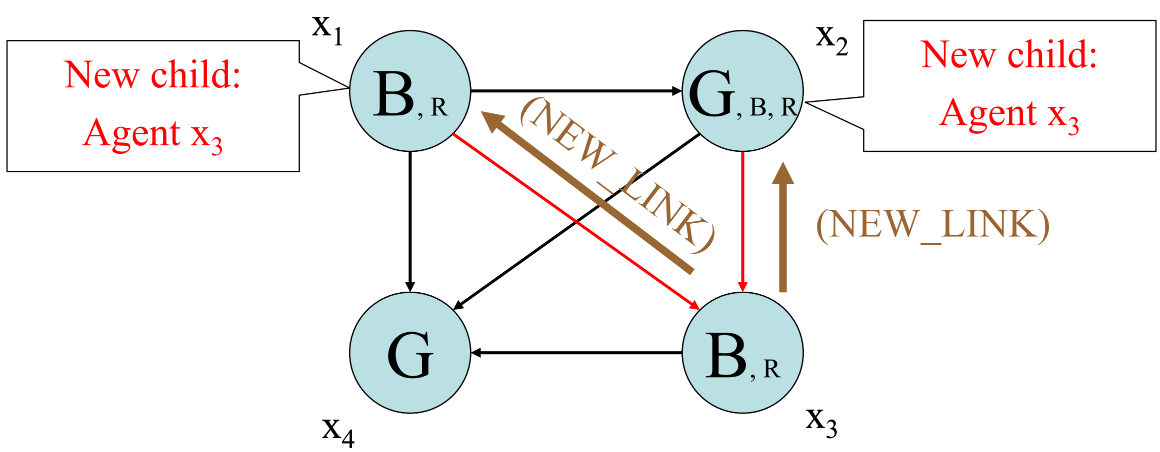
Agent x_3 checks if it needs new links to enforce it

2. Asynchronous Backtracking: The Graph Coloring Example



Agent x_3 sends NEW_LINK requests

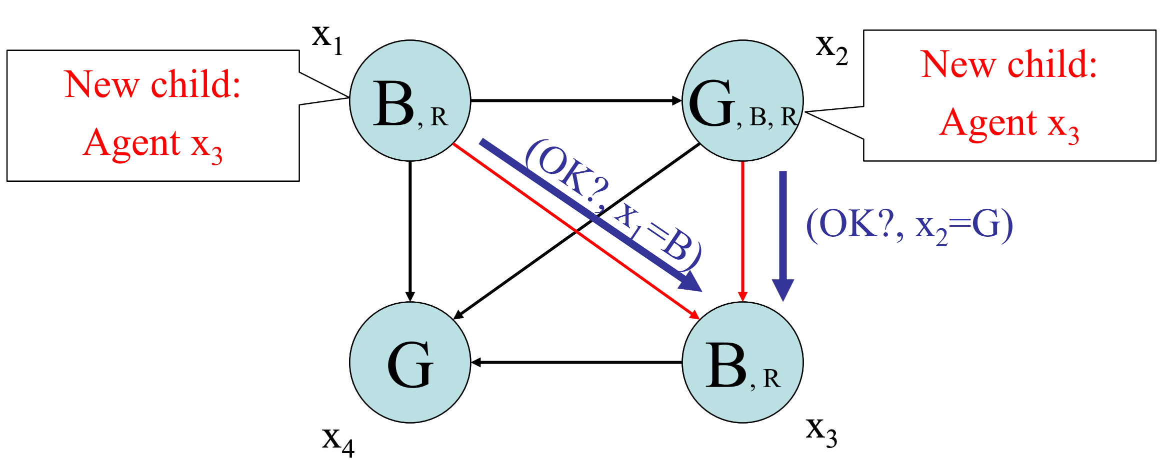
2. Asynchronous Backtracking: The Graph Coloring Example



Add child

Agents x_1 and x_2 receive the NEW_LINK requests and add Agent x_3 to their children list

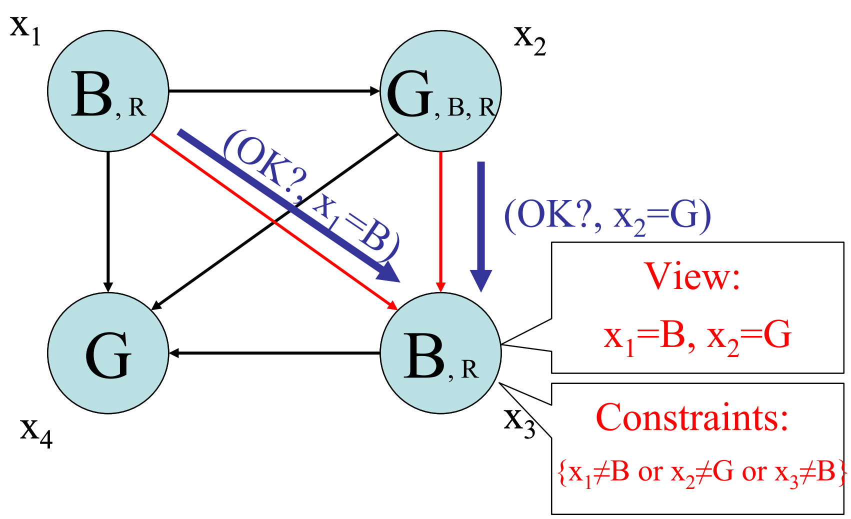
2. Asynchronous Backtracking: The Graph Coloring Example



Send OK?

Agents x_1 and x_2 send OK? to confirm new links

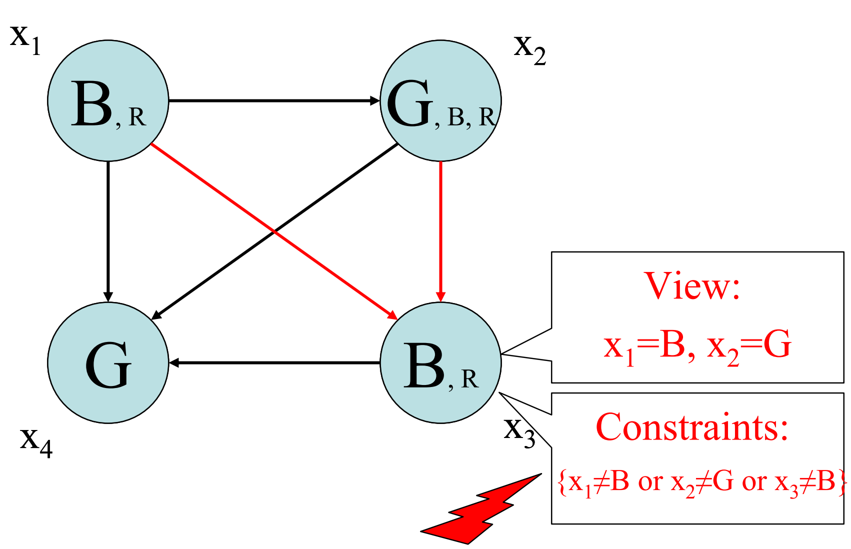
2. Asynchronous Backtracking: The Graph Coloring Example



Update view

Agent x_3 receives messages and updates its view

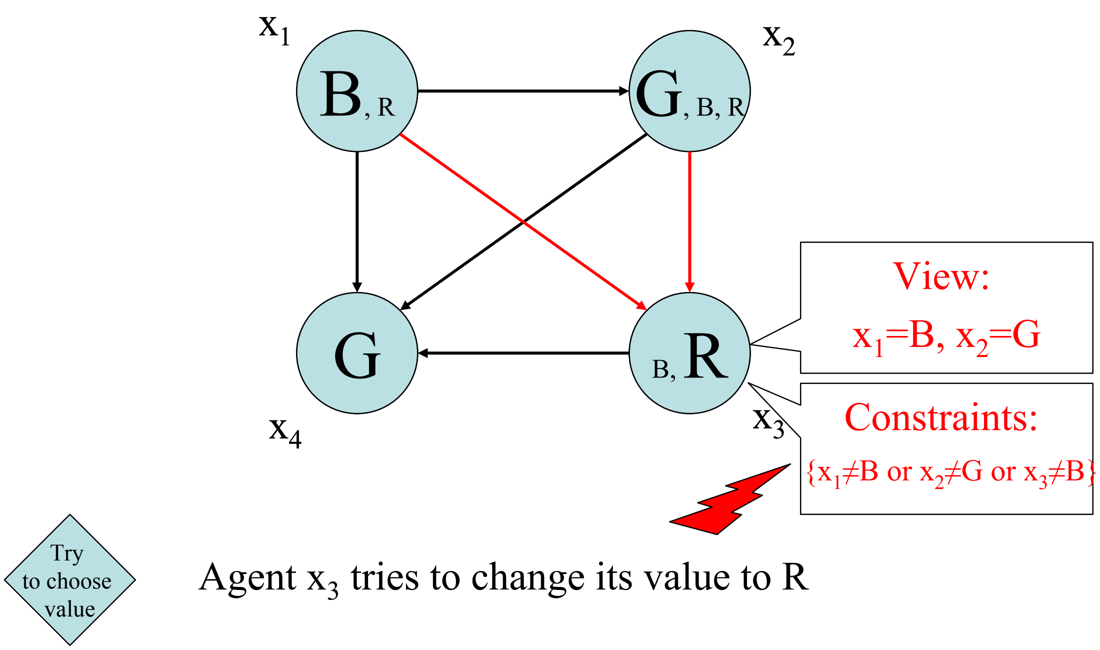
2. Asynchronous Backtracking: The Graph Coloring Example



check view

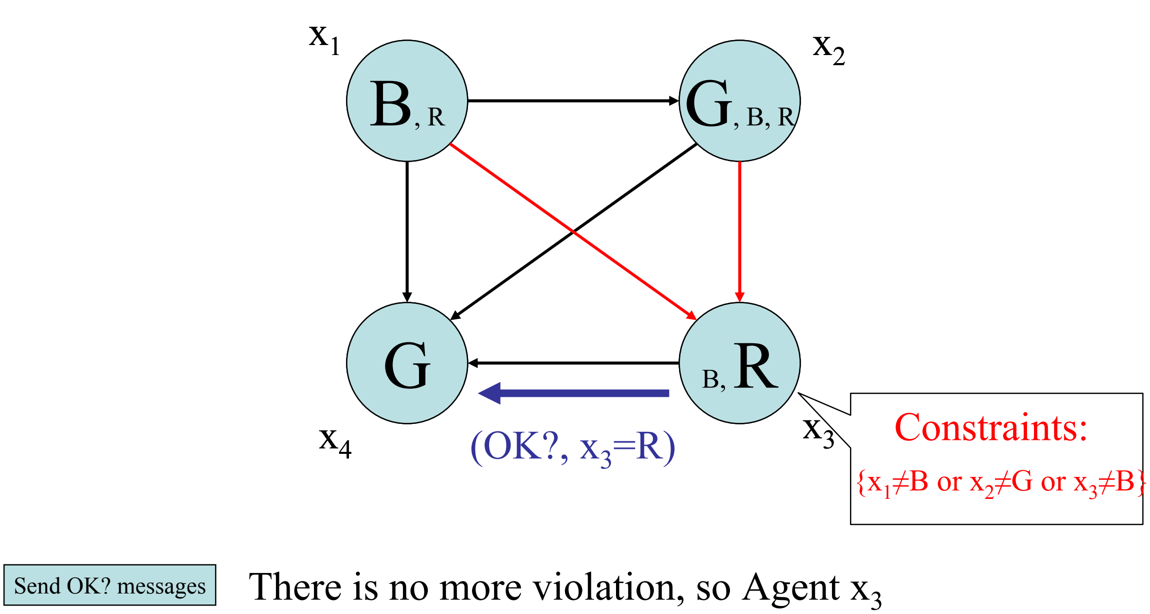
Agent x_3 checks its view, and discovers that one constraint (the new one) is violated

2. Asynchronous Backtracking: The Graph Coloring Example



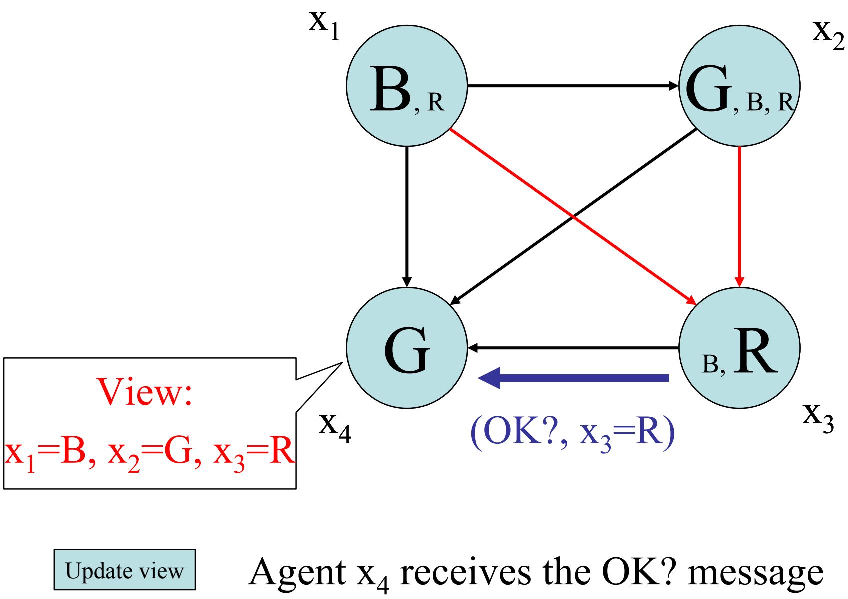
Agent x_3 tries to change its value to R

2. Asynchronous Backtracking: The Graph Coloring Example



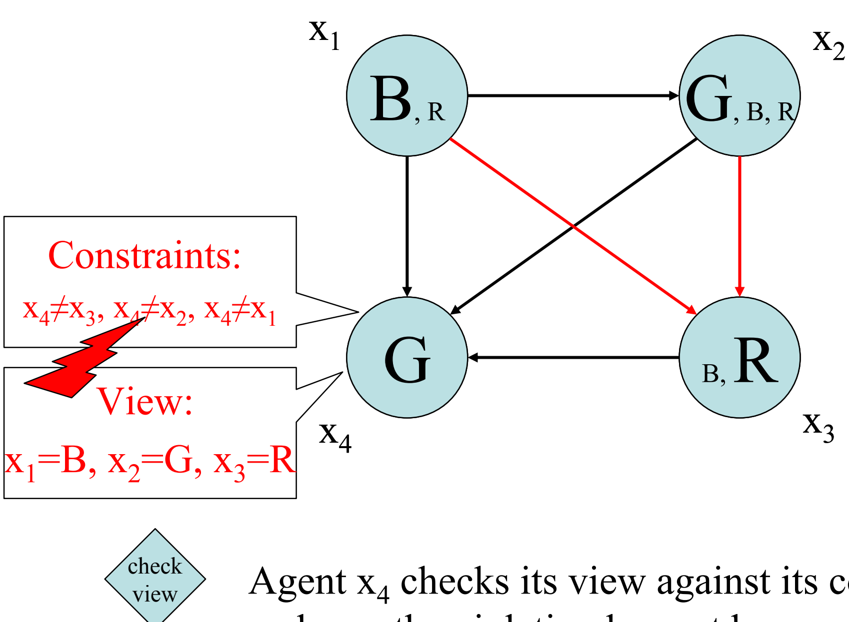
There is no more violation, so Agent x_3 communicates its new value to its children

2. Asynchronous Backtracking: The Graph Coloring Example



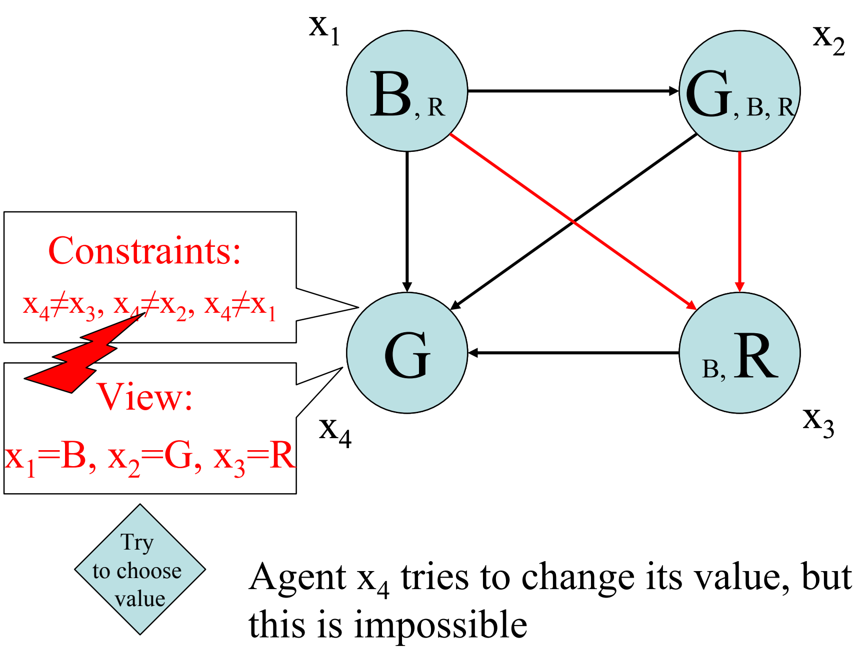
Agent x_4 receives the OK? message and updates its view

2. Asynchronous Backtracking: The Graph Coloring Example



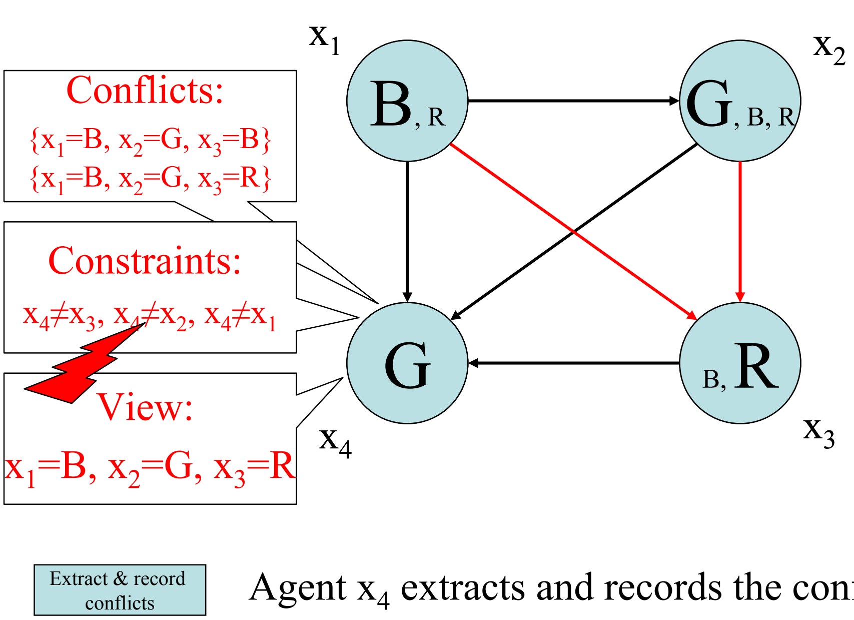
Agent x_4 checks its view against its constraints and sees the violation has not been resolved…

2. Asynchronous Backtracking: The Graph Coloring Example



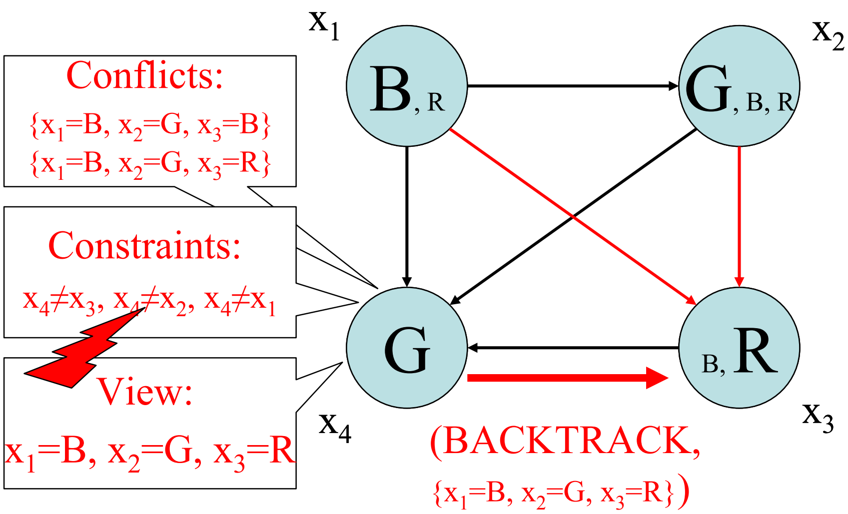
Agent x_4 tries to change its value, but this is impossible

2. Asynchronous Backtracking: The Graph Coloring Example



Agent x_4 extracts and records the conflicts

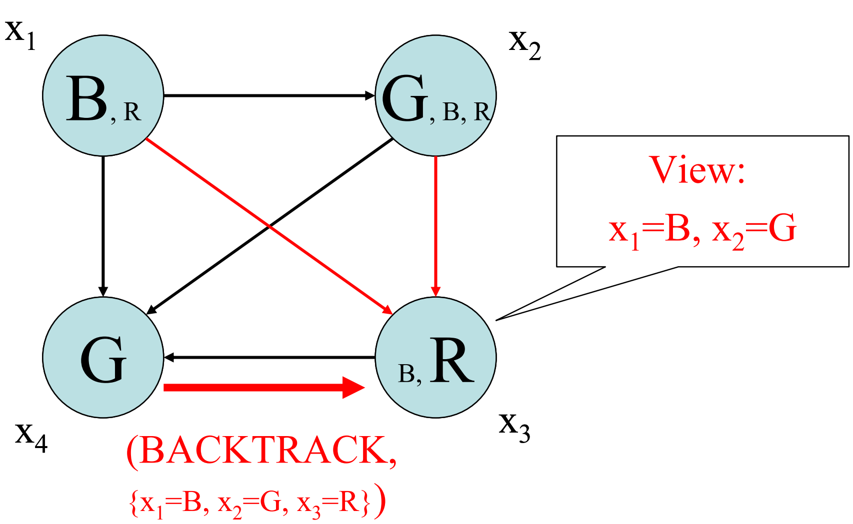
2. Asynchronous Backtracking: The Graph Coloring Example



Send BACKTRACK messages

{} is not among the new conflicts, so Agent x_4 sends BACKTRACK messages

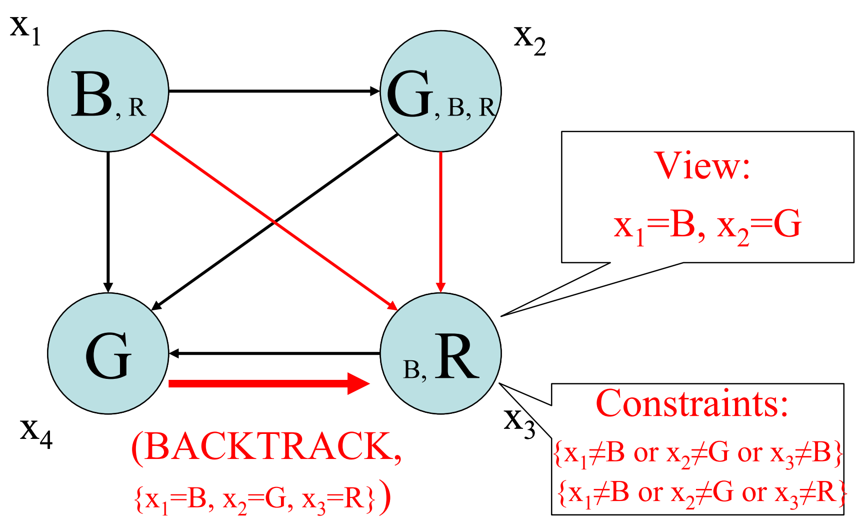
2. Asynchronous Backtracking: The Graph Coloring Example



match?

Agent x_3 receives the message and checks the conflict against its view

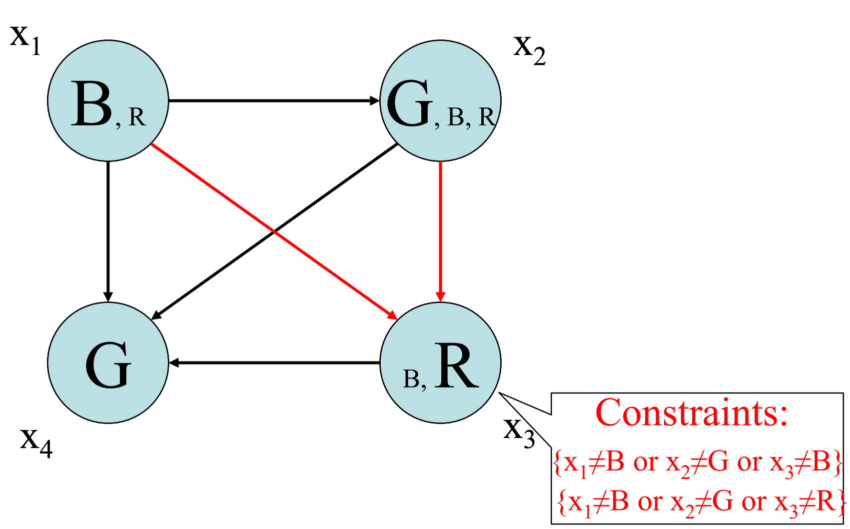
2. Asynchronous Backtracking: The Graph Coloring Example



Record new constraint

Agent x_3 records the conflict as a new constraint

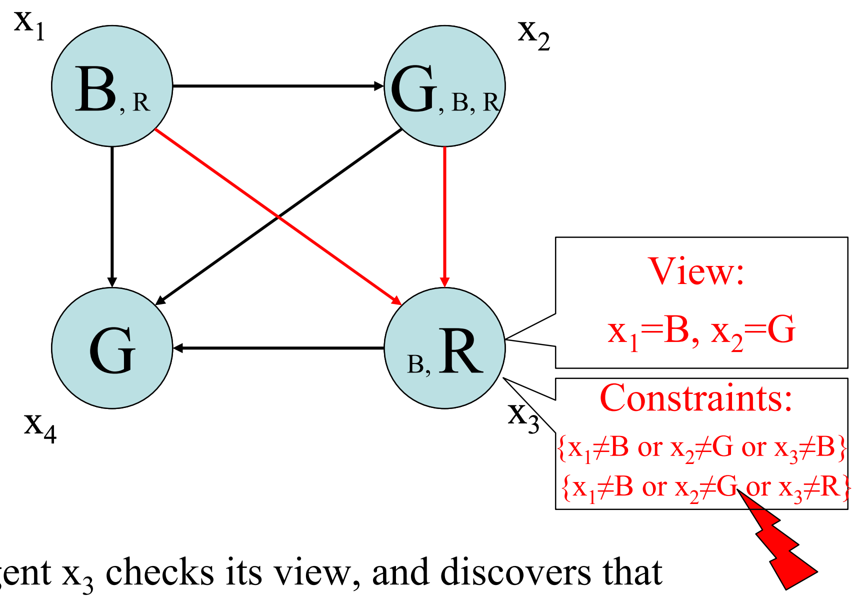
2. Asynchronous Backtracking: The Graph Coloring Example

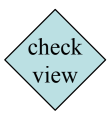


need link?

Agent x_3 needs no new link to enforce it

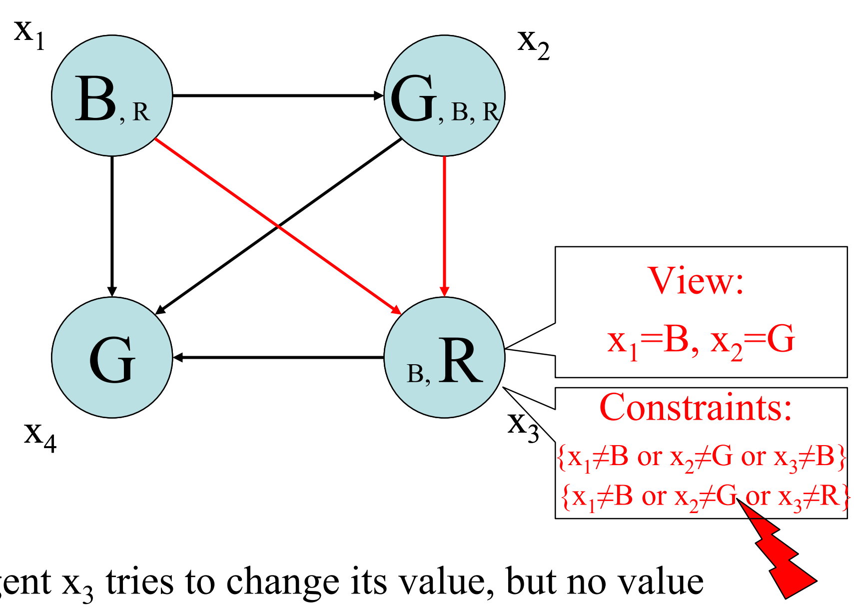
2. Asynchronous Backtracking: The Graph Coloring Example

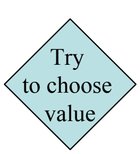




Agent x_3 checks its view, and discovers that one constraint (the new one) is violated

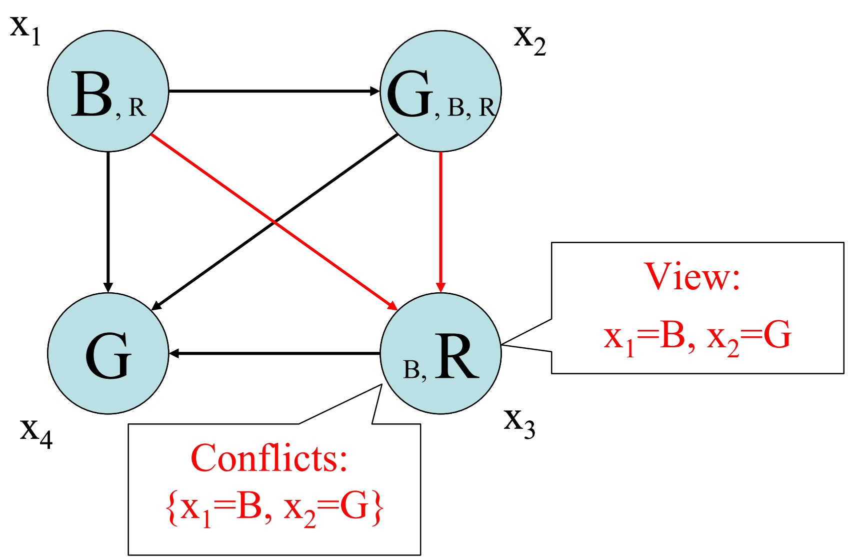
2. Asynchronous Backtracking: The Graph Coloring Example





Agent x_3 tries to change its value, but no value satisfies all the constraints

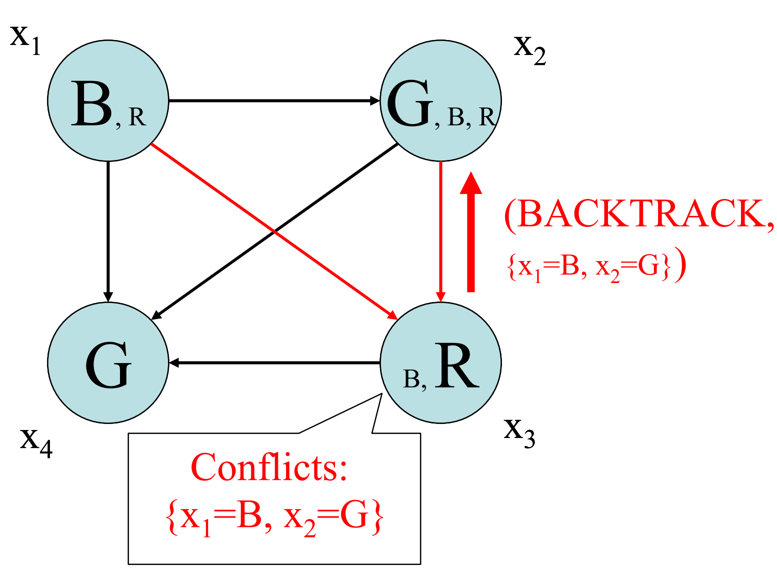
2. Asynchronous Backtracking: The Graph Coloring Example



Extract & record conflicts

Agent x_3 extracts and records new conflicts

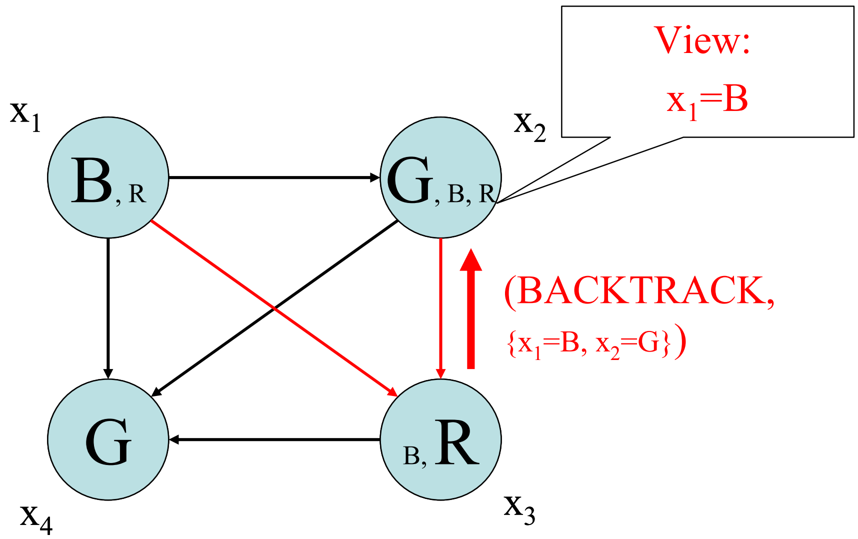
2. Asynchronous Backtracking: The Graph Coloring Example



Send BACKTRACK messages

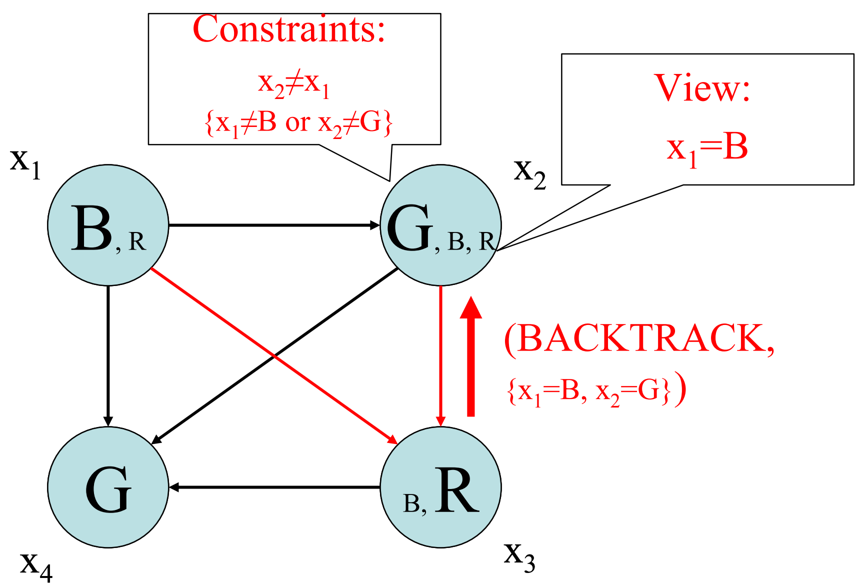
{} is not a new conflict, so Agent x_3 sends BACKTRACK messages

2. Asynchronous Backtracking: The Graph Coloring Example



Agent x_2 receives the message and checks the conflict against its view

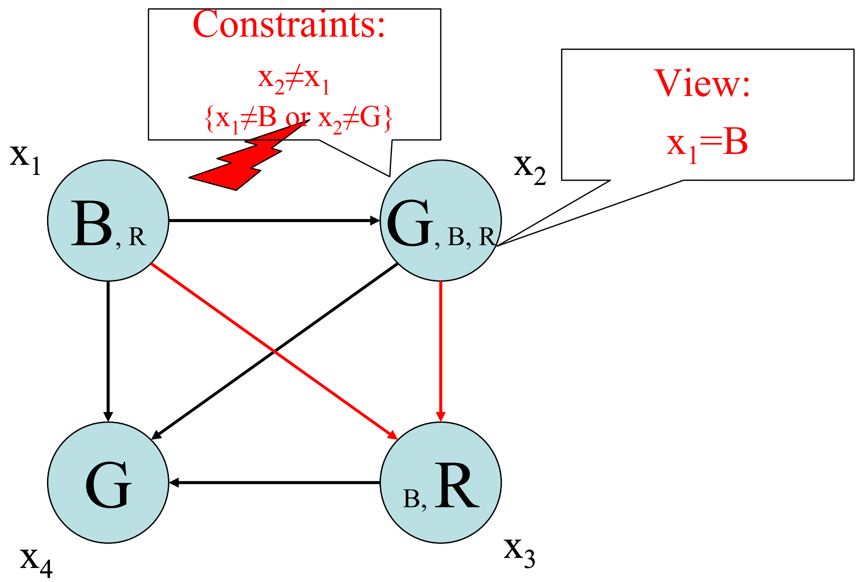
2. Asynchronous Backtracking: The Graph Coloring Example

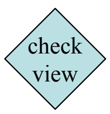


Record new constraint

Agent x_2 records the conflict as a new constraint

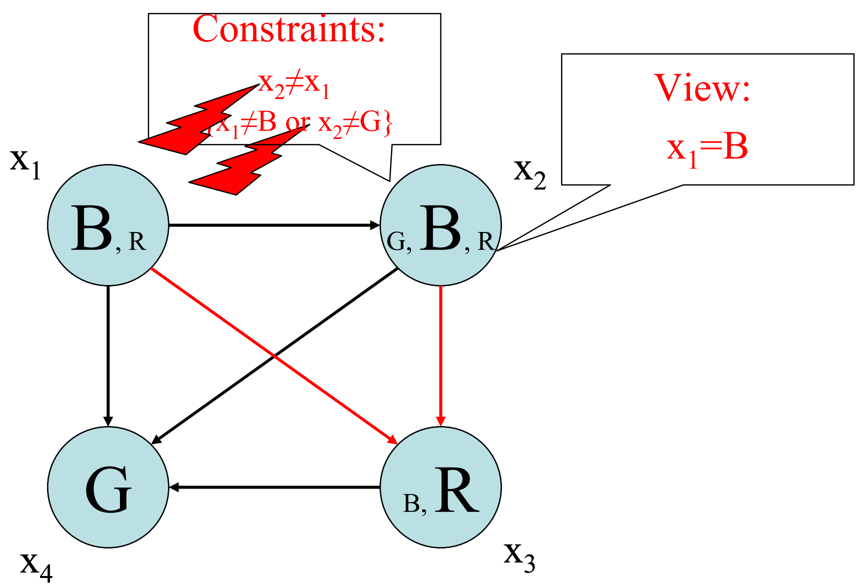
2. Asynchronous Backtracking: The Graph Coloring Example

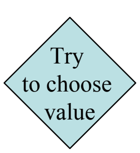




Agent x_3 checks its view, and discovers that one constraint (the new one) is violated

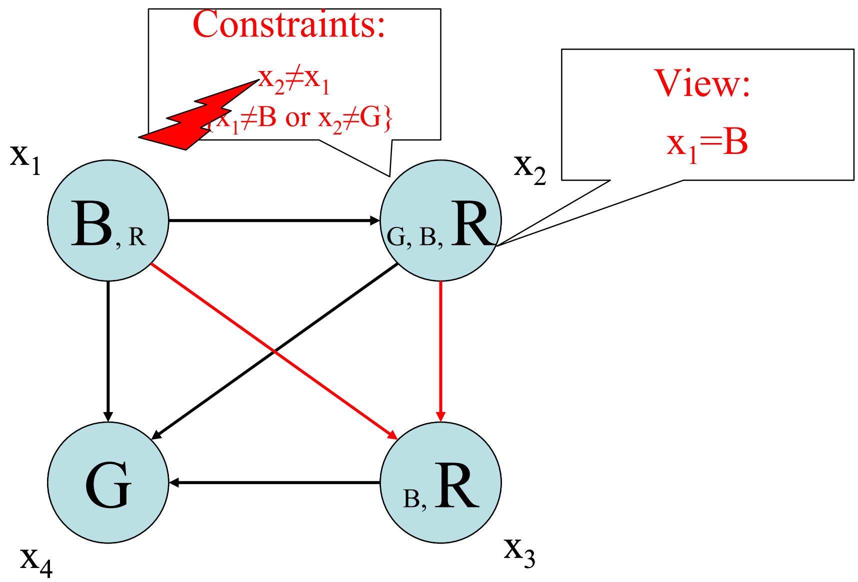
2. Asynchronous Backtracking: The Graph Coloring Example

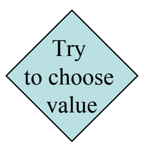




Agent x_2 tries to change its value to B, but it would violate the first constraint

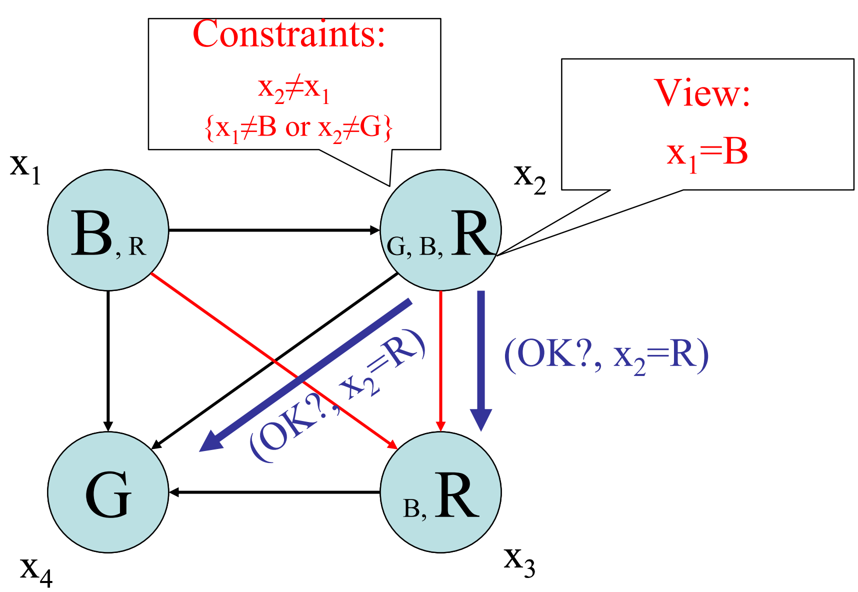
2. Asynchronous Backtracking: The Graph Coloring Example





Agent x_2 tries to change its value to R

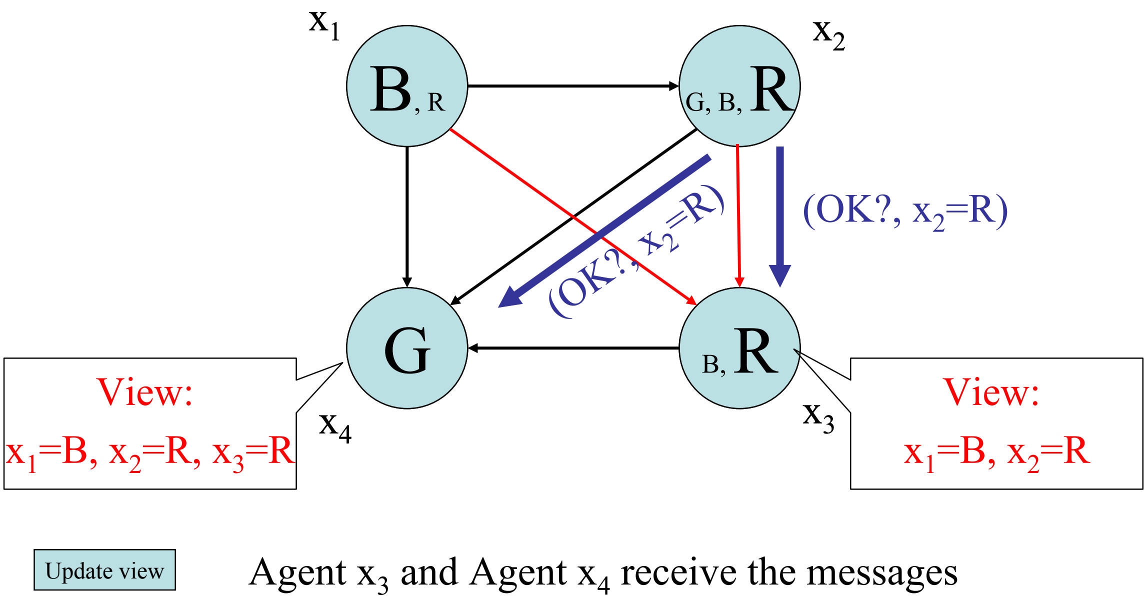
2. Asynchronous Backtracking: The Graph Coloring Example



Send OK? messages

The constraint is no longer violated, so Agent x_2 chooses value R and communicates it to its children

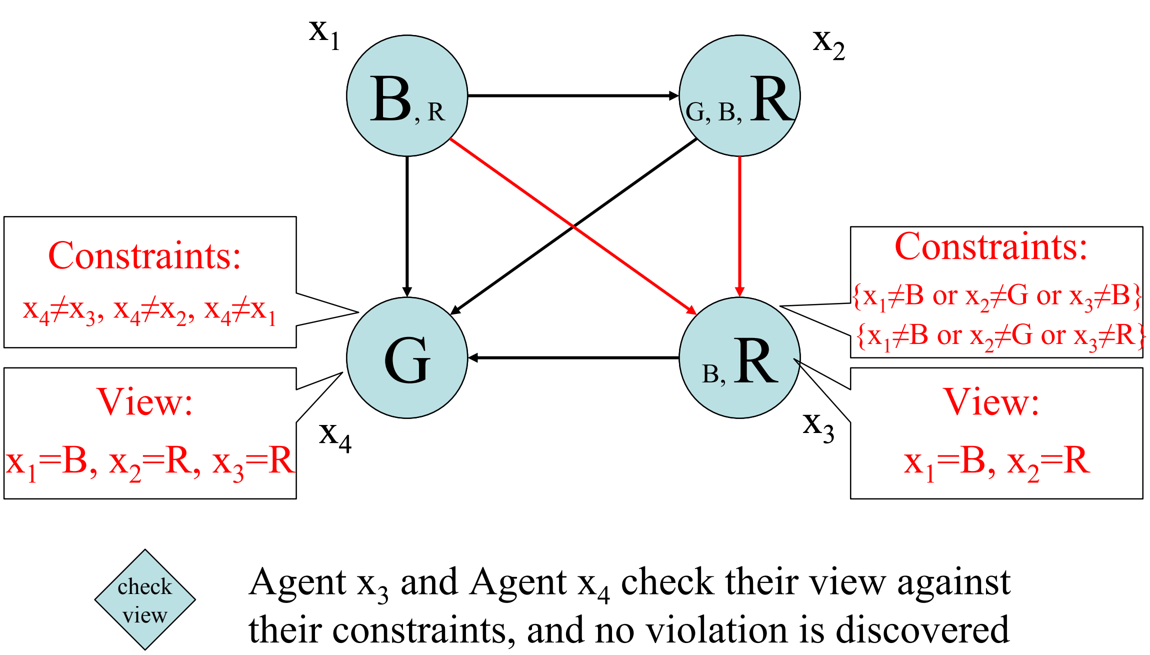
2. Asynchronous Backtracking: The Graph Coloring Example



Agent x_3 and Agent x_4 receive the messages and update their views

2. Asynchronous Backtracking: The Graph Coloring Example

SOLVED!



Agent x_3 and Agent x_4 check their view against their constraints, and no violation is discovered

Weaknesses of the Asynchronous Backtracking Algorithm

- How to better choose the assignments?
  - →Use a heuristic to make better choices
- The authors prove the algorithm always reaches a stable state within a finite number of steps, BUT it still lacks a termination procedure
  - →Use a "Distributed Snapshot" external procedure

K. M. Chandy and L. Lamport, *Distributed Snapshots: Determining Global States of Distributed Systems*, ACM Transactions on Computer Systems, 1985

Weaknesses of the Asynchronous Backtracking Algorithm (cont.)

- Need of a judicious priority ordering among the agents
  - →Do it beforehand? (might be difficult + need of a centralizing agent…)
  - →Dynamic priority ordering: let the agents come up with a judicious ordering themselves, as they encounter conflicts

Weaknesses of the Asynchronous Backtracking Algorithm (cont.)

- How to extract conflicts?
  - →Open to all conflict extraction policies
  - →There is a trade off between taking the time to extract minimal conflicts, and trying to speed up the algorithm by using the agent's view as a super-conflict but wasting time by backtracking more often

possible
Try to choose value
impossible
Extract & record conflicts

3. The Asynchronous Weak-Commitment Search Algorithm

OK? message
Wait
BACKTRACK message
match?
no
NEW_CONST
NO_SOLUTION
yes
Record constraint and neighbor
Terminate
Record new constraint
Send OK?
Update view
Wait
Send NEW_CONST messages
good
check view
violation!

3. The Asynchronous Weak-Commitment Search Algorithm

- Use a local min-conflict heuristic to guide the choices of assignments
- Judiciously change the priority ordering every time the search needs to backtrack

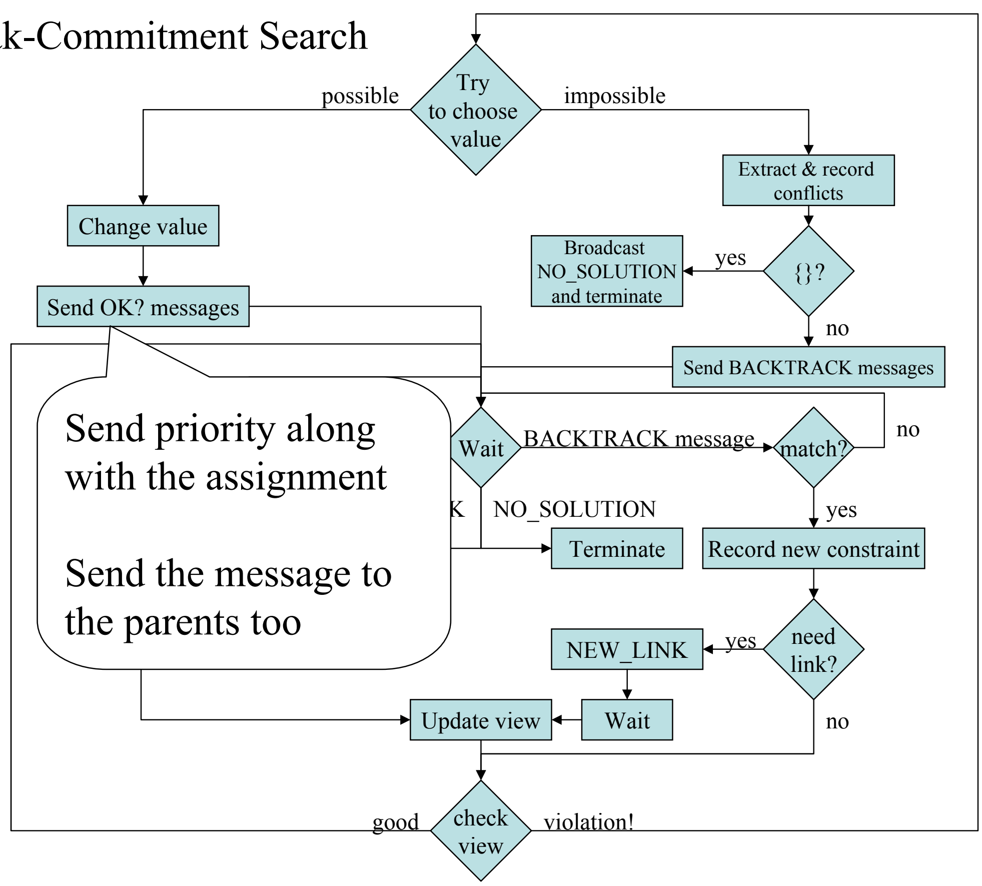
3. The Asynchronous Weak-Commitment Search Algorithm (cont.)

- “Weak-Commitment” Search:
  - A partial assignment to the variables is constructed step by step by extending it to variables with lower priority
  - The group of agents “weakly commits” itself to the partial assignment because the partial assignment is abandoned as soon as the algorithm needs to backtrack
  - The priority ordering is then modified so that the agent which failed to find a value to its variable consistent with the constraints “promotes itself” (i.e. it changes its priority value to locally become the agent with the highest priority)

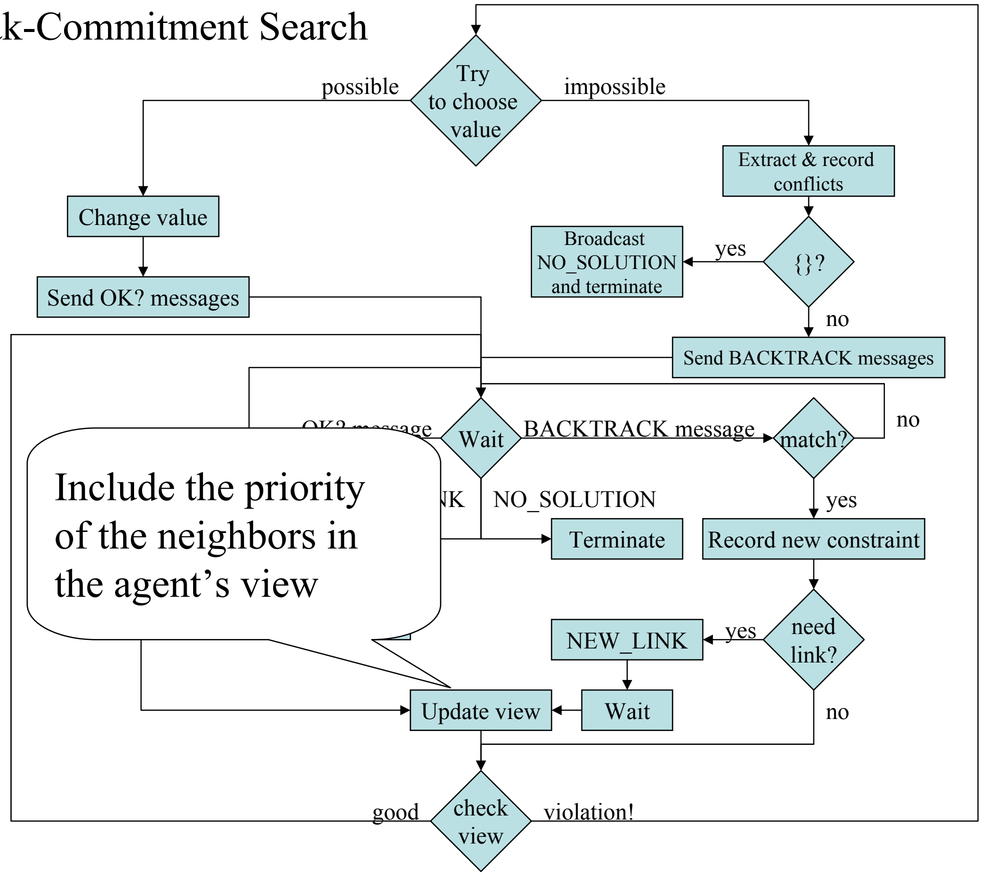
3. The Asynchronous Weak-Commitment Search Algorithm (cont.)

- ATTENTION! Tricky point:
  - Every time an agent discovers a known conflict in its view, it will abandon the partial solution
  - However, if, due to message delays, the agent's view is obsolete, it will abandon the partial assignment too early and perform an unnecessary change in its priority value
  - To avoid reacting to such unstable situations, the agent records the conflicts it has already sent, and it will temporarily ignore a conflict if it has already sent it before

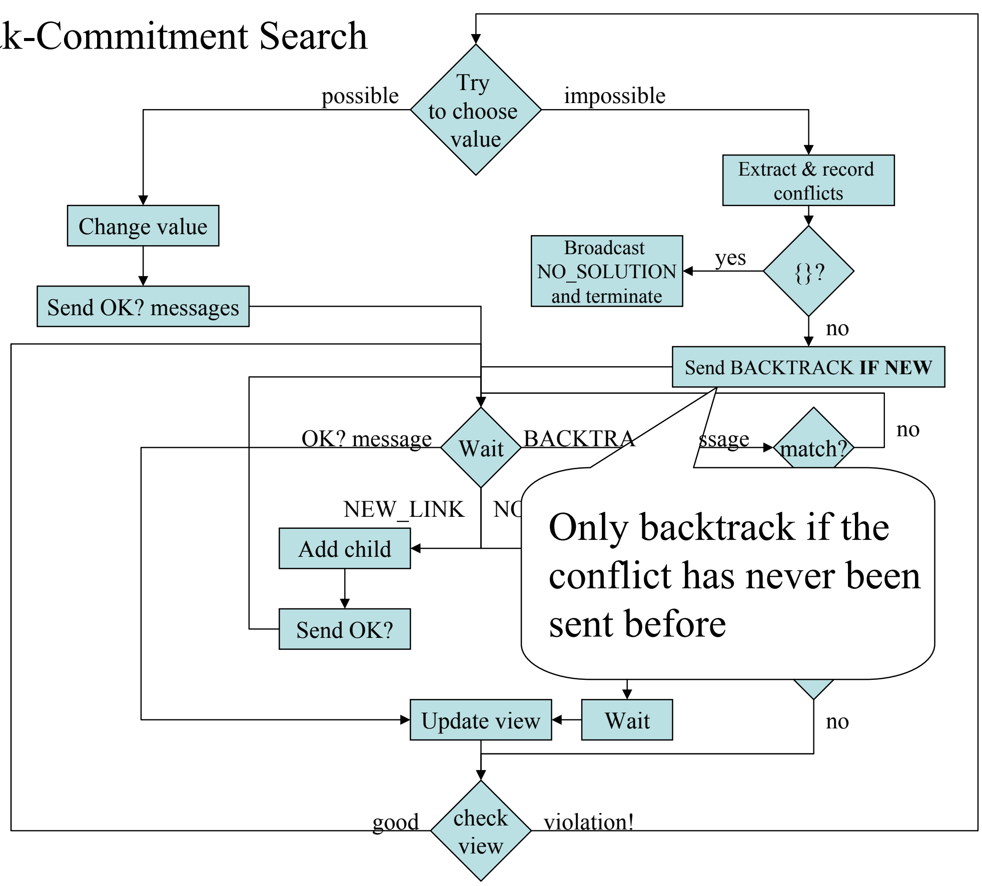
3. Weak-Commitment Search



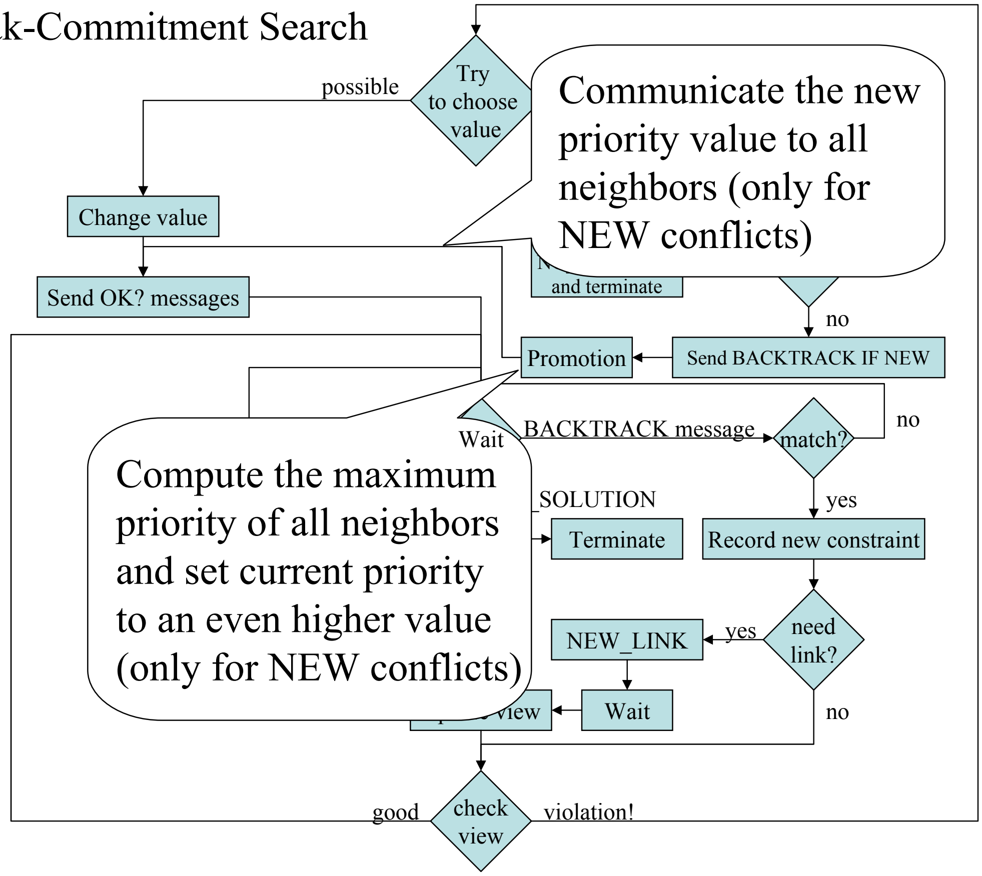
3. Weak-Commitment Search



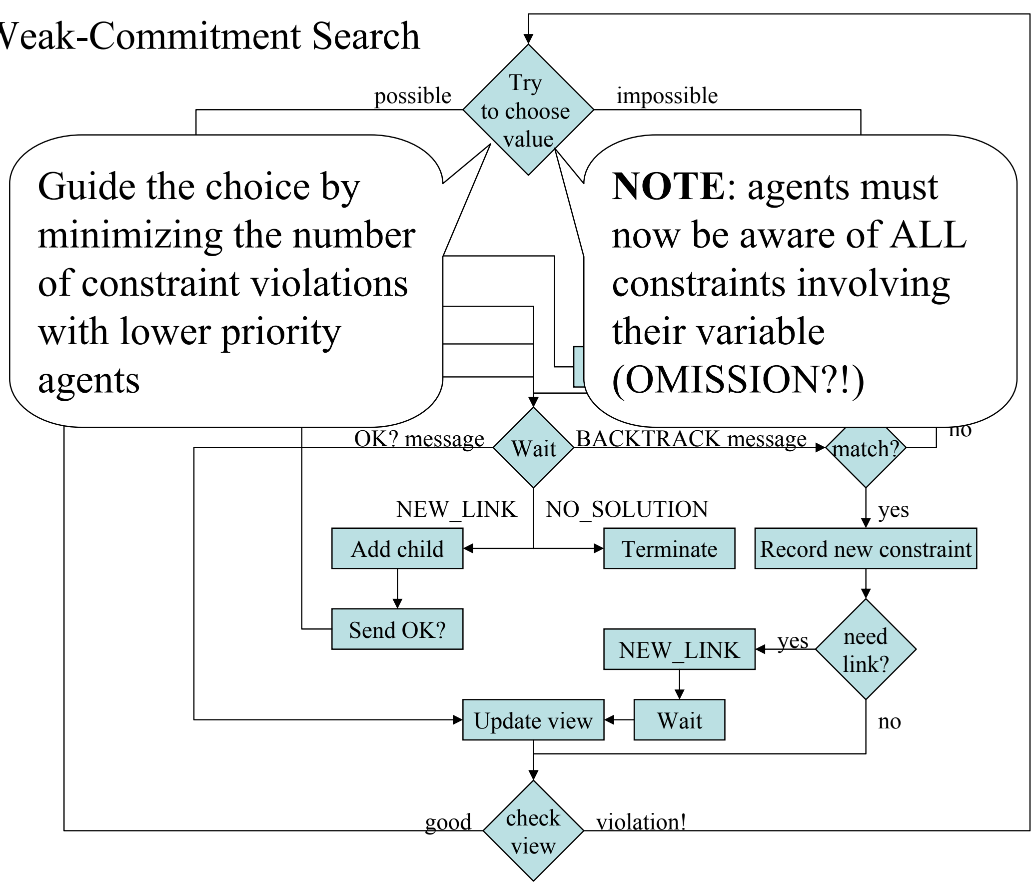
3. Weak-Commitment Search



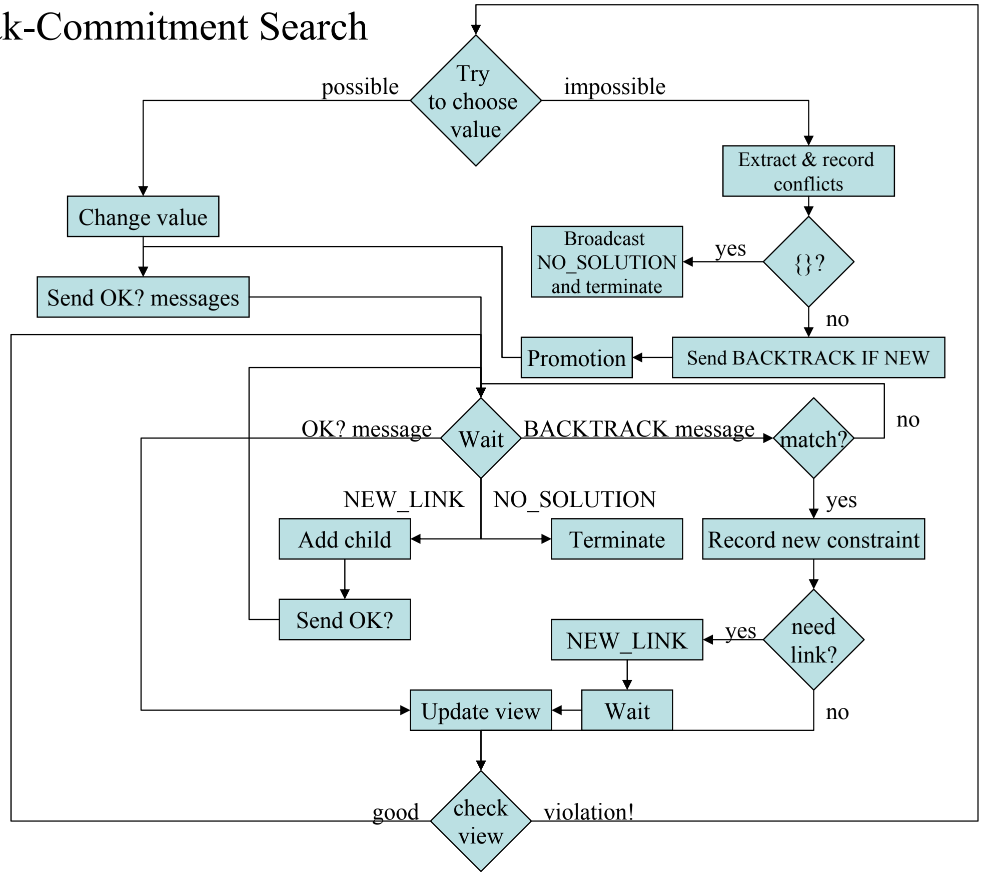
3. Weak-Commitment Search



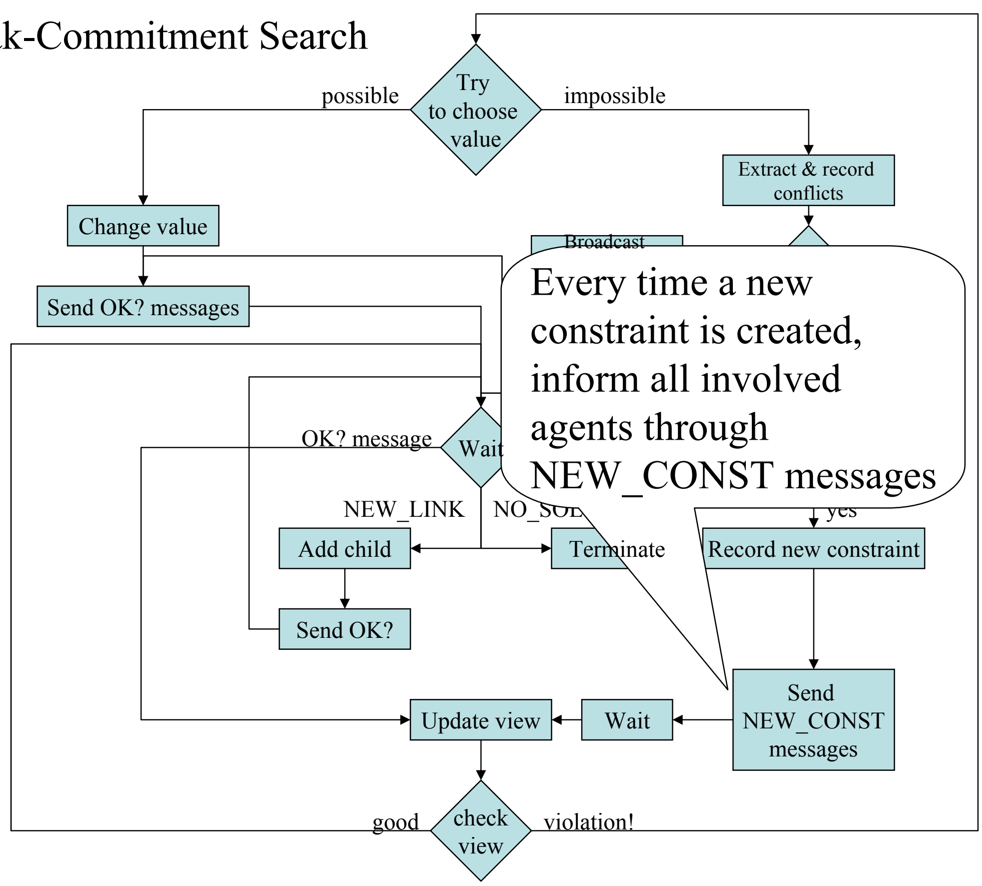
3. Weak-Commitment Search



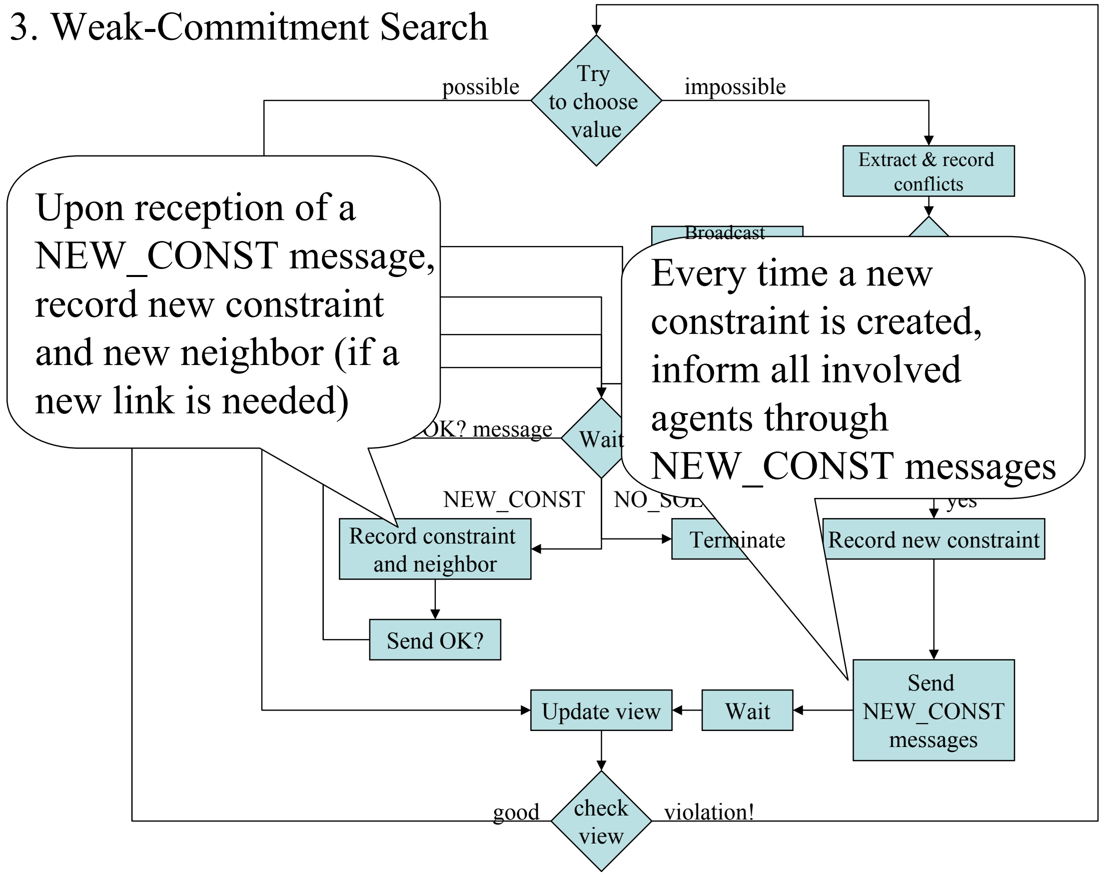
3. Weak-Commitment Search



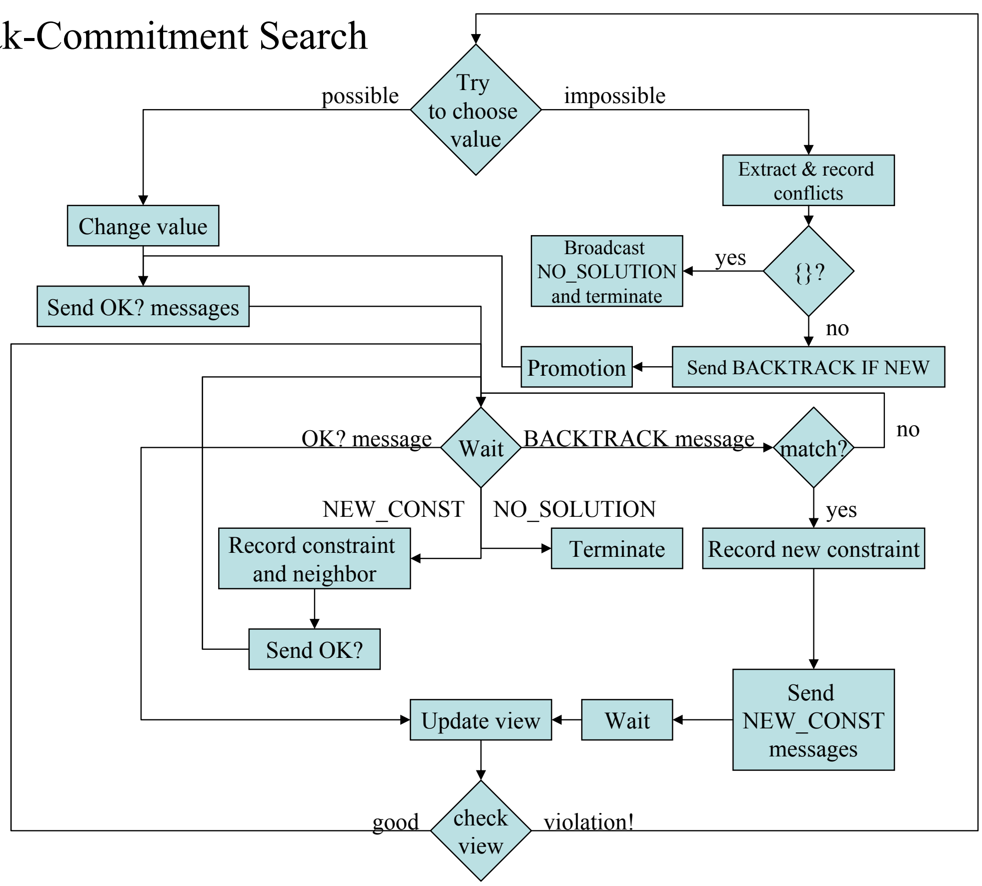
3. Weak-Commitment Search



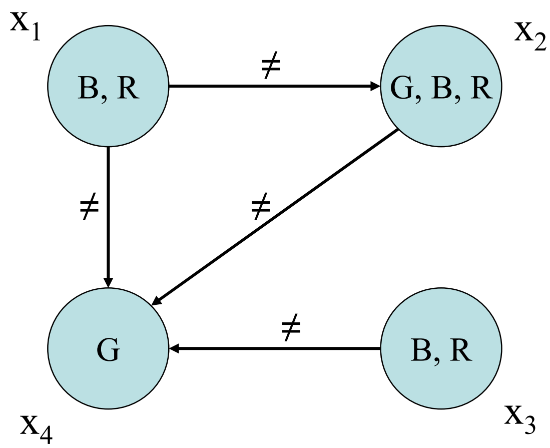
3. Weak-Commitment Search



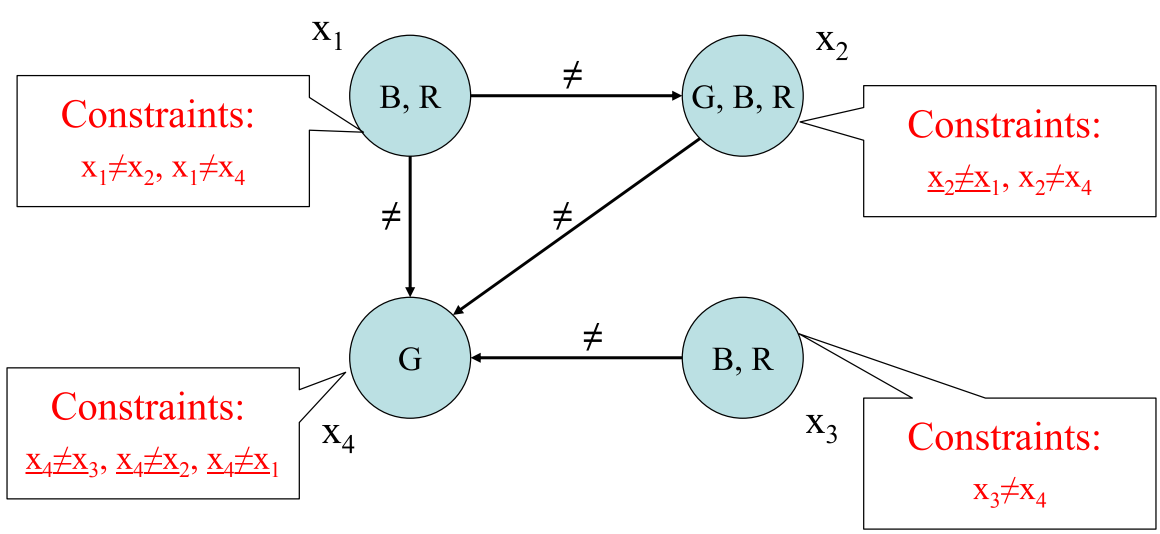
3. Weak-Commitment Search



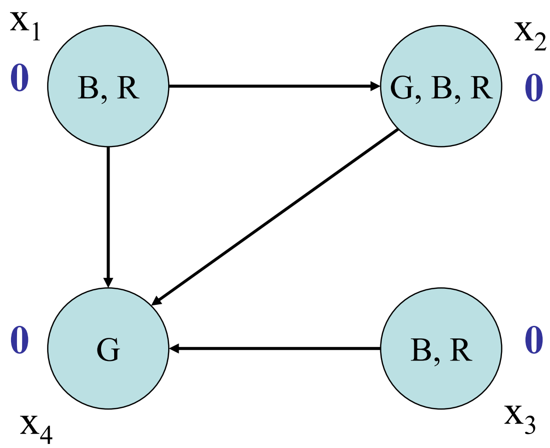
3. Weak-Commitment Search: The Graph Coloring Example



3. Weak-Commitment Search: The Graph Coloring Example

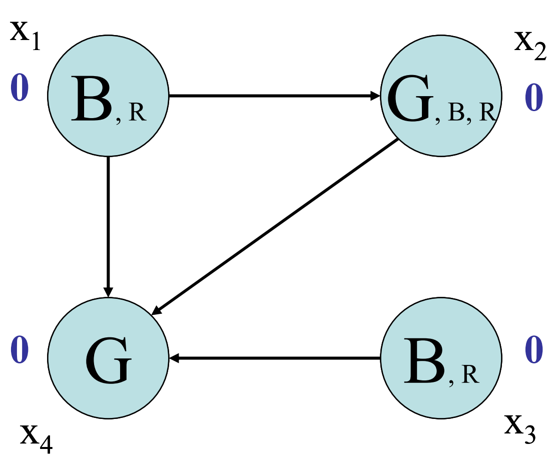


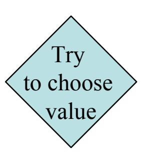
3. Weak-Commitment Search: The Graph Coloring Example



Initial priority values are all set to 0. Two agents with identical priorities are ordered with respect to their index

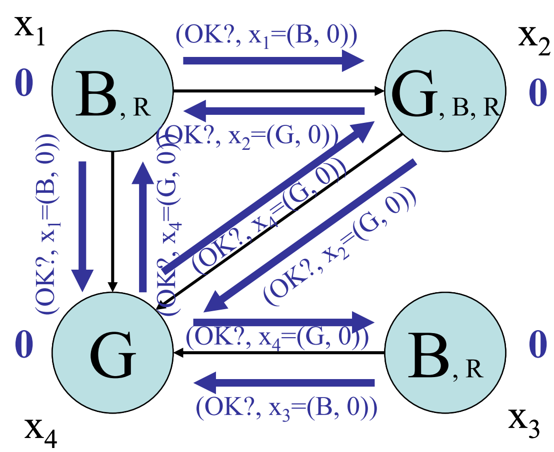
3. Weak-Commitment Search: The Graph Coloring Example





Each agent chooses an assignment to its variable (at the first time step, we cannot use the heuristic because agents still have empty views)

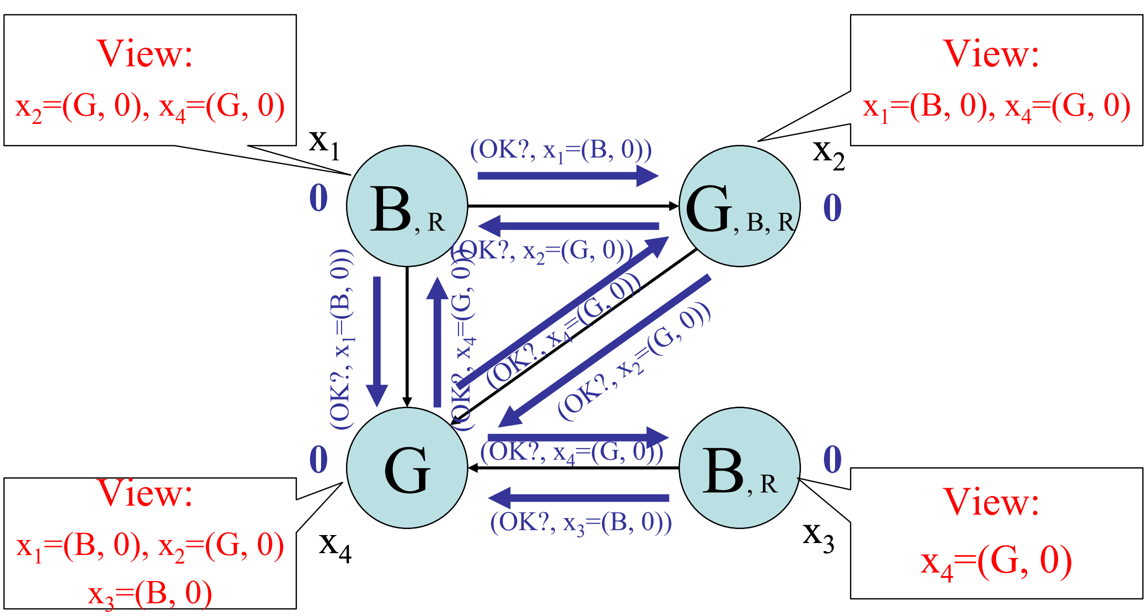
3. Weak-Commitment Search: The Graph Coloring Example



Send OK? messages

Each agent sends OK? messages to ALL of its neighbors

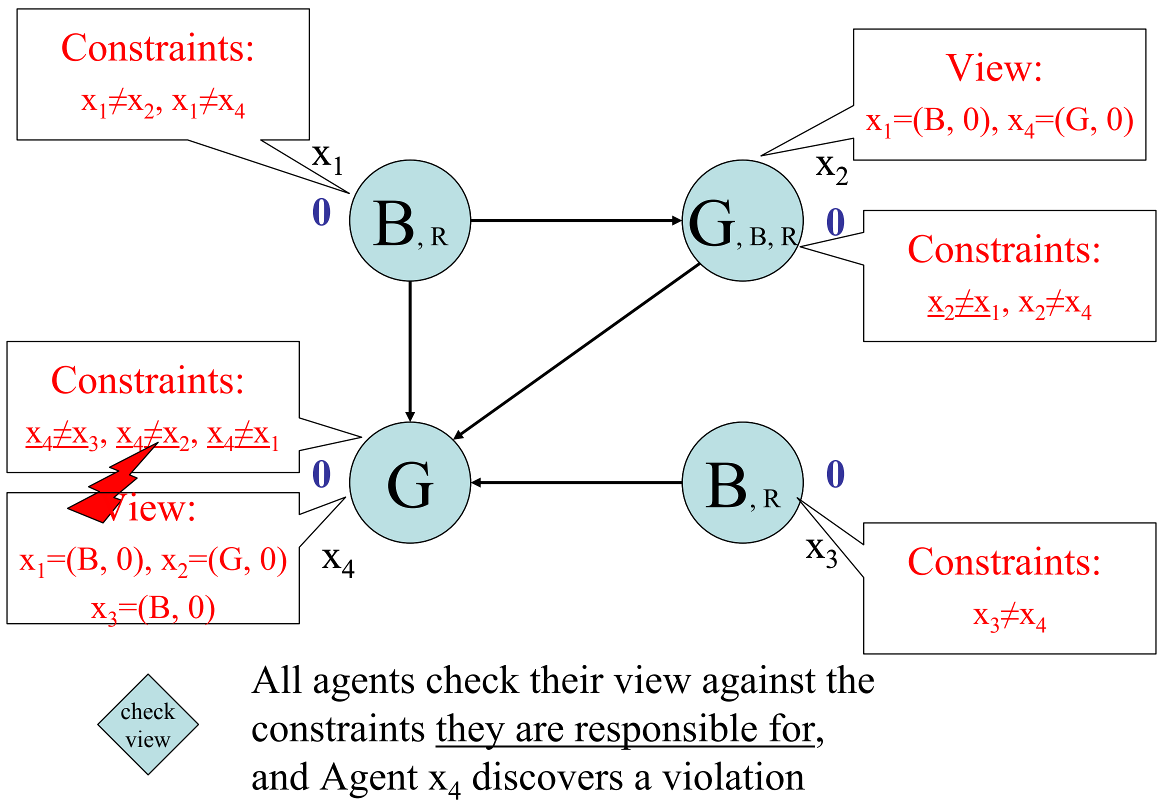
3. Weak-Commitment Search: The Graph Coloring Example



Update view

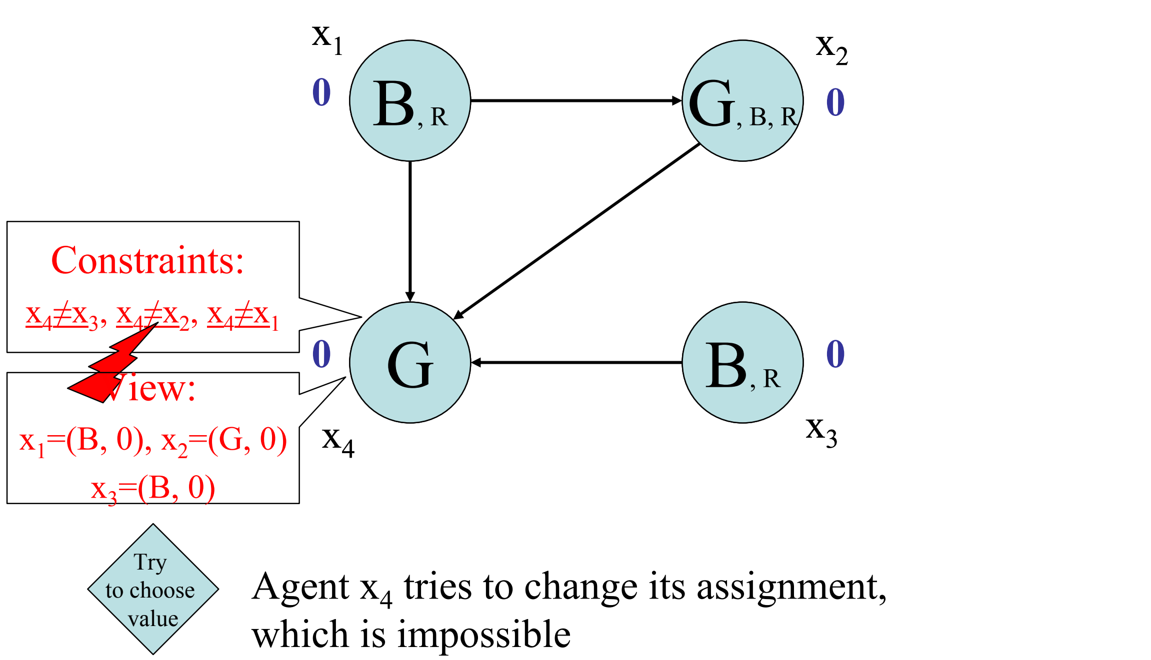
All agents update their view

3. Weak-Commitment Search: The Graph Coloring Example



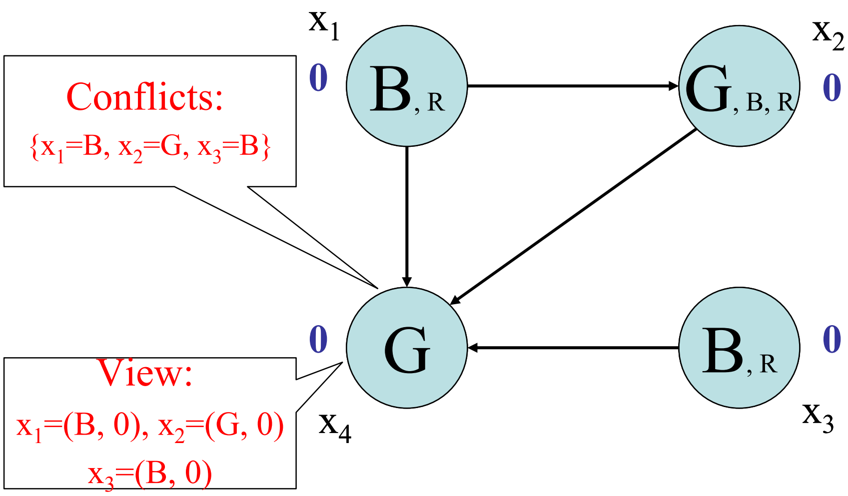
All agents check their view against the constraints they are responsible for, and Agent x_4 discovers a violation

3. Weak-Commitment Search: The Graph Coloring Example



Agent x_4 tries to change its assignment, which is impossible

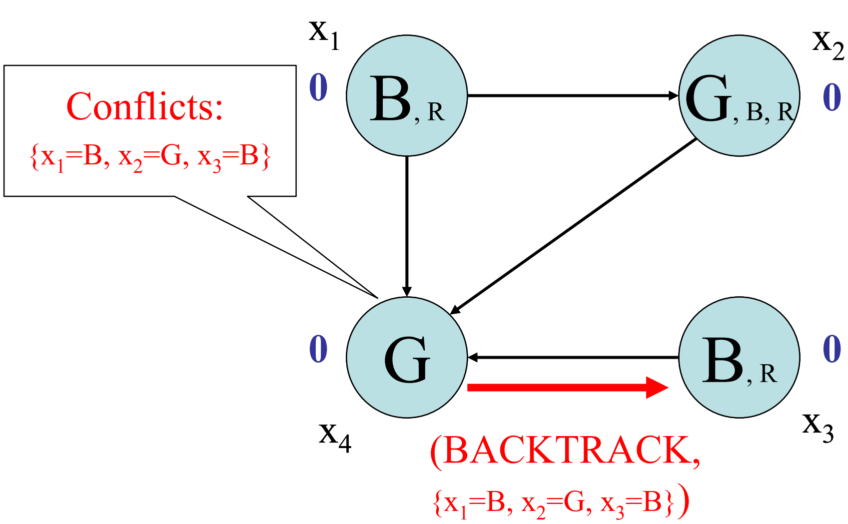
3. Weak-Commitment Search: The Graph Coloring Example



Extract & record conflicts

Agent x_4 extracts and records the conflicts

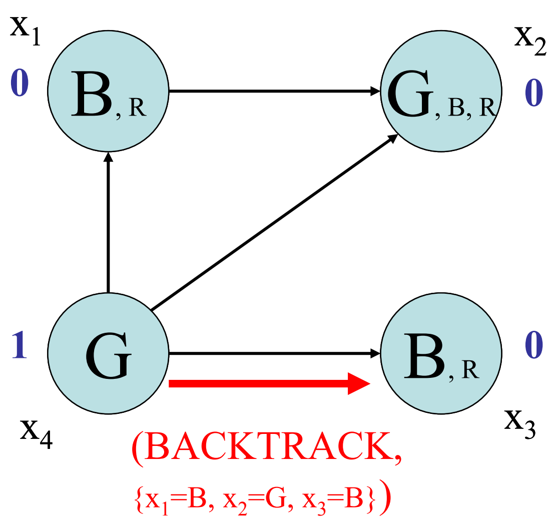
3. Weak-Commitment Search: The Graph Coloring Example



Send BACKTRACK messages

{} is not among the new conflicts, and no new conflict has already been sent, so Agent x_4 sends BACKTRACK messages

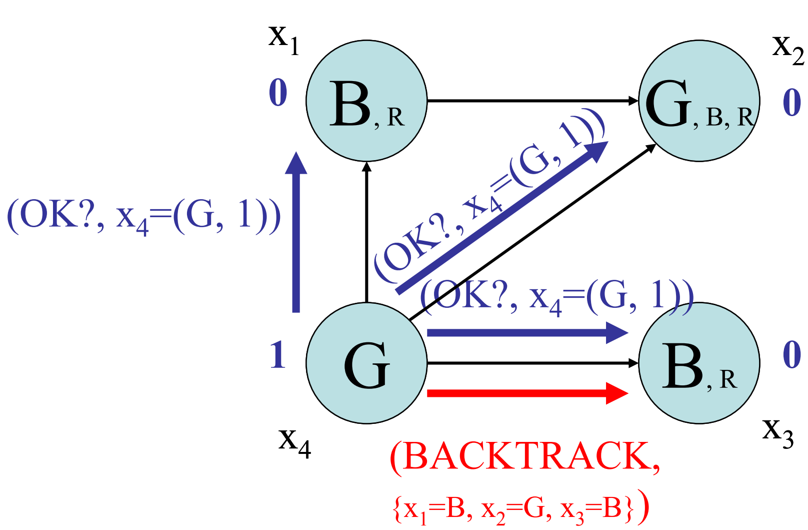
3. Weak-Commitment Search: The Graph Coloring Example



Promotion

Agent x_4 promotes itself, changing its priority value from 0 to 1

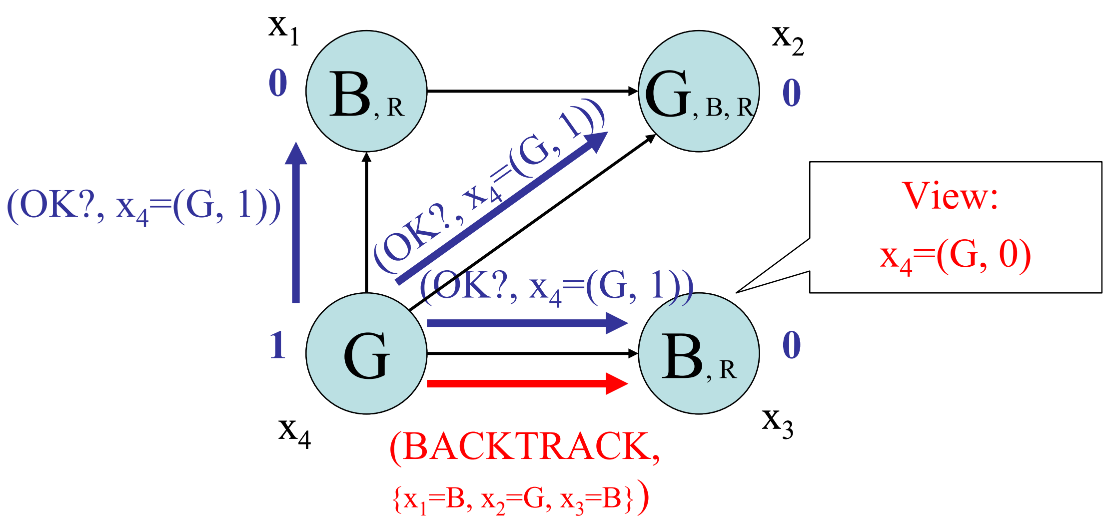
3. Weak-Commitment Search: The Graph Coloring Example

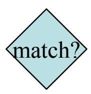


Send OK? messages

Agent x_4 communicates its new priority value to ALL its neighbors

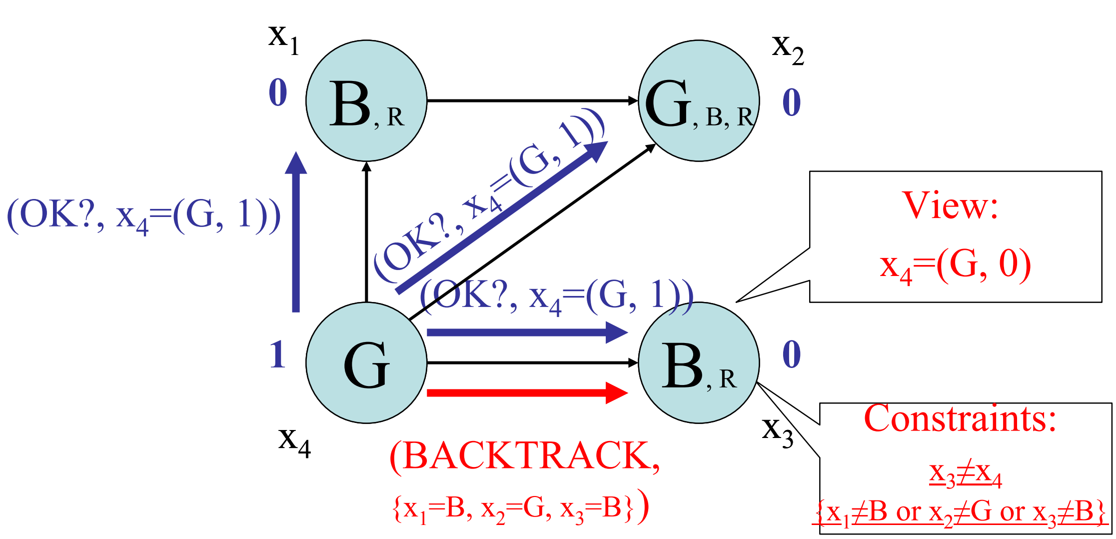
3. Weak-Commitment Search: The Graph Coloring Example

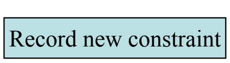




CONCURRENTLY, Agent x_3 receives the BACKTRACK message and checks the conflict against its view

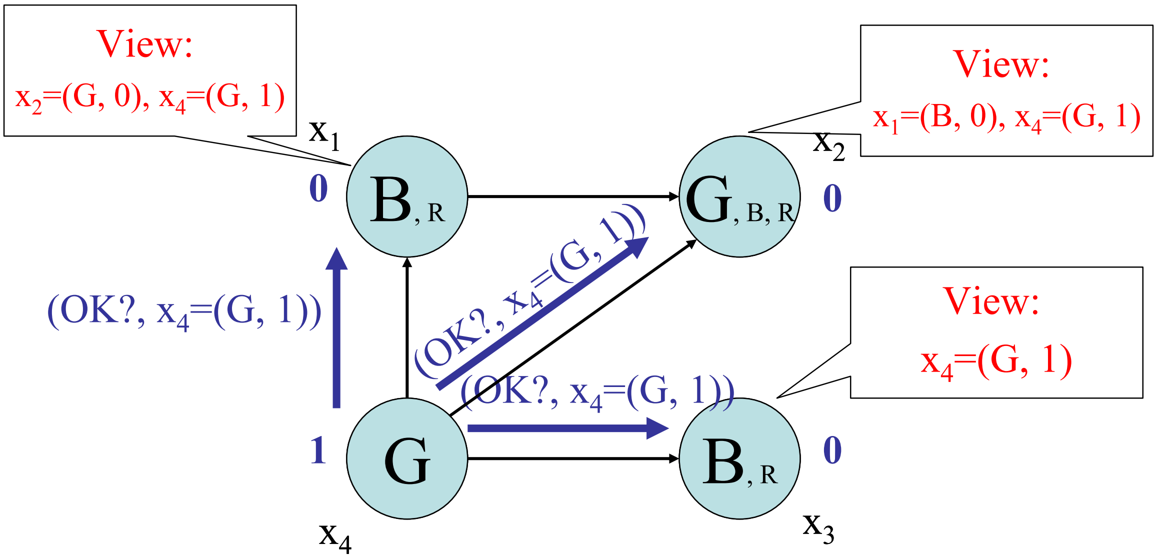
3. Weak-Commitment Search: The Graph Coloring Example





Agent x_3 records the conflict as a new constraint it will be <u>responsible for</u>

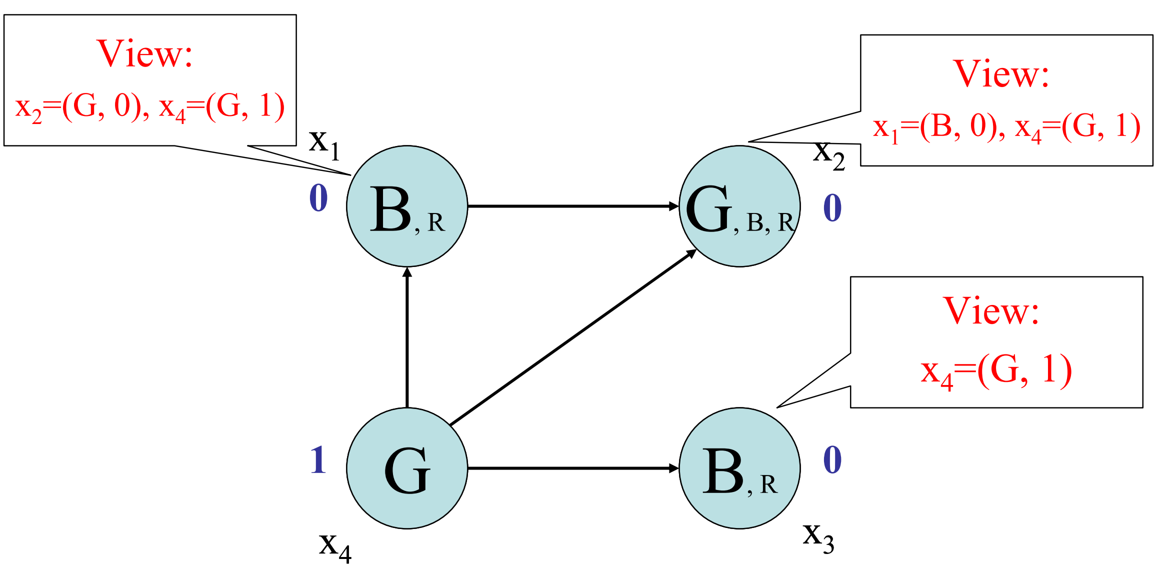
3. Weak-Commitment Search: The Graph Coloring Example



Update view

CONCURRENTLY, all agents receive the OK? messages and update their view

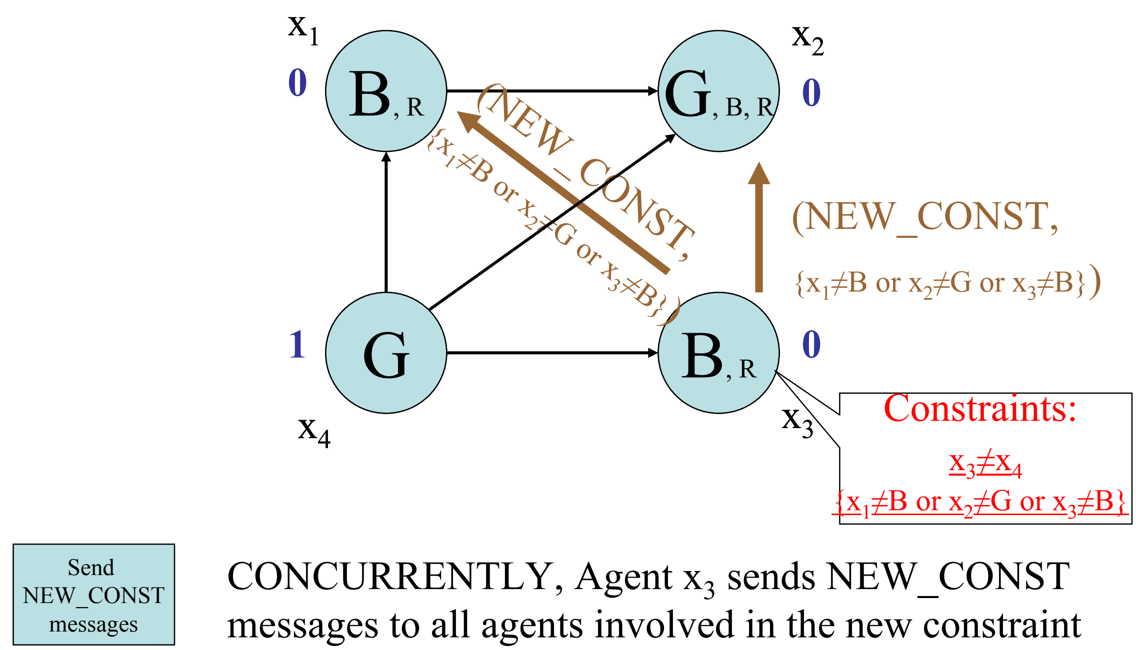
3. Weak-Commitment Search: The Graph Coloring Example



check view

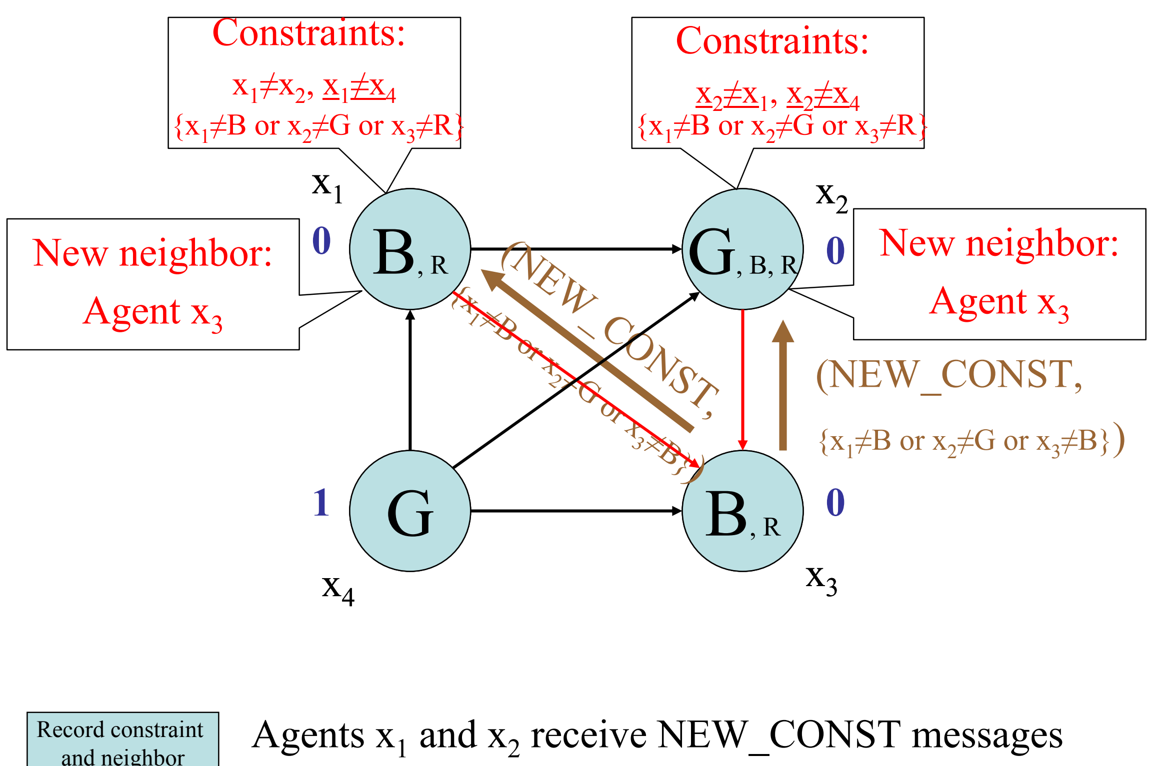
(Only the priority changed, so no new violation is discovered)

3. Weak-Commitment Search: The Graph Coloring Example



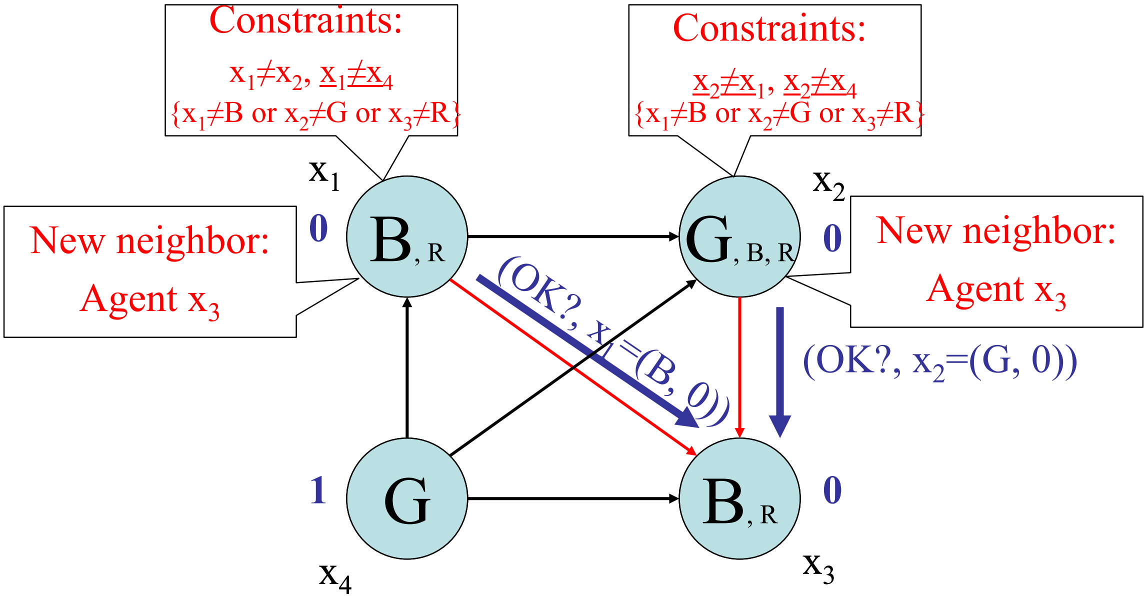
CONCURRENTLY, Agent x_3 sends NEW_CONST messages to all agents involved in the new constraint

3. Weak-Commitment Search: The Graph Coloring Example



Agents x_1 and x_2 receive NEW_CONST messages and record new constraint and new neighbor

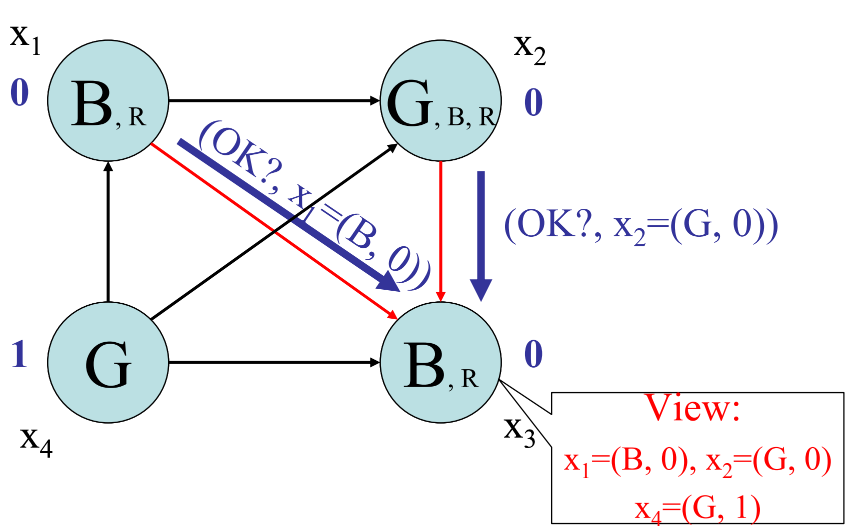
3. Weak-Commitment Search: The Graph Coloring Example



Send OK? messages

Agents x_1 and x_2 respond to the NEW_CONST through OK? messages

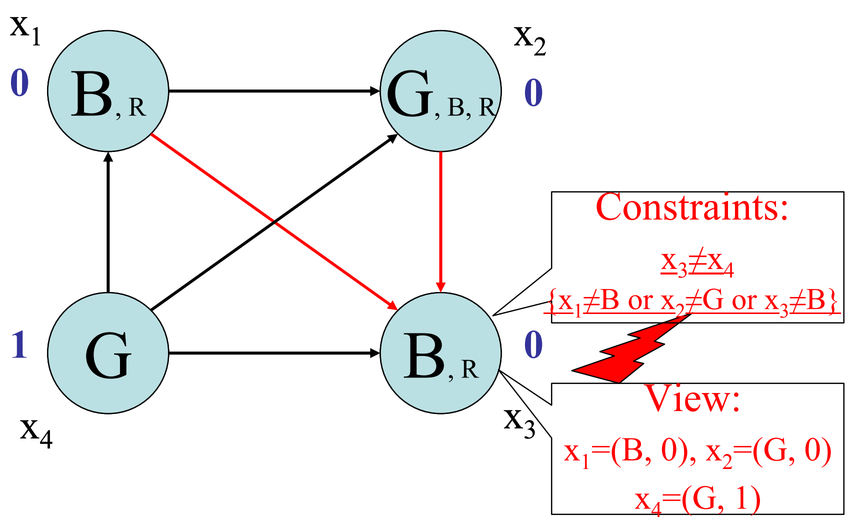
3. Weak-Commitment Search: The Graph Coloring Example

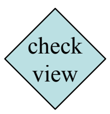


Update view

Agent x_3 receives messages and updates its view

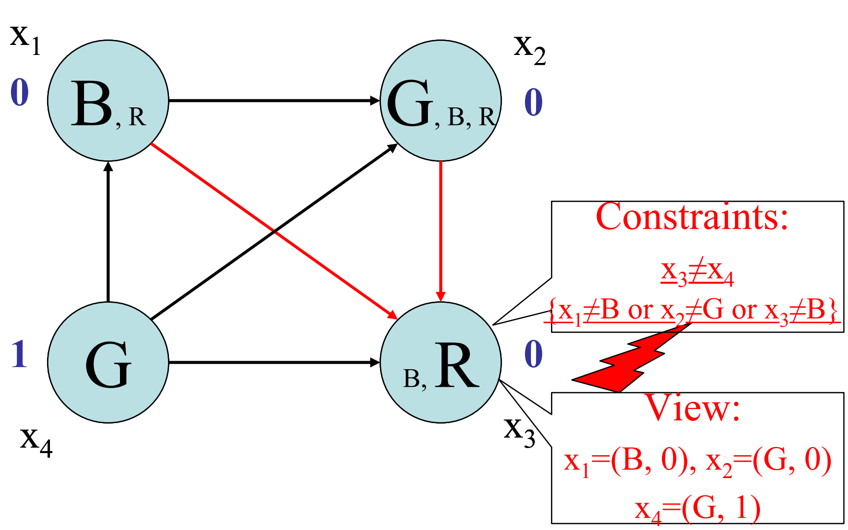
3. Weak-Commitment Search: The Graph Coloring Example

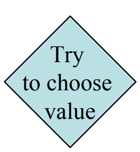




Agent x_3 checks its view, and discovers that one constraint it is responsible for (the new one) is violated

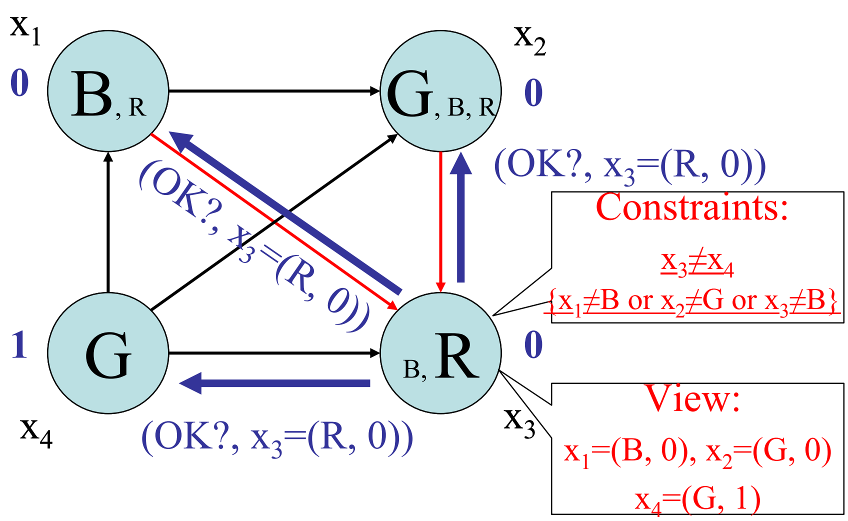
3. Weak-Commitment Search: The Graph Coloring Example





Agent x_3 tries to change its value to R, deleting all violations of constraints it is responsible for, and minimizing the number of violations of others

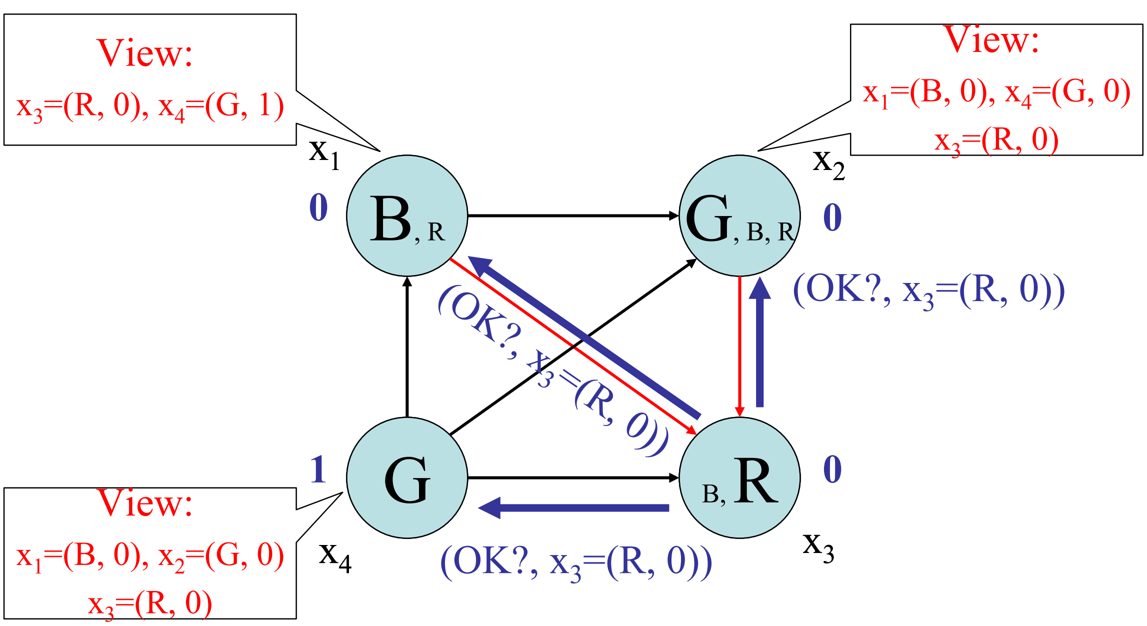
3. Weak-Commitment Search: The Graph Coloring Example



Send OK? messages

There is no more violations of constraints it is responsible for, so Agent x_3 communicates its new value to ALL neighbors

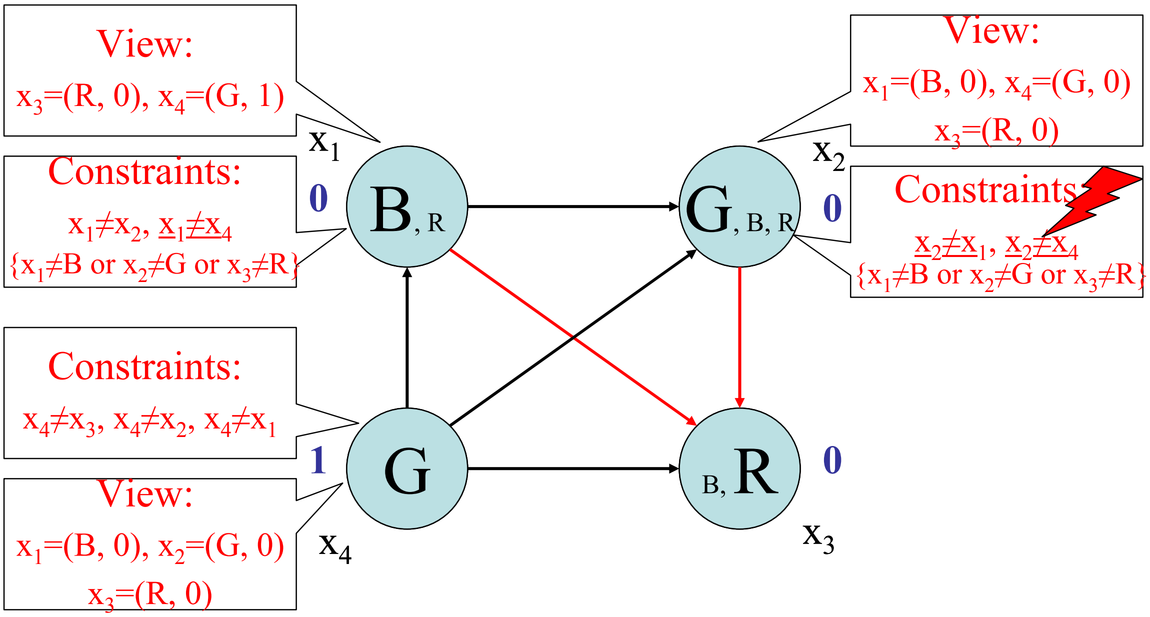
3. Weak-Commitment Search: The Graph Coloring Example

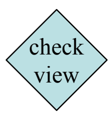


Update view

All agents receive the OK? messages and update their view

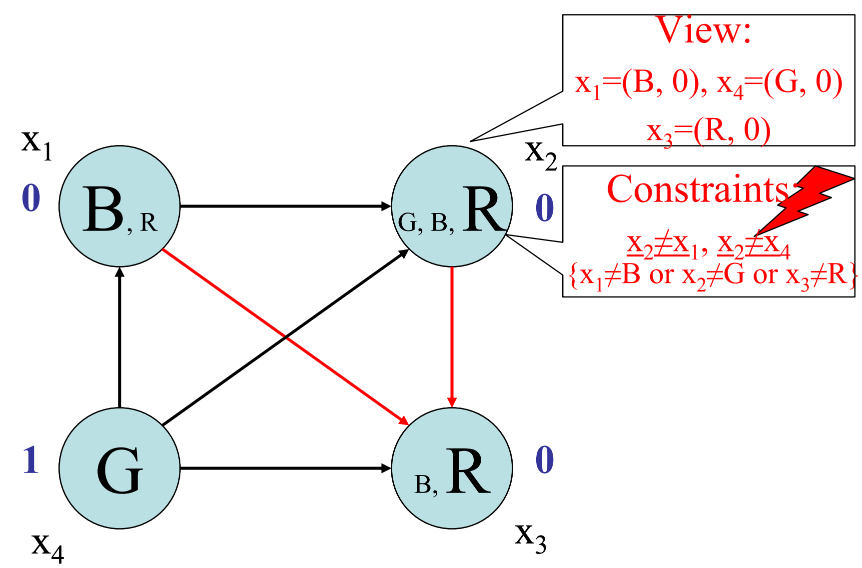
3. Weak-Commitment Search: The Graph Coloring Example

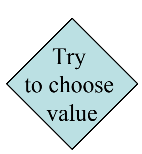




Agent x_1, x_2 and x_4 check their view against the constraints they are responsible for, and Agent x_2 discovers a violation

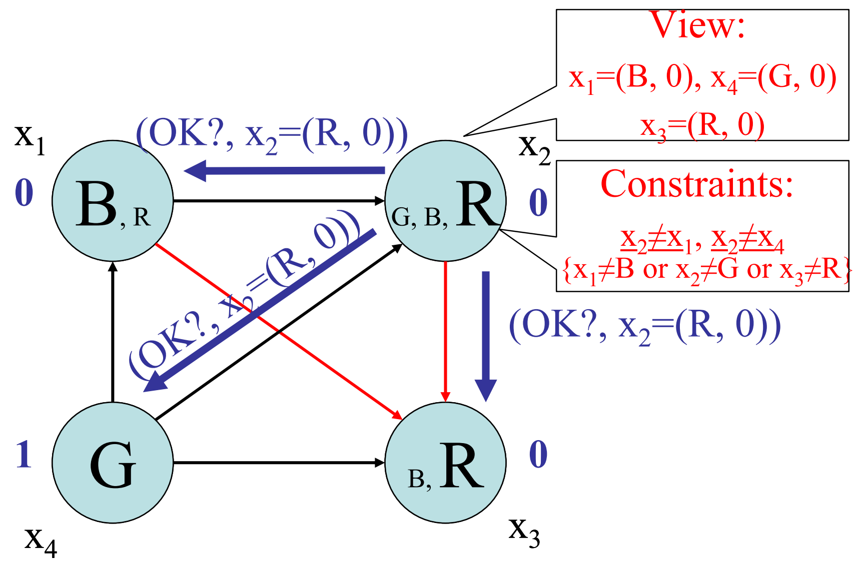
3. Weak-Commitment Search: The Graph Coloring Example





Agents x_2 tries to change its value to R, deleting all violations of constraints it is responsible for, and minimizing the number of violations of others

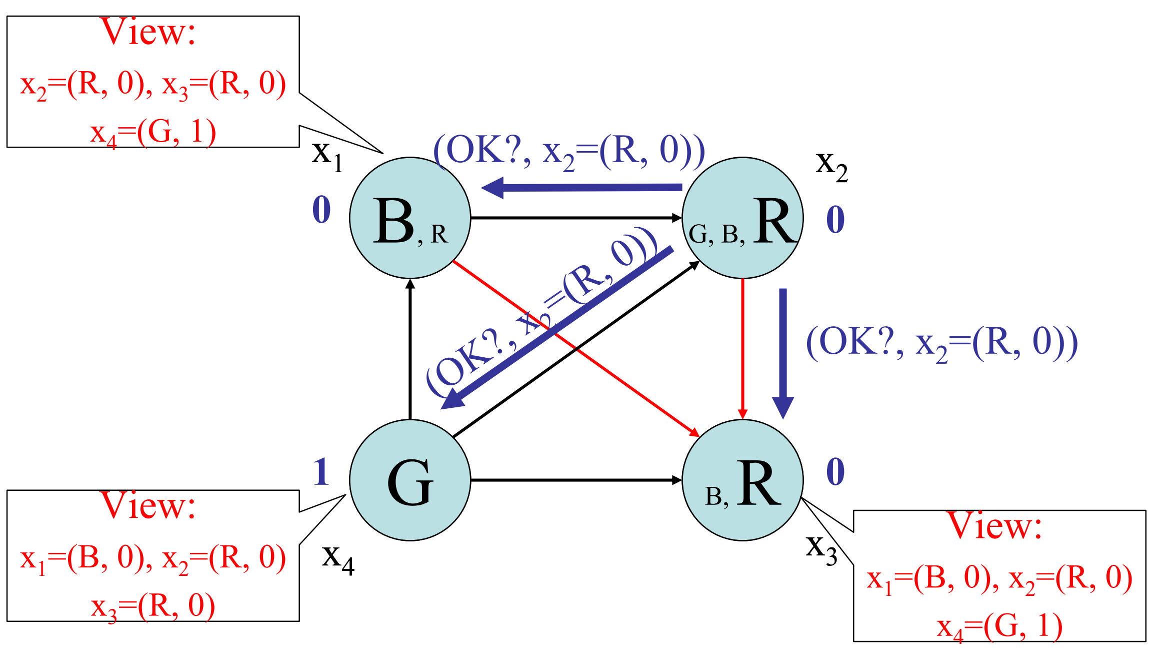
3. Weak-Commitment Search: The Graph Coloring Example



Send OK? messages

There is no more violations of constraints <u>it is responsible for</u>, so Agent x_3 communicates its new value to ALL neighbors

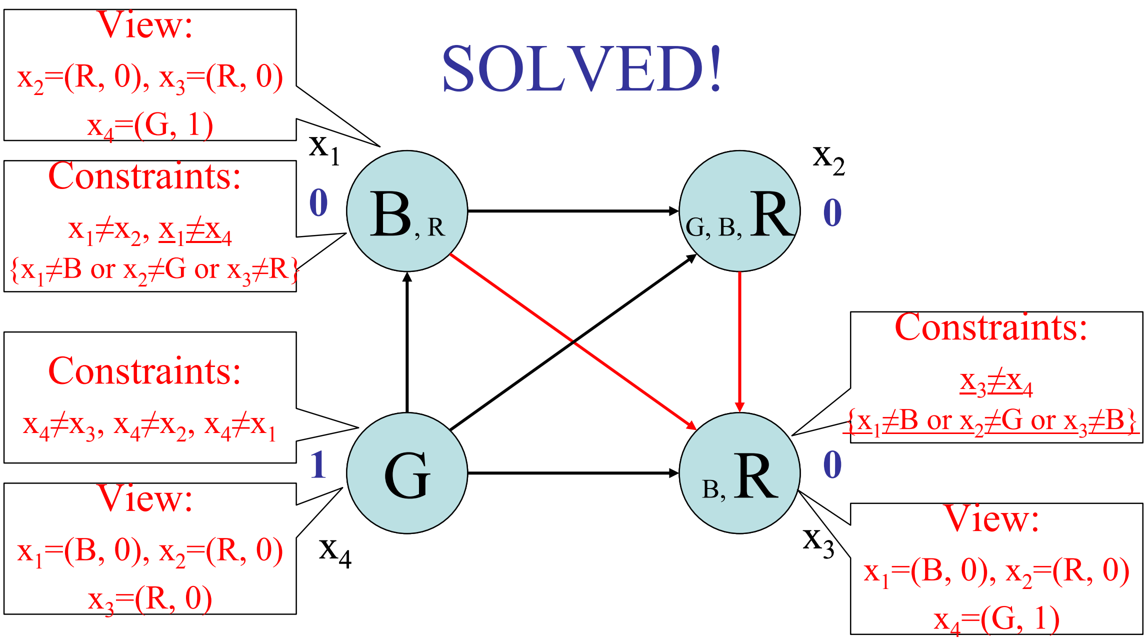
3. Weak-Commitment Search: The Graph Coloring Example

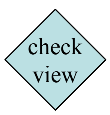


Update view

All agents receive the OK? messages and update their view

3. Weak-Commitment Search: The Graph Coloring Example





Agents x_1, x_3 and x_4 check their view against the constraints they are responsible for, and no new violation is discovered

4. Conclusion

- Perform much better than the trivial algorithms

4. Conclusion

M. Yokoo, E. Durfee, T. Ishida and K. Kuwabara, *Distributed Constraint Satisfaction Problem: Formalization and Algorithms*, IEEE Transactions on Knowledge and Data Engineering, VOL. 10, NO. 5, Sept/Oct 1998. Fig. 7.

4. Conclusion

M. Yokoo, E. Durfee, T. Ishida and K. Kuwabara, *Distributed Constraint Satisfaction Problem: Formalization and Algorithms*, IEEE Transactions on Knowledge and Data Engineering, VOL. 10, NO. 5, Sept/Oct 1998. Table 1.

4. Conclusion

- Perform much better than the trivial algorithms
- Single-variable agents => Task Allocation Problem

W.-M. Shen and B. Salemi, *Distributed and Dynamic Task Reallocation in Robot Organizations*

References

- M. Yokoo, E. Durfee, T. Ishida and K. Kuwabara, *Distributed Constraint Satisfaction Problem: Formalization and Algorithms*, IEEE Transactions on Knowledge and Data Engineering, VOL. 10, NO. 5, Sept/Oct 1998
- K. Sycara, S. Roth, N. Nadeh and M. Fox, *Distributed Constrained Heuristic Search*, IEEE Transactions on Systems, Man, and Cybernetics, VOL. 21, NO. 6, Nov/Dec 1991
- K. M. Chandy and L. Lamport, *Distributed Snapshots: Determining Global States of Distributed Systems*, ACM Transactions on Computer Systems, 1985
- W.-M. Shen and B. Salemi, *Distributed and Dynamic Task Reallocation in Robot Organizations*