سیستم‌های چندعاملی

درس ۲۶

ارضای قید توزیع‌شده

Distributed Constraint Satisfaction

کاظم فولادی قلعه
دانشکده مهندسی، دانشکدگان فارابی
دانشگاه تهران

http://courses.fouladi.ir/mas

مسئله‌ی ارضای قید

معرفی

مسئله‌ی ارضای قید

Constraint Satisfaction Problem (CSP)

مجموعه‌ای از متغیرها با دامنه‌ی مشخص وجود دارند.
هدف تعیین مقدار برای این متغیرهاست به گونه‌ای که یک مجموعه از قیدها ارضا شوند.

انواع زیادی از مسائل می‌توانند به صورت CSP تعریف شوند.

مسائل CSP را می‌توان توسط الگوریتم‌های عمومی جستجو حل کرد،
اما به دلیل ساختار خاص این مسائل، الگوریتم‌های کارآمدتری برای آنها وجود دارد.

مسئله‌ی ارضای قید

تعریف ریاضی

مؤلفه‌های سه‌گانه‌ی تعریف مسئله‌ی «ارضای قید»

X	**متغیرها** Variables	مجموعه‌ای از متغیرها $\{X_1, X_2, \ldots, X_n\}$
D	**دامنه‌ها** Domains	مجموعه‌ای از دامنه‌ها (مقادیر مجاز متغیرها) هر دامنه به صورت $\{v_1, v_2, \ldots, v_k\}$ $\{D_1, D_2, \ldots, D_n\}$
C	**قیدها** Constraints	مجموعه‌ای از قیدها: مشخص‌کننده‌ی مقادیر مجاز ترکیبات متغیرها هر قید به صورت رابطه‌ای روی تعدادی متغیر (نمایش لیستی رابطه / رابطه‌ی انتزاعی) قید صریح (explicit): به صورت مجموعه‌ای از مقادیر مجاز قید ضمنی (implicit): با استفاده از تابعی که ارضا شدن قیدها را بررسی می‌کند.

The constraints are defined by predicates, $p_k(x_{k1}, x_{k2}, \ldots, x_{kj})$
where each p_k is the function
$p_k : D_{k1} \times D_{k2} \times \ldots \times D_{kj} \rightarrow \{0, 1\}$

انتساب

انتساب مقدار به متغیر و انواع آن

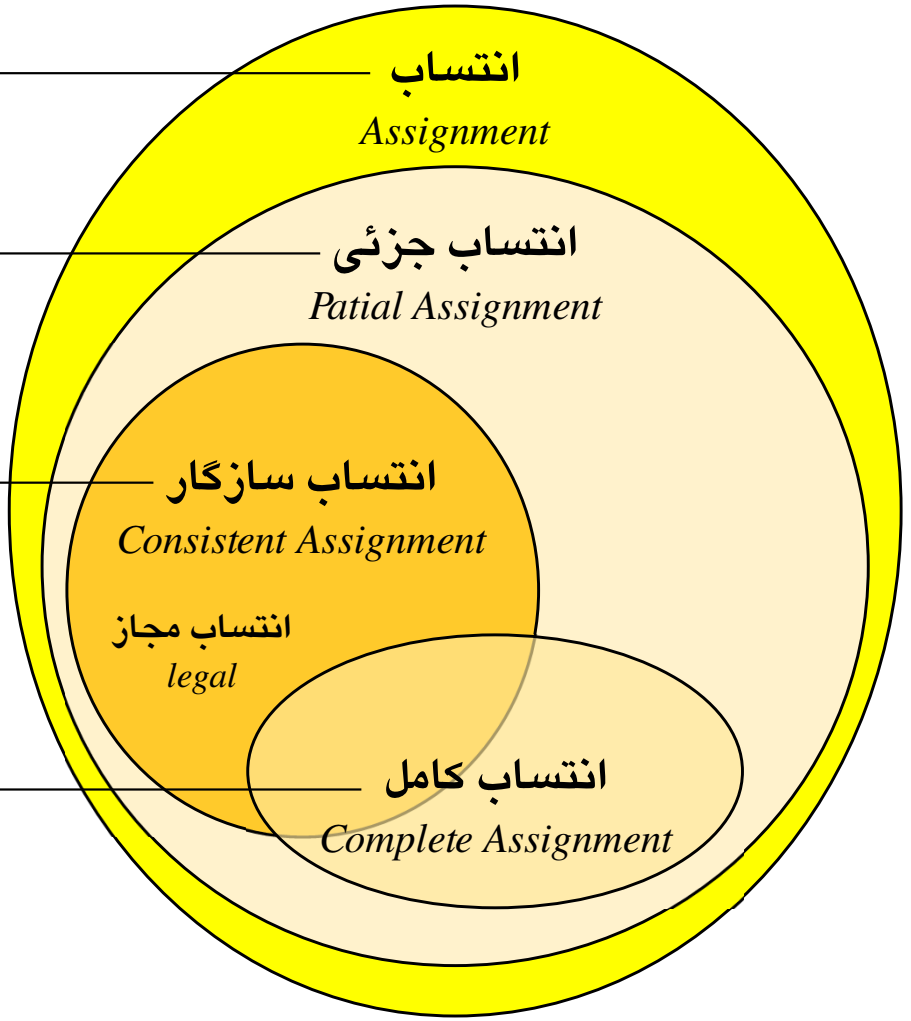
ASSIGNMENT

نسبت‌دهی مقادیر به تمام یا بعضی از متغیرها

یک انتساب که در آن بعضی متغیرها نسبت‌دهی شده باشند.

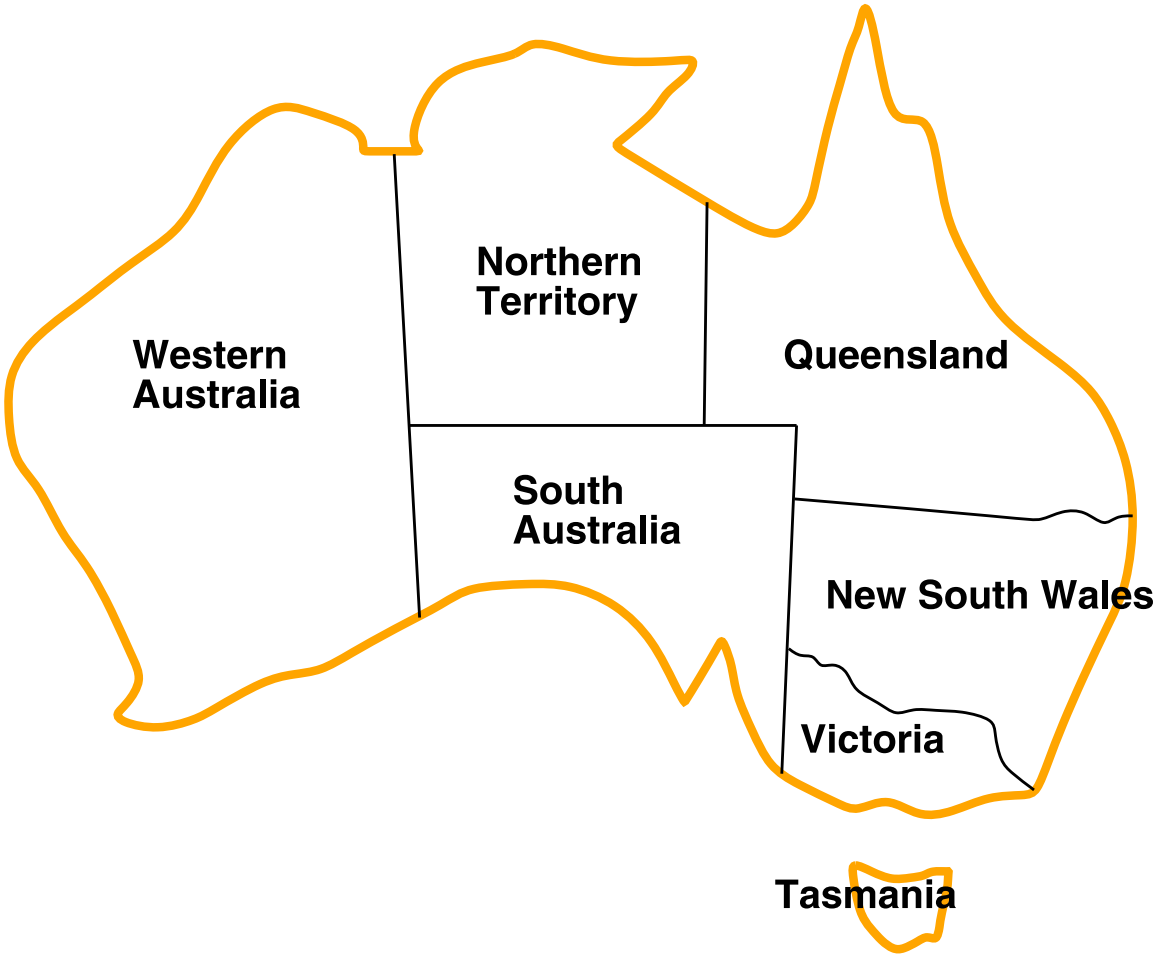
یک انتساب که هیچ قیدی را نقض نکرده باشد

یک انتساب که در آن همه‌ی متغیرها نسبت‌دهی شده باشند.



مسئله‌ی «رنگ‌آمیزی نقشه»

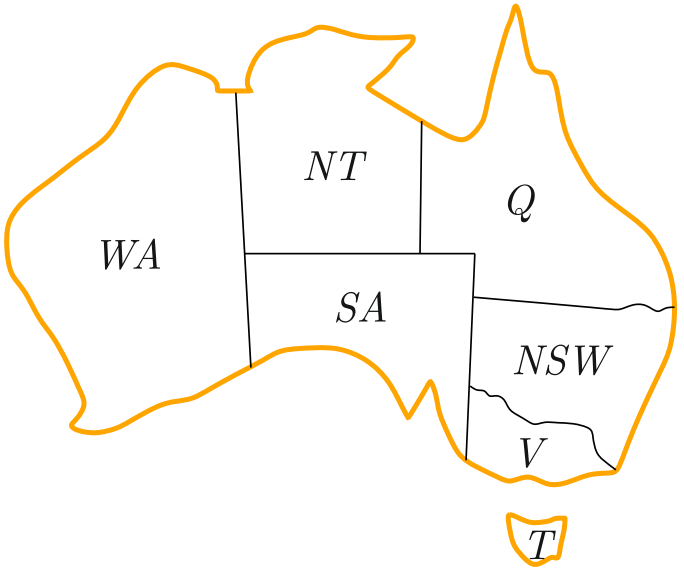
MAP-COLORING



انتساب رنگ به استان‌ها به طوری که استان‌های مجاور هم‌رنگ نباشند.

مسئله‌ی ارضای قید

مثال: مسئله‌ی رنگ‌آمیز نقشه

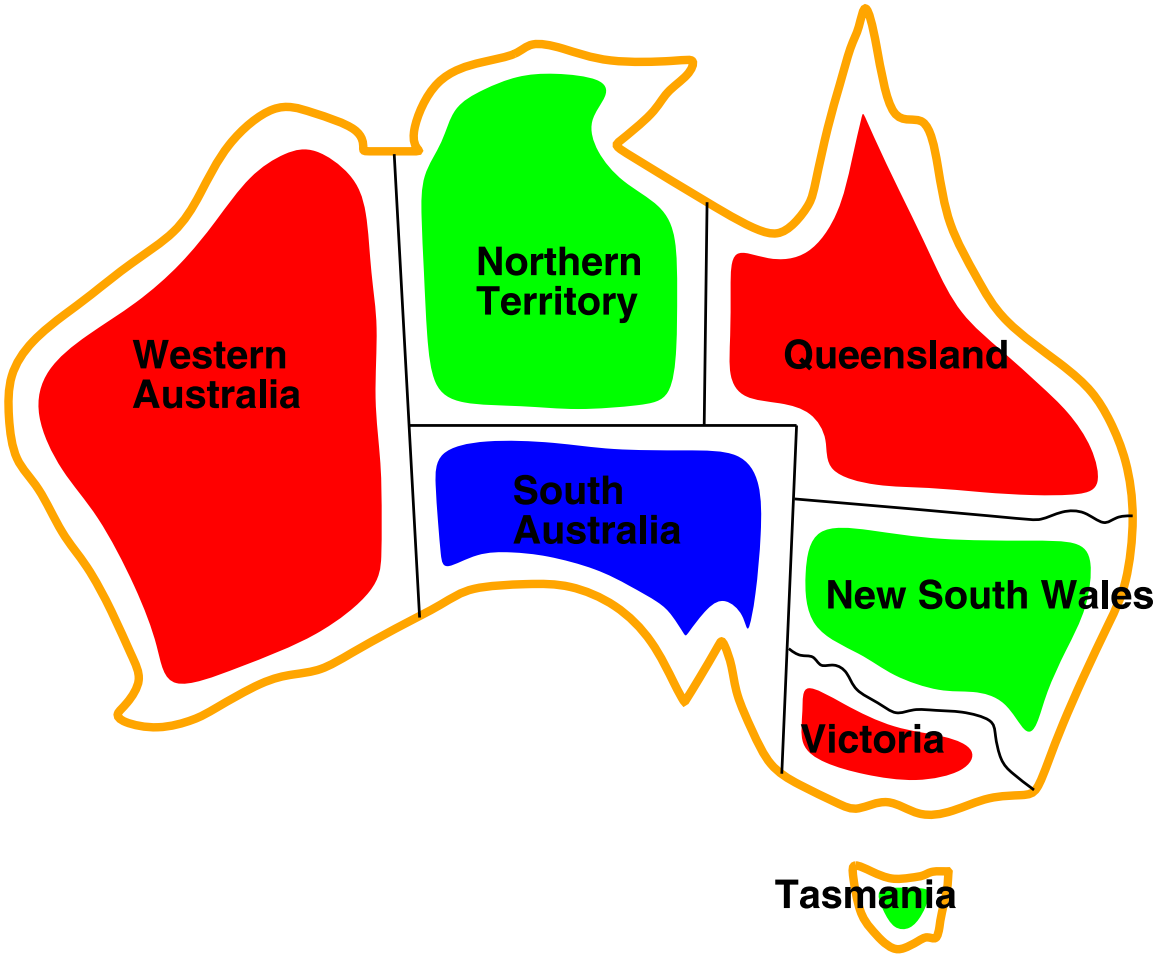


مؤلفه‌های تعریف مسئله‌ی «ارضای قید»

X	**متغیرها** Variables	استان‌ها: $WA,\ NT,\ Q,\ NSW,\ V,\ SA,\ T$
D	**دامنه‌ها** Domains	رنگ‌ها: $D_i = \{red, blue, green\}$
C	**قیدها** Constraints	ناحیه‌های مجاور باید رنگ‌های متفاوت داشته باشند. e.g., $WA \neq NT,\quad V \neq NSW,\quad \ldots$

مسئله‌ی ارضای قید

مثال: مسئله‌ی رنگ‌آمیز نقشه: راه‌حل



راه‌حل‌ها: انتساب‌هایی هستند که همه‌ی قیدها را ارضا کنند، مانند:

$$\{WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green\}$$

گراف قید

CONSTRAINT GRAPH

گراف قید *Constraint Graph*	
کمان‌ها *Arcs*	**گره‌ها** *Nodes*
نشان‌دهنده‌ی **قیدها**	نشان‌دهنده‌ی **متغیرها**

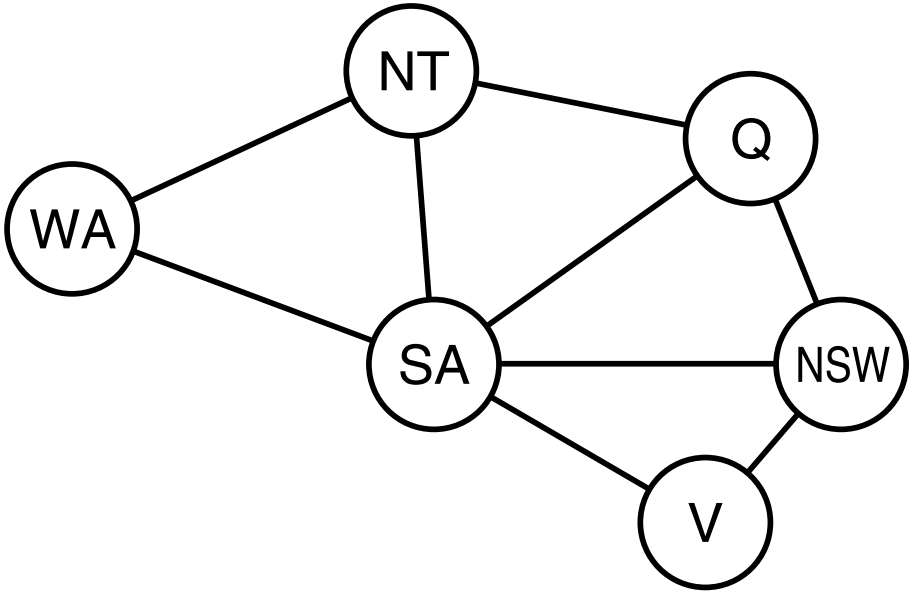
قابل استفاده برای CSP دودویی

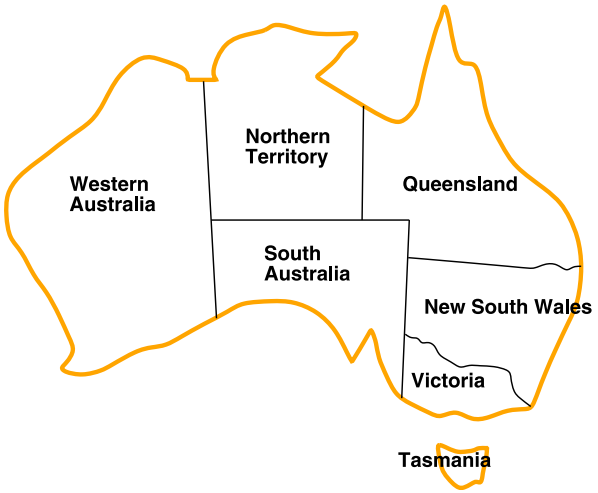
الگوریتم‌های CSP همه‌منظوره، از ساختار گراف برای تسریع جستجو استفاده می‌کنند.

گراف قید

مثال

CONSTRAINT GRAPH





حل مسئله‌ی ارضای قید با جستجوی عمق–اول

A centralized depth first search algorithm for the CSP. Variable g holds the partial assignment of variable values. The algorithm is called with DEPTH-FIRST-SEARCH-CSP$(1, \emptyset)$.

DEPTH-FIRST-SEARCH-CSP(i, g)

```
1  if i > n
2      then return g
3  for v ∈ D_i
4      do if setting x_i ← v does not violate any constraint in P given g
5          then g' ← DEPTH-FIRST-SEARCH-CSP(i + 1, g + {x_i ← v})
6               if g' ≠ ∅
7                   then return g'
8
9  return ∅
```

حل مسئله‌ی ارضای قید توزیع‌شده

ارضای قید چندعاملی

هر عامل مسئول مقدار یکی از متغیرها و قیدهای مربوط به آن است.

حل مسئله‌ی ارضای قید با الگوریتم فیلترینگ

FILTERING ALGORITHM

```
FILTERING()
1  for j ∈ {neighbors of i}        ▷ i is this agent.
2        do REVISE(x_i, x_j)
```

The filtering algorithm. Each agent i executes FILTERING().

```
HANDLE-NEW-DOMAIN(j, D')
1  D_j ← D'
2  REVISE(x_i, x_j)
```

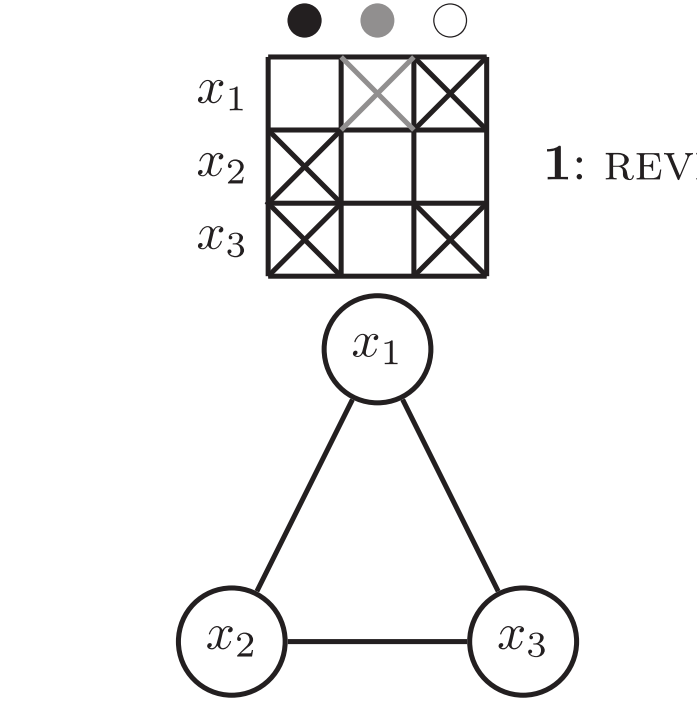
هر عامل دامنه‌اش را به عامل‌های همسایه‌اش اطلاع می‌دهد و سپس مقادیری که نمی‌توانند قیدها را ارضا کنند، از دامنه‌اش حذف می‌کند.

```
REVISE(x_i, x_j)
1  old-domain ← D_i
2  for v_i ∈ D_i
3        do if there is no v_j ∈ D_j consistent with v_i
4              then D_i ← D_i − v_i
5  if old-domain ≠ D_i
6     then ∀_{k∈{neighbors of i}} k.HANDLE-NEW-DOMAIN(i, D_i)
```

حل مسئله‌ی ارضای قید با الگوریتم فیلترینگ

مثال

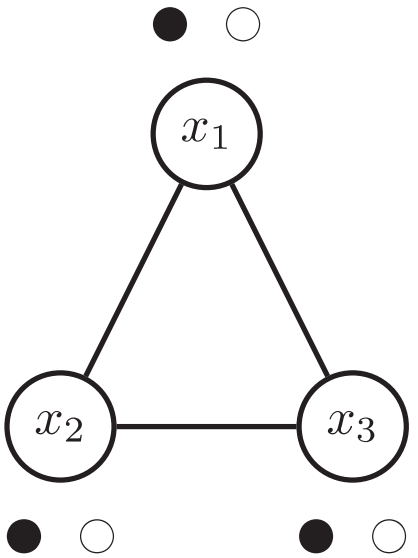
Filtering example. The agents start out with some prohibited colors as indicated by the black crosses. On the first step x_1 does his REVISE and eliminates the color gray from consideration. It then tells everyone else about this. Then x_2 does its revise and eliminates the color gray from its domain.



حل مسئله‌ی ارضای قید با الگوریتم فیلترینگ

مثال

Example of a problem that does not have a solution and the filtering algorithm cannot that fact.



حل مسئله‌ی ارضای قید با الگوریتم سازگاری مبتنی بر هایپر رزولوشن

قاعده‌ی هایپر رزولوشن

HYPER-RESOLUTION BASED CONSISTENCY ALGORITHM

$$\frac{\begin{array}{c} A_1 \vee A_2 \vee \cdots \vee A_m \\ \neg(A_1 \wedge A_{11} \wedge \cdots) \\ \neg(A_2 \wedge A_{21} \wedge \cdots) \\ \vdots \\ \neg(A_m \wedge A_{m1} \wedge \cdots) \end{array}}{\neg(A_{11} \wedge \cdots \wedge A_{21} \wedge \cdots \wedge A_{m1} \wedge \cdots)}.$$

حل مسئله‌ی ارضای قید با الگوریتم سازگاری مبتنی بر هایپر رزولوشن

مثال

HYPER-RESOLUTION BASED CONSISTENCY ALGORITHM

Time	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$
	$x_1 = \bigcirc \vee x_1 = \bullet$	$x_2 = \bigcirc \vee x_2 = \bullet$	$x_3 = \bigcirc \vee x_3 = \bullet$
	$\neg(x_1 = \bigcirc \wedge x_2 = \bigcirc)$	$\neg(x_1 = \bigcirc \wedge x_2 = \bigcirc)$	$\neg(x_1 = \bigcirc \wedge x_2 = \bigcirc)$
	$\neg(x_1 = \bullet \wedge x_2 = \bullet)$	$\neg(x_1 = \bullet \wedge x_2 = \bullet)$	$\neg(x_1 = \bullet \wedge x_2 = \bullet)$
1	$\neg(x_2 = \bigcirc \wedge x_3 = \bullet)$	$\neg(x_2 = \bigcirc \wedge x_3 = \bullet)$	$\neg(x_2 = \bigcirc \wedge x_3 = \bullet)$
2	$\neg(x_2 = \bullet \wedge x_3 = \bigcirc)$	$\neg(x_3 = \bullet)$	$\neg(x_2 = \bullet \wedge x_3 = \bigcirc)$
3		$\neg(x_2 = \bullet \wedge x_3 = \bigcirc)$	$\neg(x_3 = \bigcirc)$
4		$\neg(x_3 = \bigcirc)$	$\neg(x_3 = \bullet)$

Example databases for a sample run of the hyper-resolution based consistency algorithm as applied to the graph of figure 2.5. Only a few of the nogoods produced by the graph's constraints are shown due to space constraints. You can infer the rest. New statements are added starting at time 1.

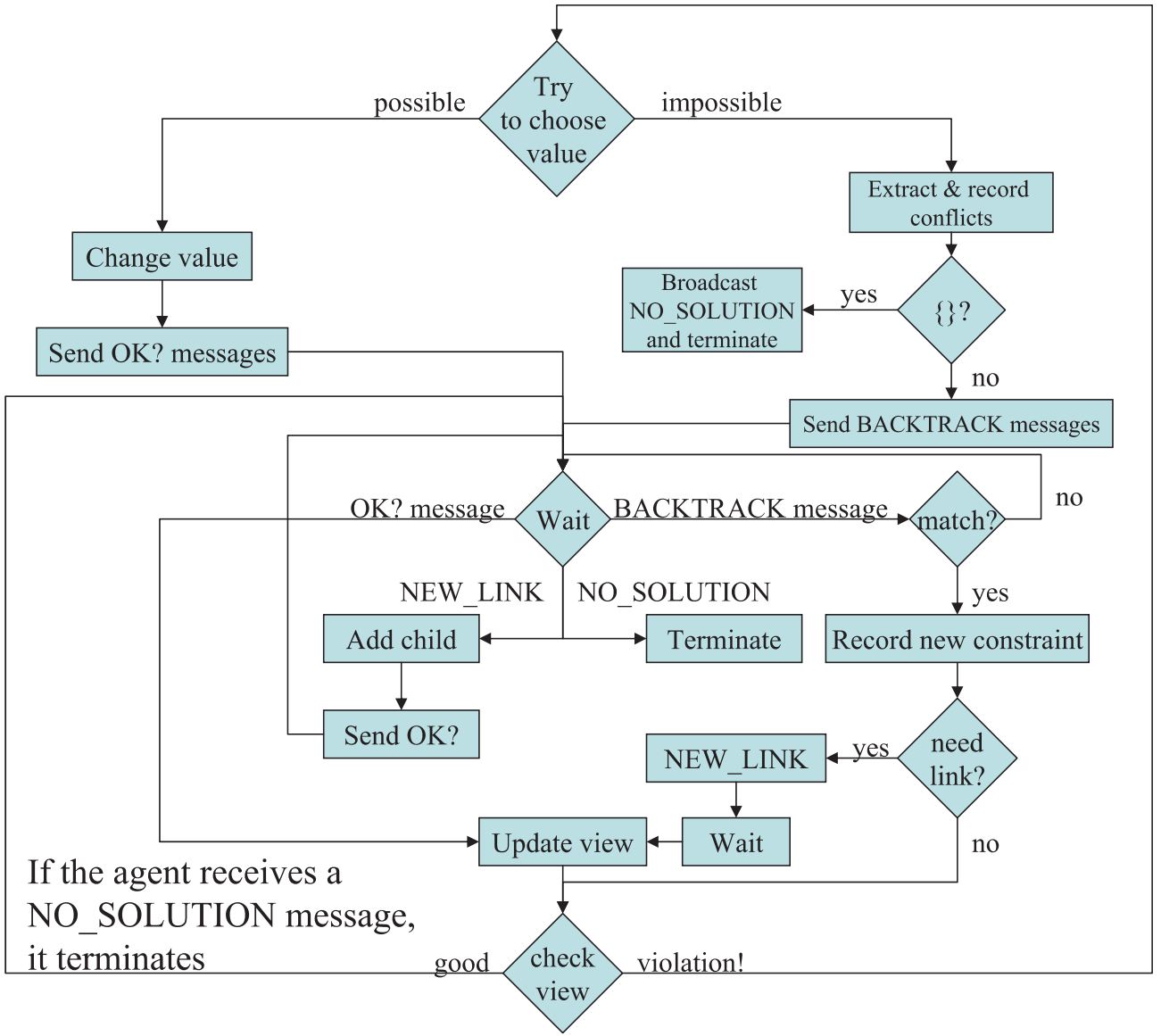
حل مسئله‌ی ارضای قید با الگوریتم سازگاری مبتنی بر هایپر رزولوشن

HYPER-RESOLUTION BASED CONSISTENCY ALGORITHM

procedure ReviseHR(NG_i, NG_j^*)
repeat
  $NG_i \leftarrow NG_i \bigcup NG_j^*$
  let NG_i^* denote the set of new Nogoods that i can derive from NG_i and his domain using hyper-resolution
  if NG_i^* *is nonempty* **then**
    $NG_i \leftarrow NG_i \bigcup NG_i^*$
    send the Nogoods NG_i^* to all neighbors of i
    if $\{\} \in NG_i^*$ **then**
      stop
until *there is no change in* i*'s set of Nogoods* NG_i

الگوریتم عقب‌گرد ناهمگام

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

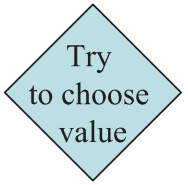


الگوریتم عقب‌گرد ناهمگام

۱ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

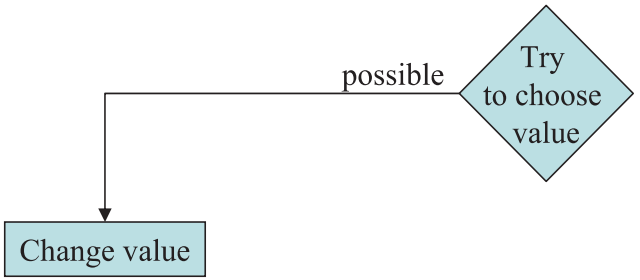
عامل برای متغیرش یک مقدار انتخاب می‌کند که
قیدهایی که مسئول برقراری آنهاست را ارضا کند.



الگوریتم عقب‌گرد ناهمگام

۲ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

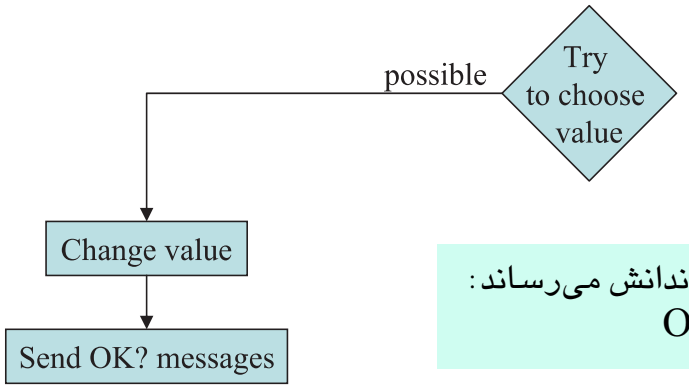


اگر حداقل یک مقدار وجود دارد که آن قیدها را ارضا می‌کند، عامل یکی را برمی‌گزیند و نسبت‌دهی آن به متغیرش را انجام می‌دهد.

الگوریتم عقب‌گرد ناهمگام

۳ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

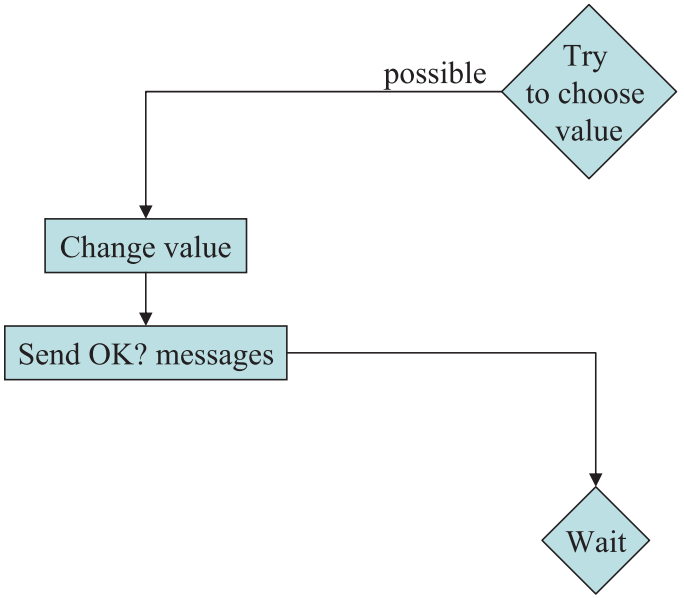


عامل انتساب جدیدش را به اطلاع فرزندانش می‌رساند:
با ارسال پیام‌های OK?

الگوریتم عقب‌گرد ناهمگام

۴ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

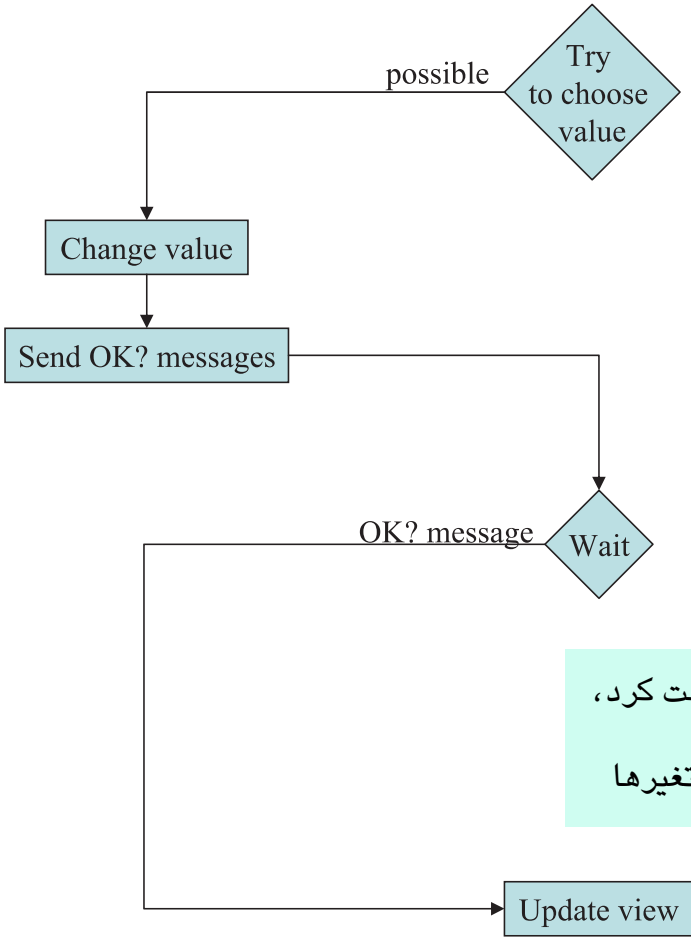


عامل سپس منتظر پاسخ به پیام OK? خودش از سوی فرزندانش می‌ماند.
(همچنین دیگر پیام‌های OK? از سوی والدهایش)

الگوریتم عقب‌گرد ناهمگام

۵ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



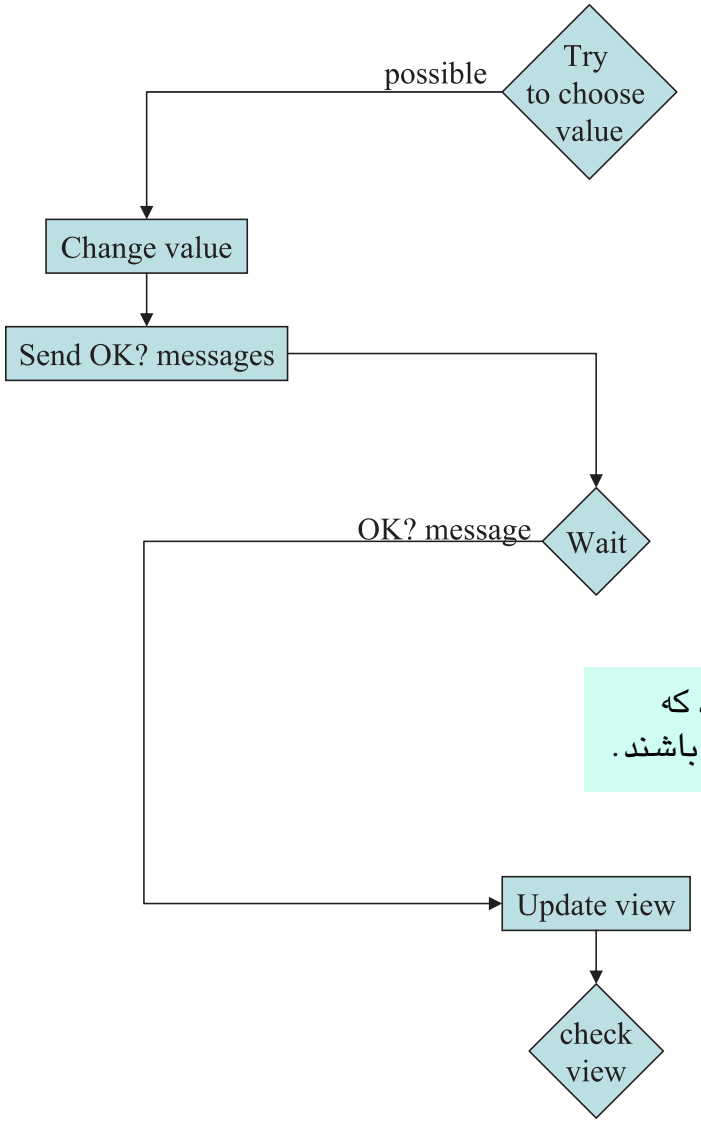
A's view	
B	1
C	?
D	3
E	4
F	?
G	?
H	?

اگر عامل یک پیام OK? را از یکی از والدهایش دریافت کرد،
«دید» خودش را به‌هنگام می‌کند.
یعنی: به‌هنگام‌سازی دانایی خود از مقادیر سایر متغیرها

الگوریتم عقب‌گرد ناهمگام

۶ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



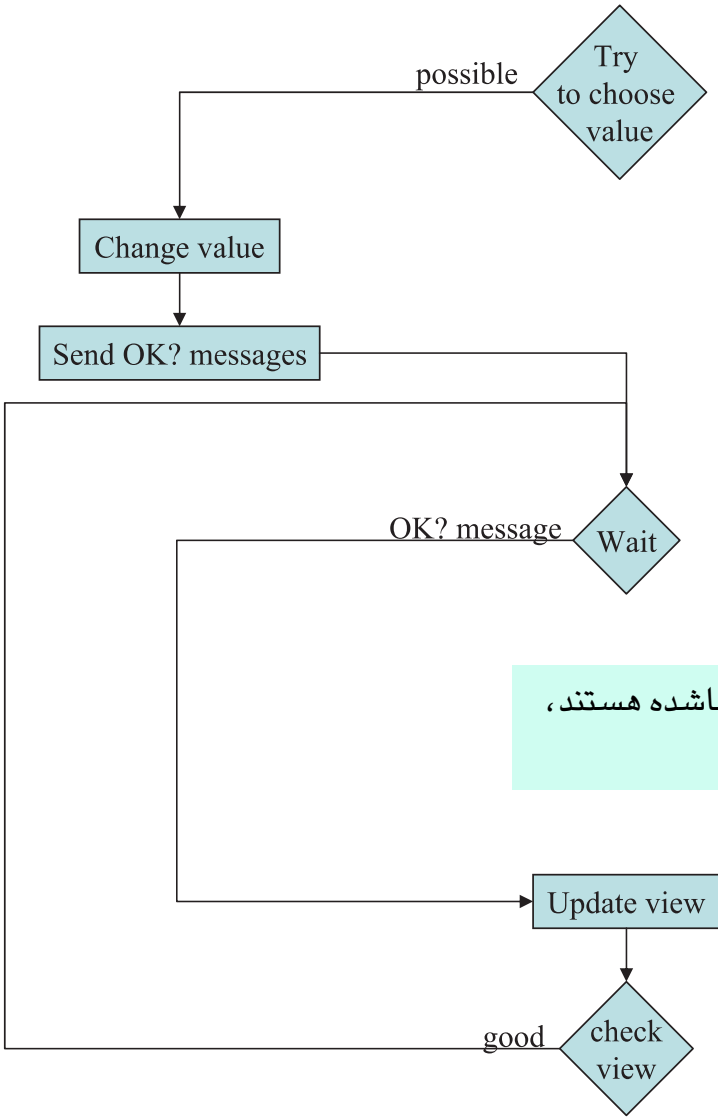
A's view	
B	1
C	?
D	3
E	4
F	?
G	?
H	?

سپس، عامل «دید» خودش را بررسی می‌کند تا مطمئن شود که مقادیر هیچ‌یک از قیدهایی که او مسئول آنهاست را نقض نکرده باشند.

الگوریتم عقب‌گرد ناهمگام

۷ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



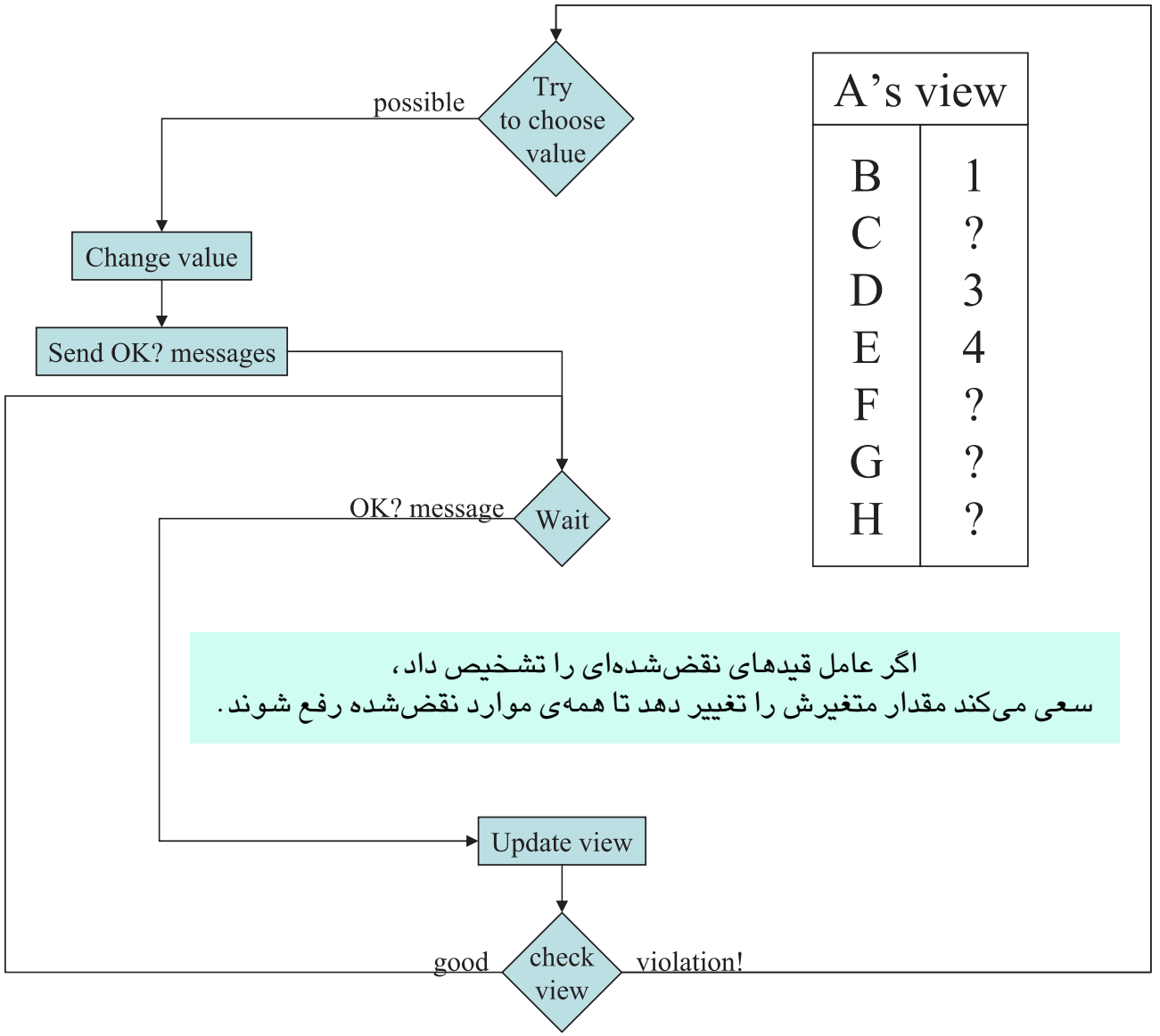
A's view	
B	1
C	?
D	3
E	4
F	?
G	?
H	?

اگر همه‌ی قیدهایی که عامل مسئول آنهاست هنوز ارضاشده هستند،
عامل منتظر پیام‌ها می‌ماند

الگوریتم عقب‌گرد ناهمگام

۸ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

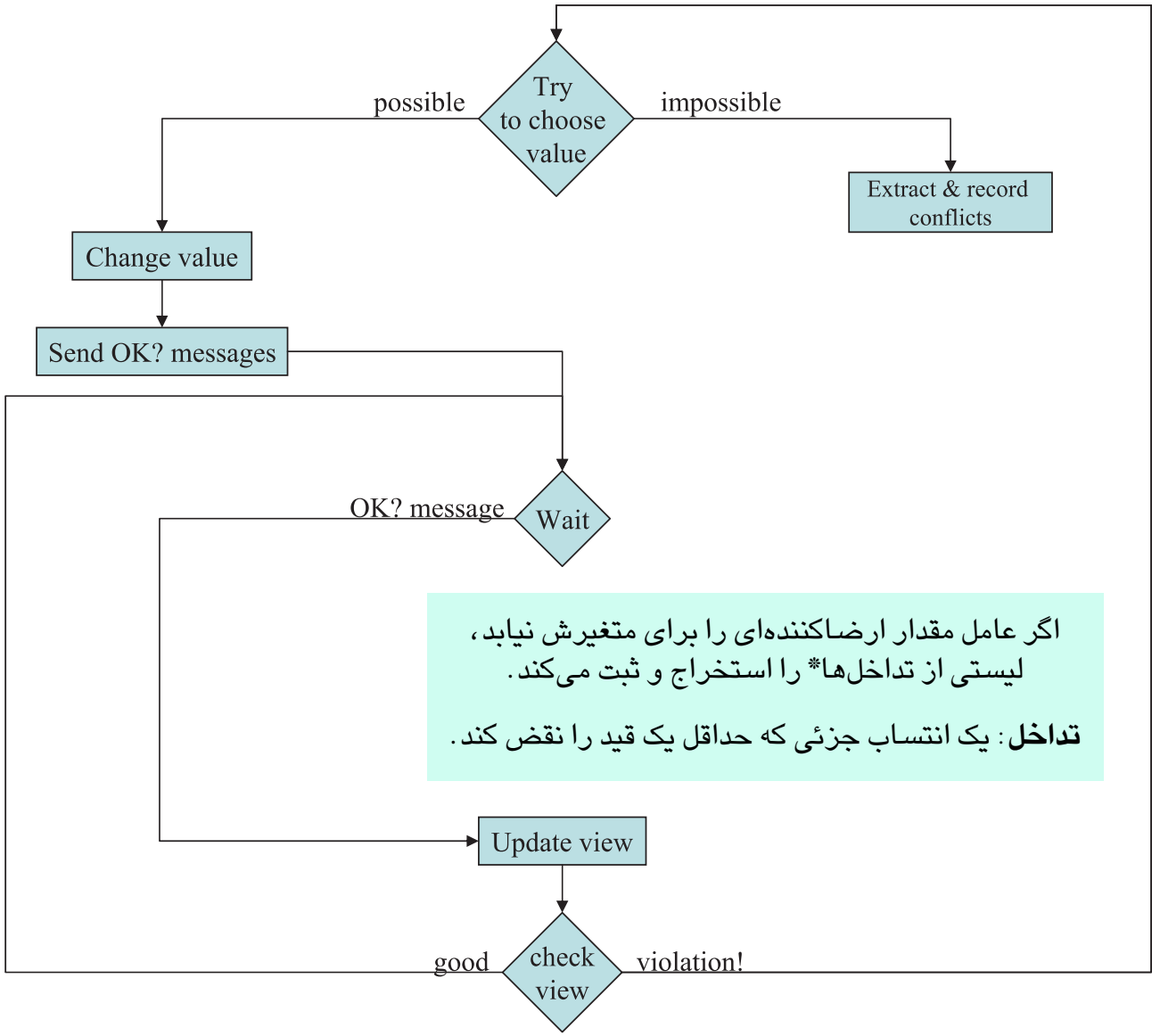


اگر عامل قیدهای نقض‌شده‌ای را تشخیص داد،
سعی می‌کند مقدار متغیرش را تغییر دهد تا همه‌ی موارد نقض‌شده رفع شوند.

الگوریتم عقب‌گرد ناهمگام

۹ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



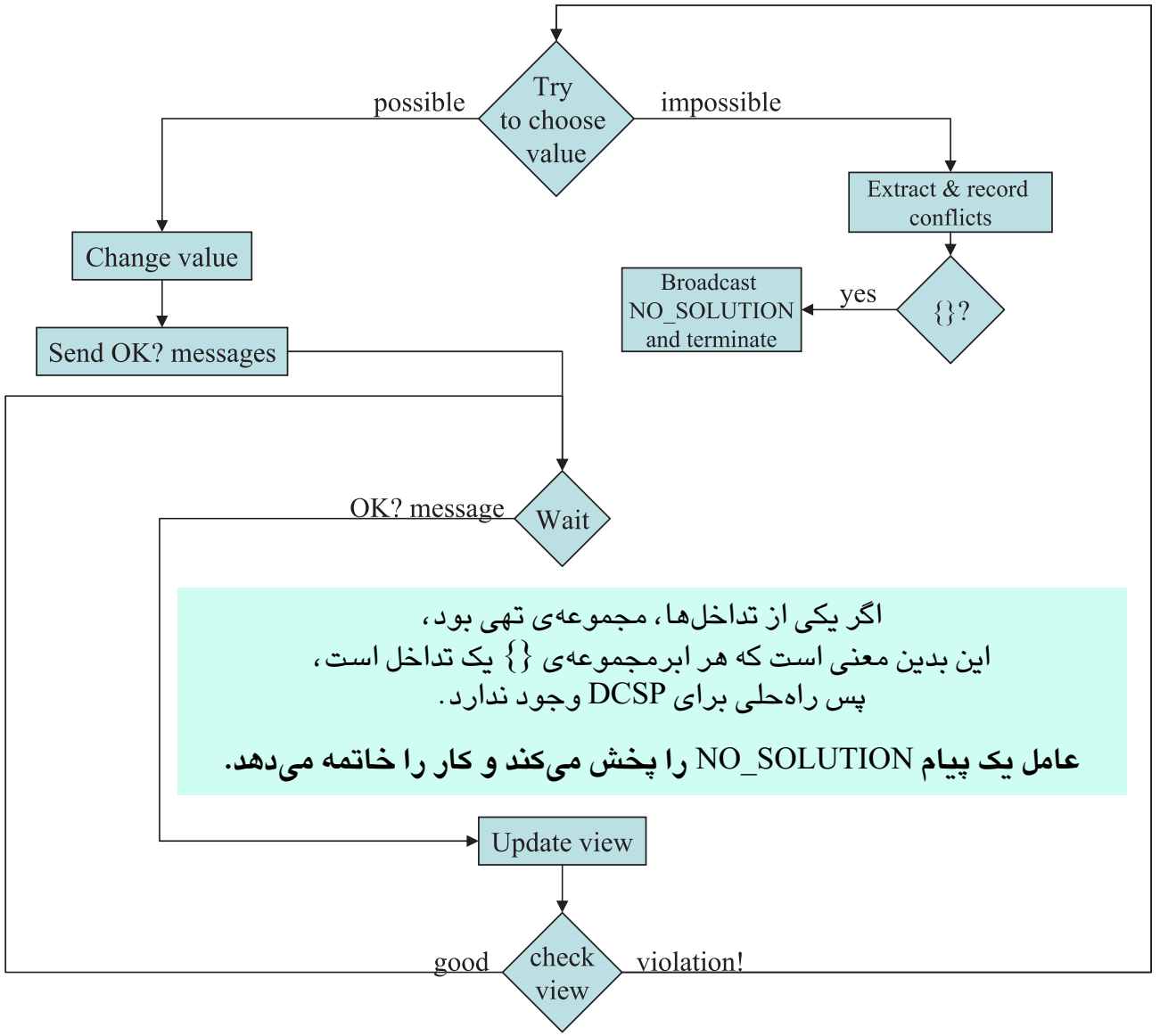
اگر عامل مقدار ارضاکننده‌ای را برای متغیرش نیابد، لیستی از تداخل‌ها* را استخراج و ثبت می‌کند.

تداخل: یک انتساب جزئی که حداقل یک قید را نقض کند.

الگوریتم عقب‌گرد ناهمگام

۱۰ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



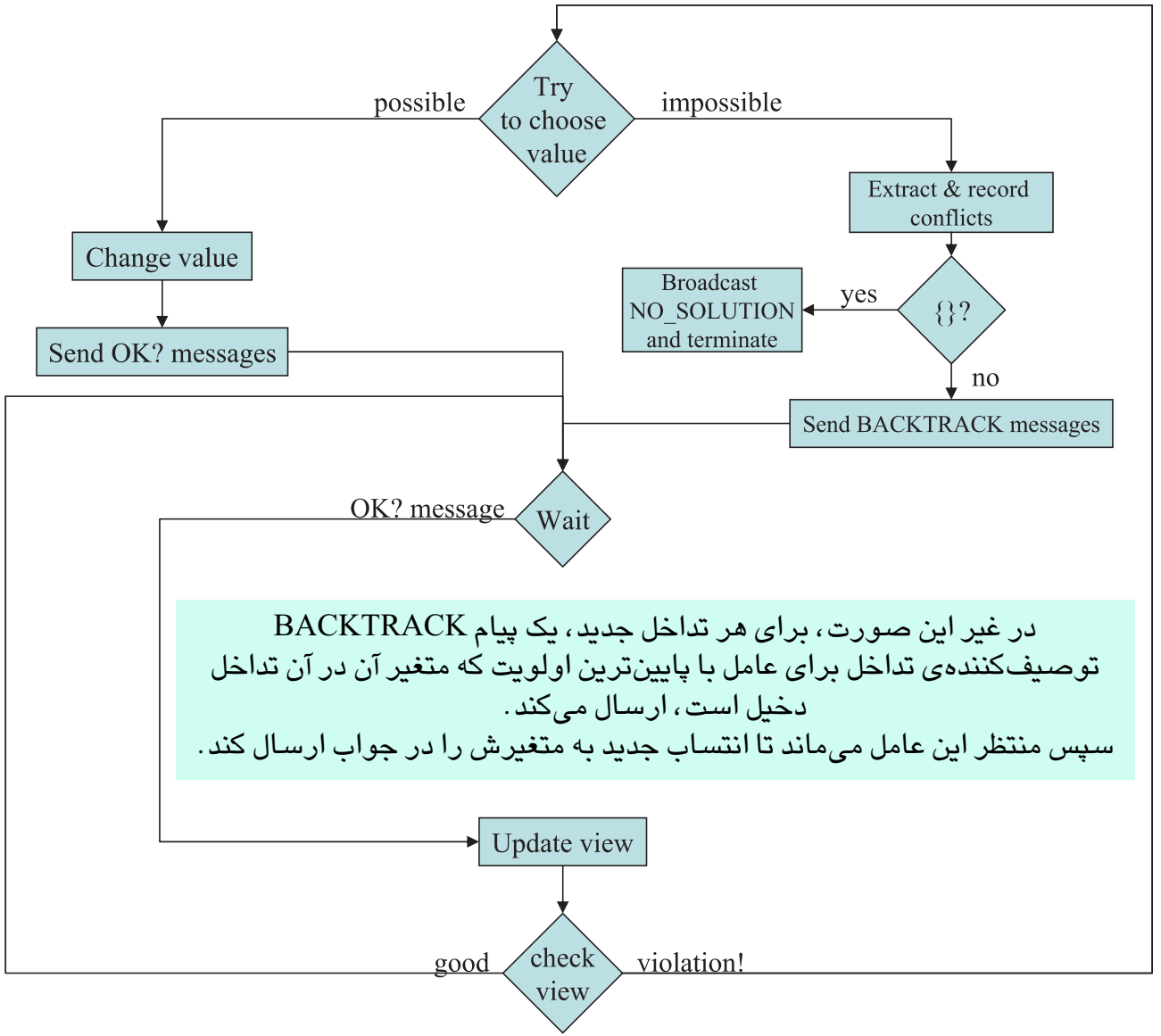
اگر یکی از تداخل‌ها، مجموعه‌ی تهی بود،
این بدین معنی است که هر ابرمجموعه‌ی {} یک تداخل است،
پس راه‌حلی برای DCSP وجود ندارد.

عامل یک پیام NO_SOLUTION را پخش می‌کند و کار را خاتمه می‌دهد.

الگوریتم عقب‌گرد ناهمگام

۱۱ از ۱۶

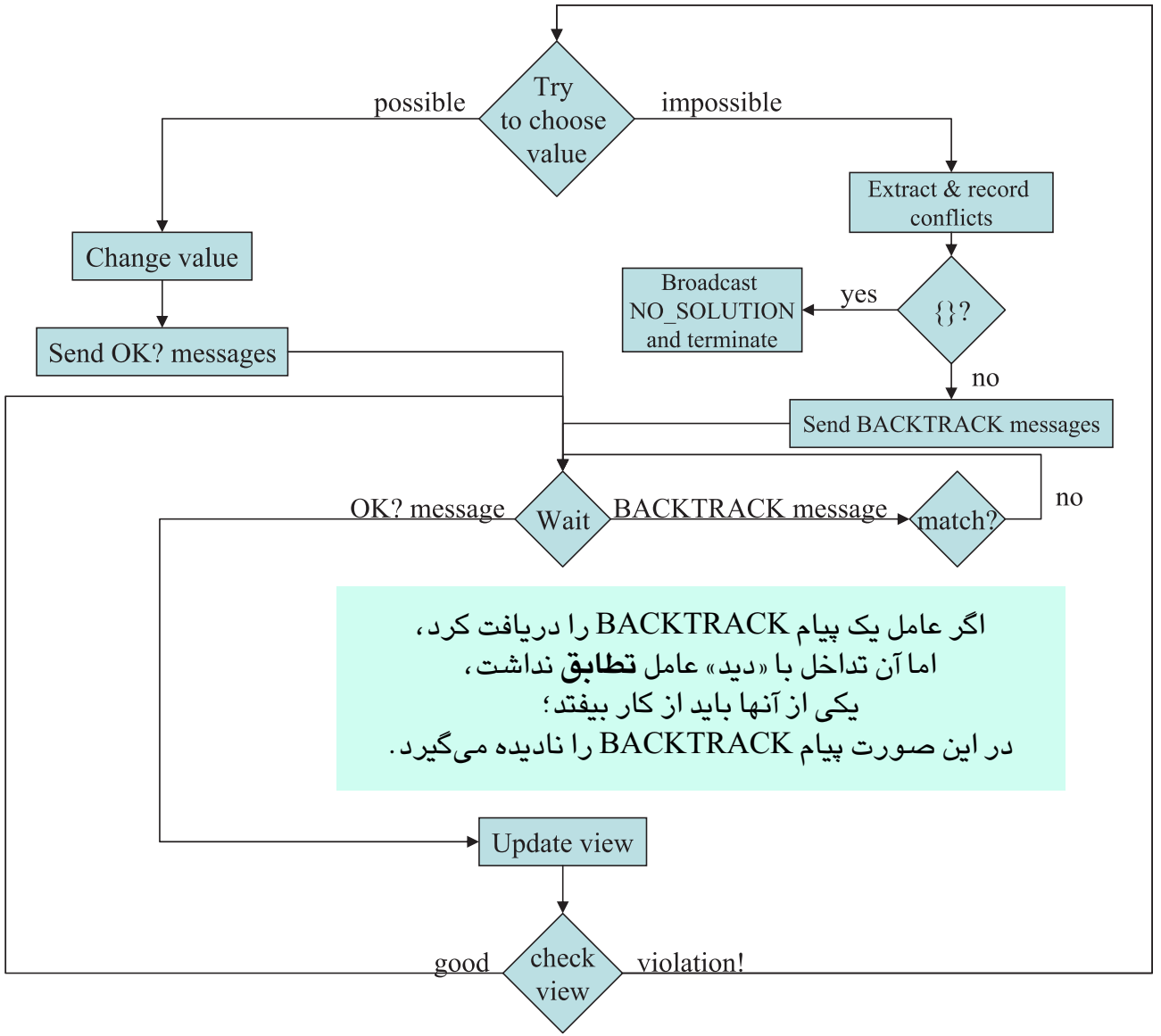
THE ASYNCHRONOUS BACKTRACKING ALGORITHM



در غیر این صورت، برای هر تداخل جدید، یک پیام BACKTRACK توصیف‌کننده‌ی تداخل برای عامل با پایین‌ترین اولویت که متغیر آن در آن تداخل دخیل است، ارسال می‌کند.
سپس منتظر این عامل می‌ماند تا انتساب جدید به متغیرش را در جواب ارسال کند.

الگوریتم عقب‌گرد ناهمگام

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

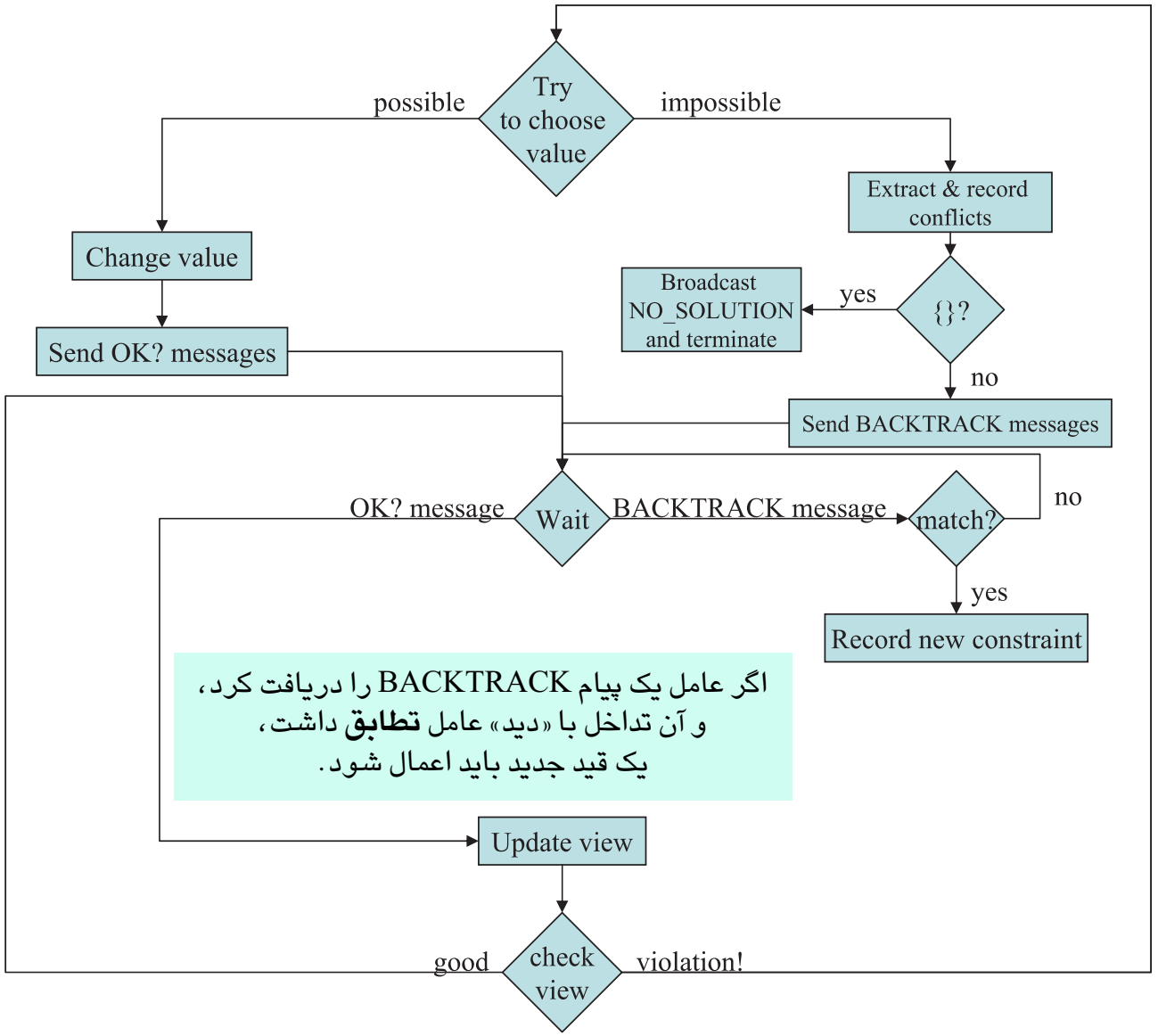


اگر عامل یک پیام BACKTRACK را دریافت کرد،
اما آن تداخل با «دید» عامل **تطابق** نداشت،
یکی از آنها باید از کار بیفتد؛
در این صورت پیام BACKTRACK را نادیده می‌گیرد.

الگوریتم عقب‌گرد ناهمگام

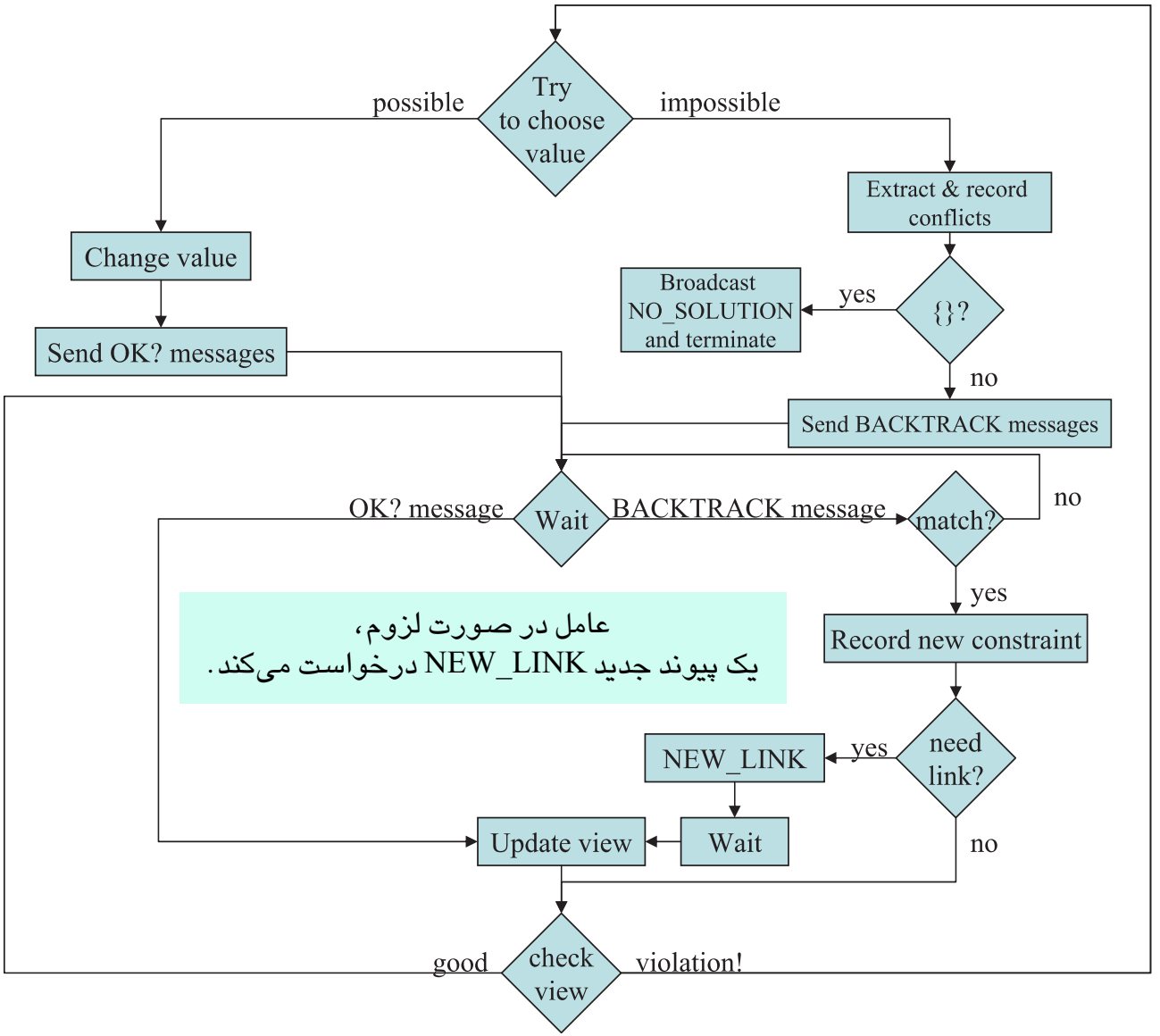
۱۳ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



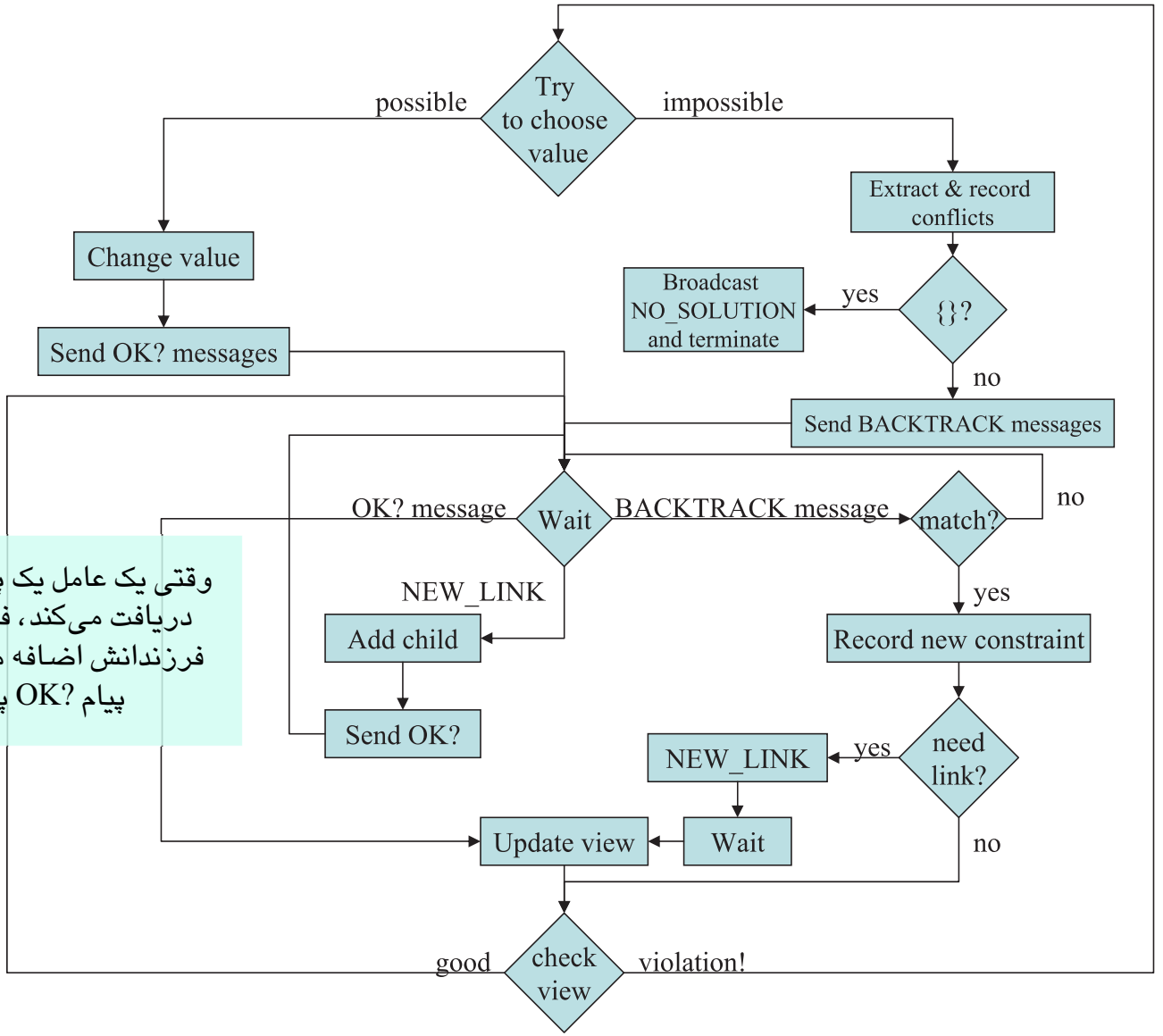
الگوریتم عقب‌گرد ناهمگام

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



الگوریتم عقب‌گرد ناهمگام

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

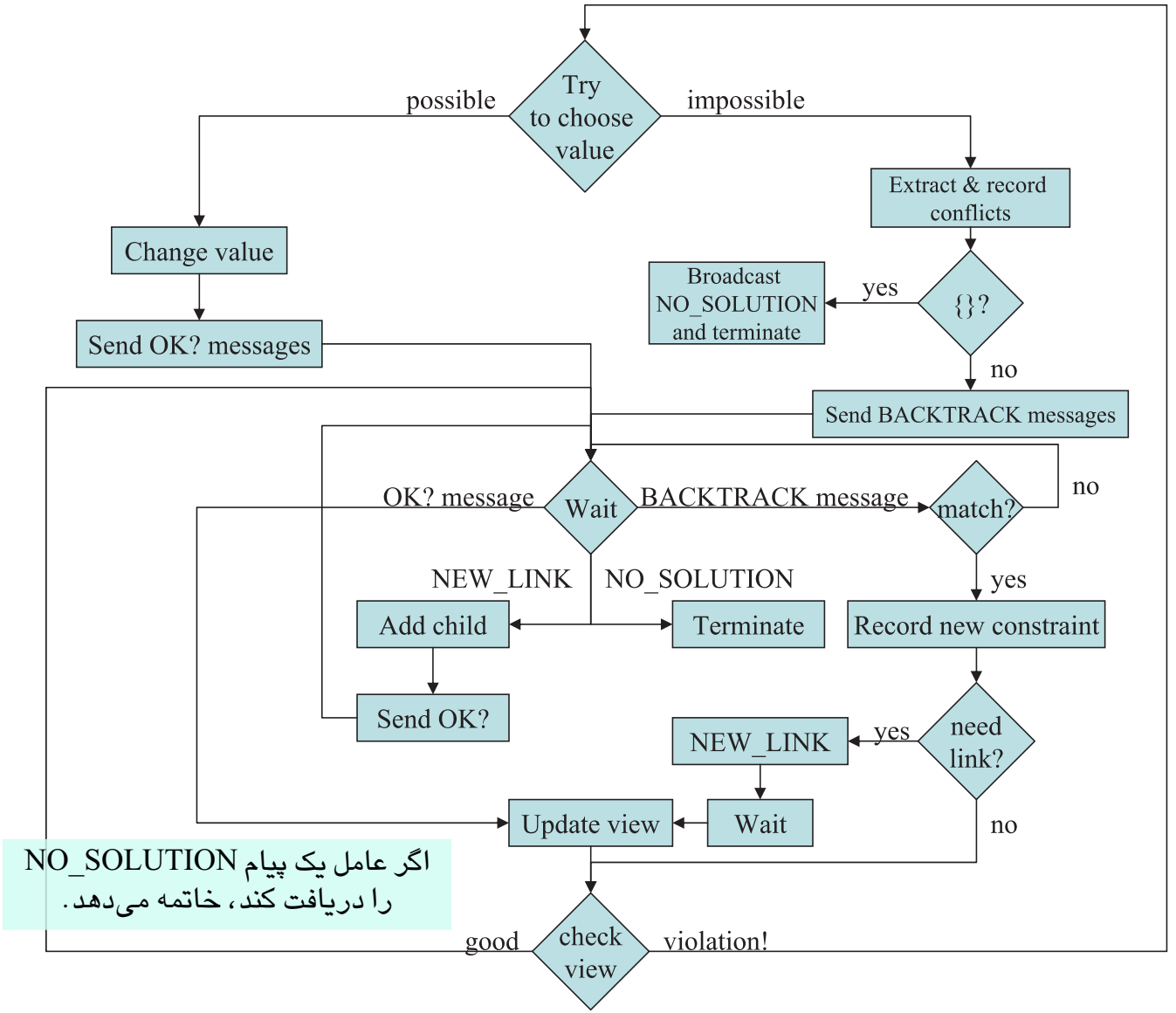


وقتی یک عامل یک پیام NEW_LINK را دریافت می‌کند، فرستنده را به لیست فرزندانش اضافه می‌کند و از طریق یک پیام OK? پاسخ می‌دهد.

الگوریتم عقب‌گرد ناهمگام

۱۶ از ۱۶

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

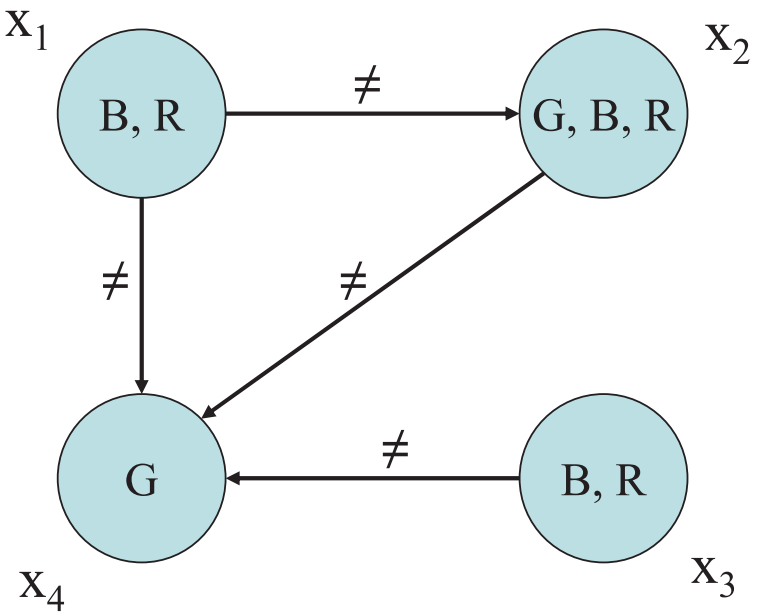


اگر عامل یک پیام NO_SOLUTION را دریافت کند، خاتمه می‌دهد.

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱ از ۴۰)

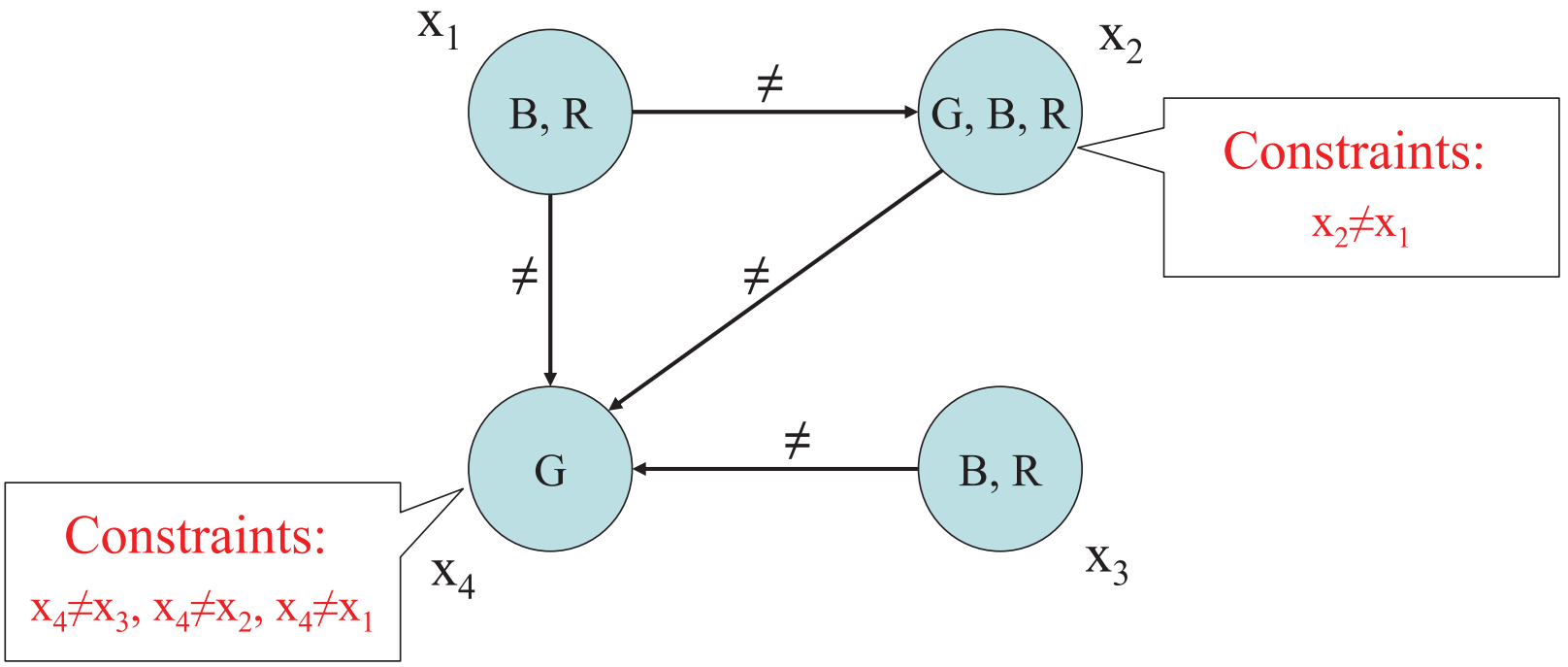
THE ASYNCHRONOUS BACKTRACKING ALGORITHM



الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲ از ۴۰)

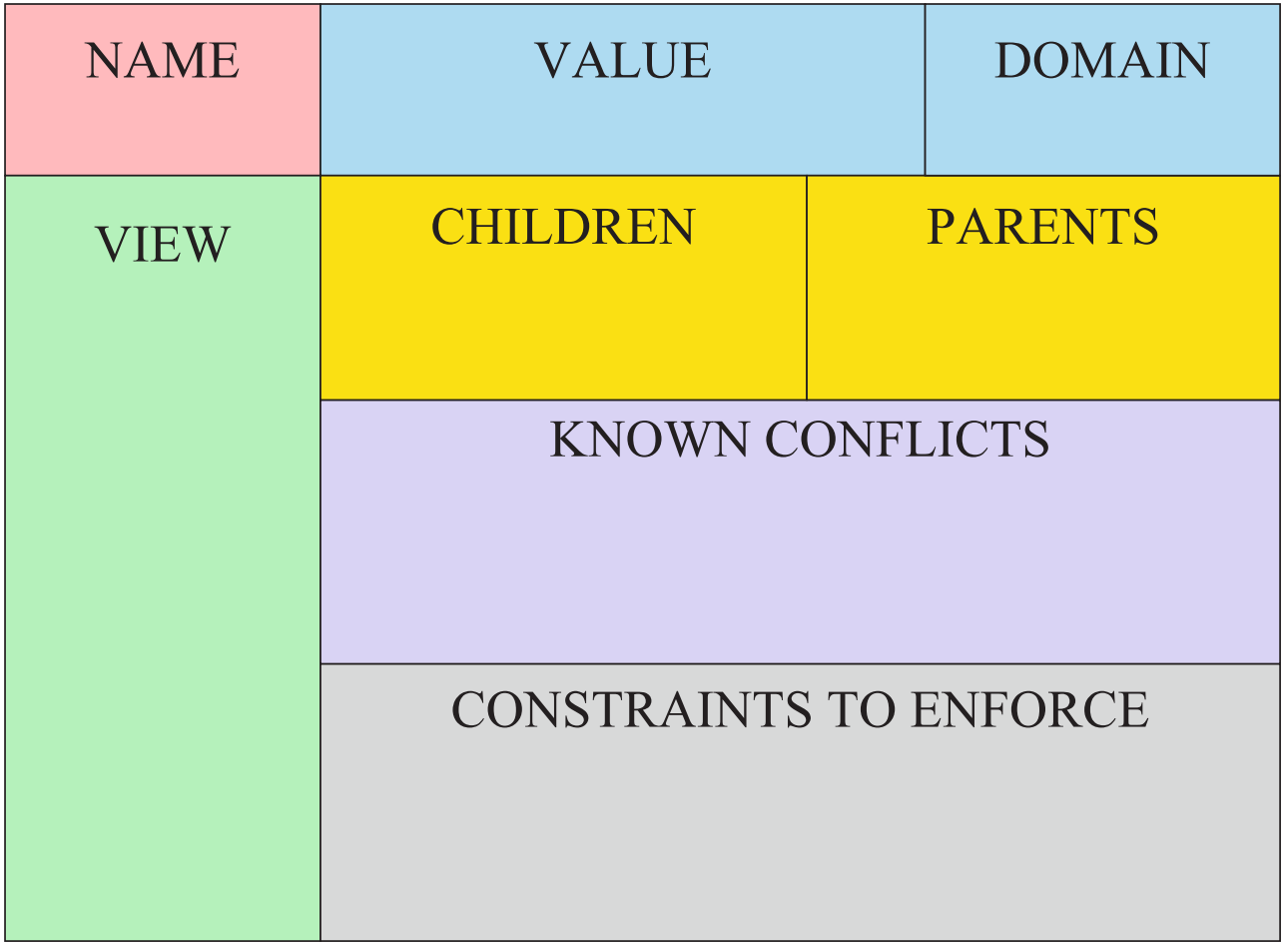
THE ASYNCHRONOUS BACKTRACKING ALGORITHM



الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳ از ۴۰) ساختمان‌داده‌ی لازم برای هر عامل/گره

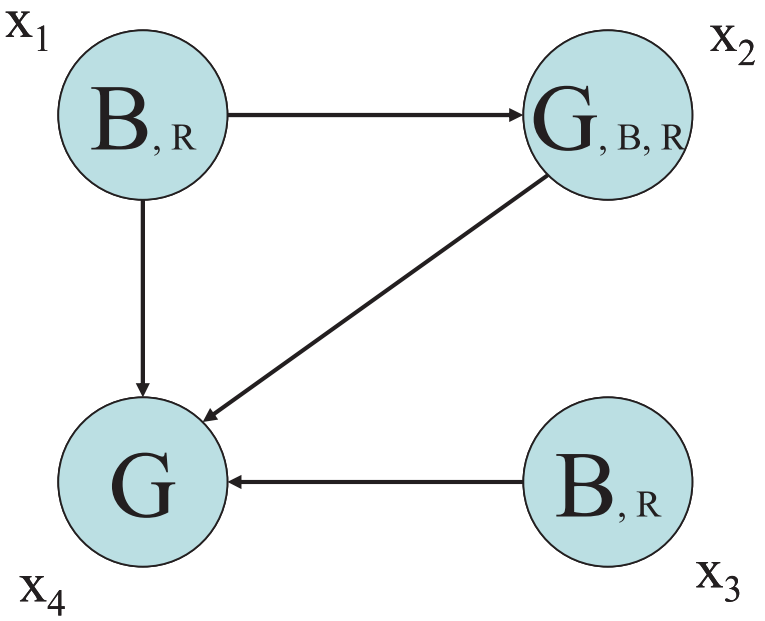
THE ASYNCHRONOUS BACKTRACKING ALGORITHM

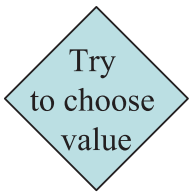


الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۴ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



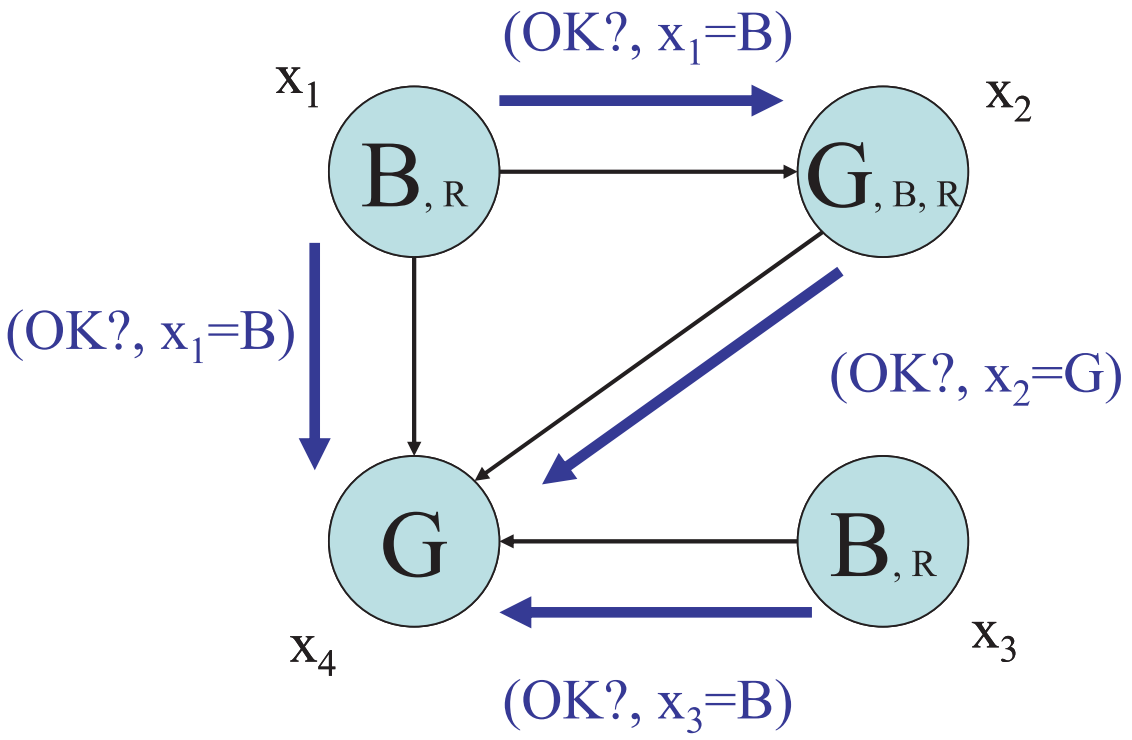


Each agent chooses an assignment to its variable

الگوریتم عقب‌گرد ناهمگام

مثال : رنگ‌آمیزی گراف (۵ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



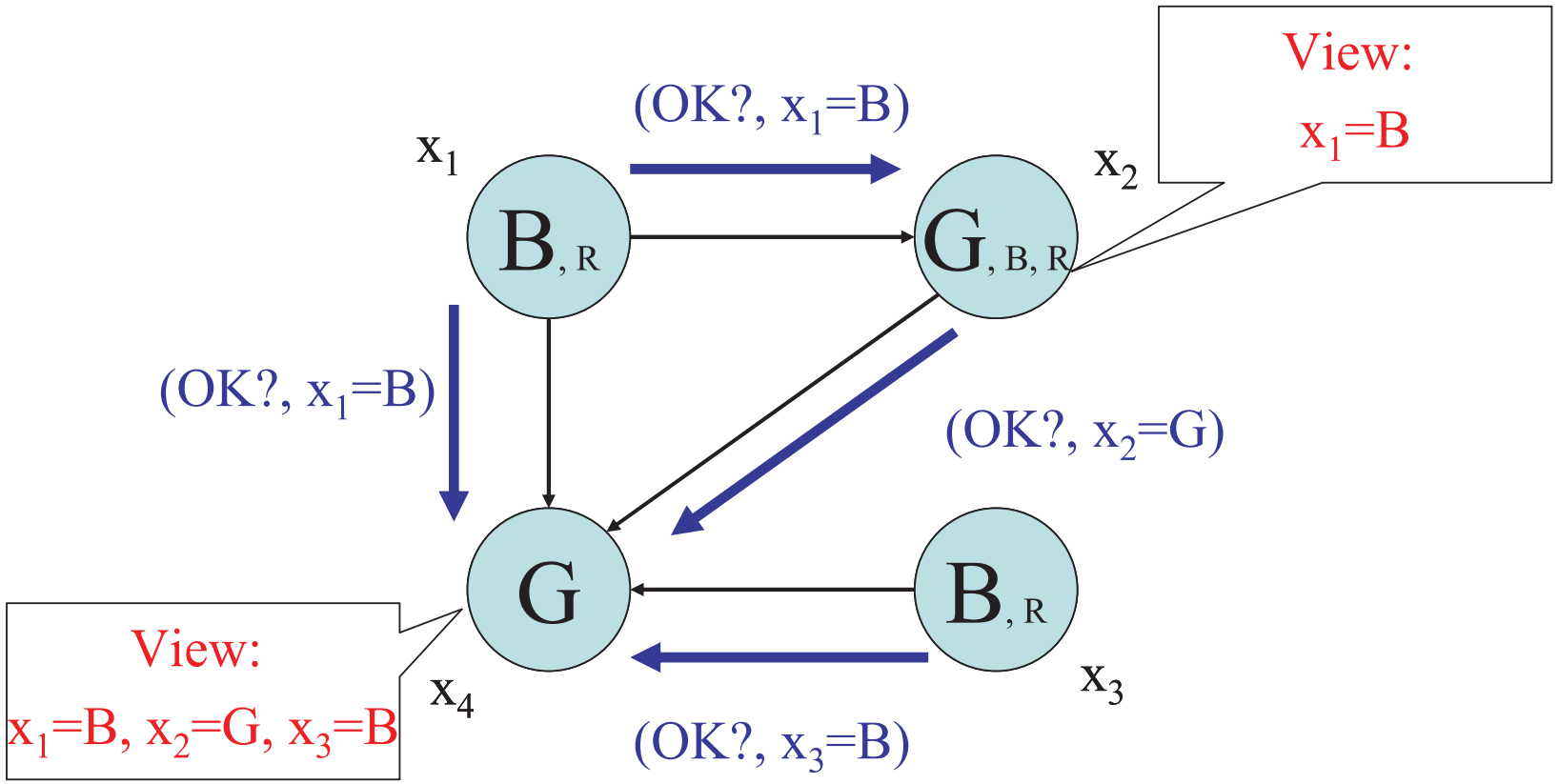
Send OK? messages

Each agent sends OK? messages to its children

الگوریتم عقب‌گرد ناهمگام

مثال : رنگ‌آمیزی گراف (۶ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



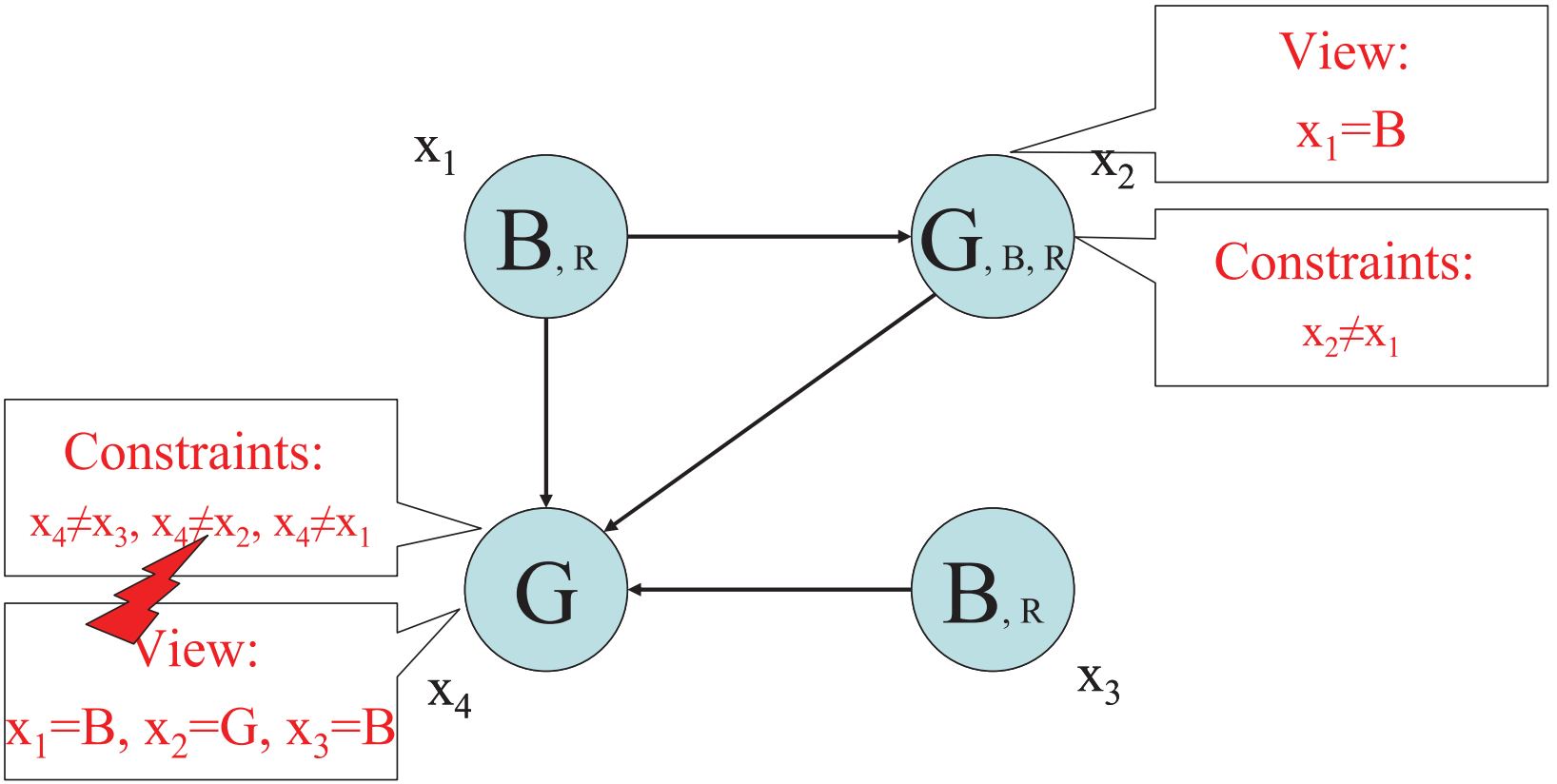
Update view

Agent x_2 and Agent x_4 update their view

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۷ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



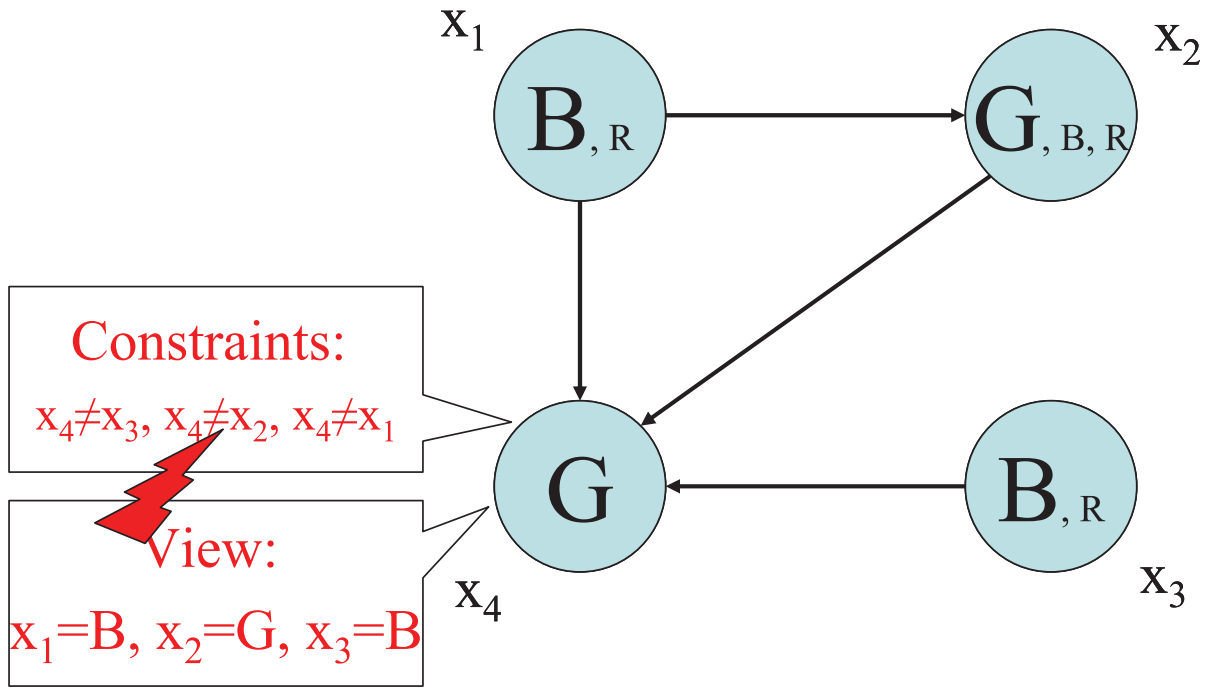
check view

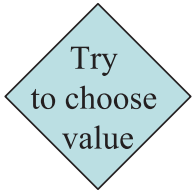
Agent x_2 and Agent x_4 check their view against their constraints, and Agent x_4 discovers a violation

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۸ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



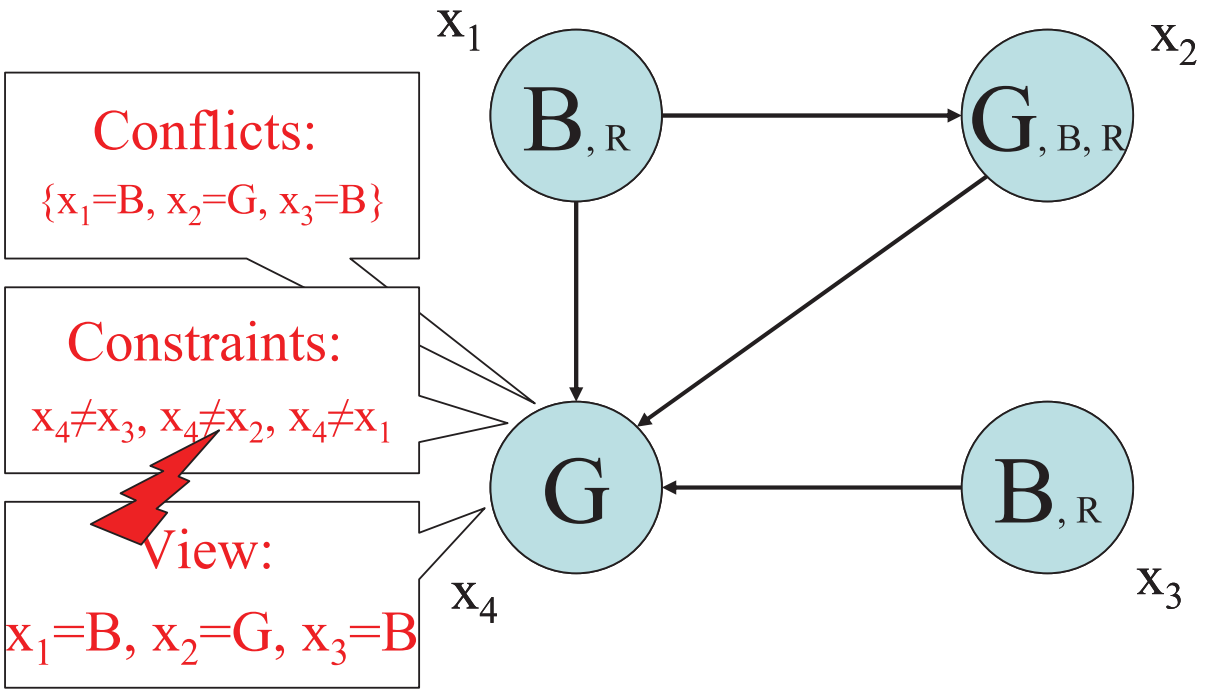


Agent x_4 tries to change its assignment, which is impossible

الگوریتم عقب‌گرد ناهمگام

مثال : رنگ‌آمیزی گراف (۹ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



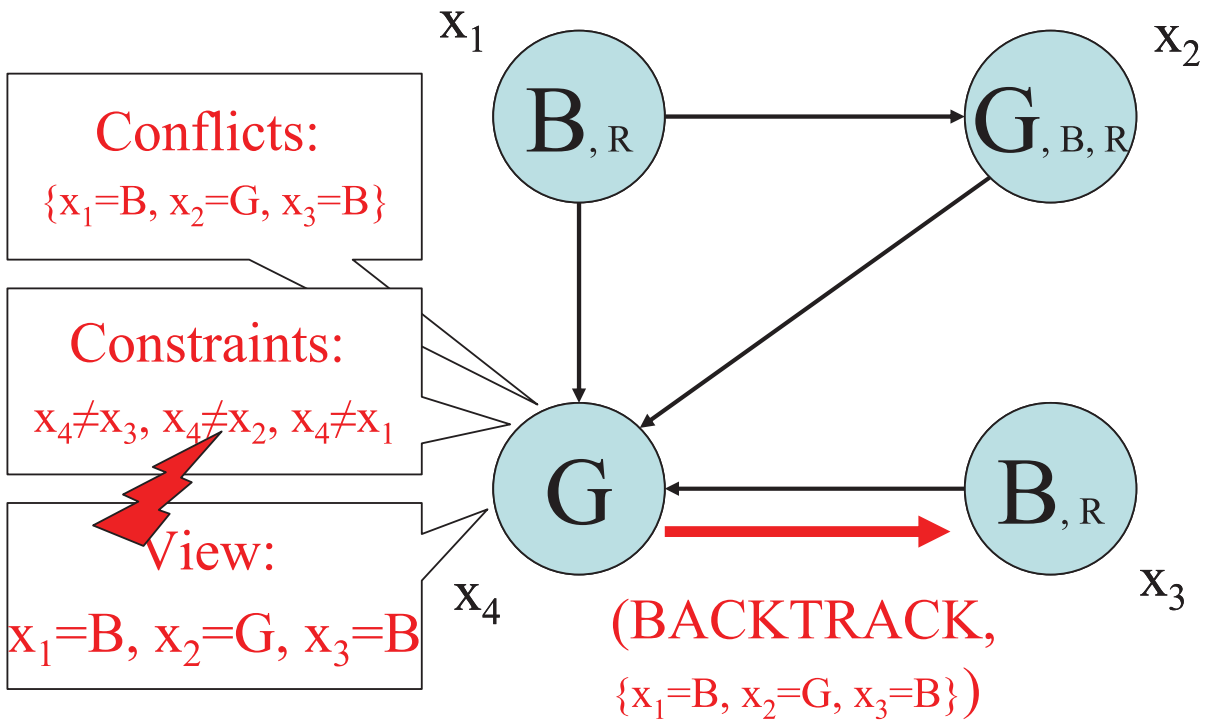
Extract & record conflicts

Agent x_4 extracts and records the conflicts

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۰ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



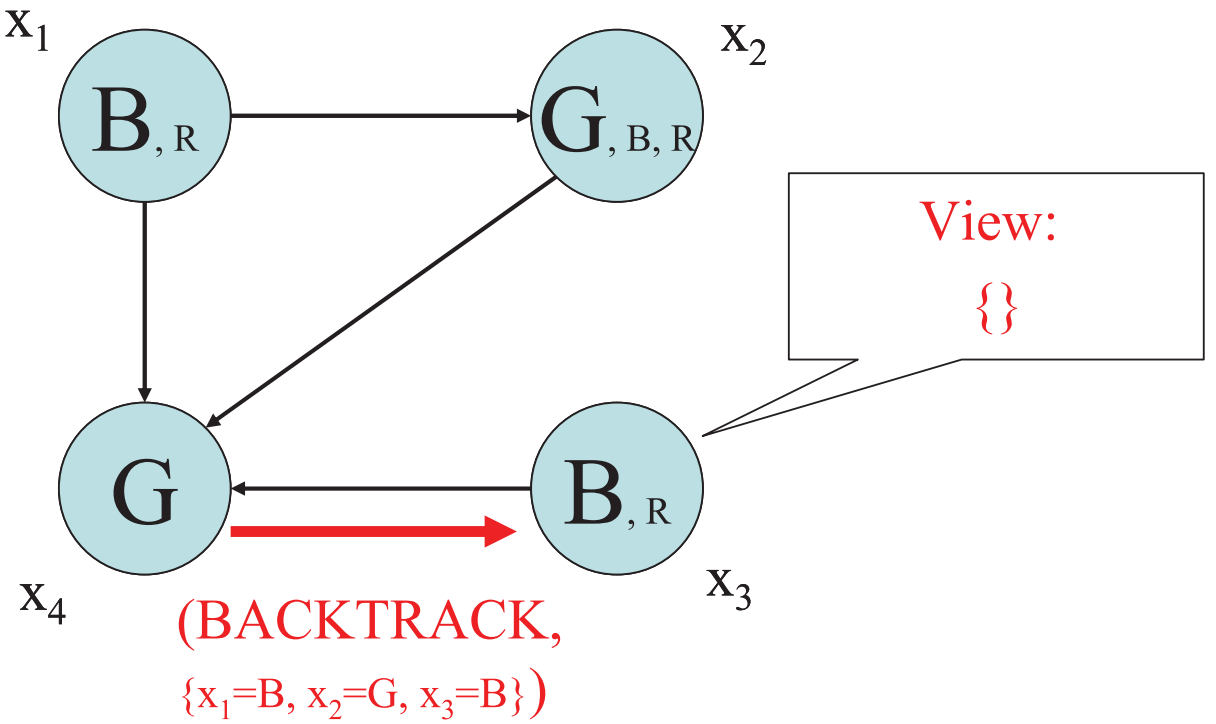
Send BACKTRACK messages

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

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۱ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

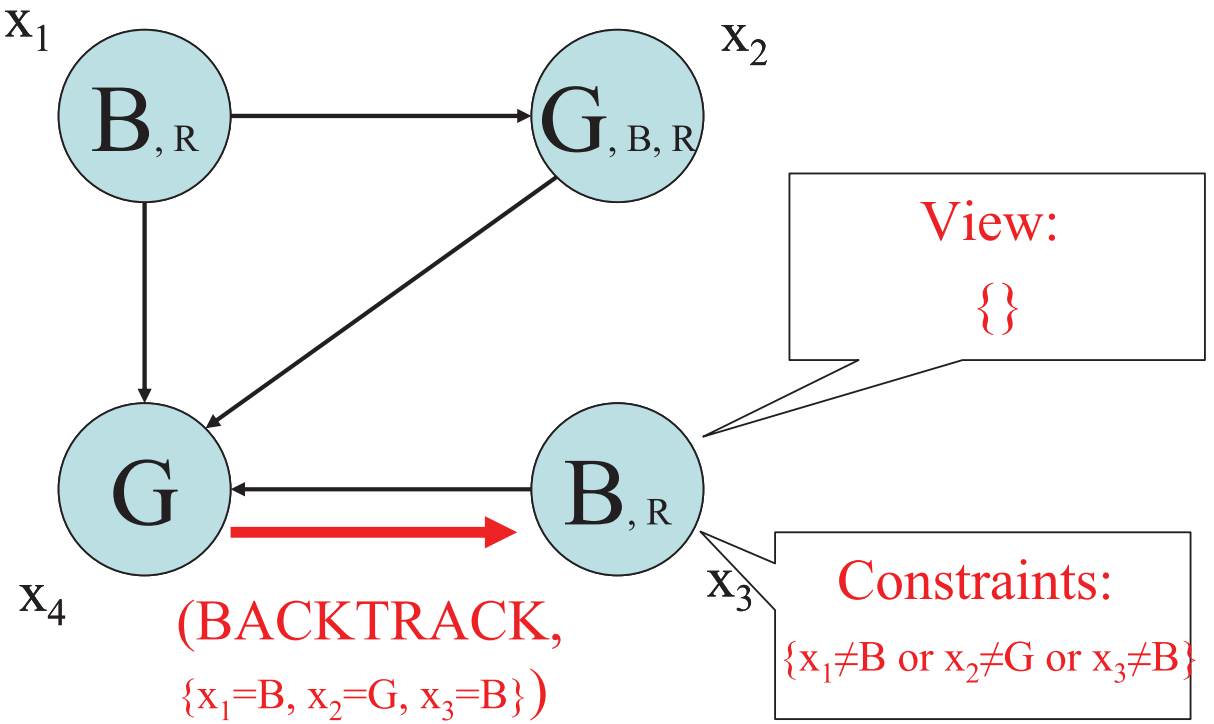


Agent x_3 receives the message and checks the conflict against its view

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۲ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

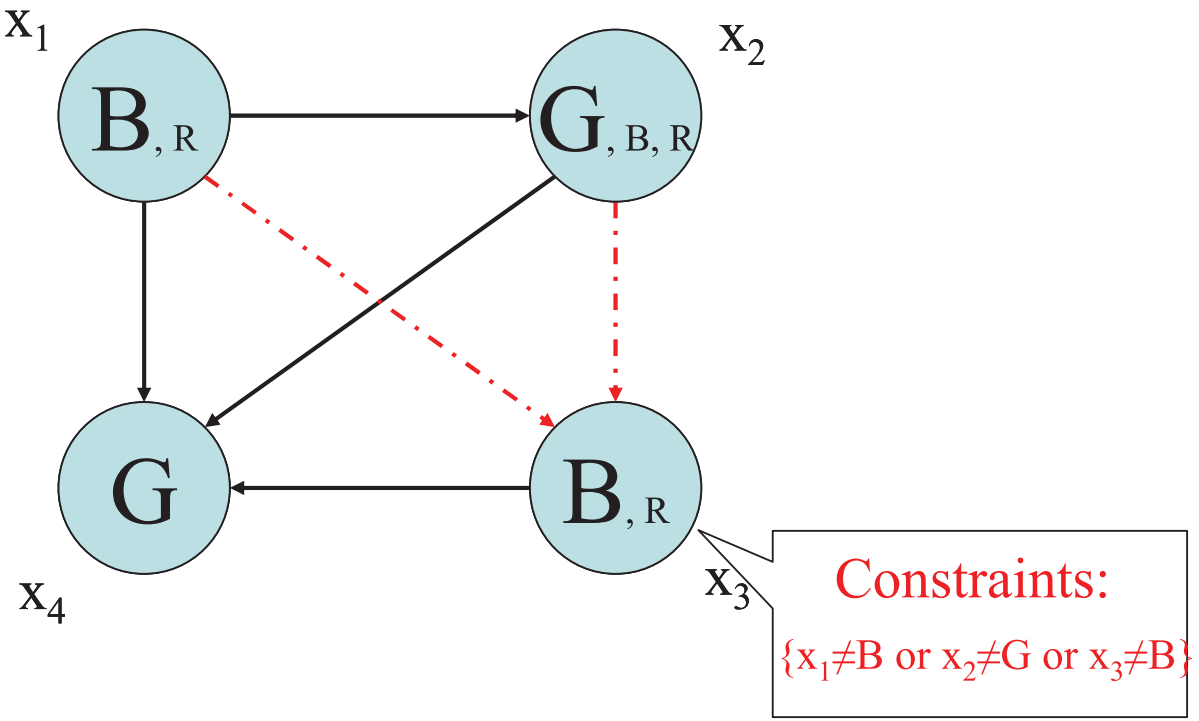


Record new constraint Agent x_3 records the conflict as a new constraint

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۳ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

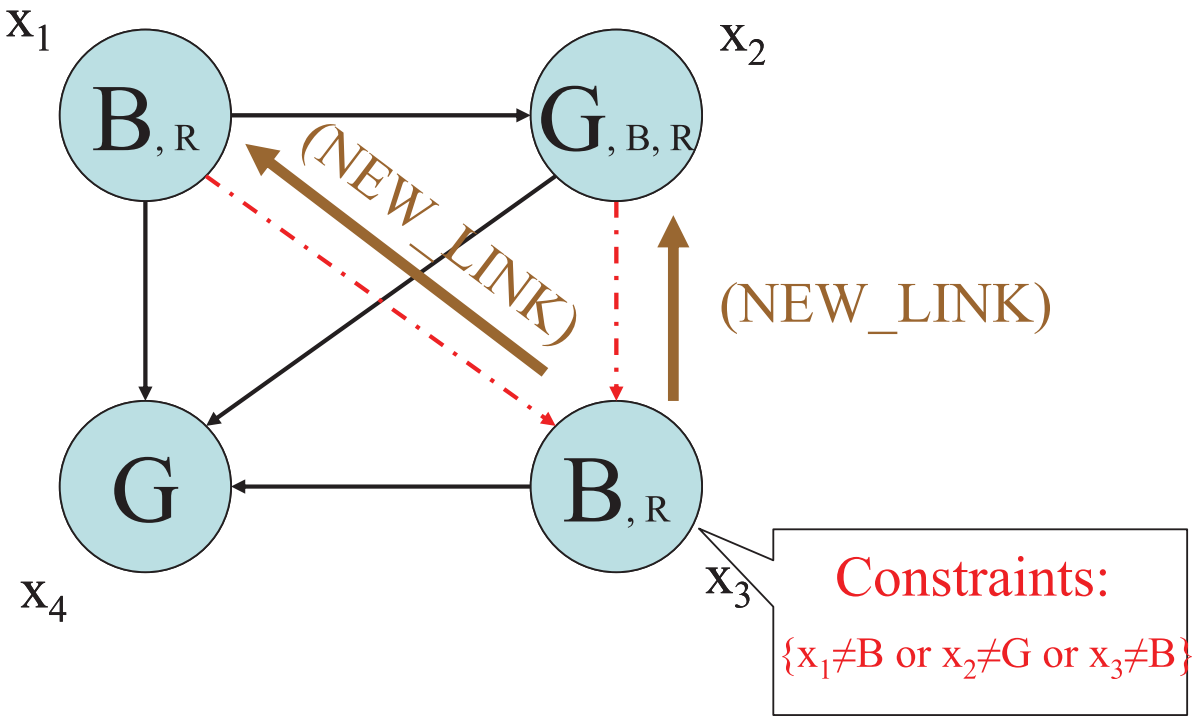


Agent x_3 checks if it needs new links to enforce it

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۴ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

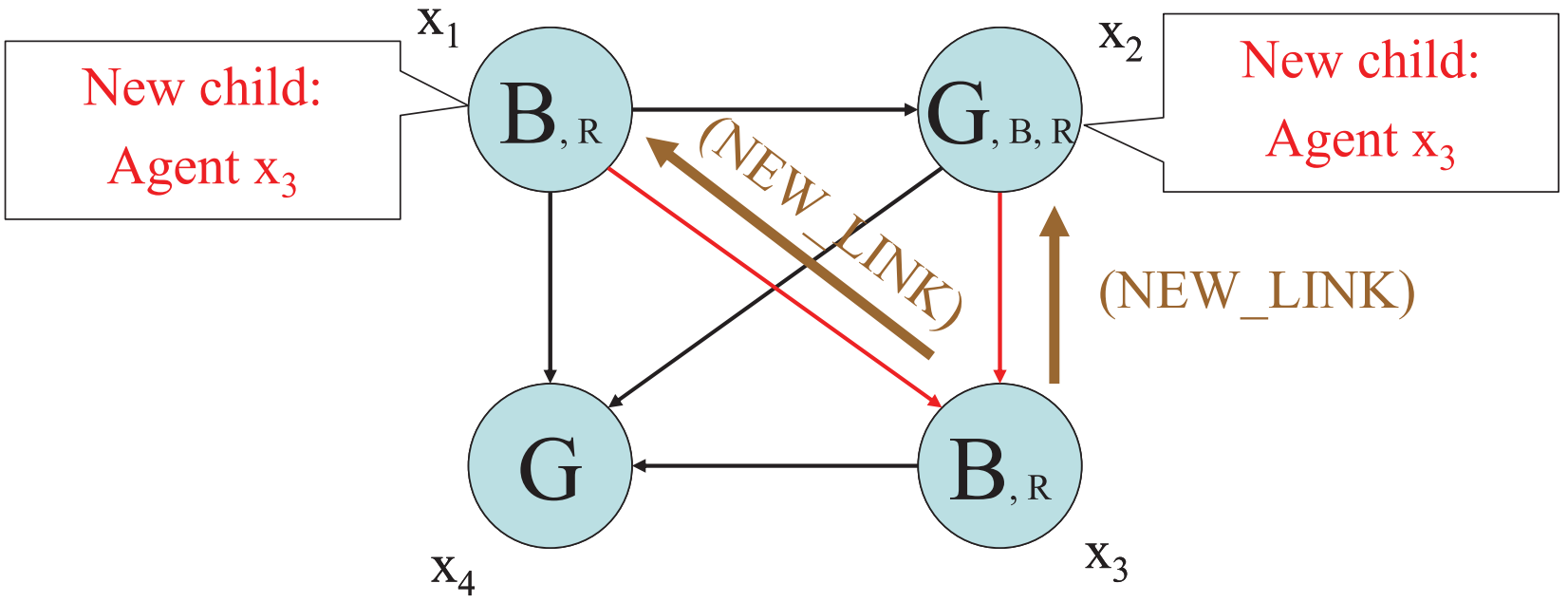


NEW_LINK Agent x_3 sends NEW_LINK requests

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۵ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



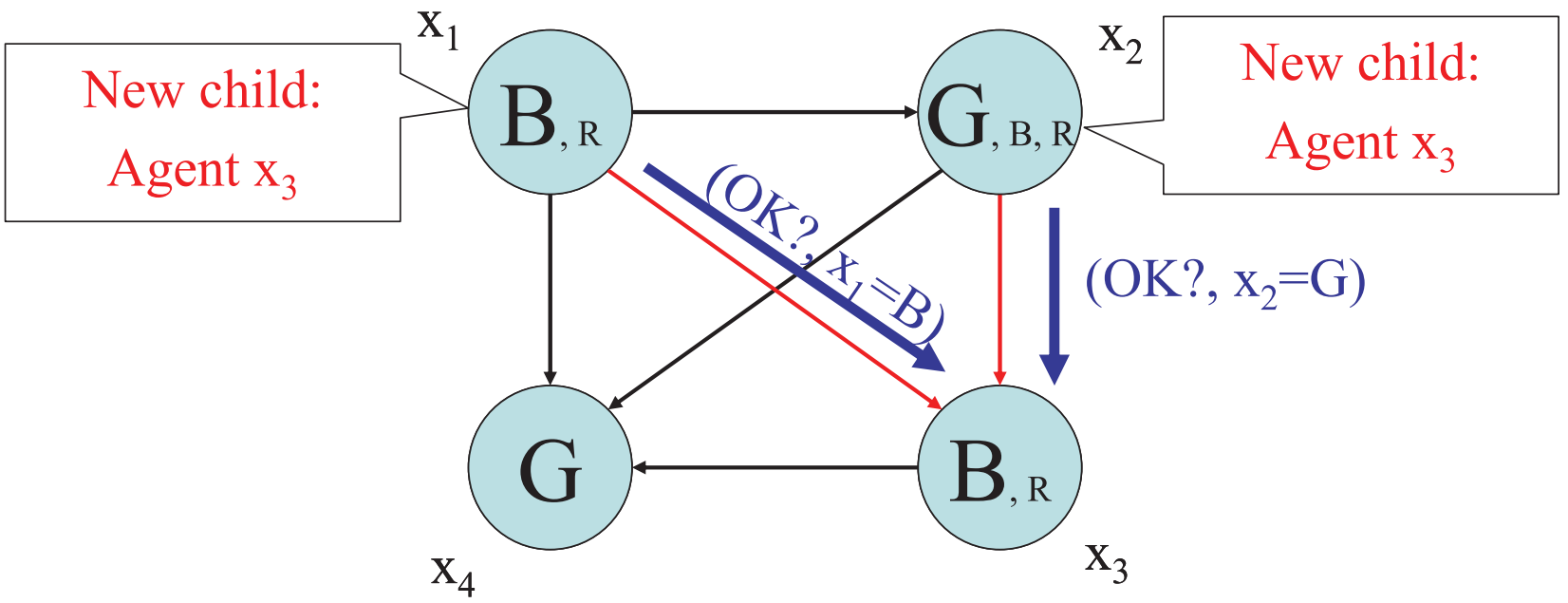
Add child

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

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۶ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



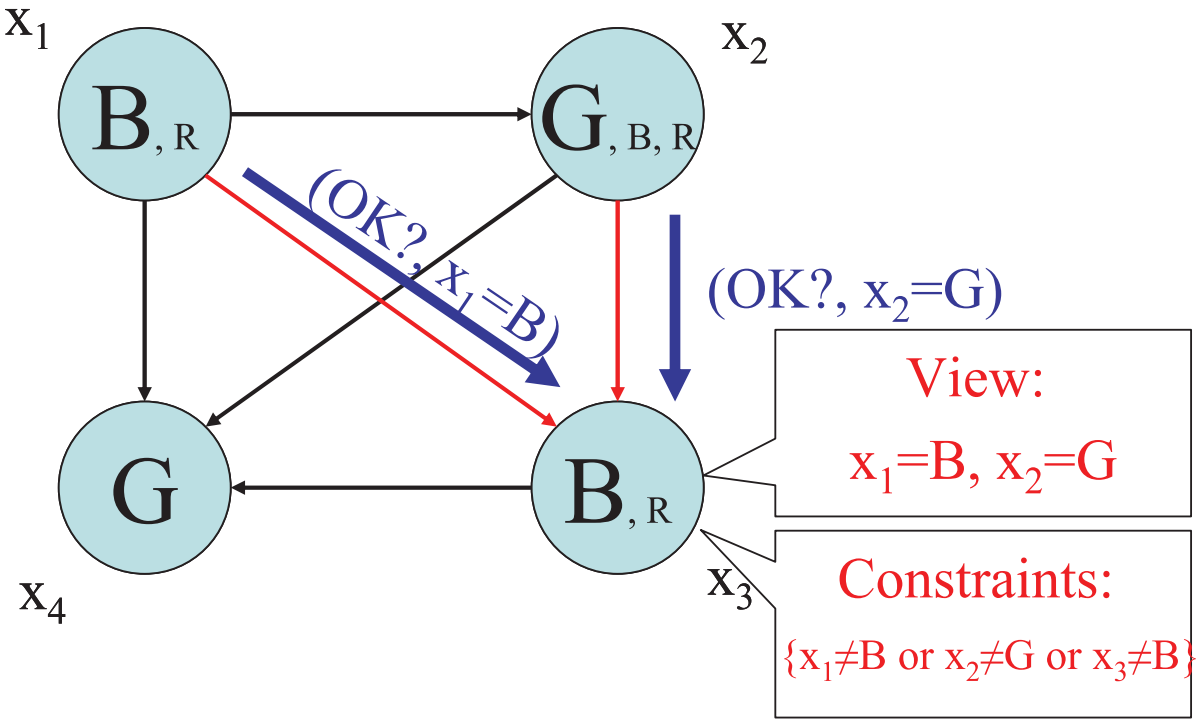
Send OK?

Agents x_1 and x_2 send OK? to confirm new links

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۷ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



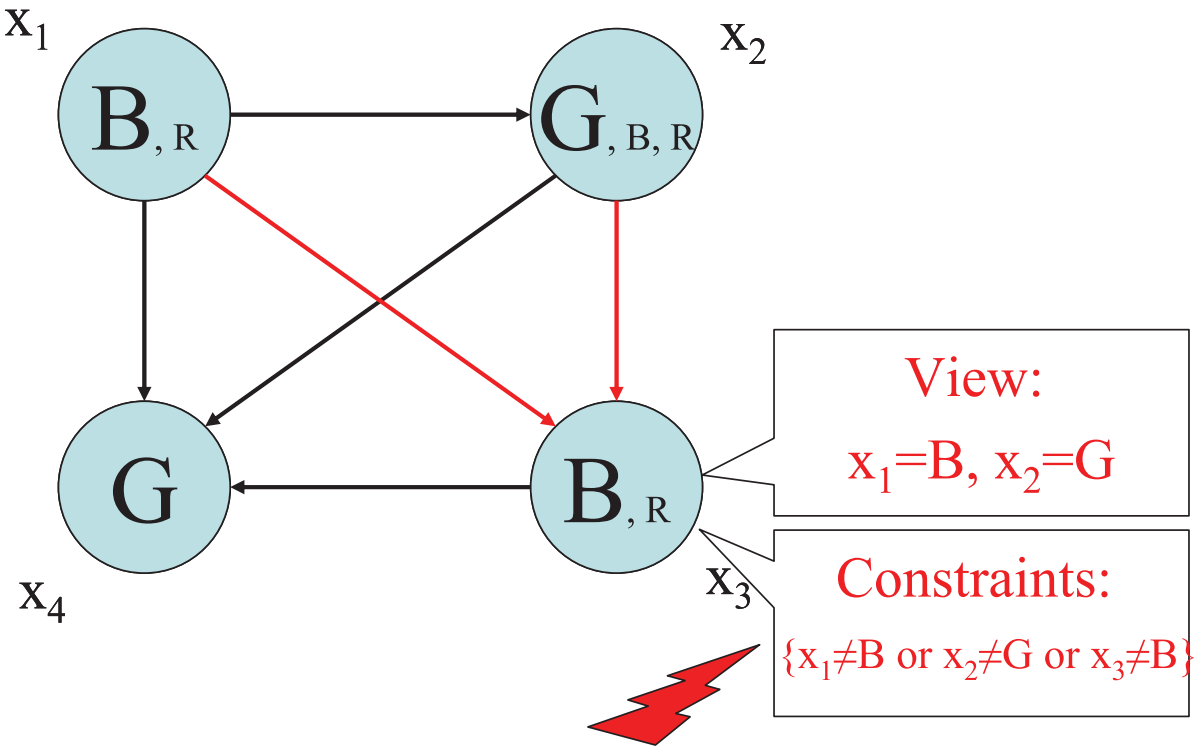
Update view

Agent x_3 receives messages and updates its view

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۸ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



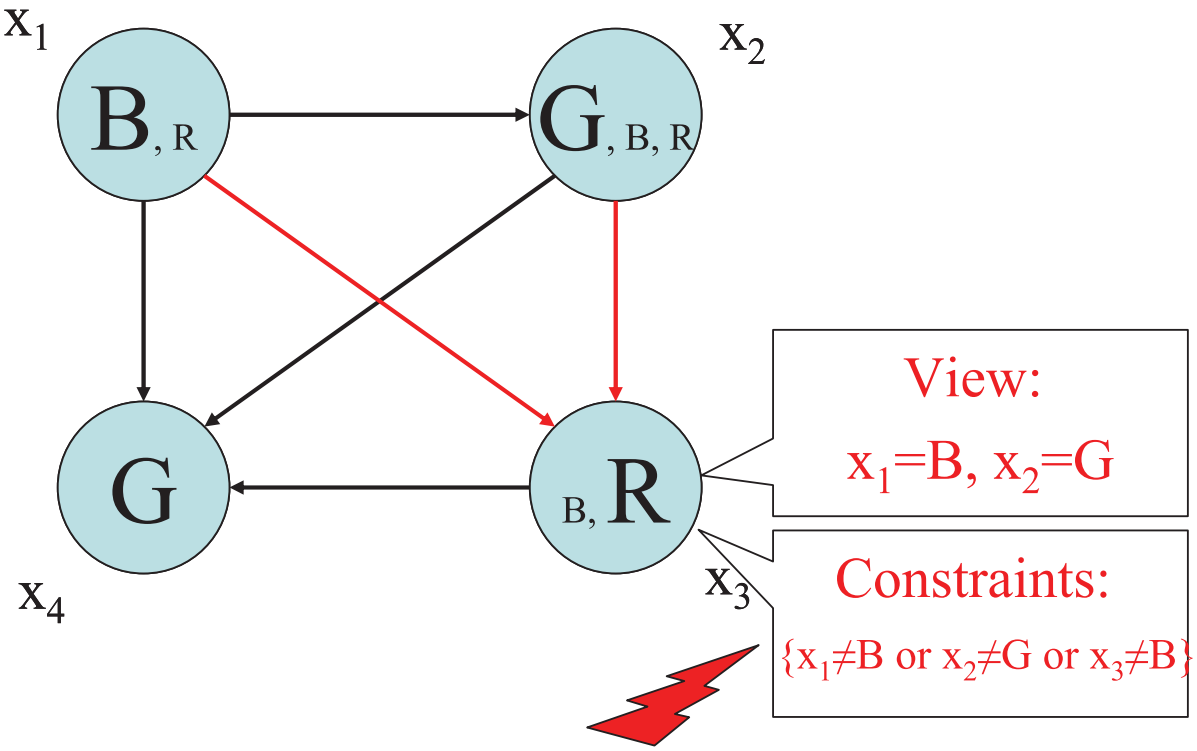
check view

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

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۱۹ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



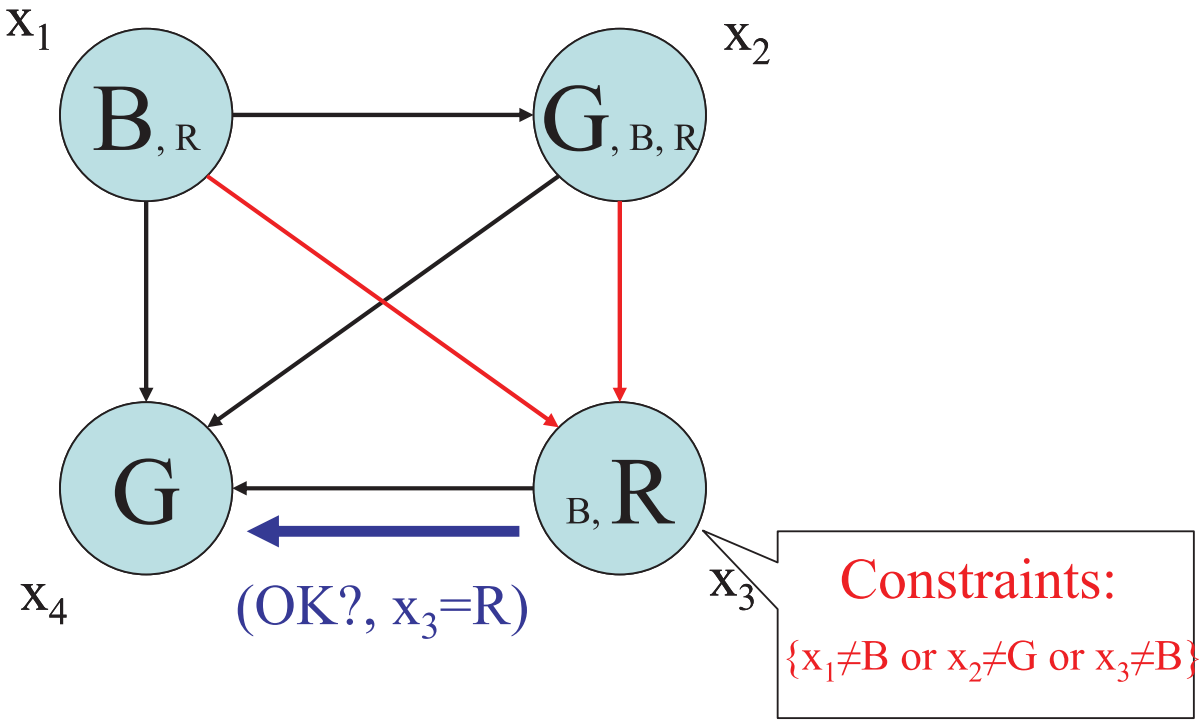
Try to choose value

Agent x_3 tries to change its value to R

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۰ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



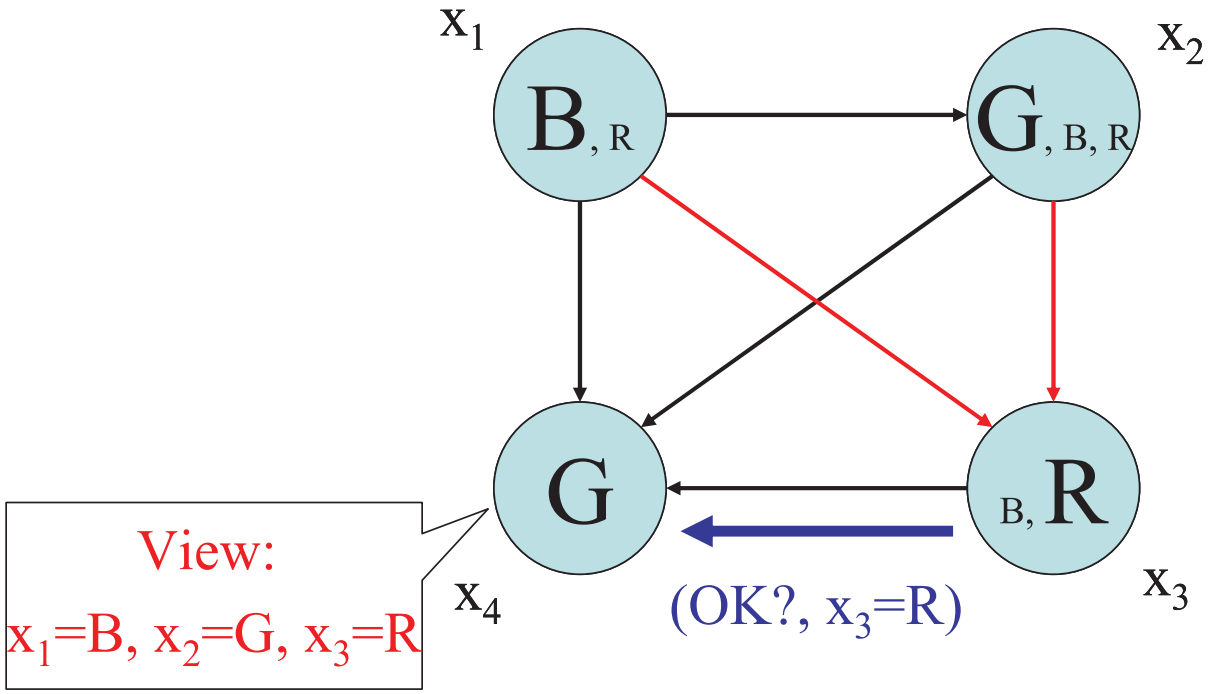
Send OK? messages

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

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۱ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



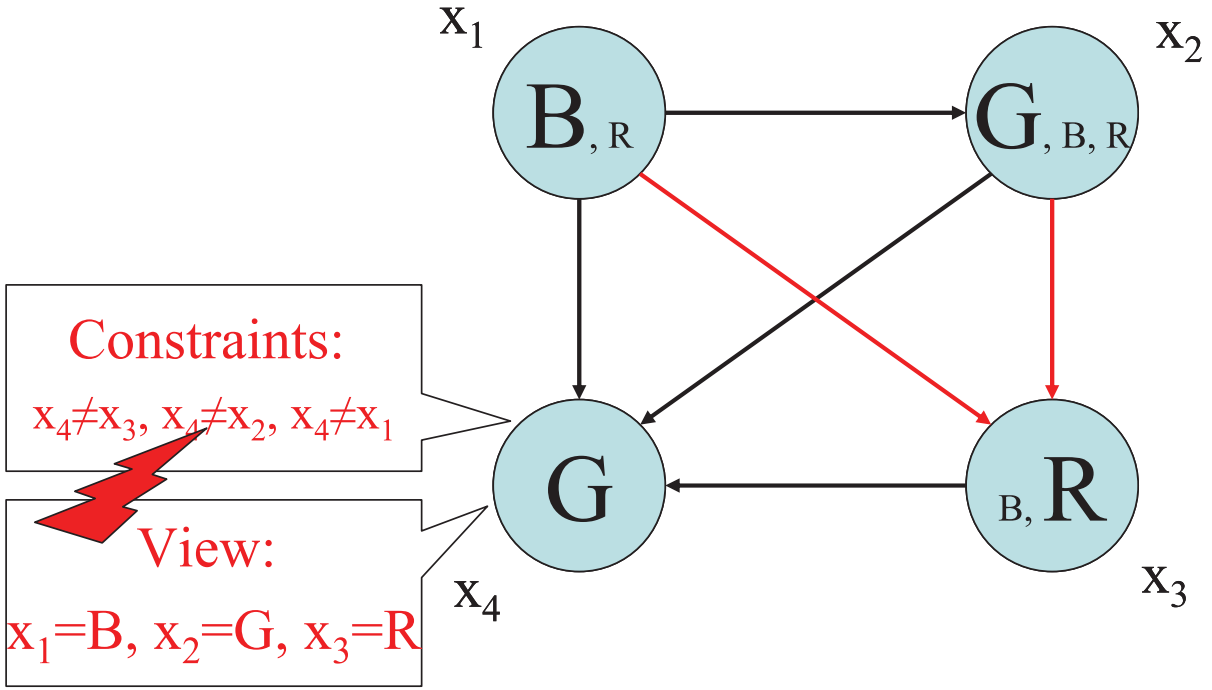
Update view

Agent x_4 receives the OK? message and updates its view

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۲ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

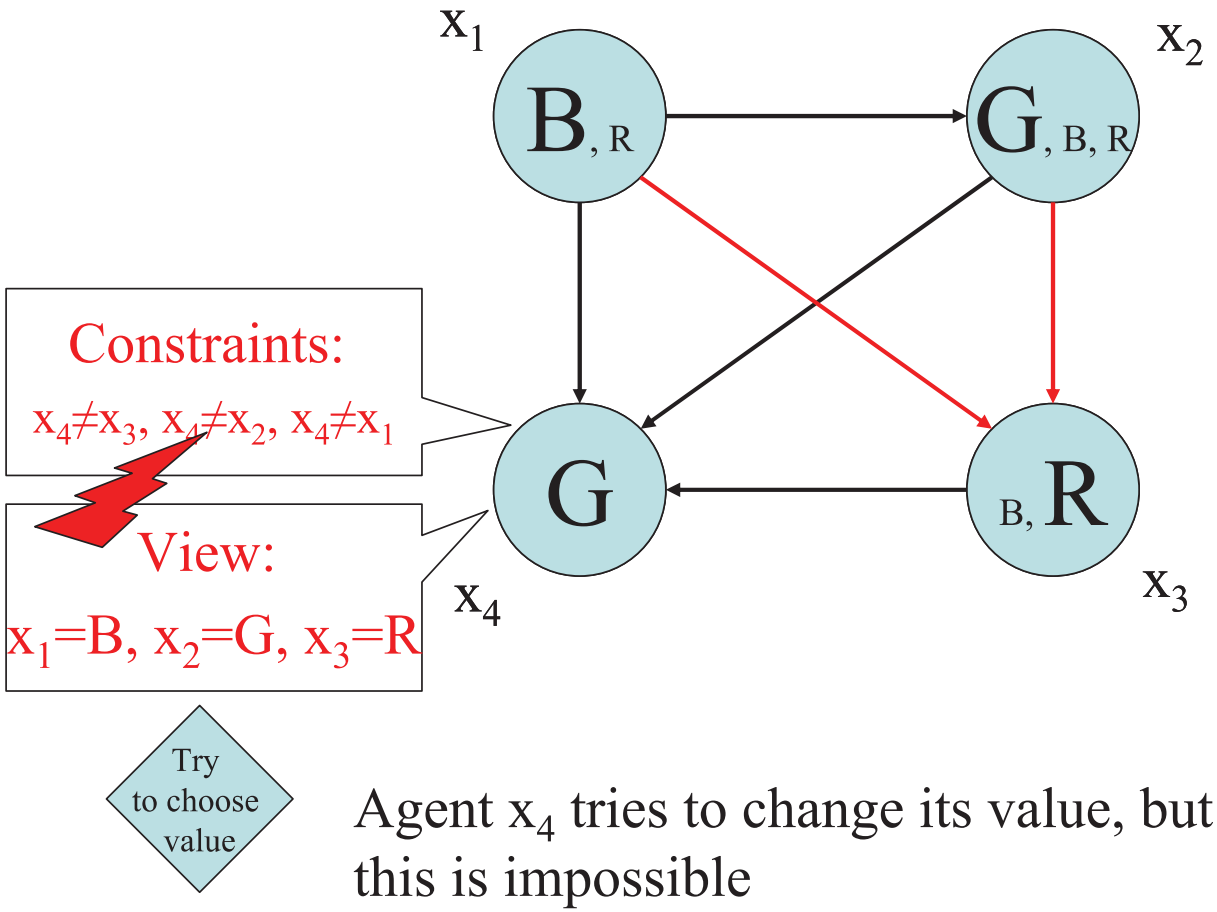


Agent x_4 checks its view against its constraints and sees the violation has not been resolved…

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۳ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

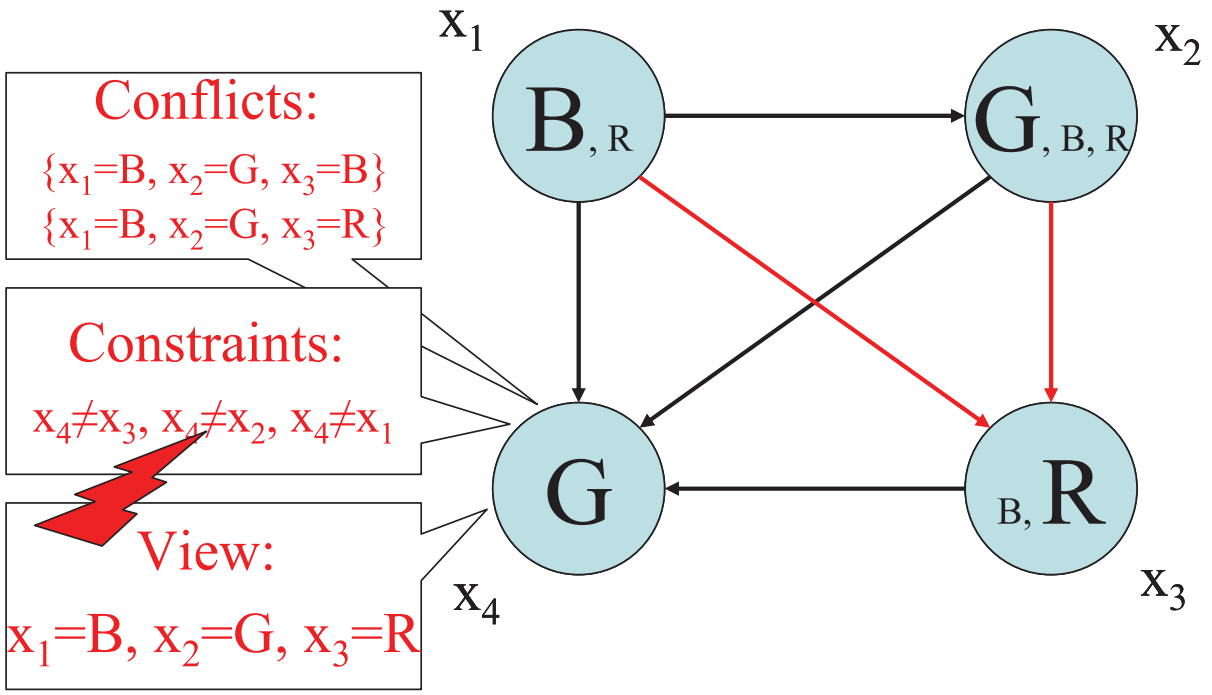


Agent x_4 tries to change its value, but this is impossible

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۴ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



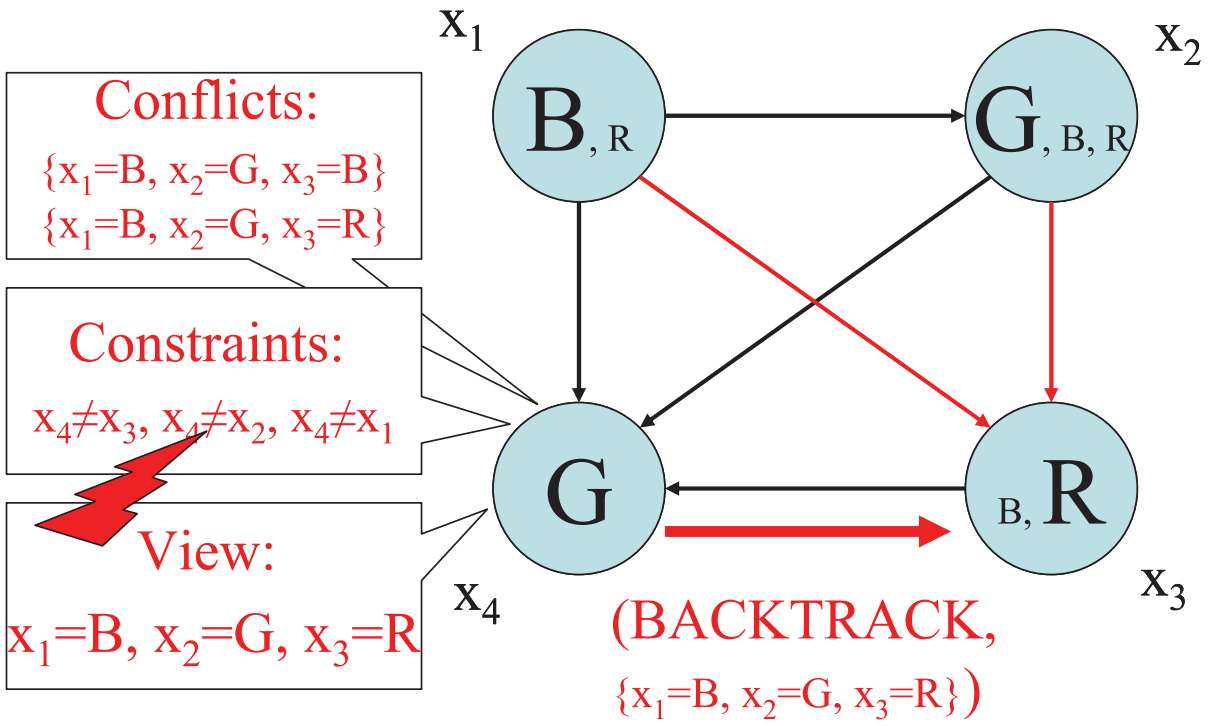
Extract & record conflicts

Agent x_4 extracts and records the conflicts

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۵ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



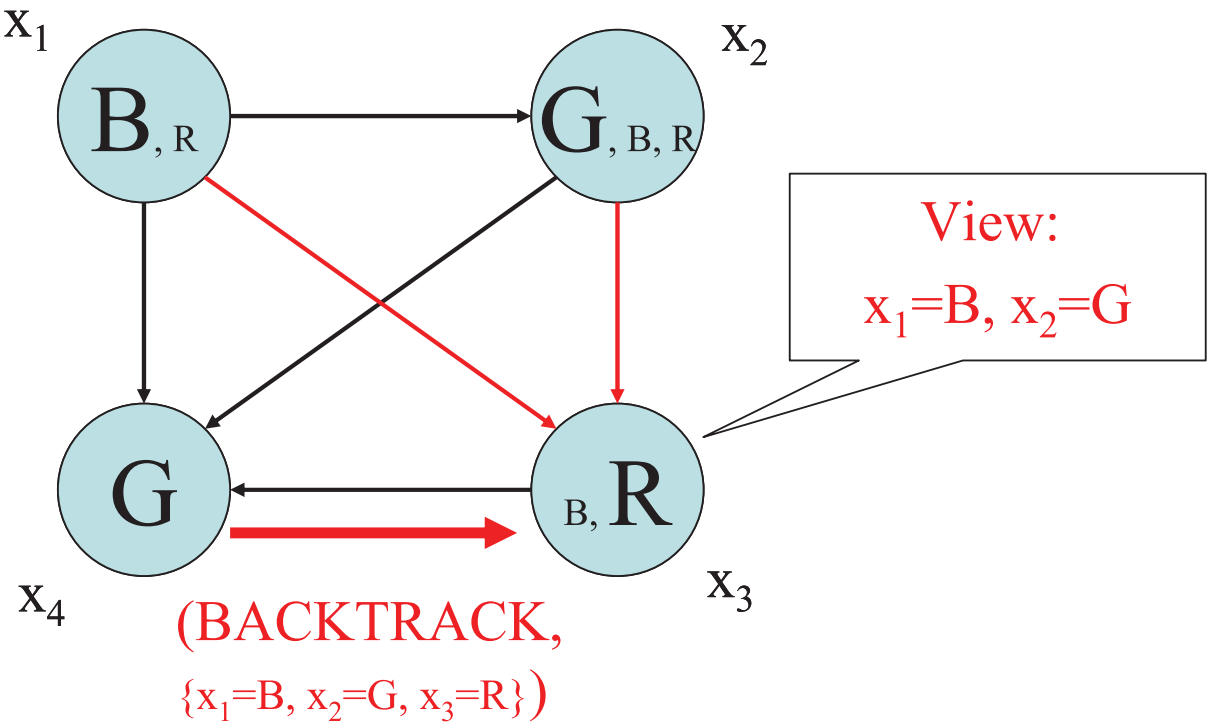
Send BACKTRACK messages

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

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۶ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

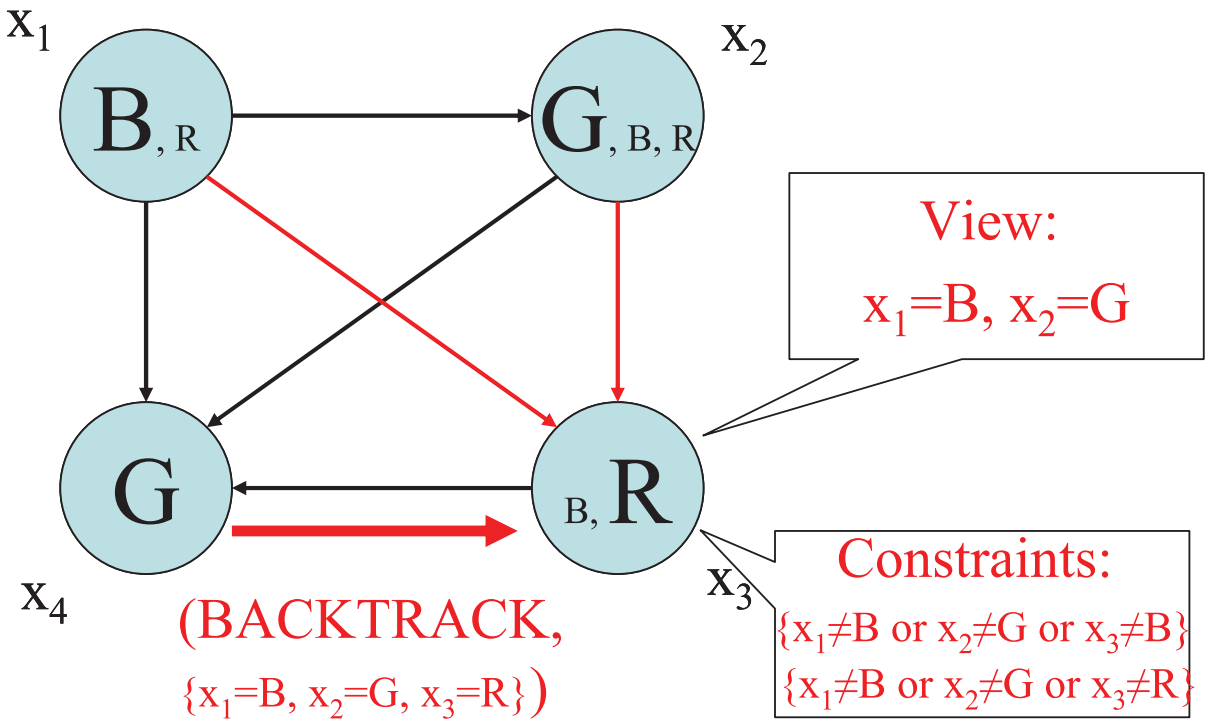


Agent x_3 receives the message and checks the conflict against its view

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۷ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



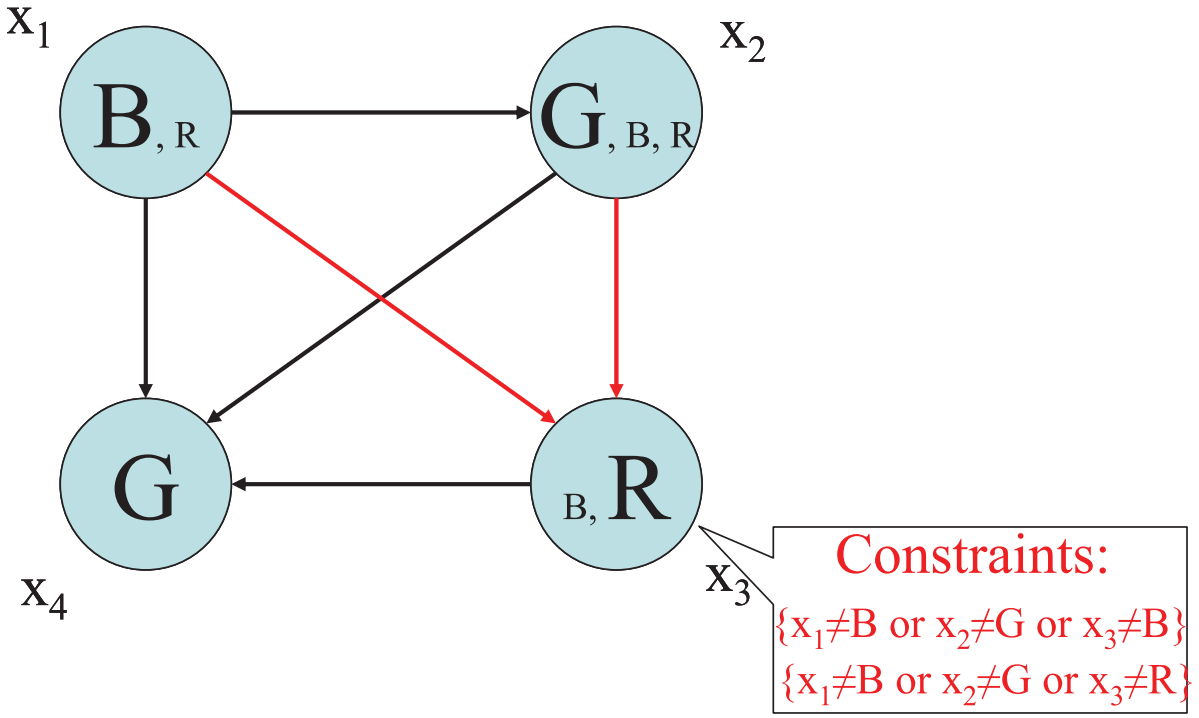
Record new constraint

Agent x_3 records the conflict as a new constraint

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۸ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



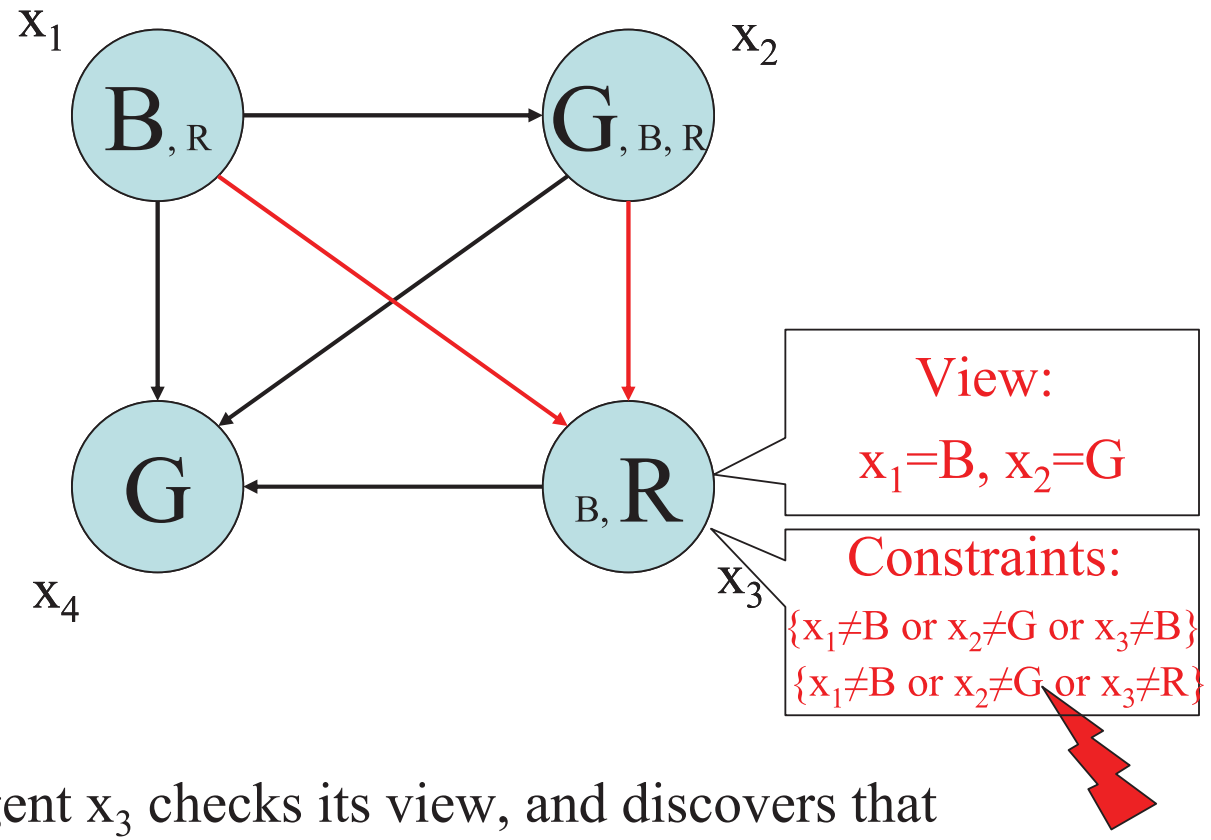
need link?

Agent x_3 needs no new link to enforce it

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۲۹ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

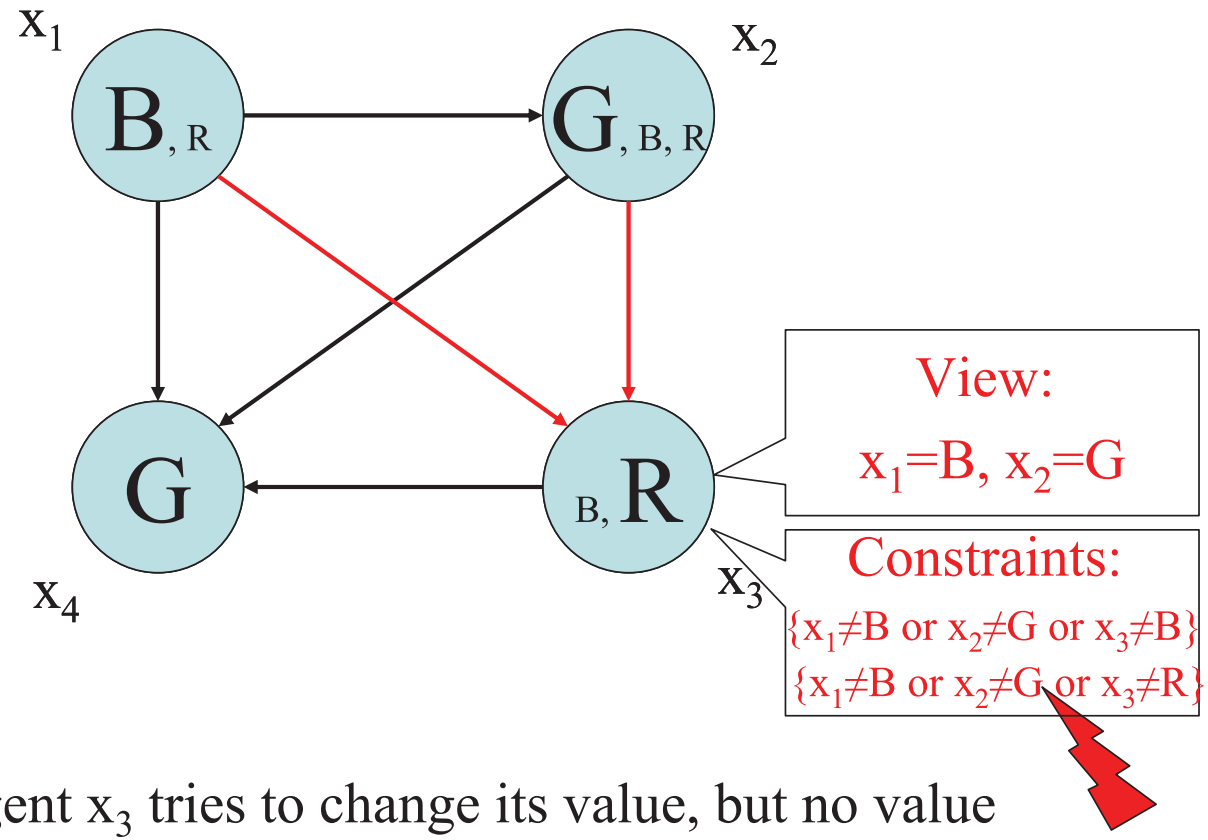


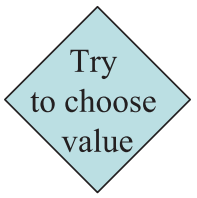
Agent x_3 checks its view, and discovers that one constraint (the new one) is violated

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۰ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



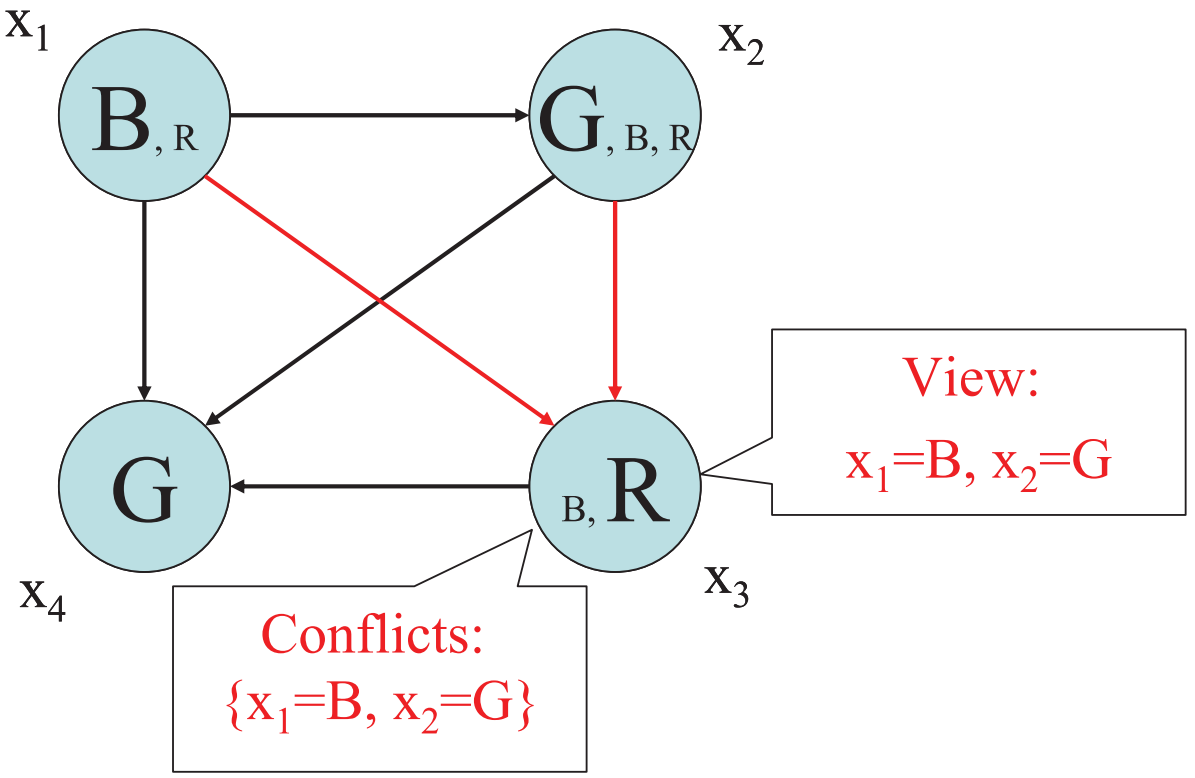


Agent x_3 tries to change its value, but no value satisfies all the constraints

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۱ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



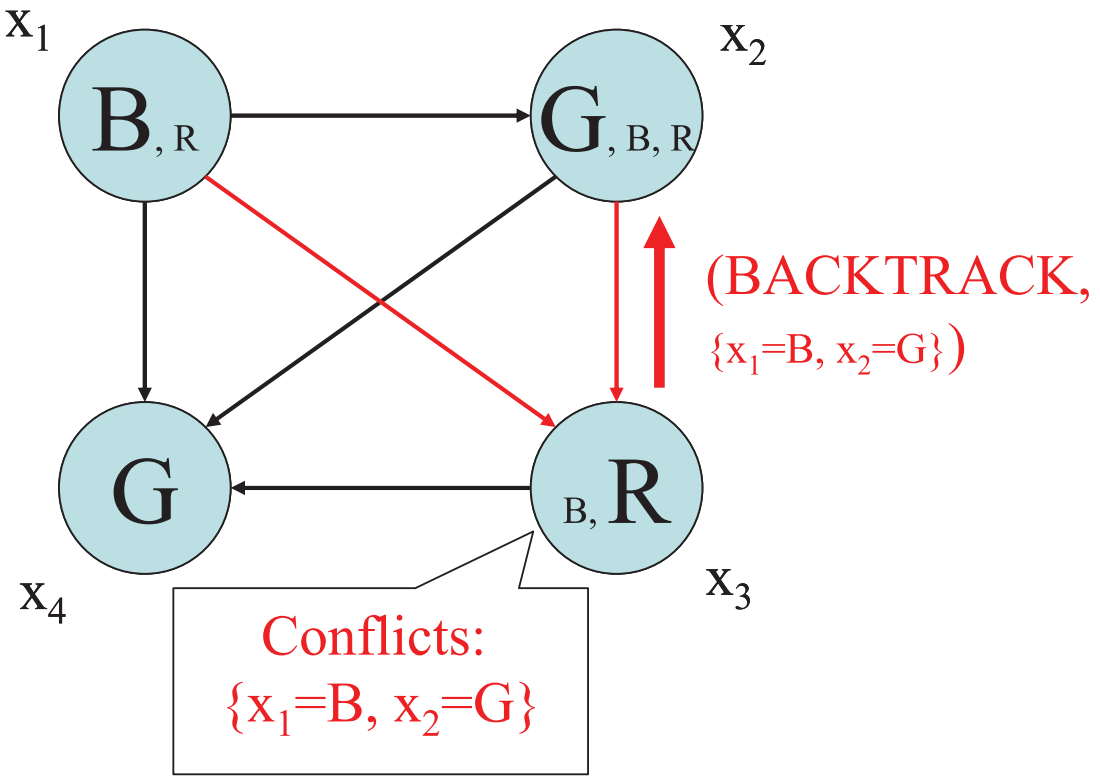
Extract & record conflicts

Agent x_3 extracts and records new conflicts

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۲ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



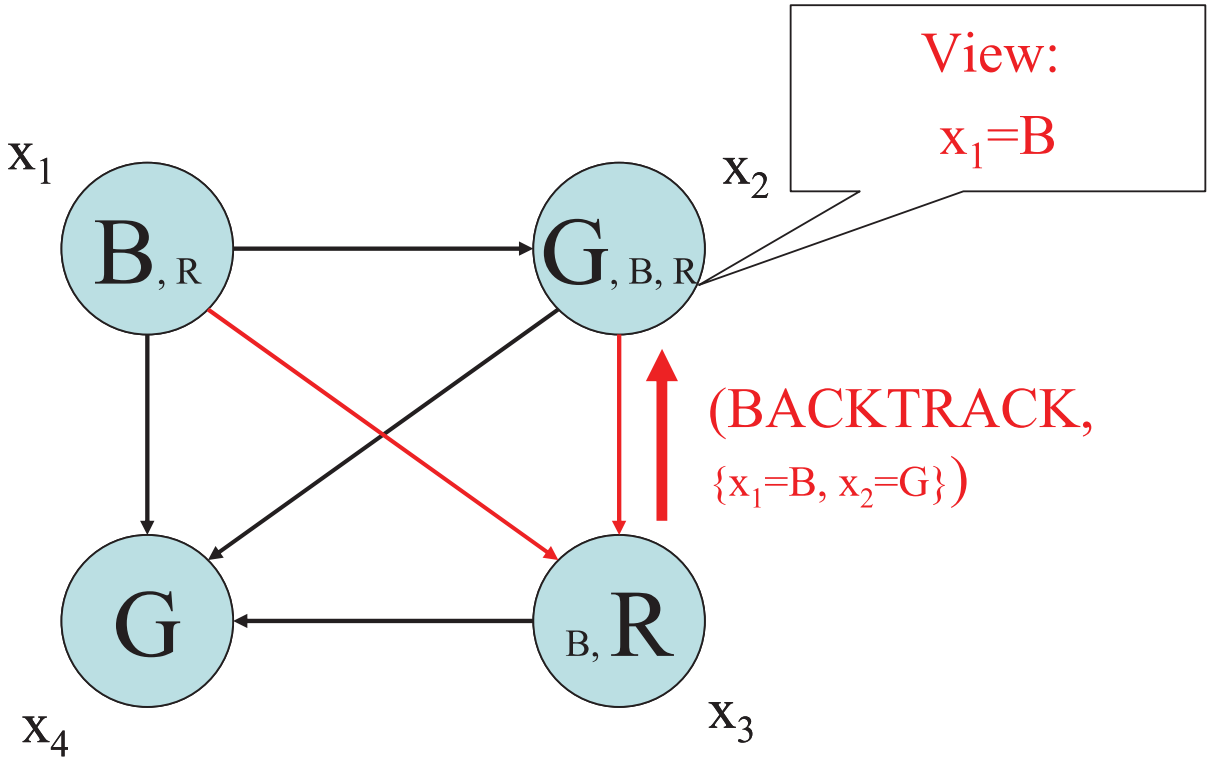
Send BACKTRACK messages

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

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۳ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

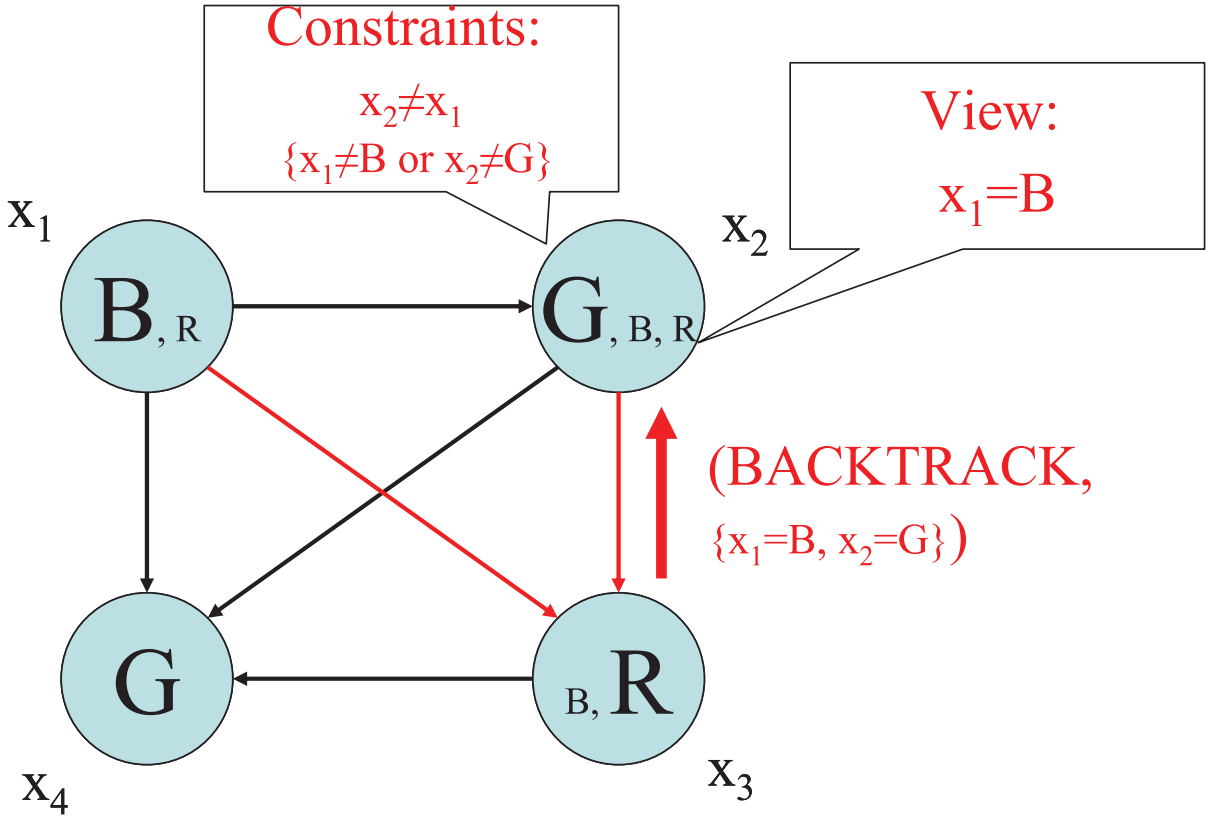


Agent x_2 receives the message and checks the conflict against its view

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۴ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

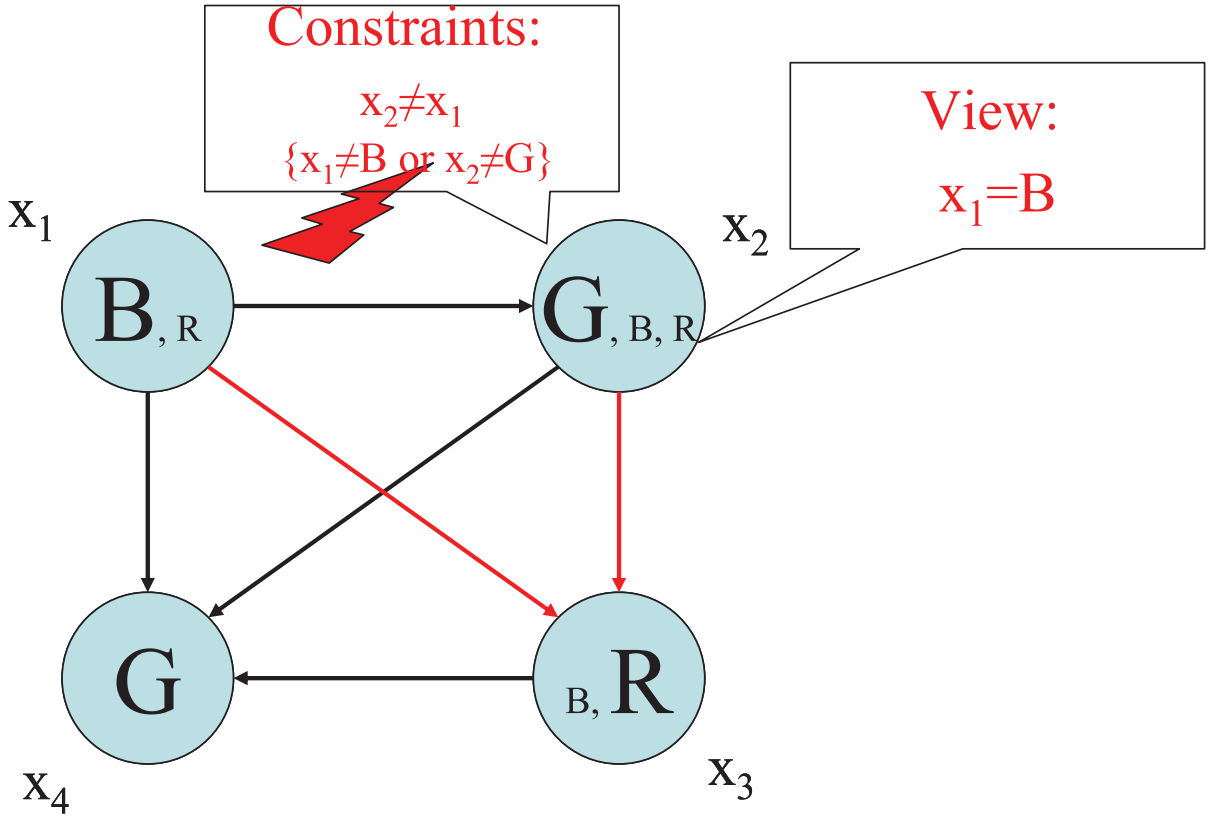


Record new constraint Agent x_2 records the conflict as a new constraint

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۵ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

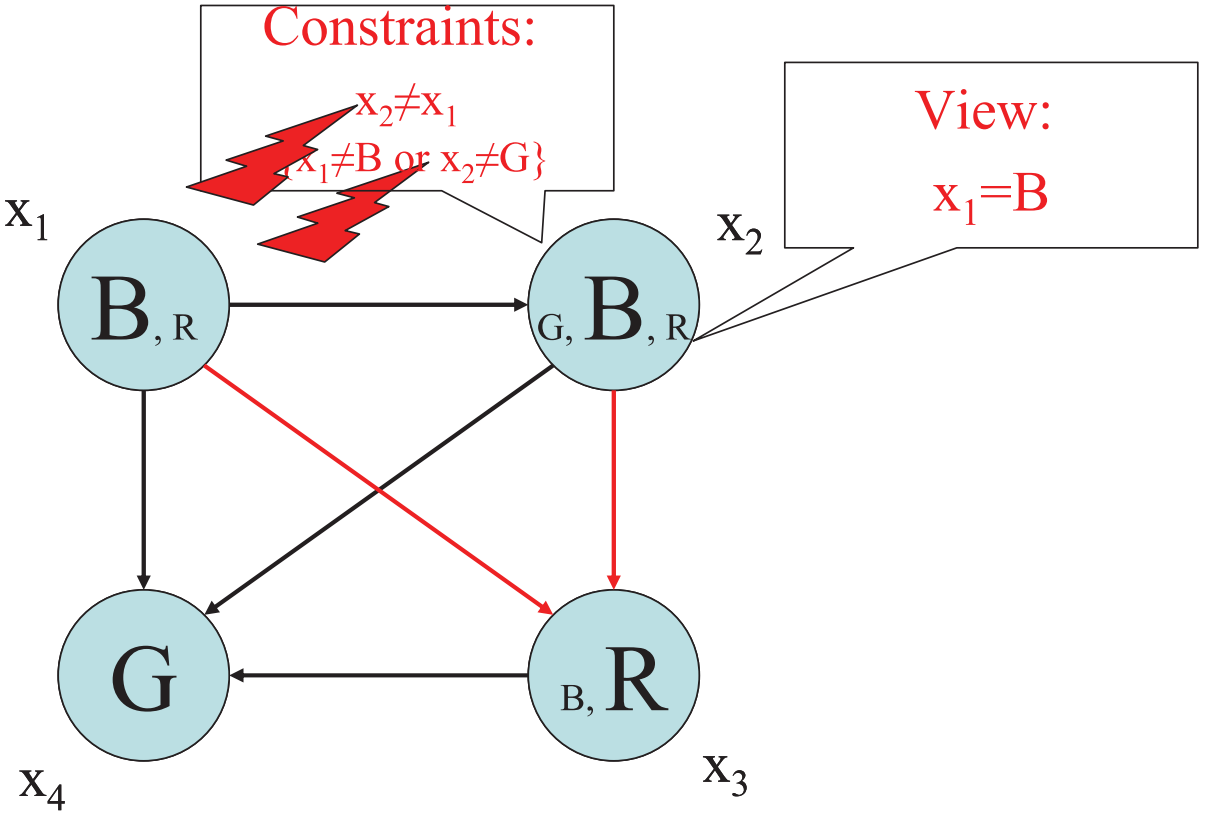


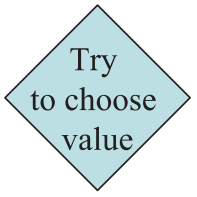
Agent x_3 checks its view, and discovers that one constraint (the new one) is violated

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۶ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



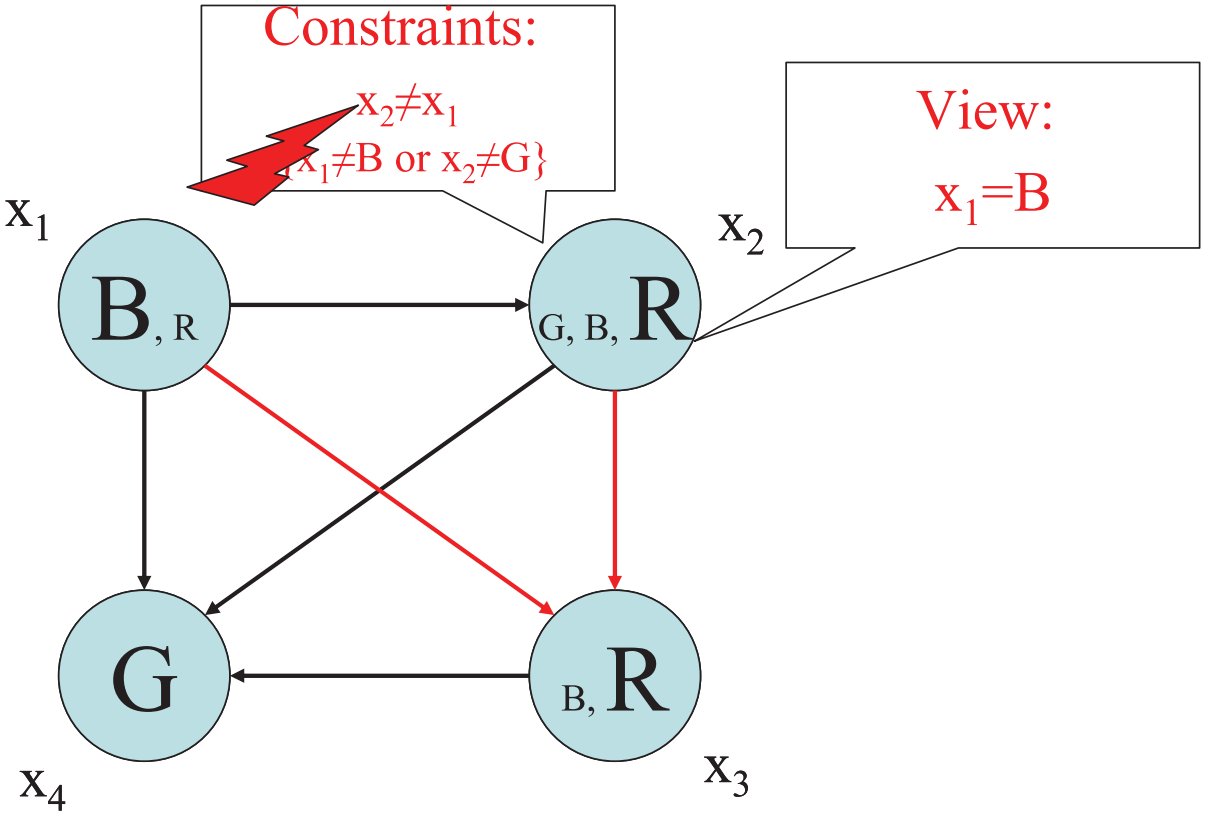


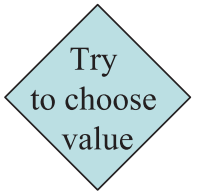
Agent x_2 tries to change its value to B,
but it would violate the first constraint

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۷ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



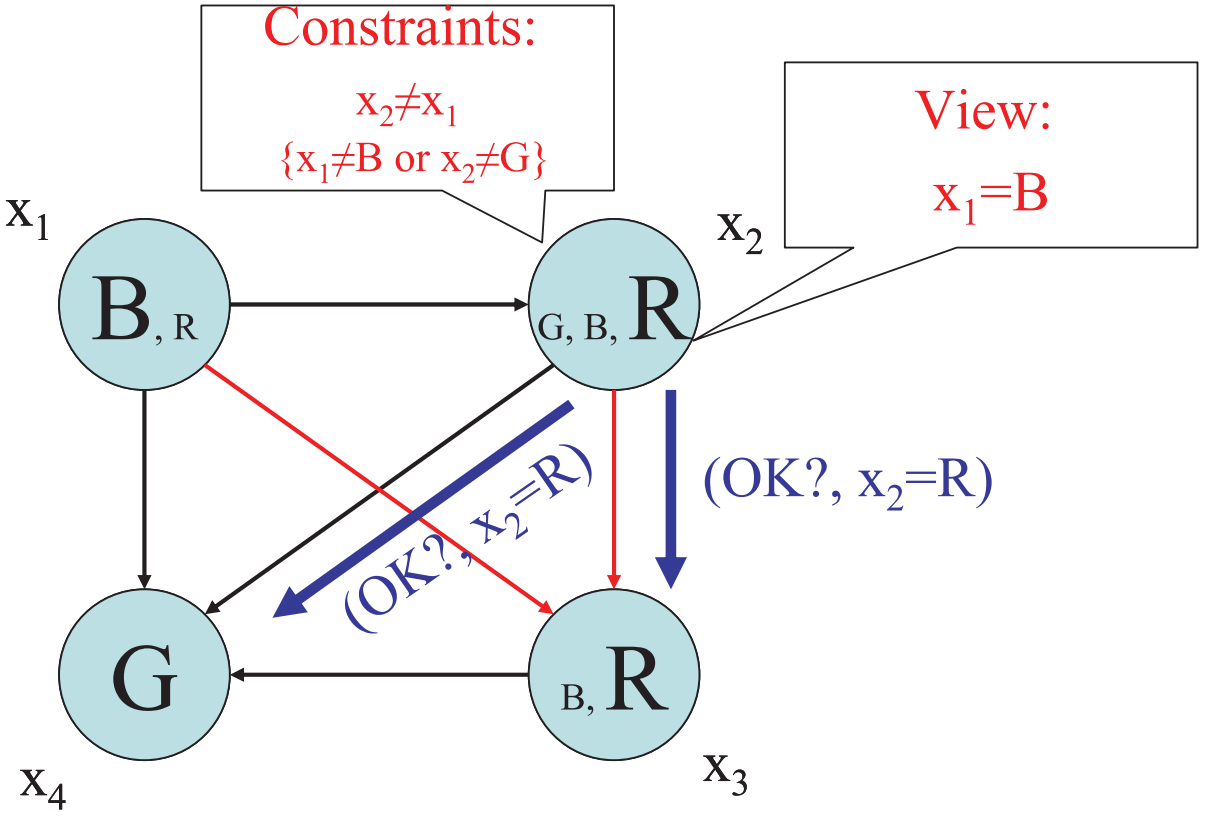


Agent x_2 tries to change its value to R

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۸ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



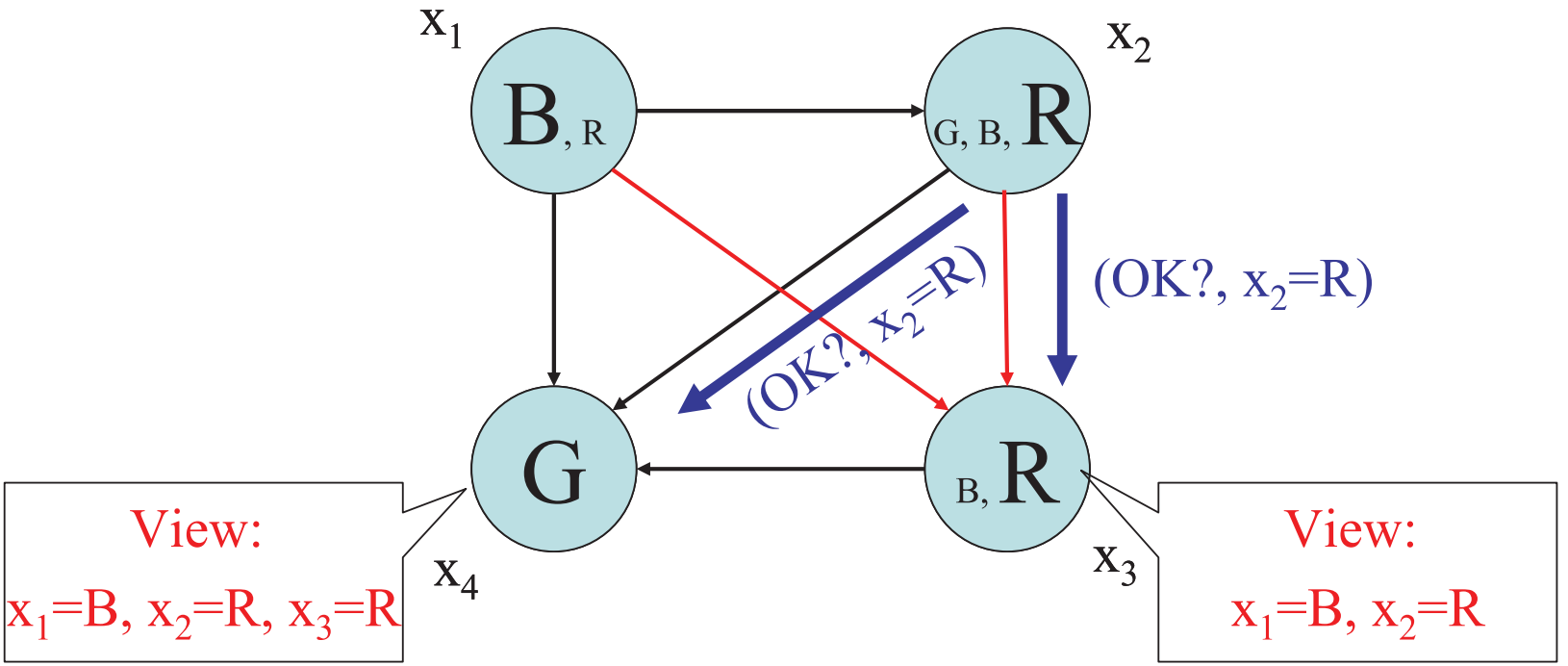
Send OK? messages

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

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۳۹ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM



Update view

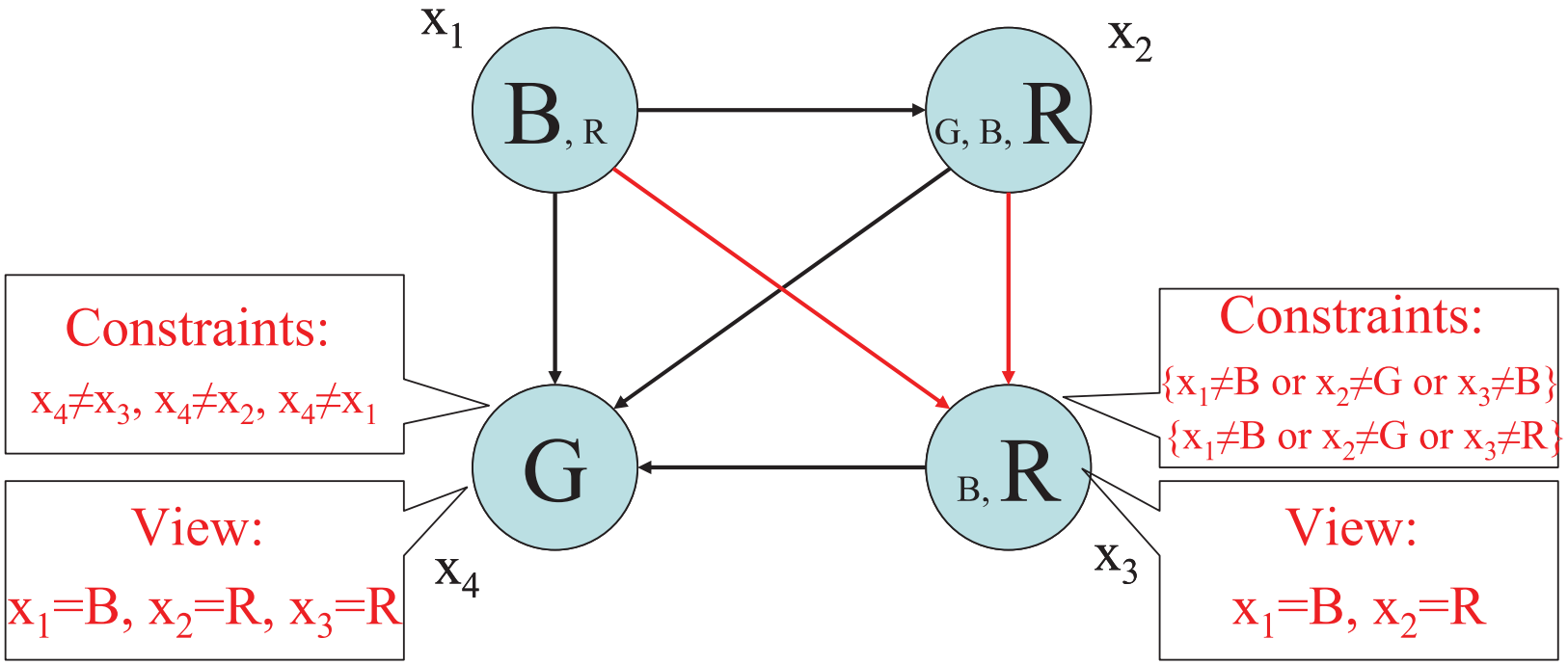
Agent x_3 and Agent x_4 receive the messages and update their views

الگوریتم عقب‌گرد ناهمگام

مثال: رنگ‌آمیزی گراف (۴۰ از ۴۰)

THE ASYNCHRONOUS BACKTRACKING ALGORITHM

SOLVED!



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

الگوریتم عقب‌گرد ناهمگام

نقاط ضعف

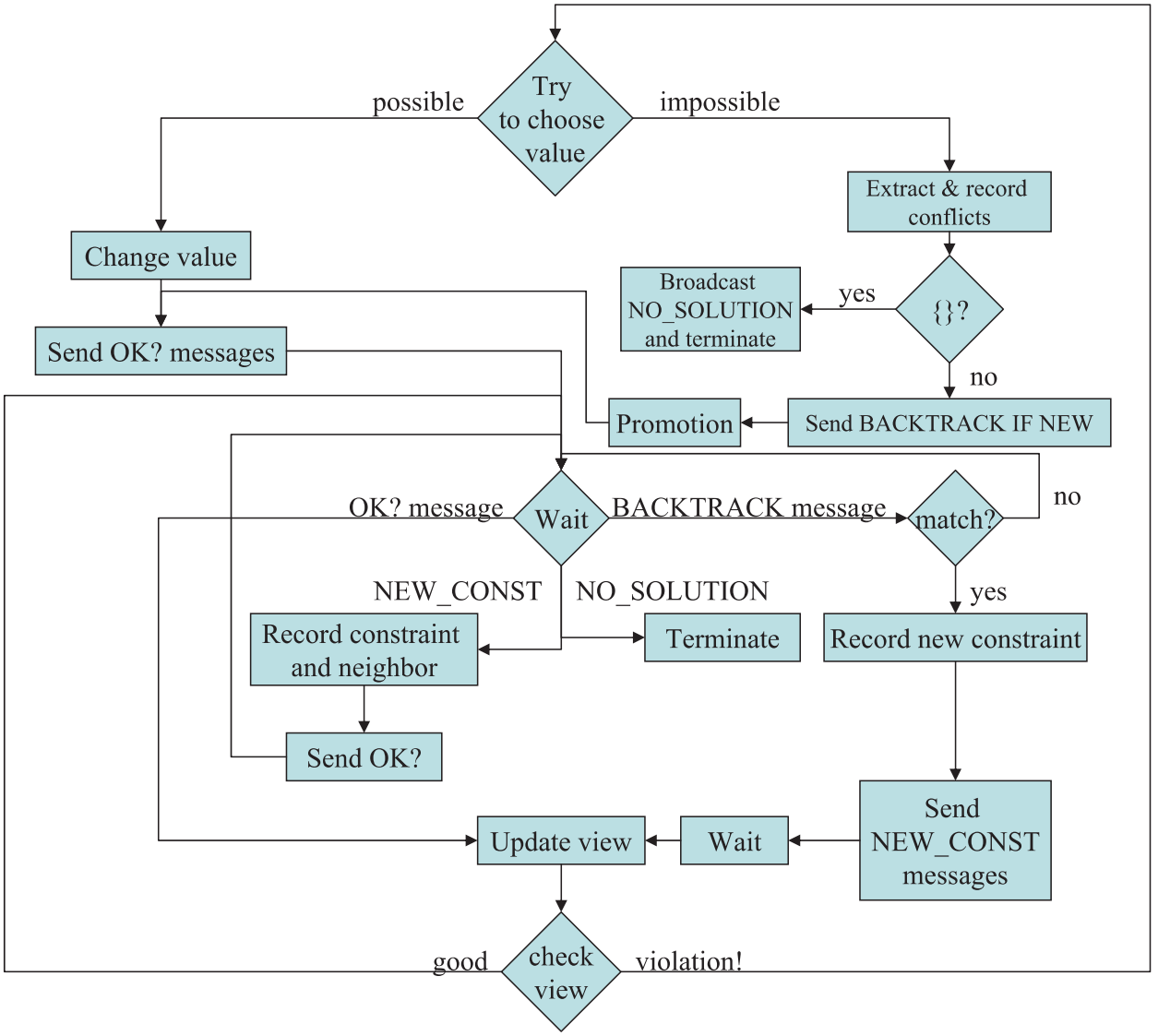
THE ASYNCHRONOUS BACKTRACKING ALGORITHM

الگوریتم جستجوی تعهد ضعیف ناهمگام

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM

الگوریتم جستجوی تعهد ضعیف ناهمگام

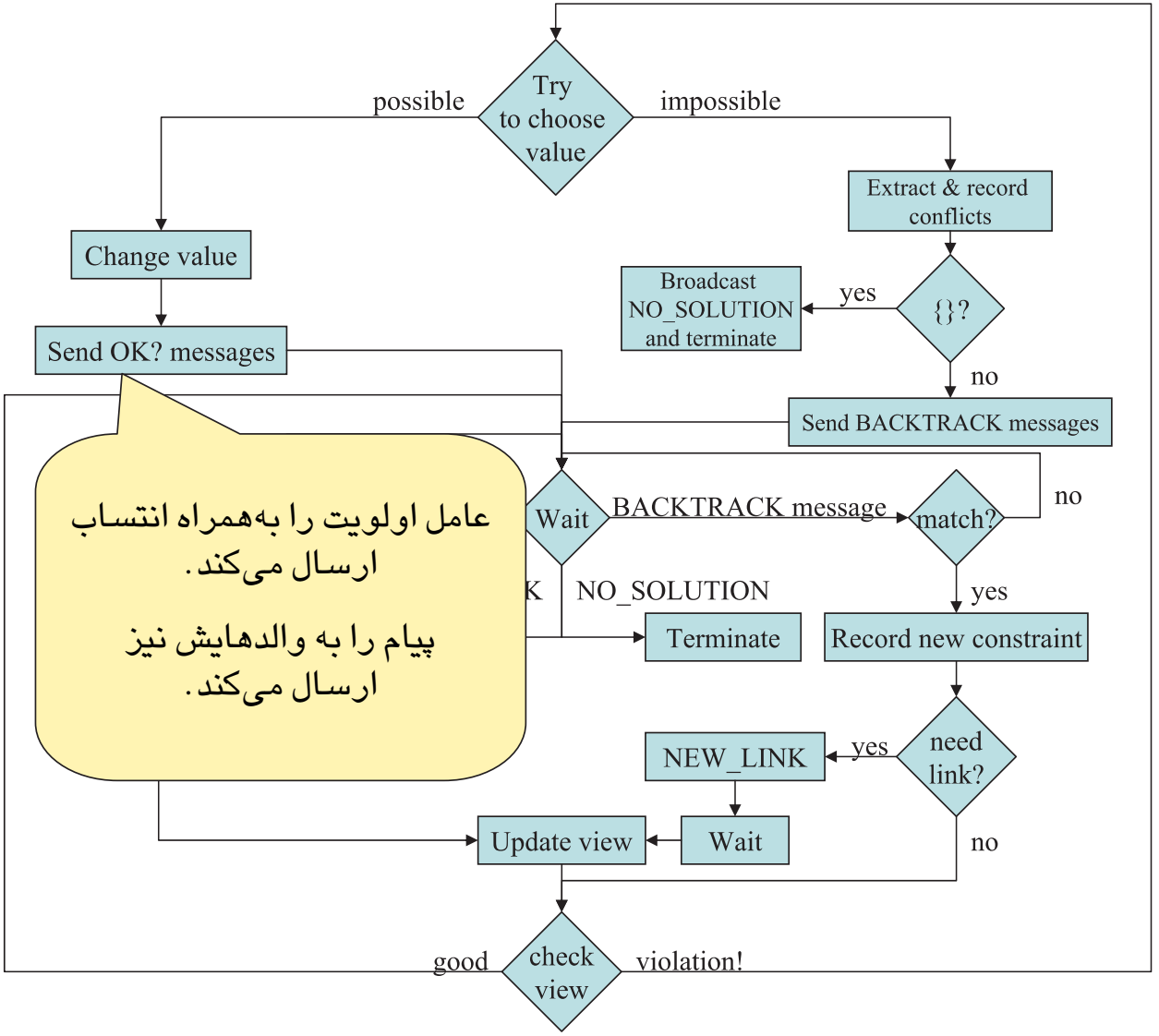
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۱ از ۹

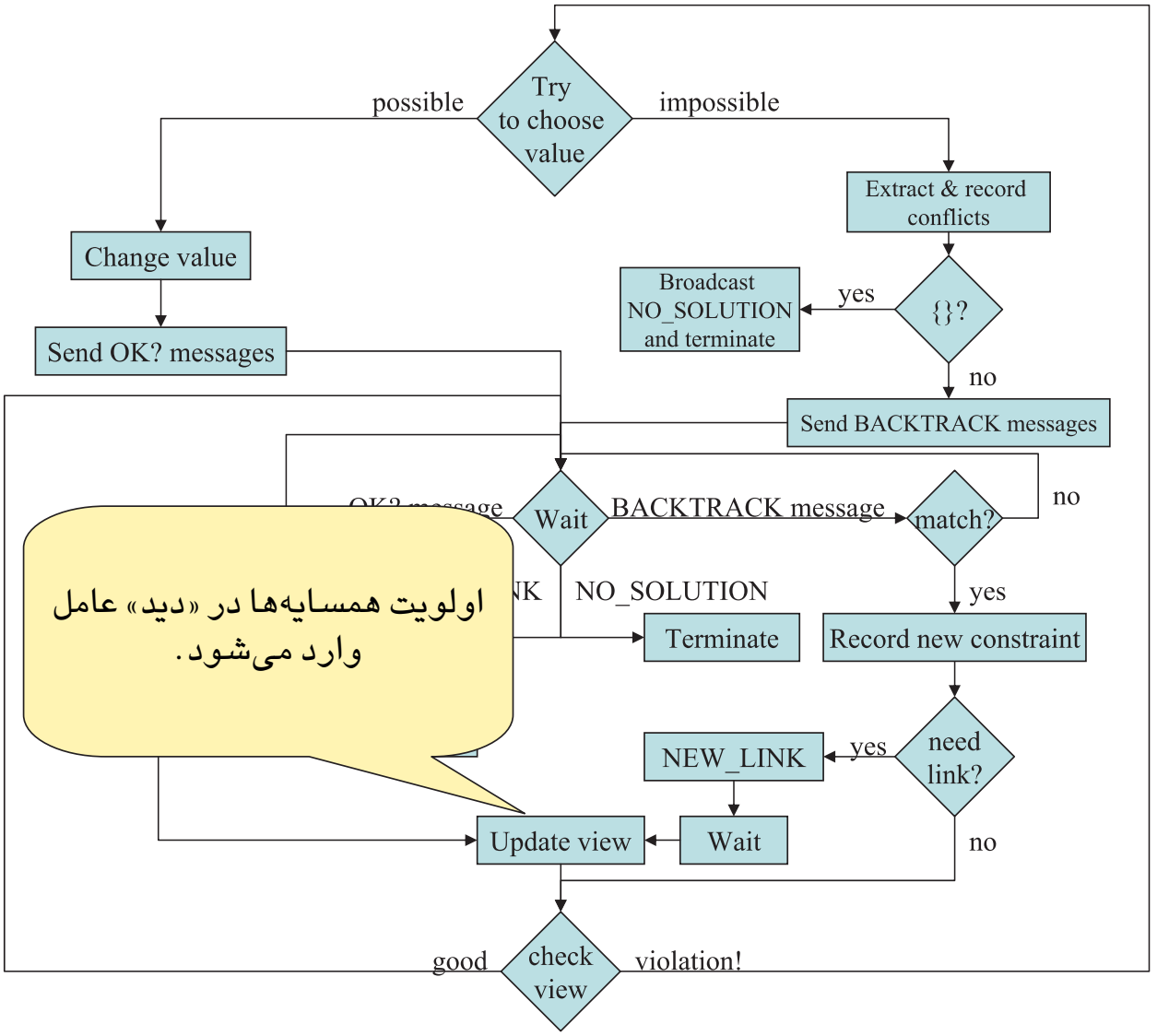
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۲ از ۹

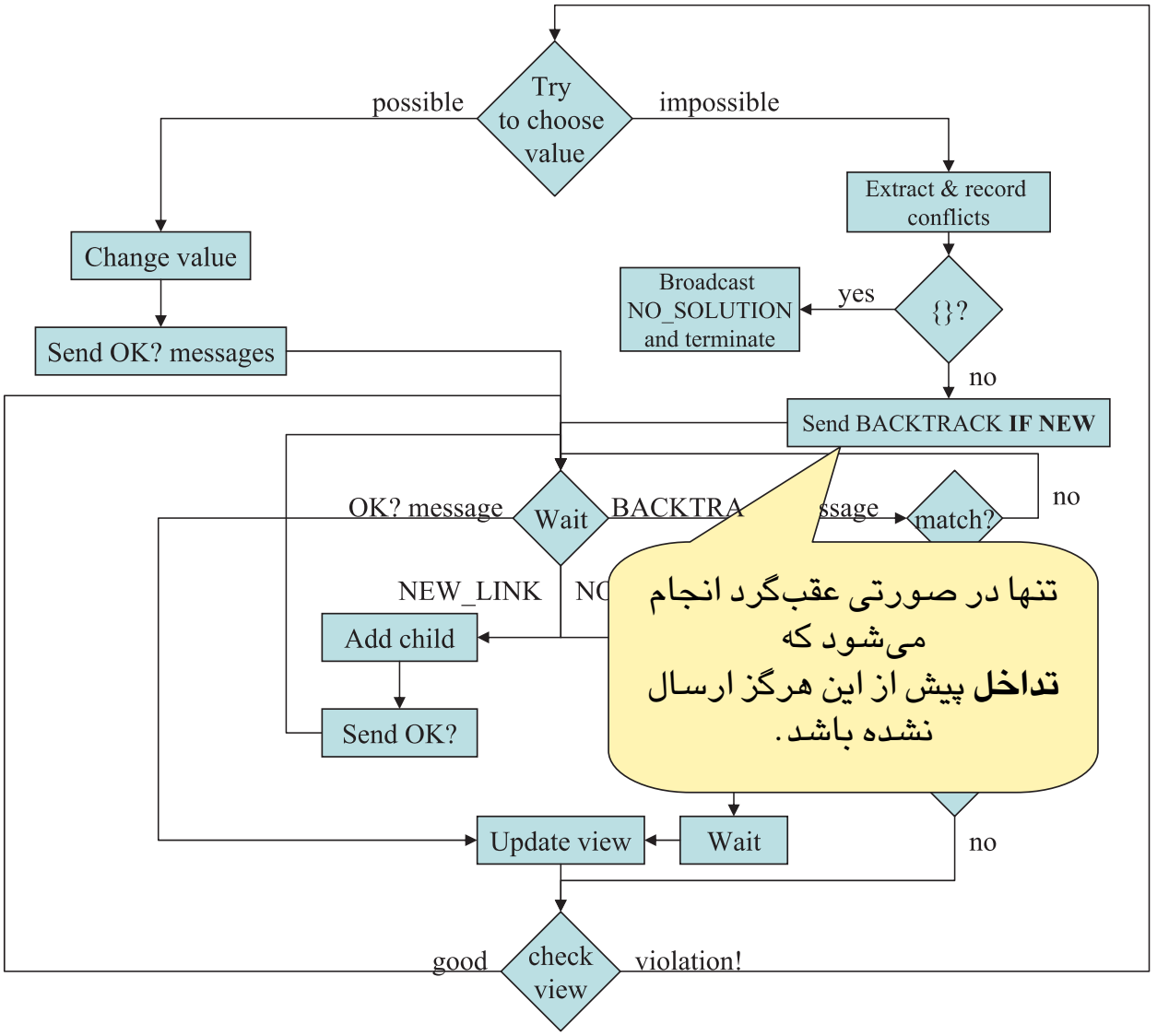
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۳ از ۹

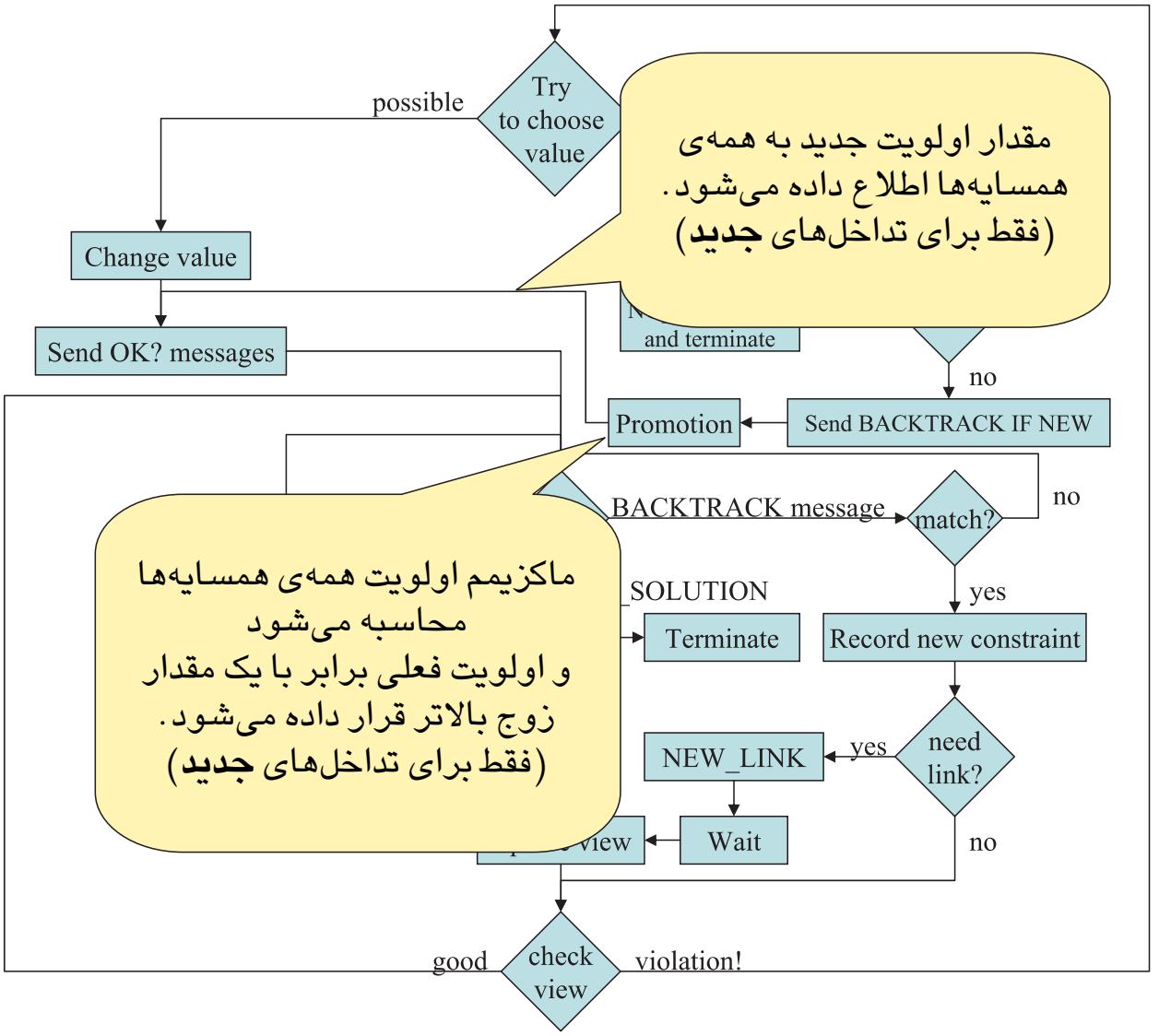
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۴ از ۹

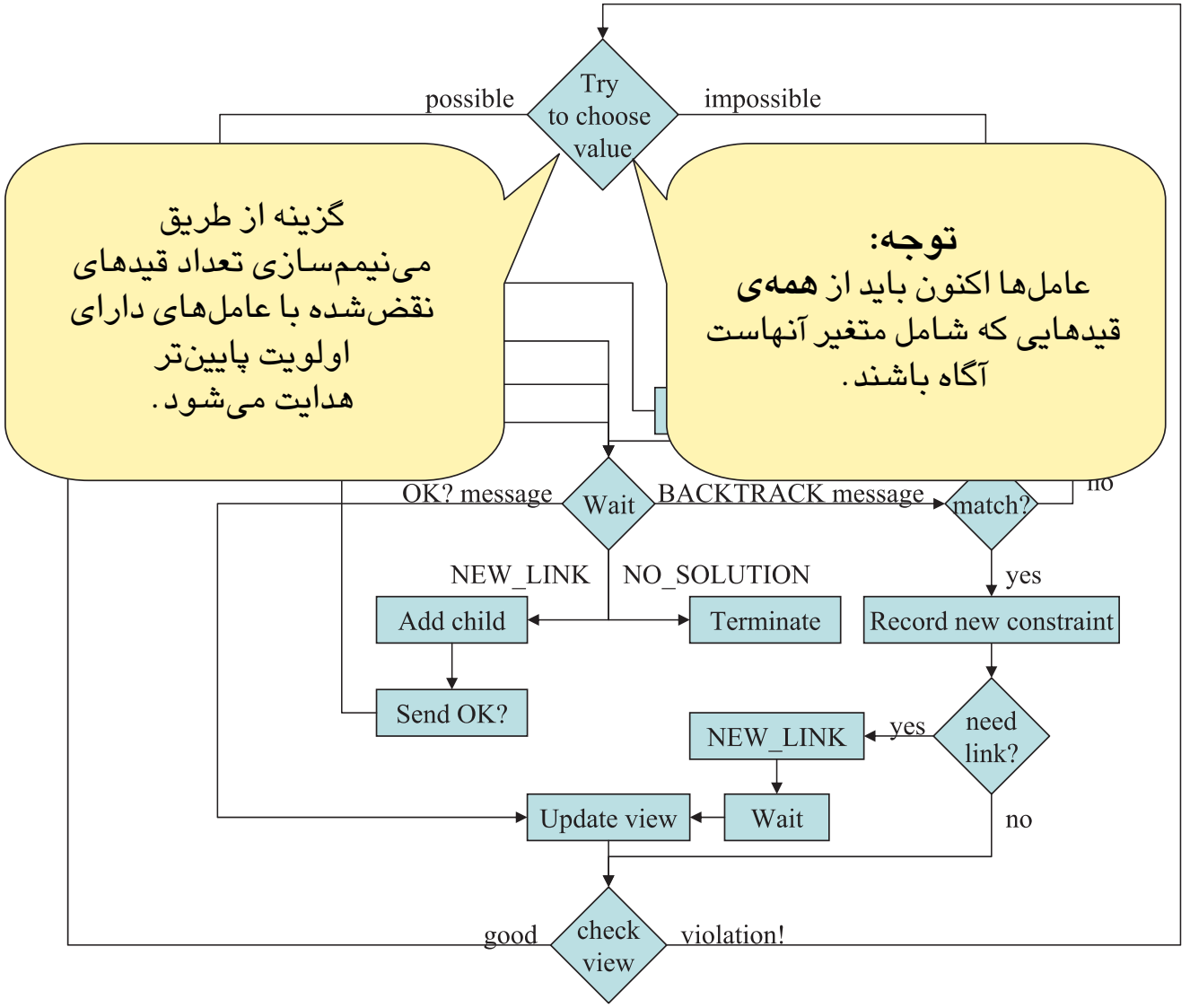
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۵ از ۹

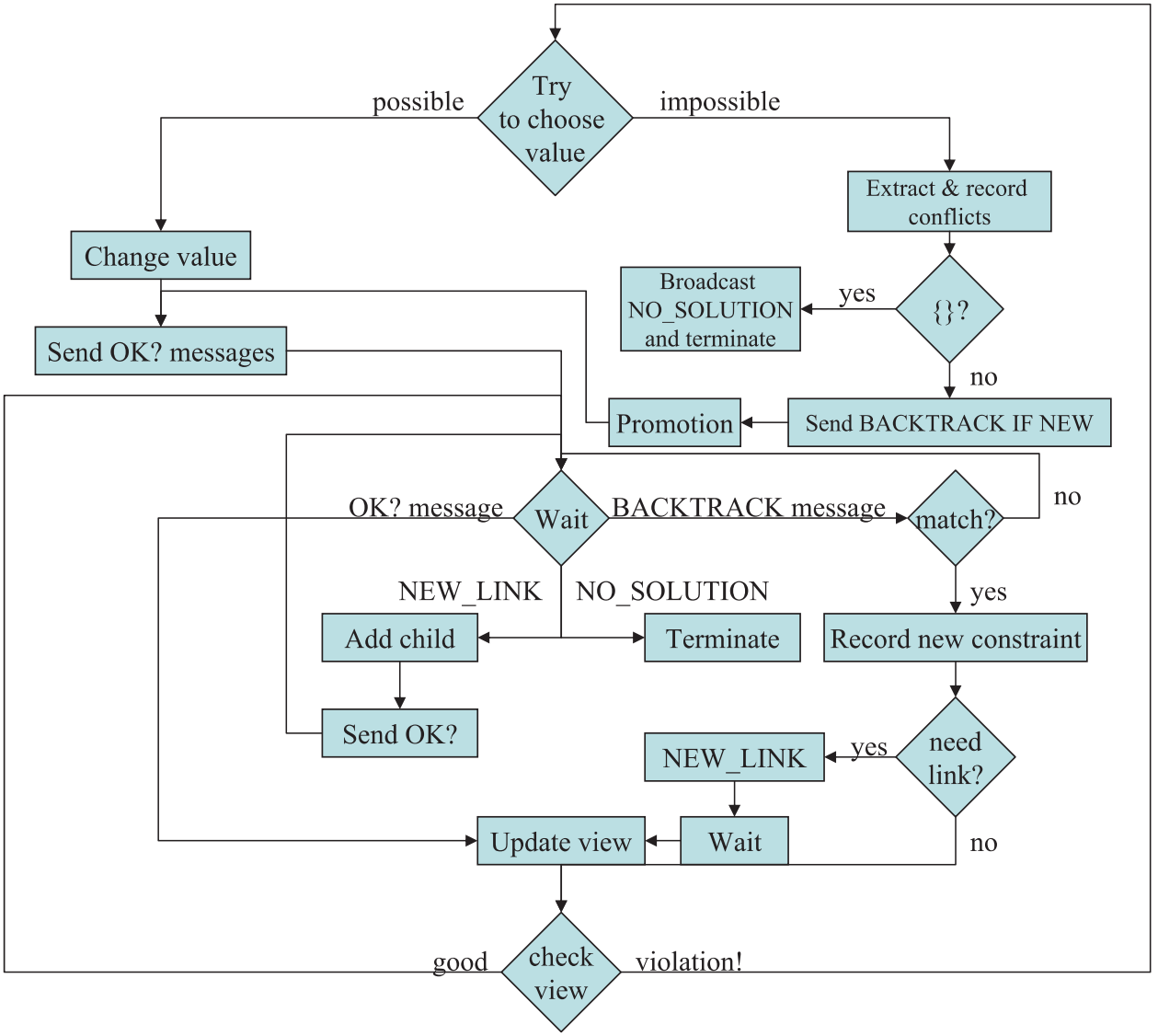
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۶ از ۹

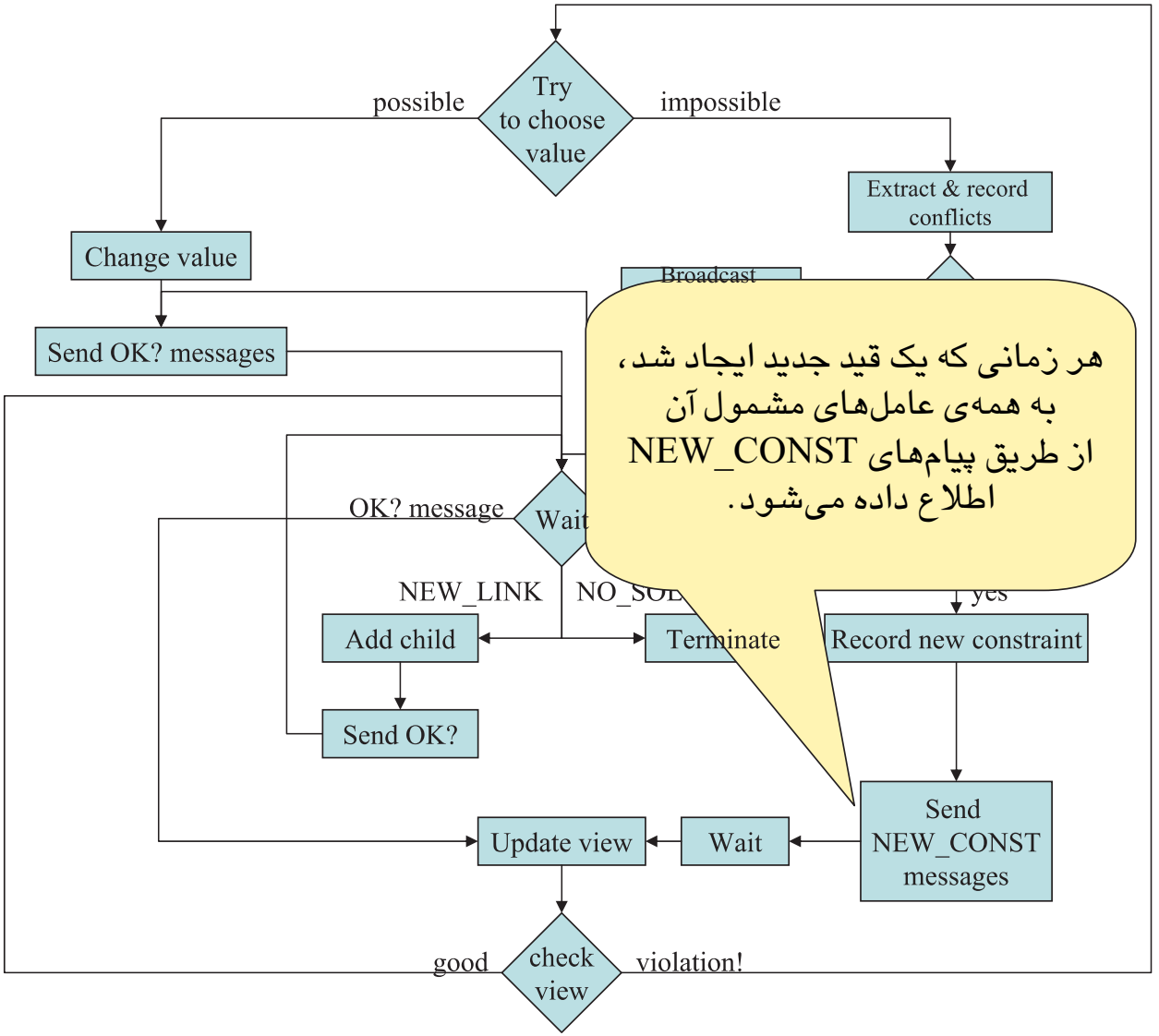
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۷ از ۹

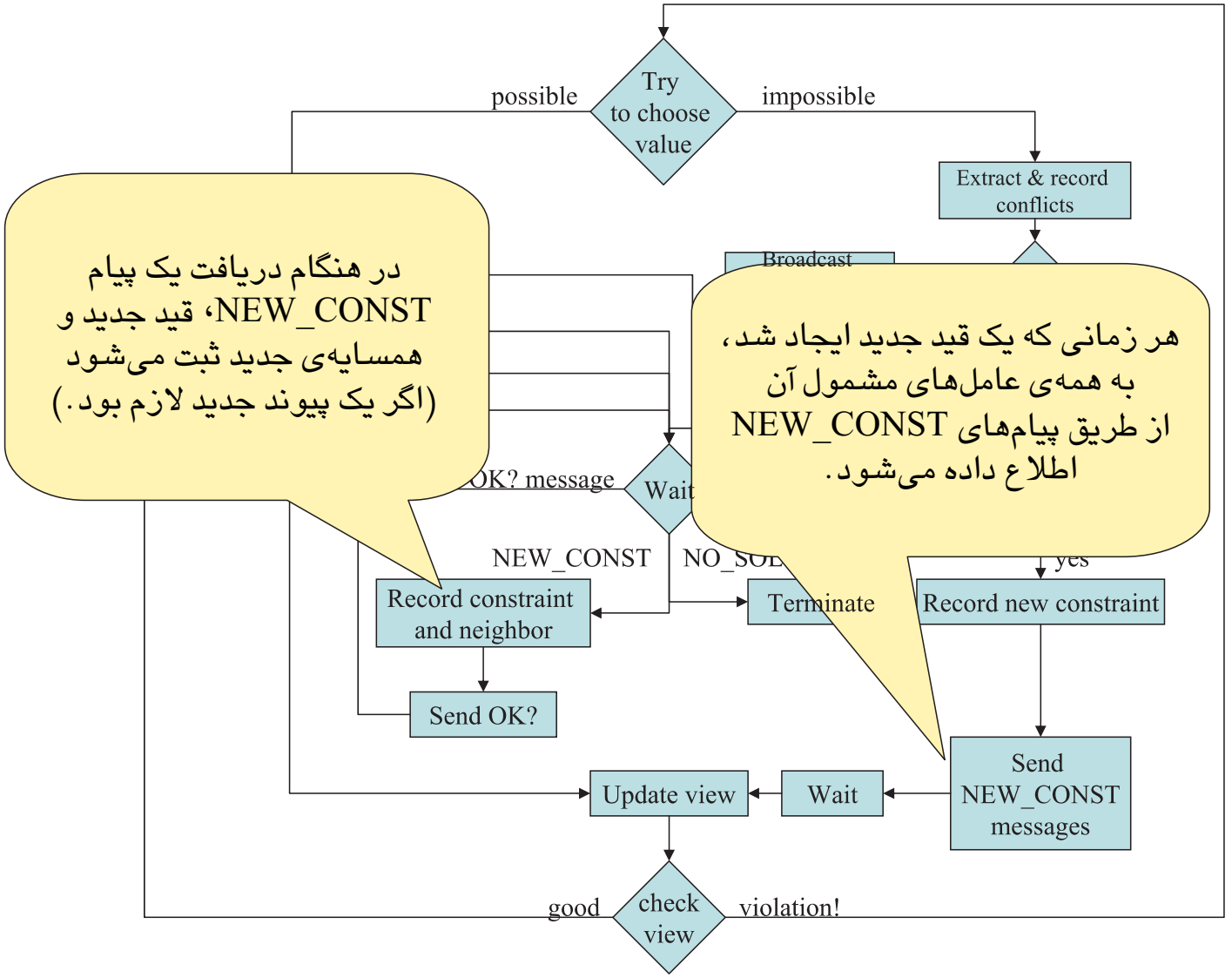
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۸ از ۹

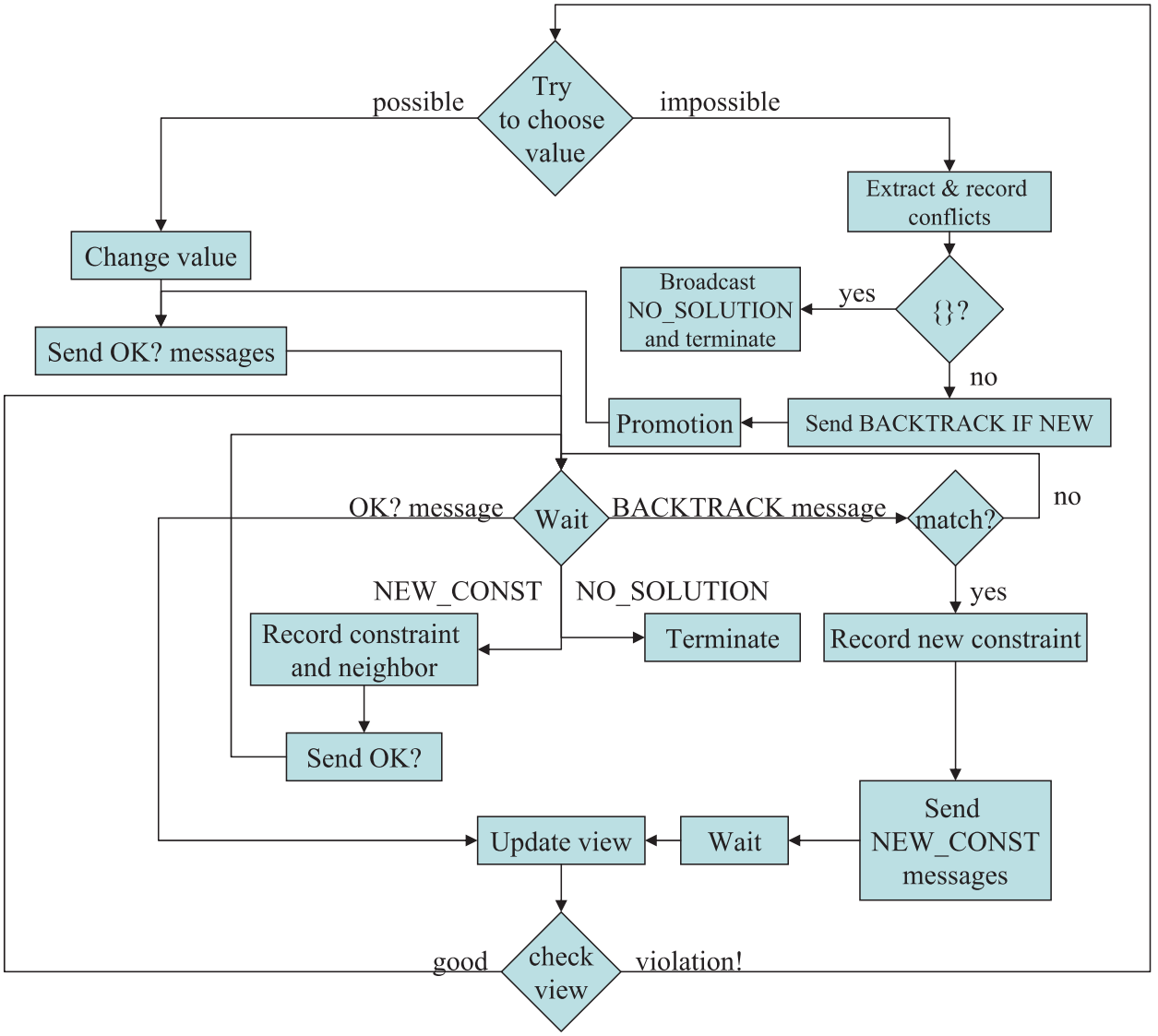
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

۹ از ۹

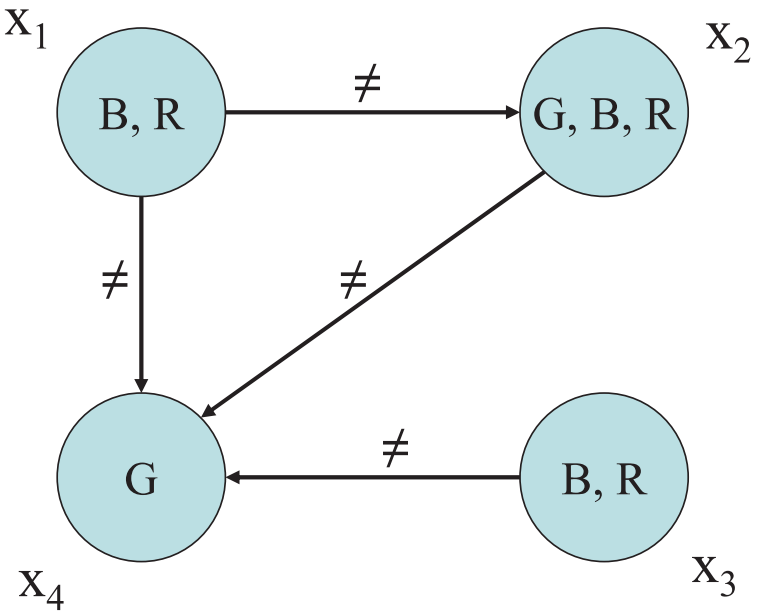
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱ از ۲۹)

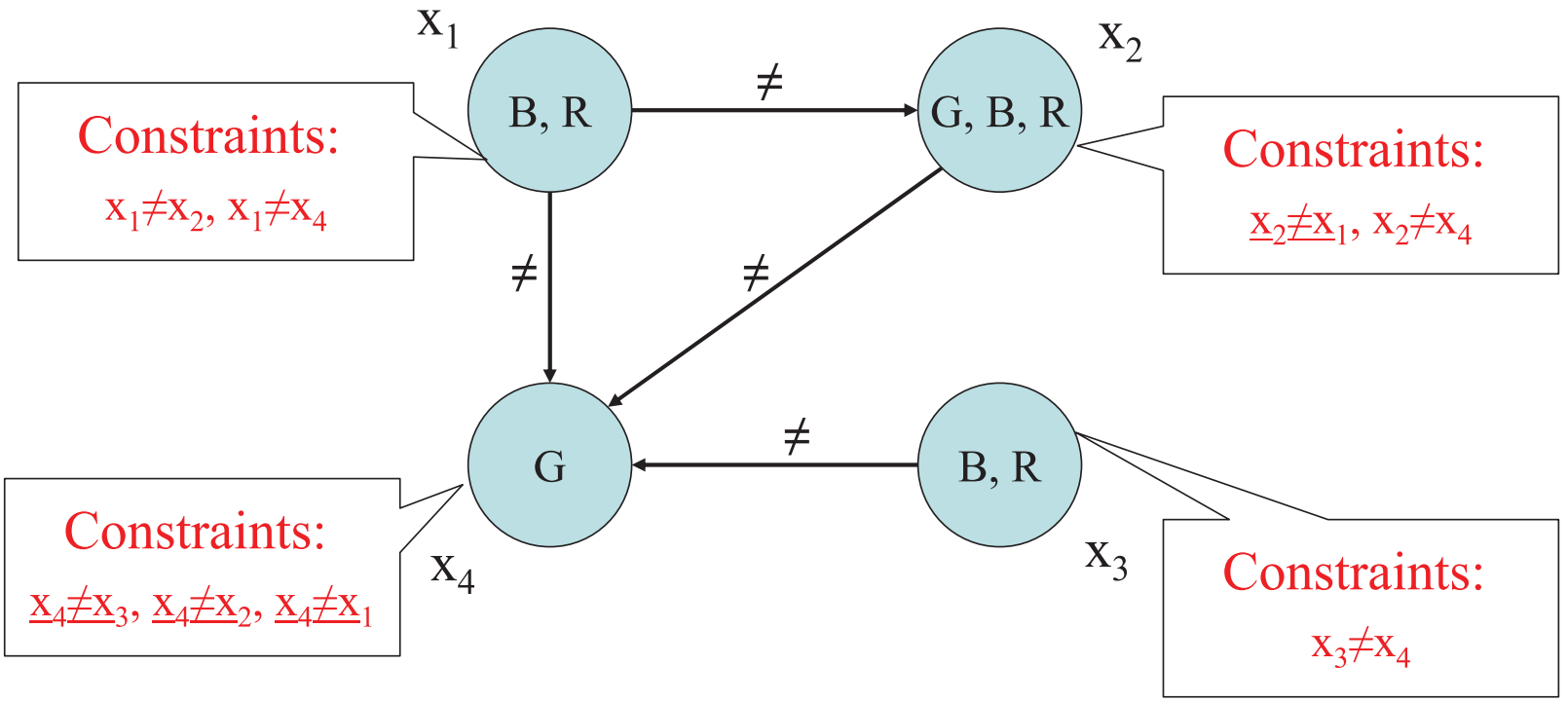
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲ از ۲۹)

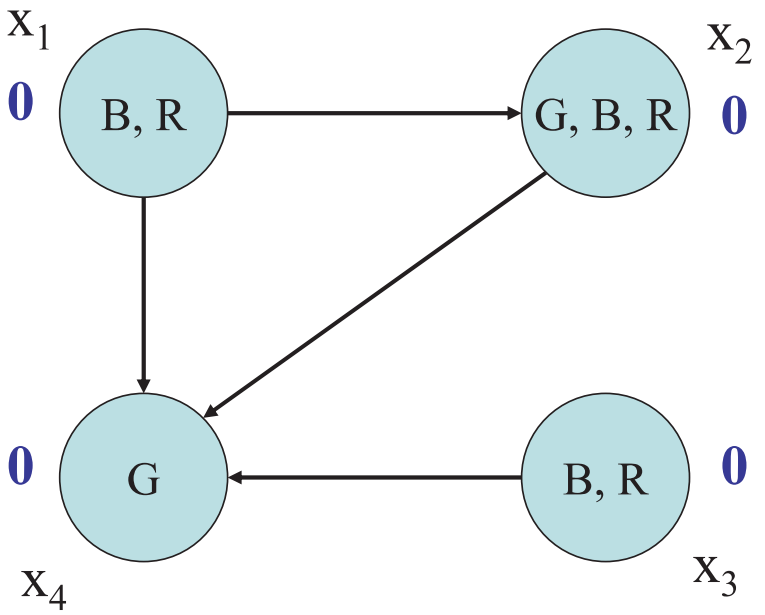
THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۳ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM

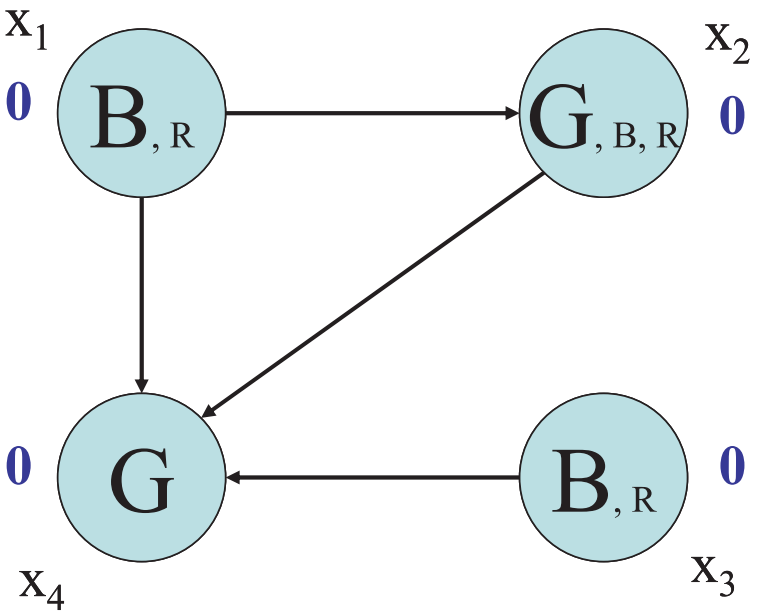


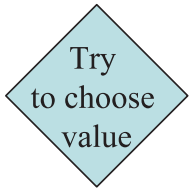
Initial priority values are all set to 0. Two agents with identical priorities are ordered with respect to their index

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۴ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



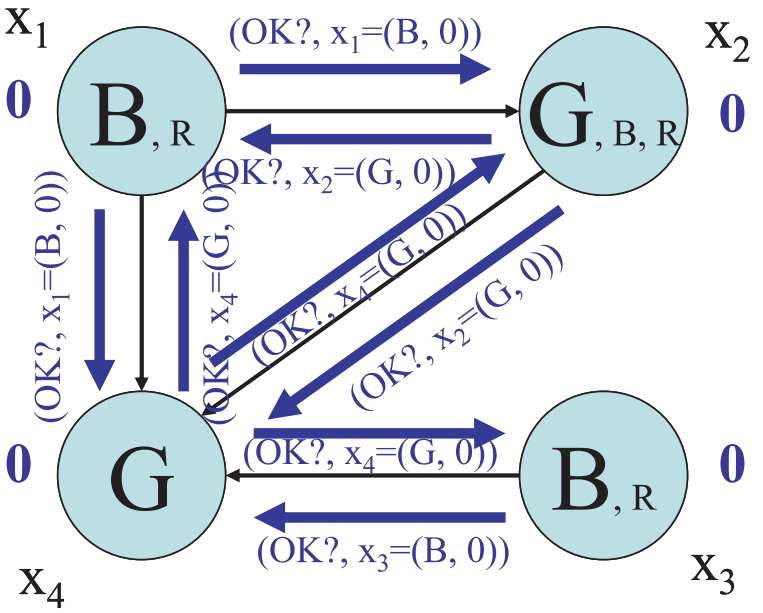


Each agent chooses an assignment to its variable (at the first time step, we cannot use the heuristic because agents still have empty views)

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۵ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



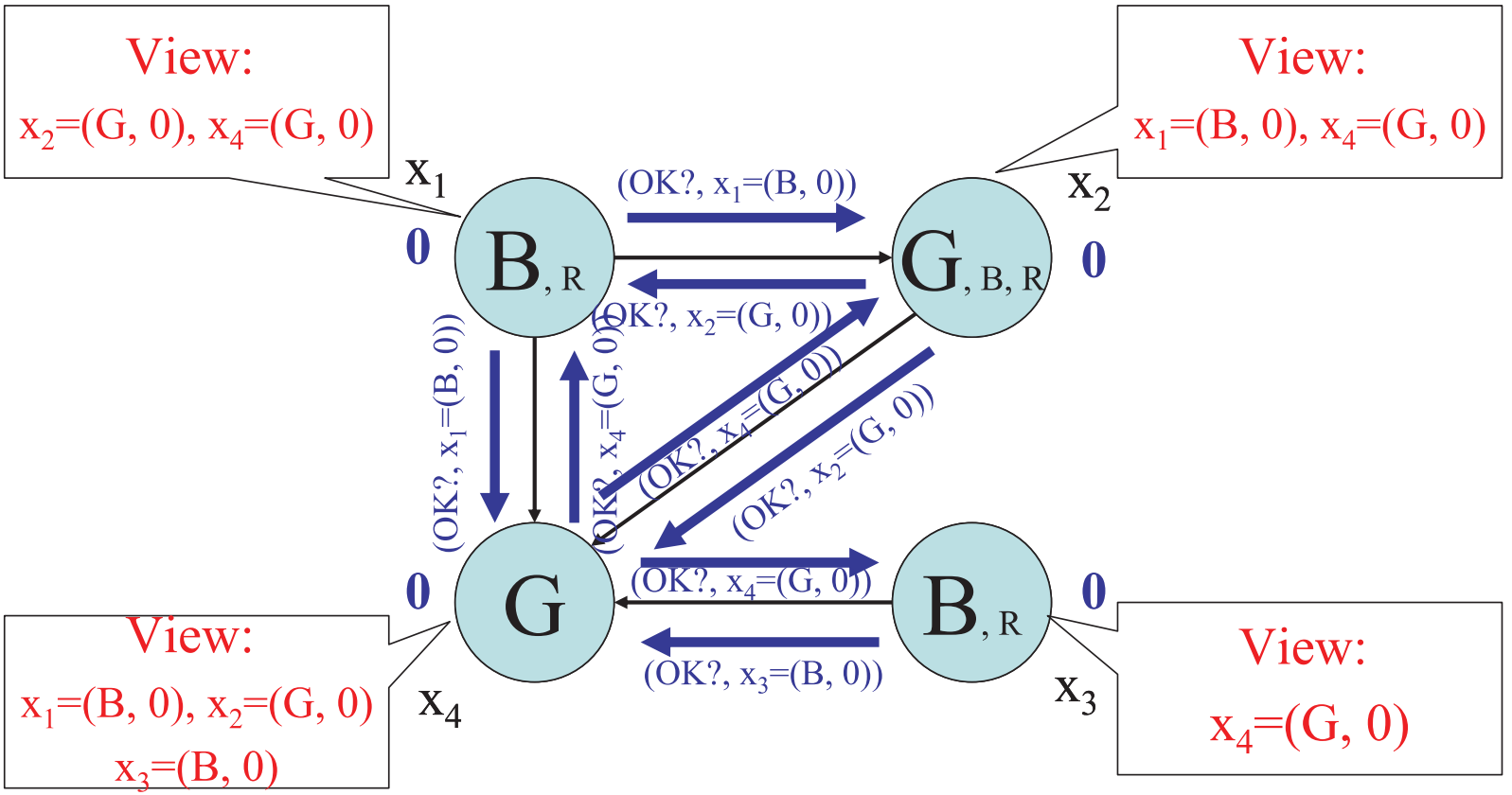
Send OK? messages

Each agent sends OK? messages to ALL of its neighbors

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۶ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



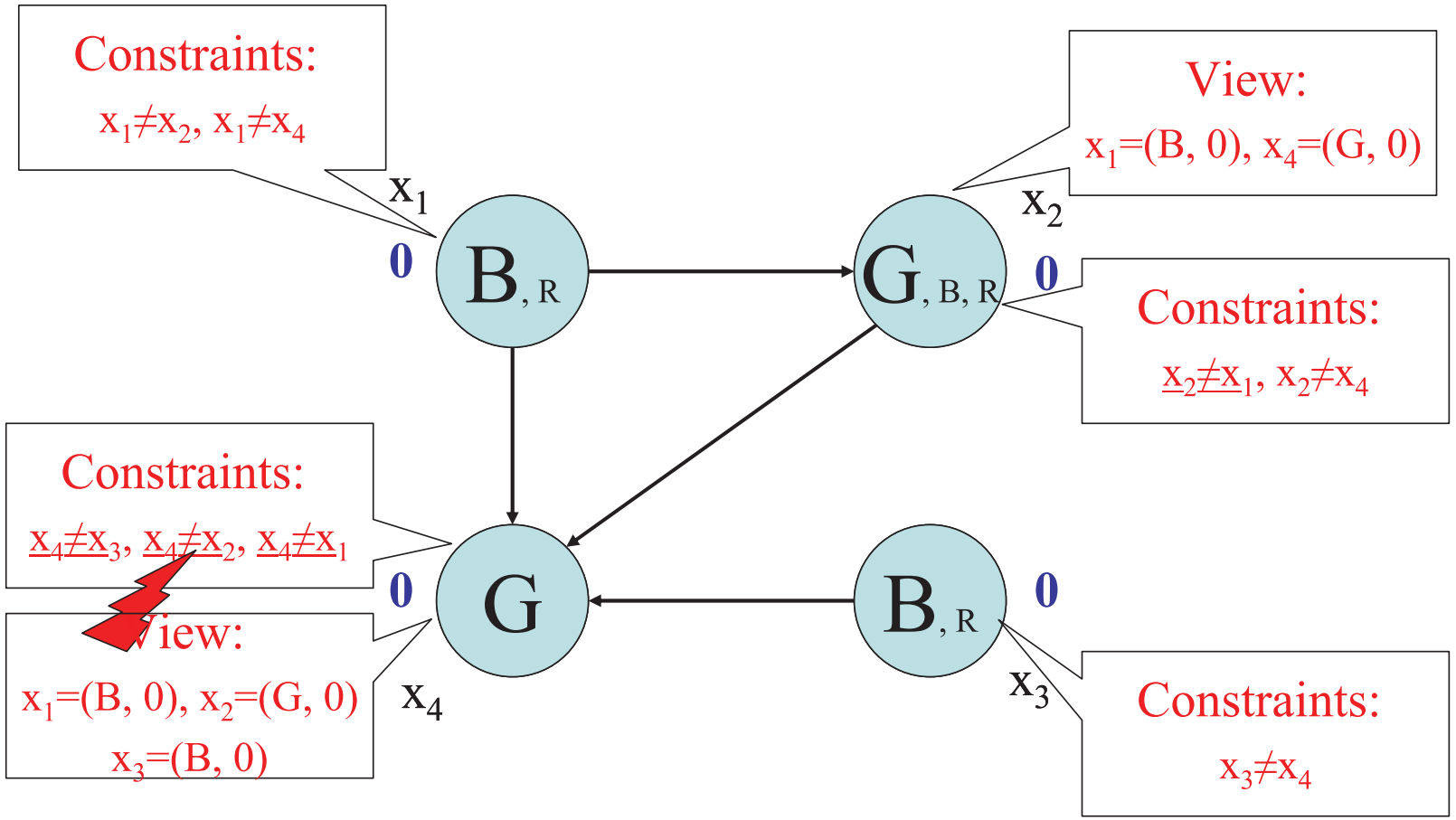
Update view

All agents update their view

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۷ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



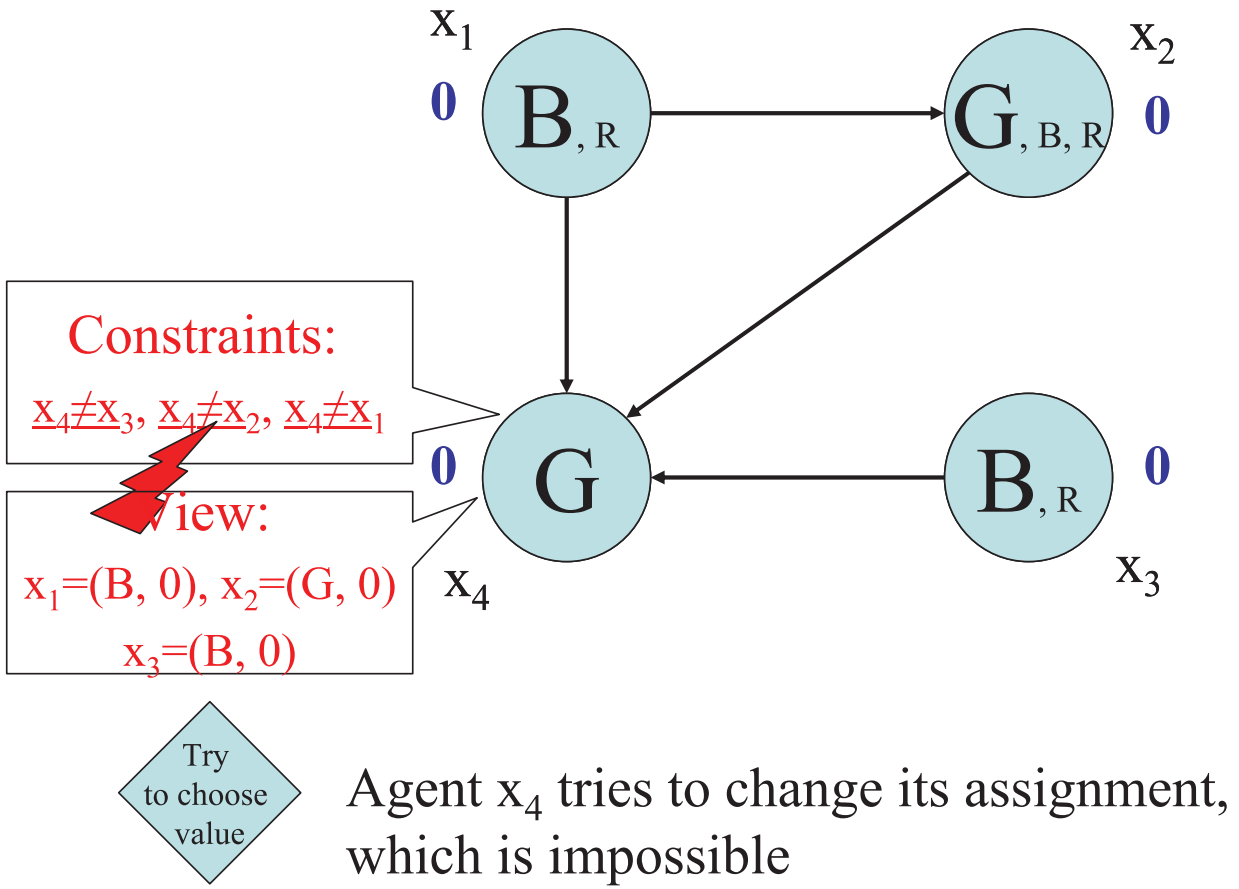
check view

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۸ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM

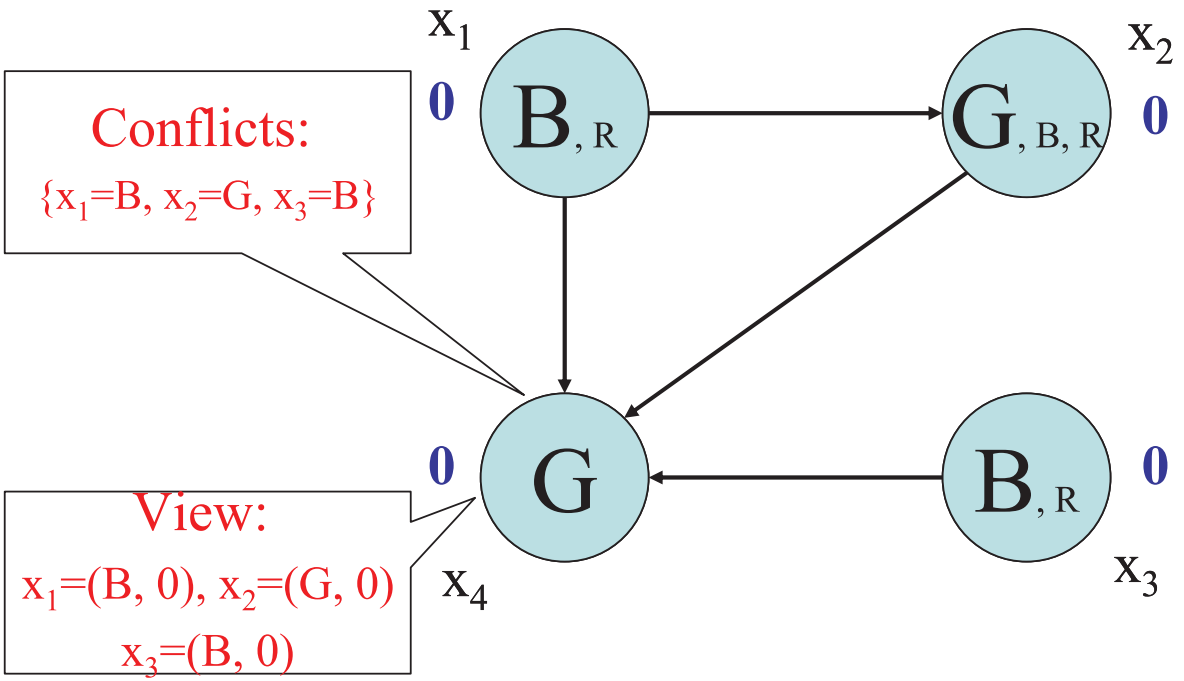


Agent x_4 tries to change its assignment, which is impossible

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۹ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



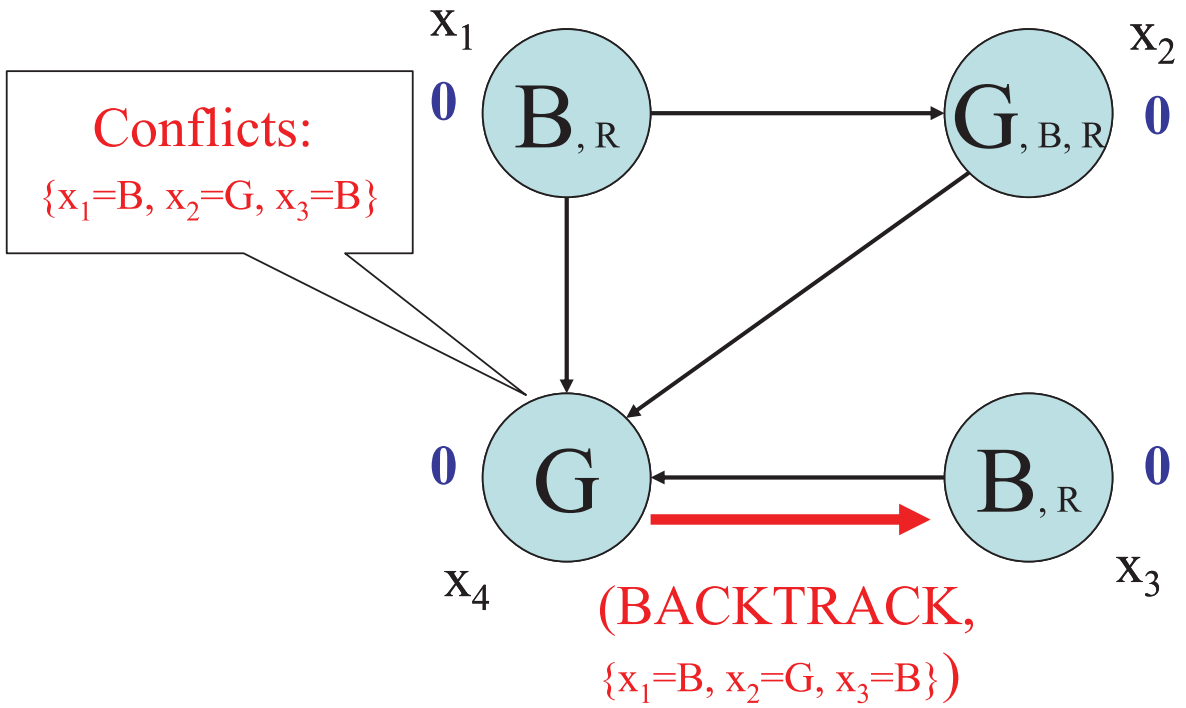
Extract & record conflicts

Agent x_4 extracts and records the conflicts

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۰ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



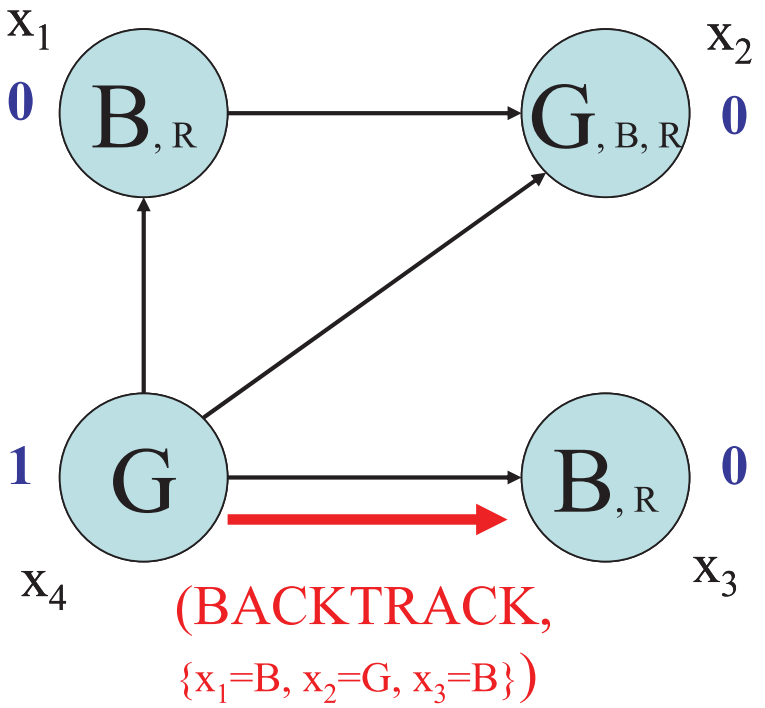
Send BACKTRACK messages

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۱ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



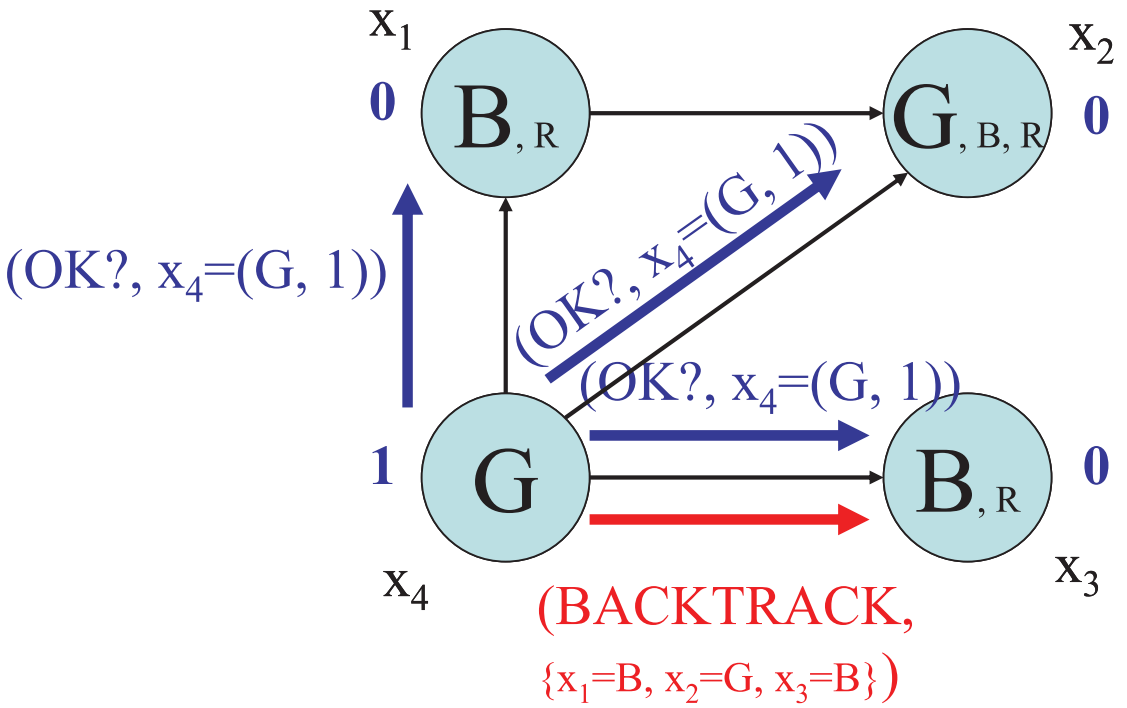
Promotion

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۲ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



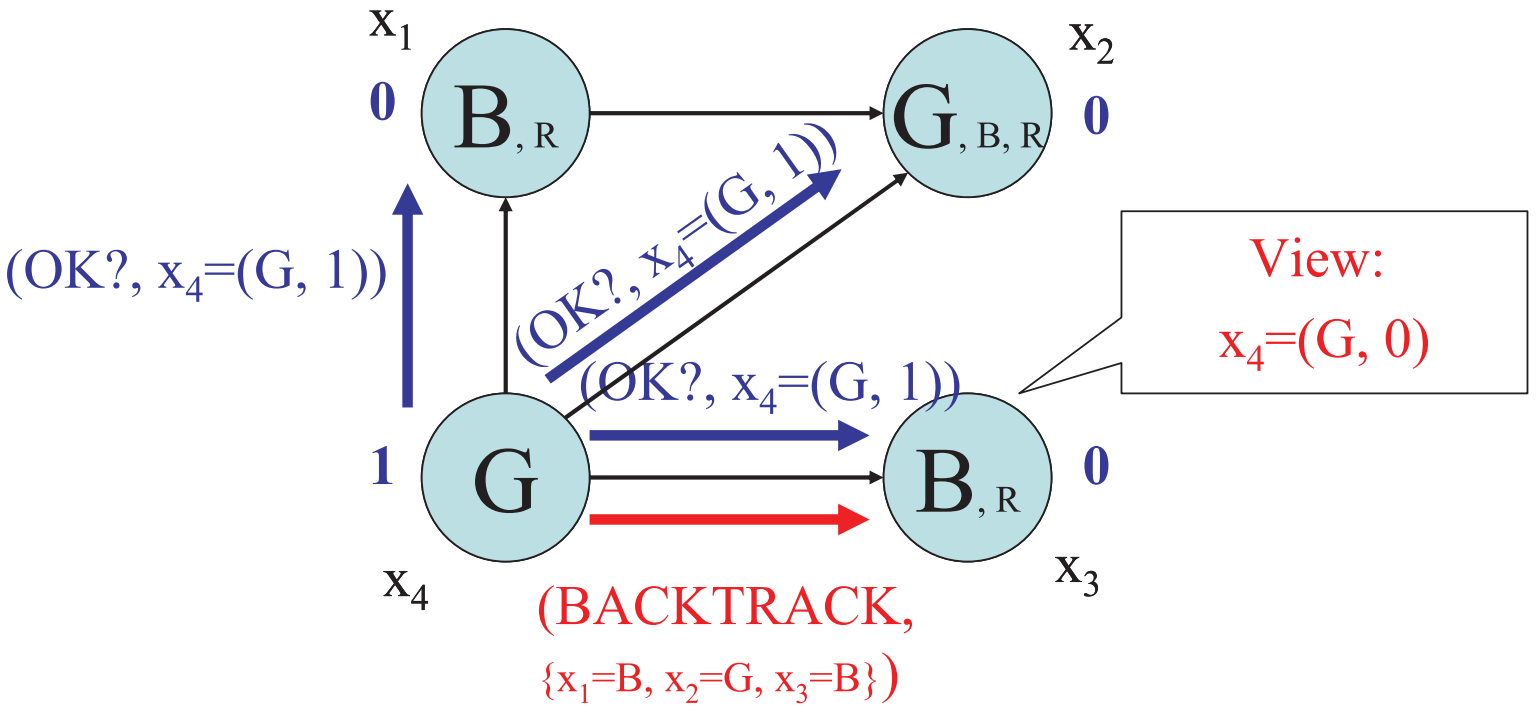
Send OK? messages

Agent x_4 communicates its new priority value to ALL its neighbors

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۳ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



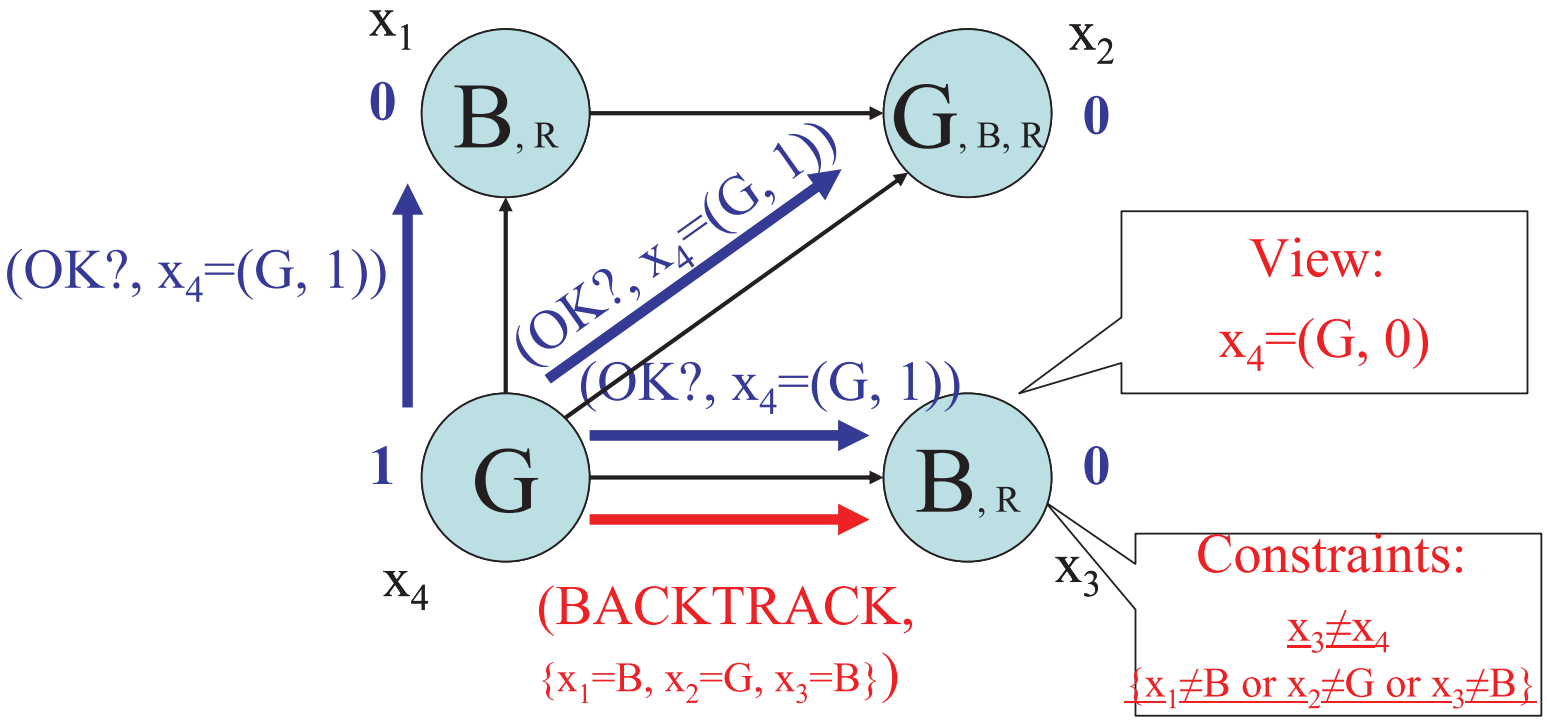
match?

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۴ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



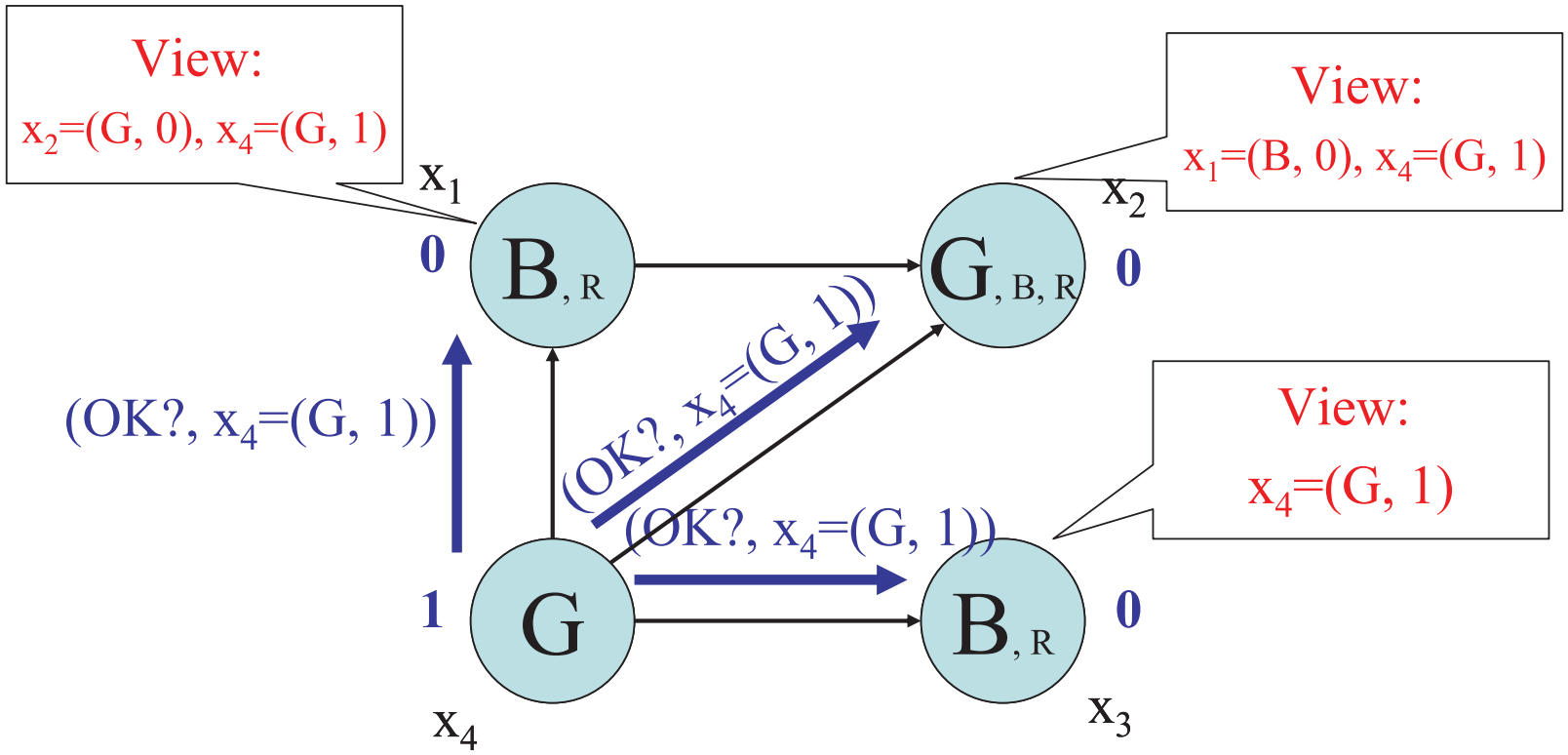
Record new constraint

Agent x_3 records the conflict as a new constraint it will be responsible for

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۵ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



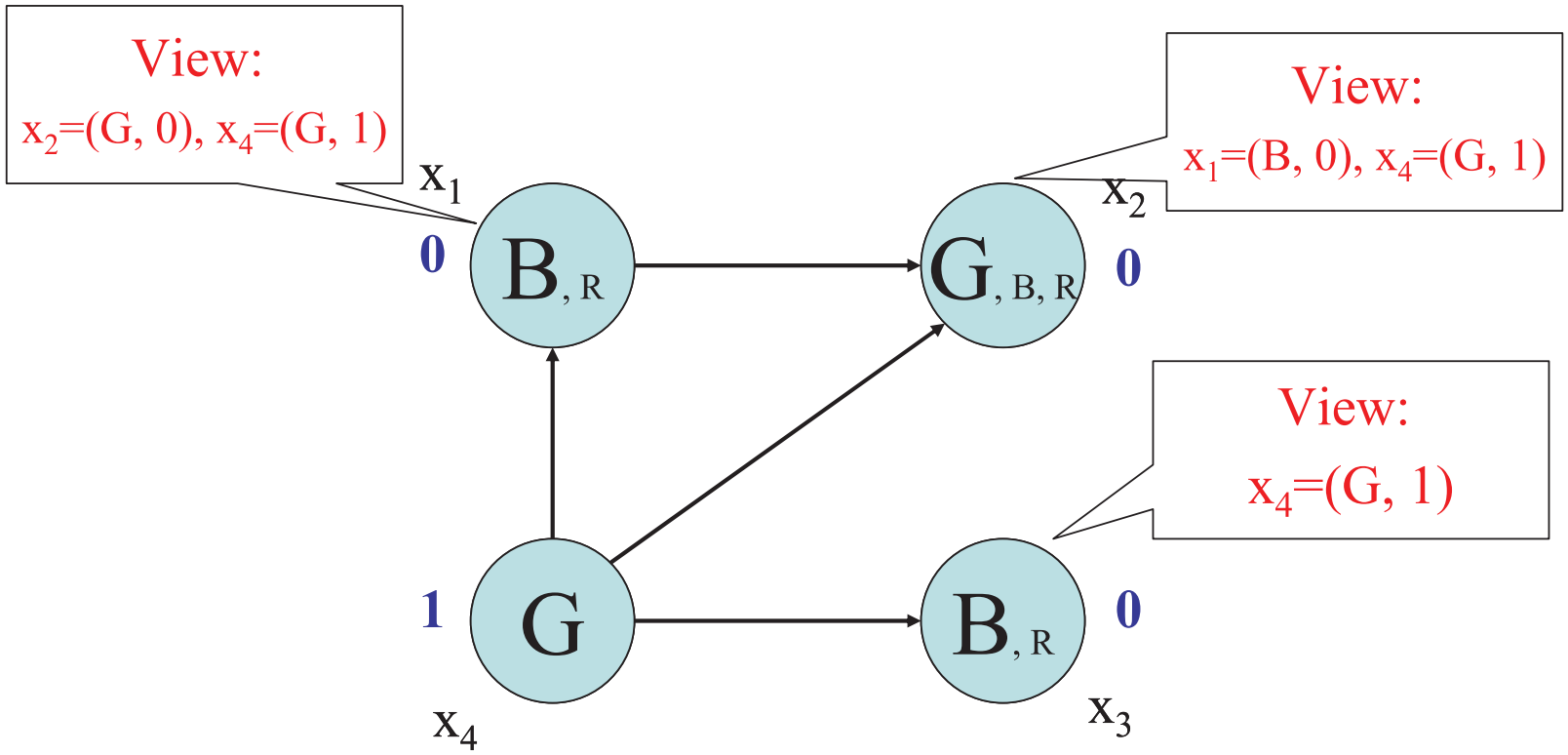
Update view

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۶ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



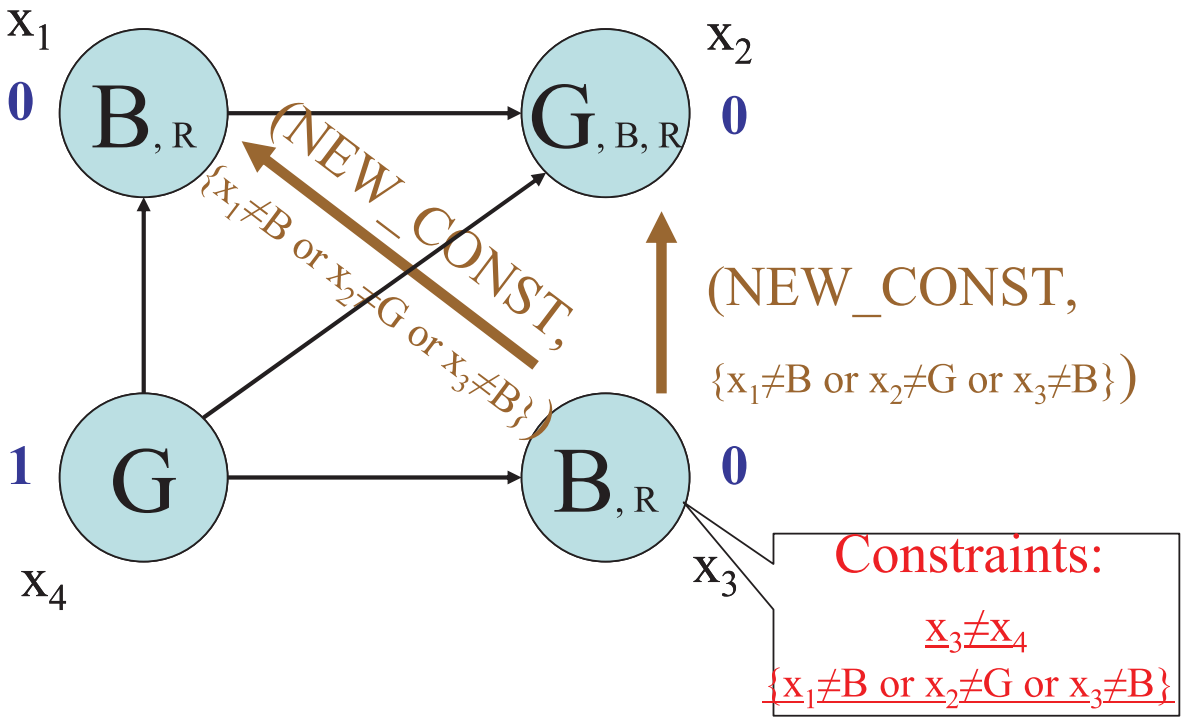
check view

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۷ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



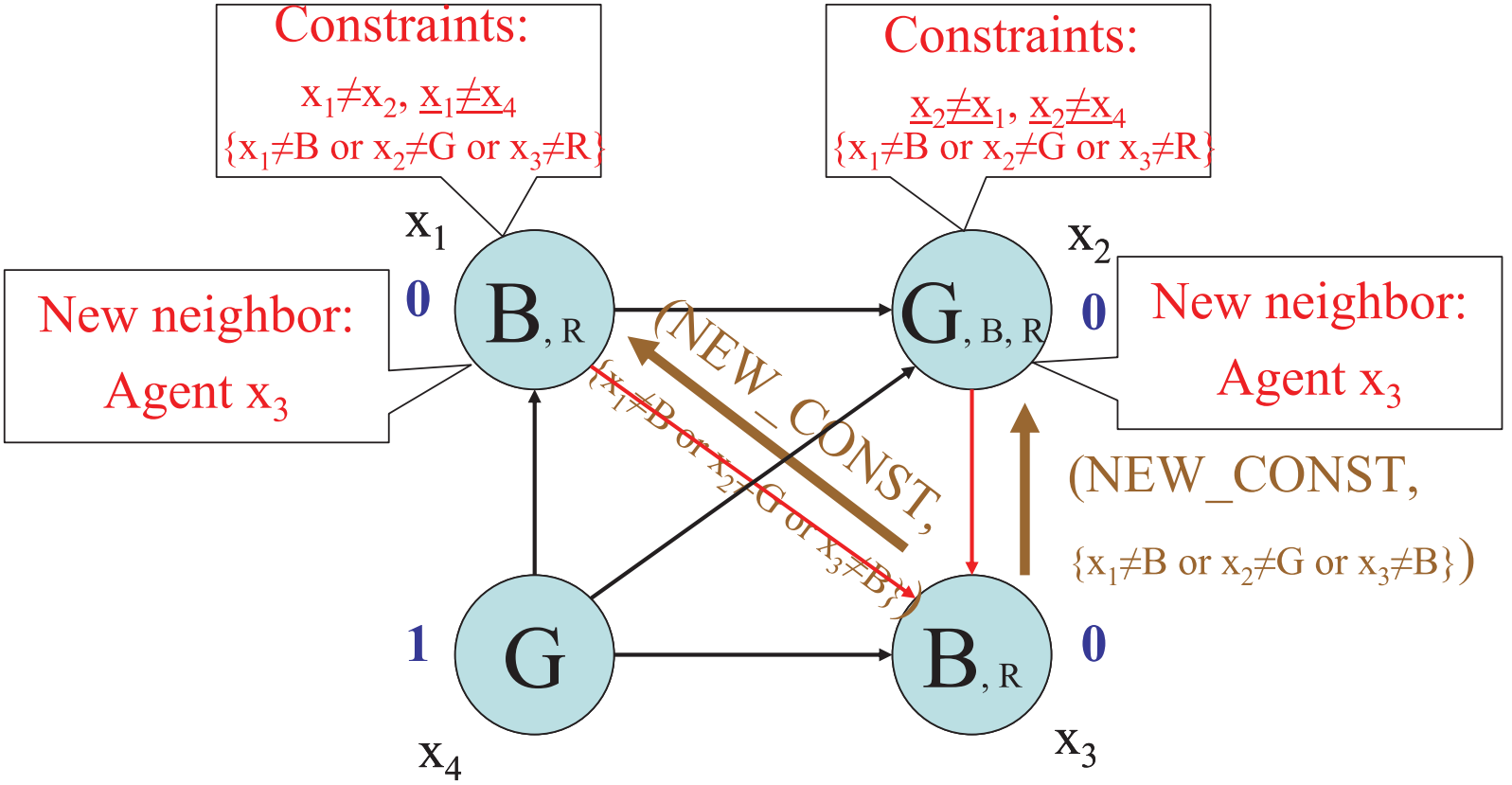
Send NEW_CONST messages

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۸ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



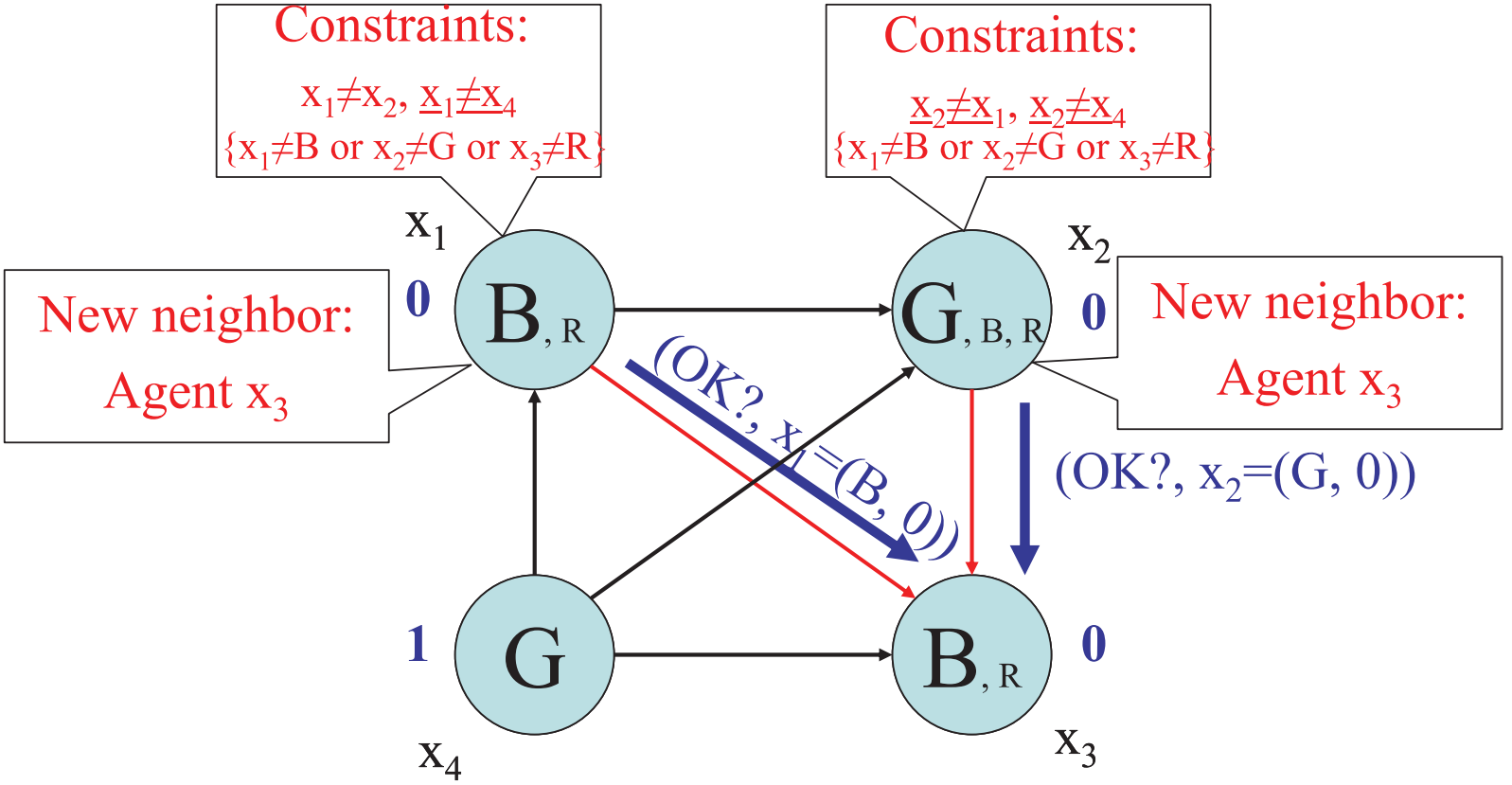
Record constraint and neighbor

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۱۹ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



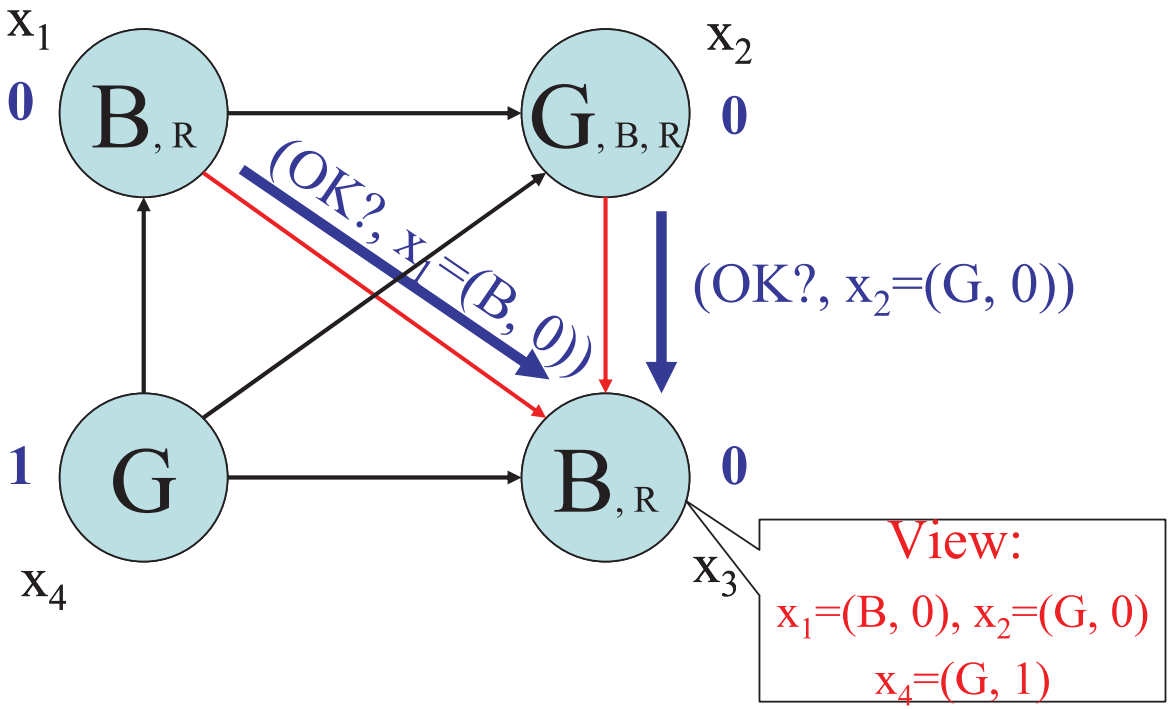
Send OK? messages

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۰ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



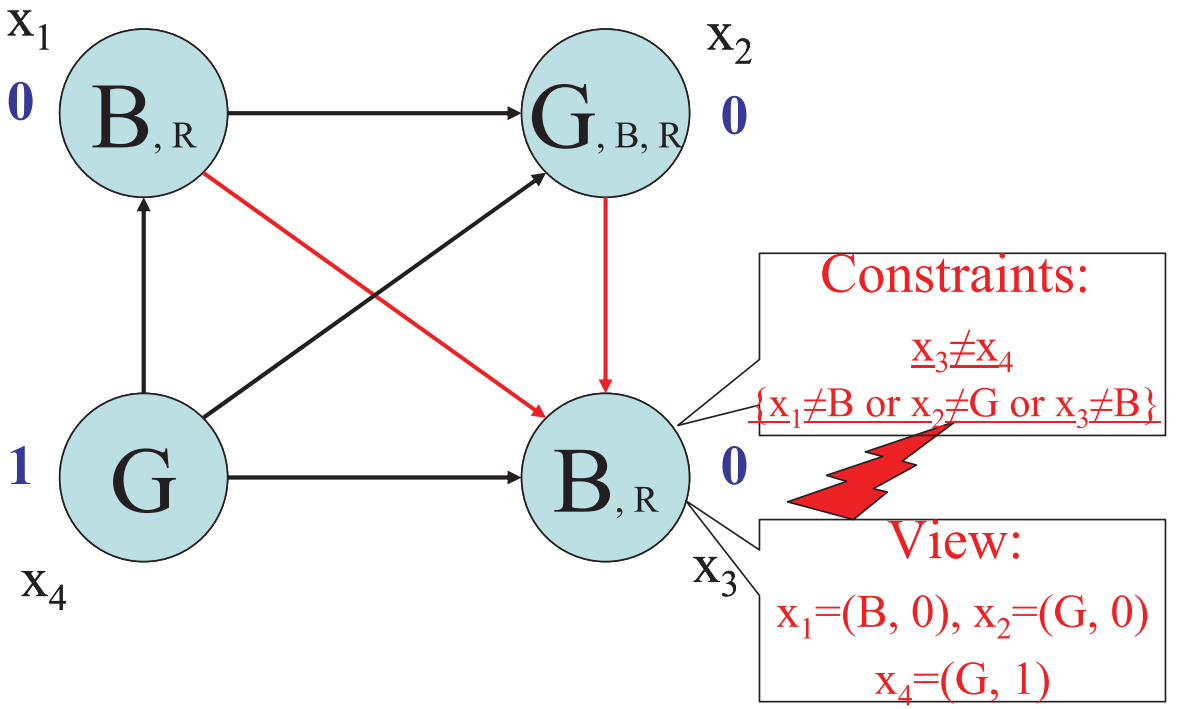
Update view

Agent x_3 receives messages and updates its view

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۱ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



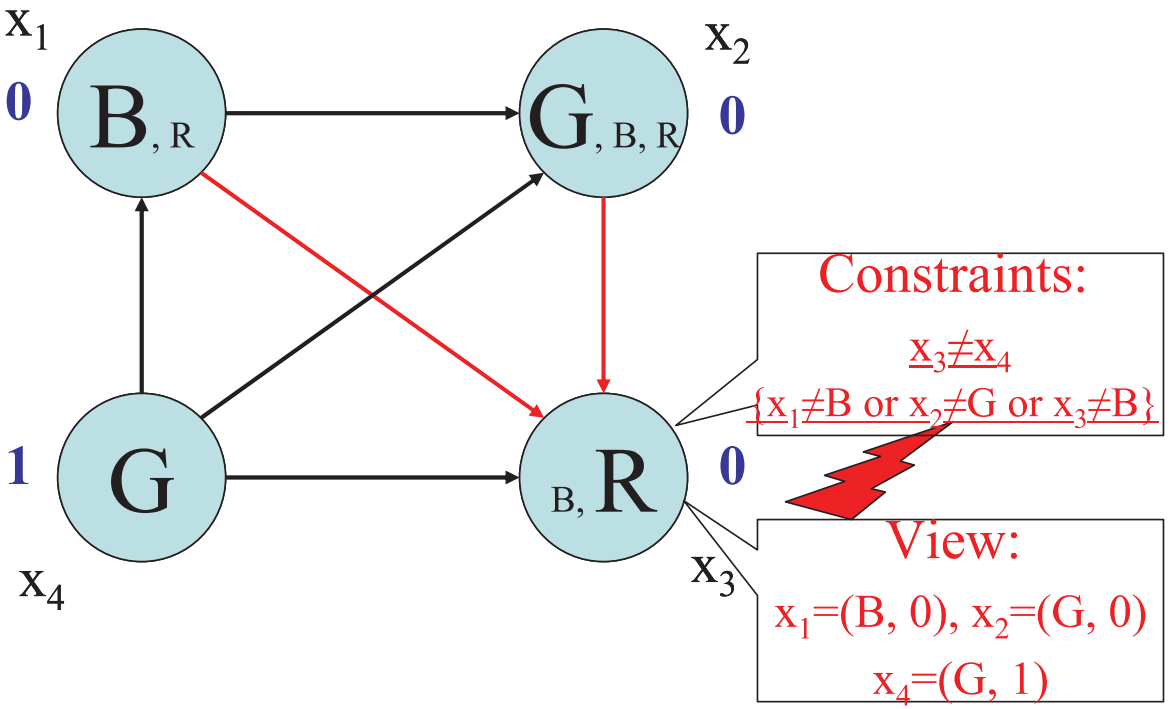
check view

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۲ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



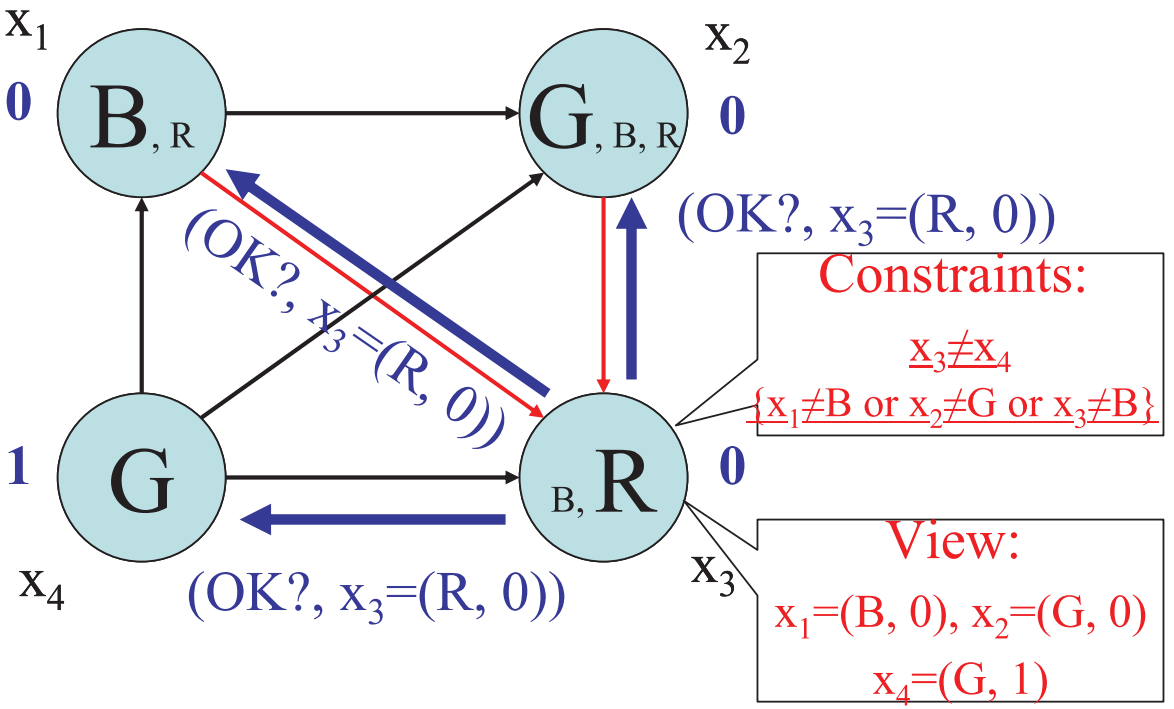
Try to choose value

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۳ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



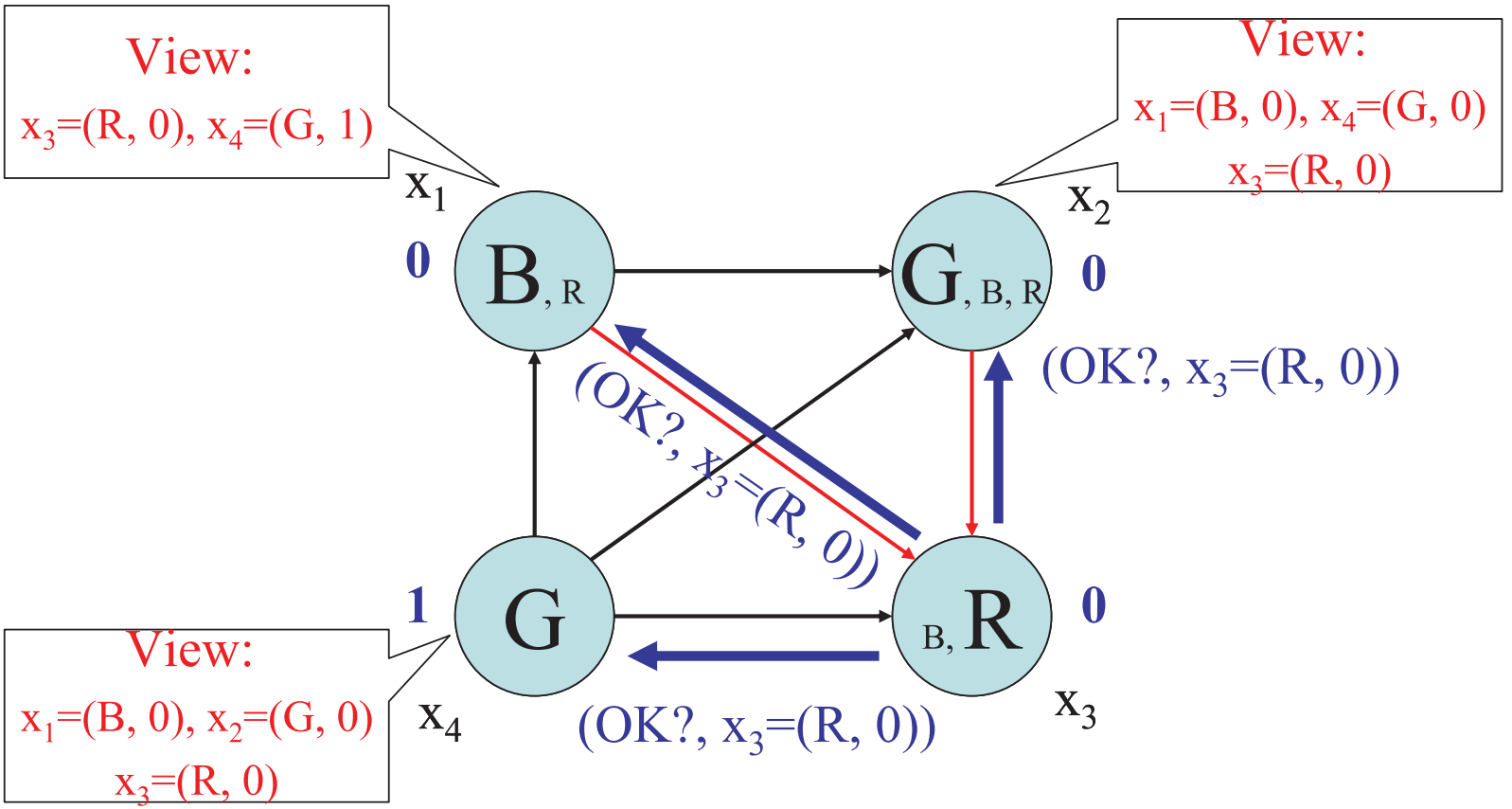
Send OK? messages

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۴ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



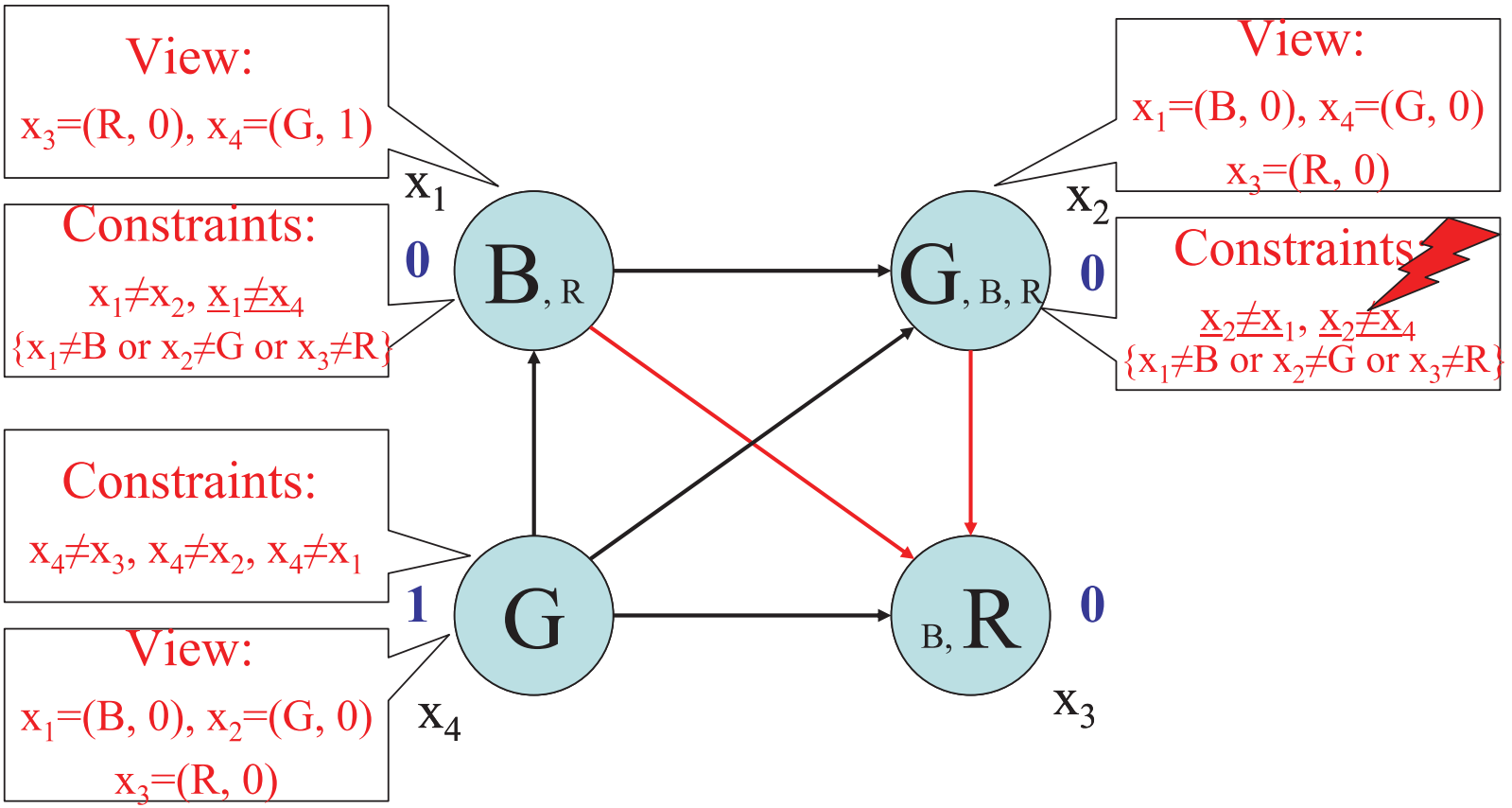
Update view

All agents receive the OK? messages and update their view

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۵ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



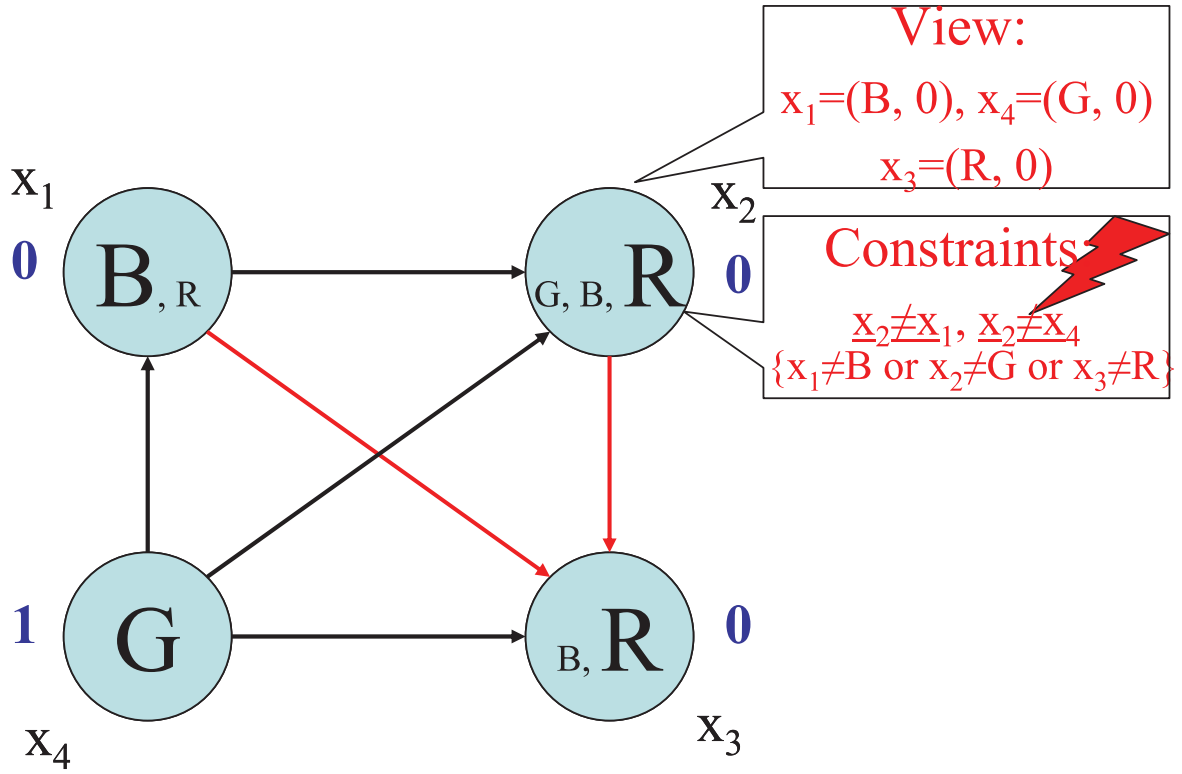
check view

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۶ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



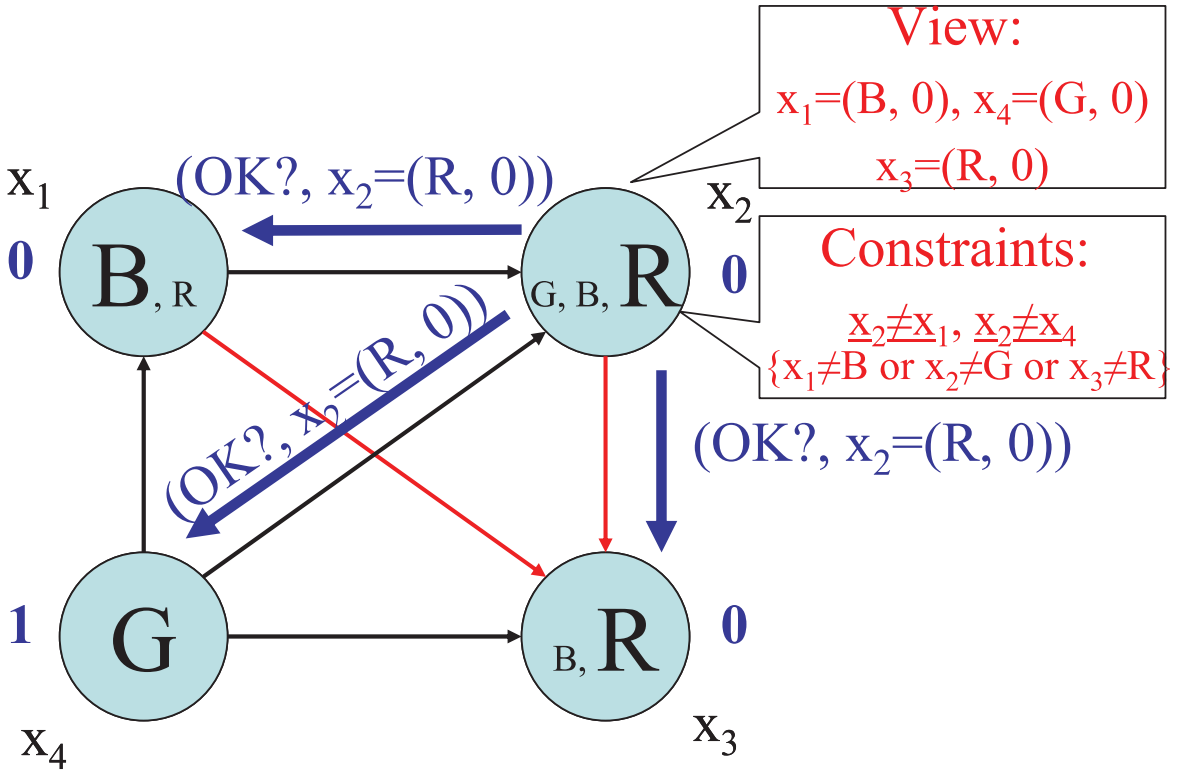
Try to choose value

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۷ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



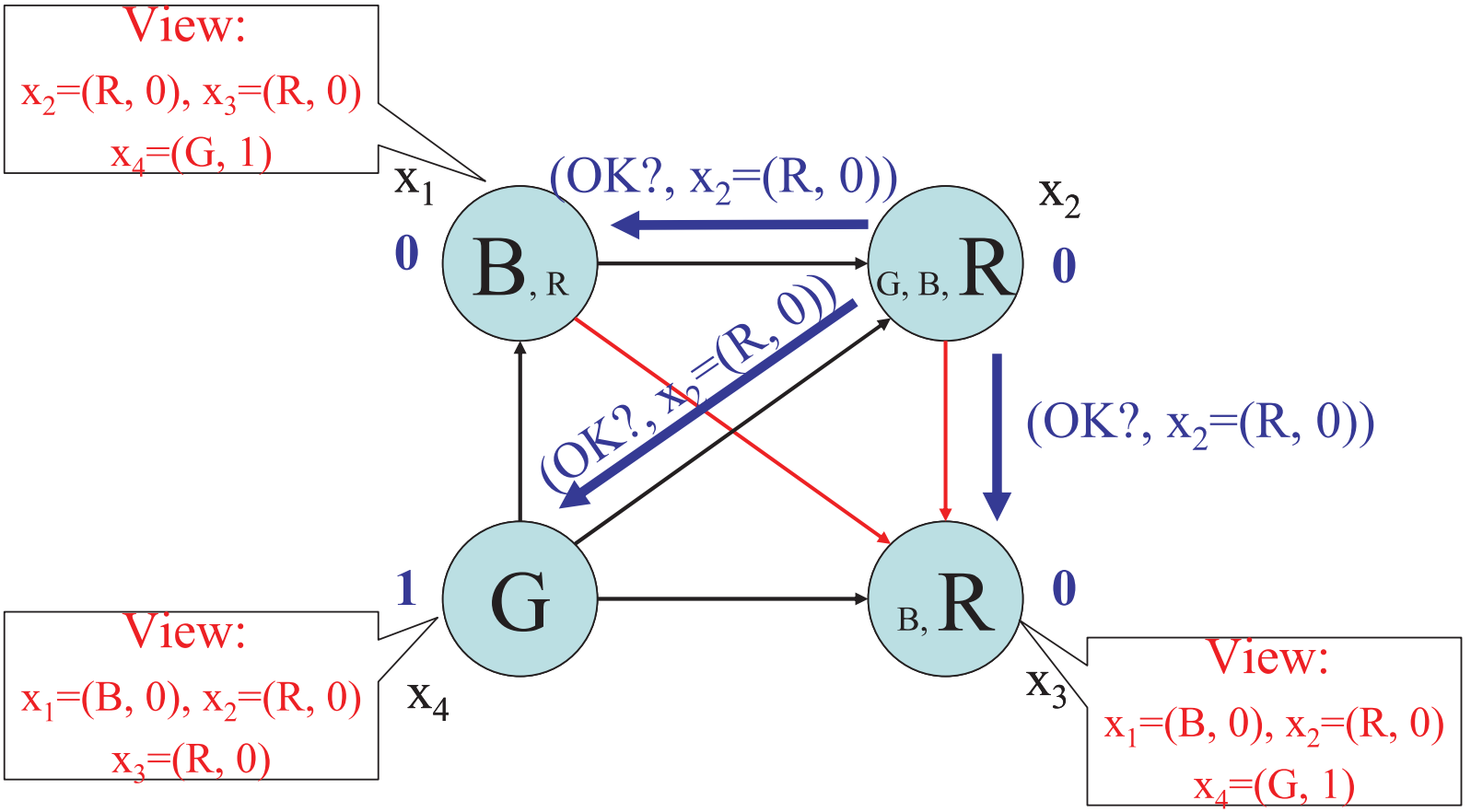
Send OK? messages

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

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۸ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



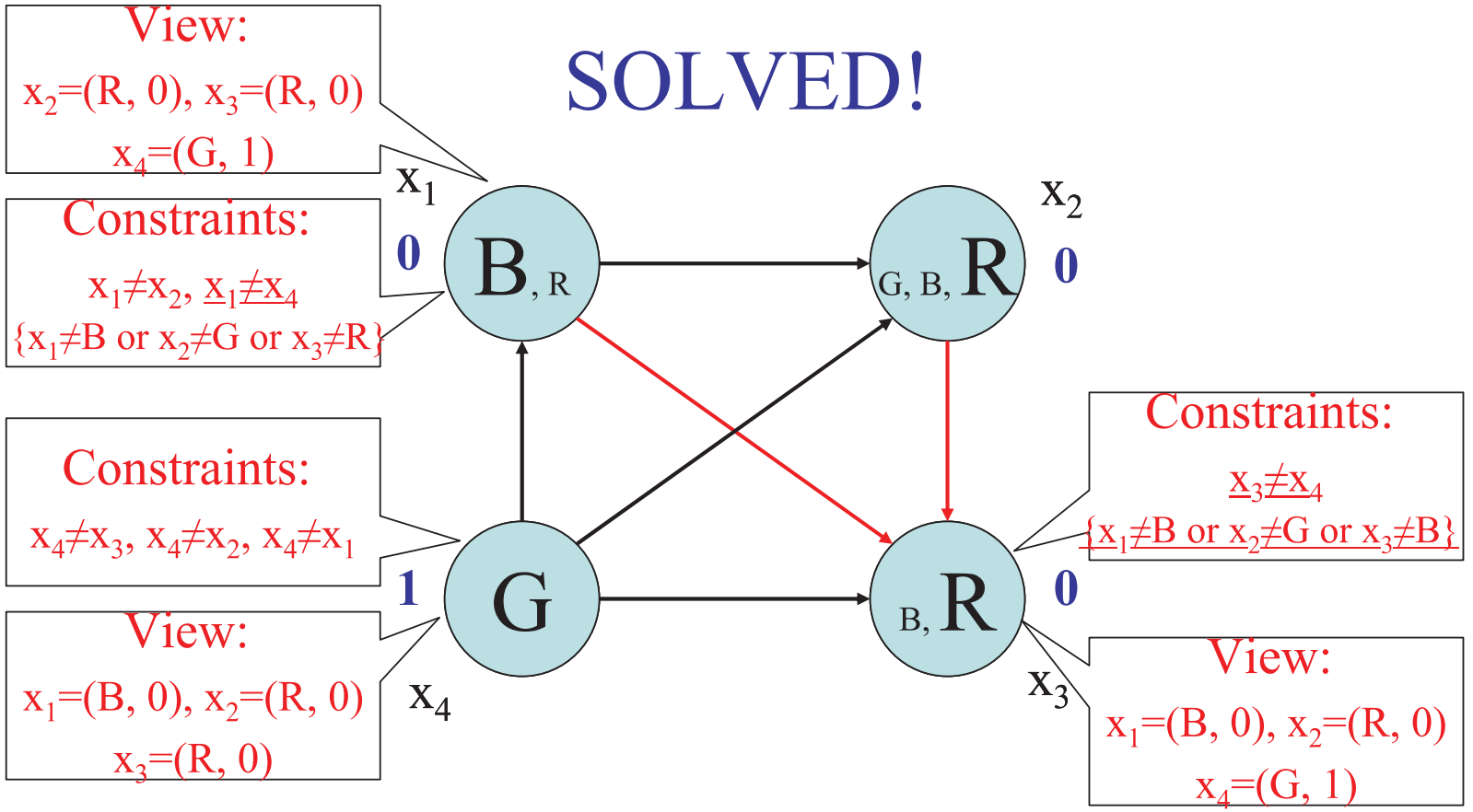
Update view

All agents receive the OK? messages and update their view

الگوریتم جستجوی تعهد ضعیف ناهمگام

مثال (۲۹ از ۲۹)

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM



check view

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

مسائل بهینه‌سازی قید توزیع‌شده

DISTRIBUTED CONSTRAINT OPTIMIZATION PROBLEMS

Given

Variables $\{x_1, x_2, \ldots, x_n\}$, each assigned to an agent

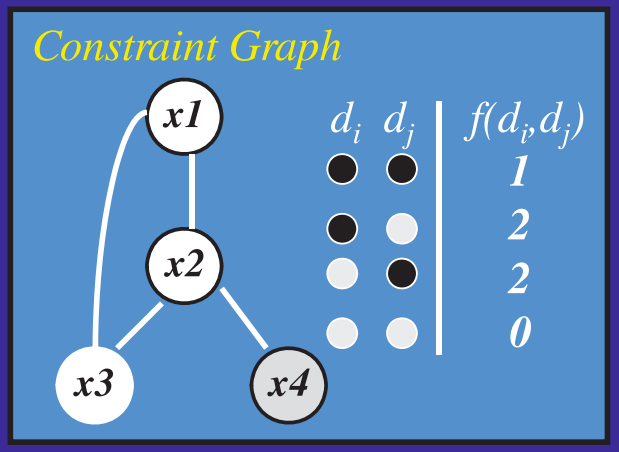
Finite, discrete domains $D_1, D_2, \ldots, D_n$,

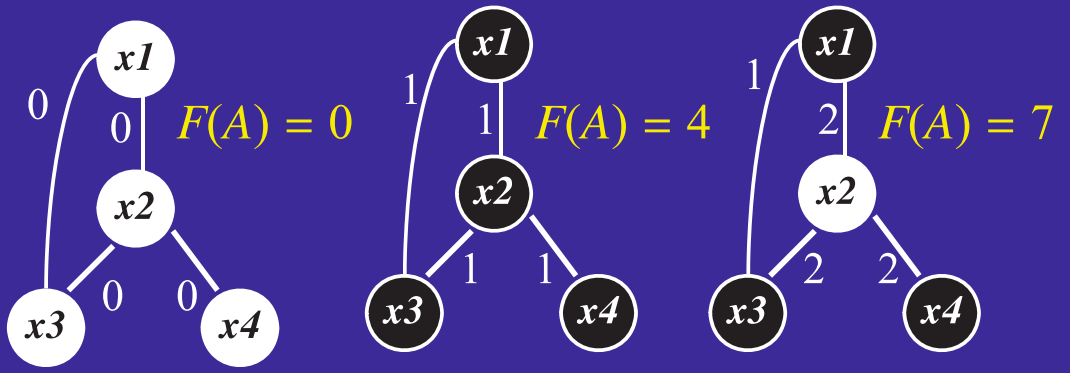
For each x_i, x_j, valued constraint $f_{ij}: D_i \times D_j \rightarrow N$.

Goal

Find complete assignment A that minimizes $F(A)$ where,

$$F(A) = \sum f_{ij}(d_i,d_j),\ x_i \leftarrow d_i,\ x_j \leftarrow d_j \text{ in } A$$



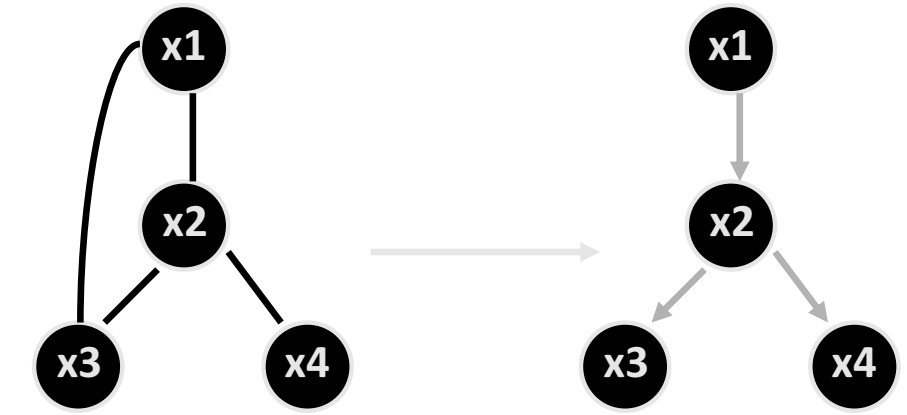


الگوریتم جستجوی تعهد ضعیف ناهمگام

THE ASYNCHRONOUS WEAK-COMMITMENT SEARCH ALGORITHM

الگوریتم بهینه‌سازی توزیع‌شده‌ی ناهمگام

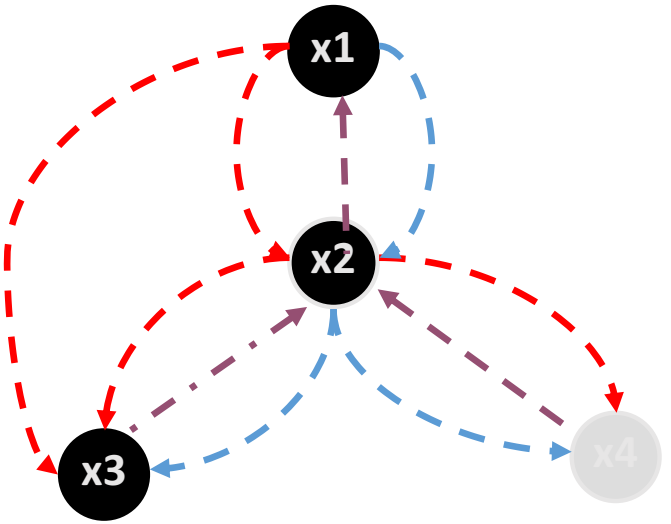
ADOPT (ASYNCHRONOUS DISTRIBUTED OPTIMIZATION)



الگوریتم بهینه‌سازی توزیع‌شده‌ی ناهمگام

ADOPT (Asynchronous Distributed Optimization)

- Agents are ordered in a tree
  - constraints between ancestors/descendents
  - no constraints between siblings



- Basic Algorithm:
  - choose value with min cost
  - Loop until termination-condition true:
    - When receive message:
      - choose value with min cost
      - send **VALUE** message to descendents
      - send **COST** message to parent
      - send **THRESHOLD** message to child

VALUE messages

COST messages

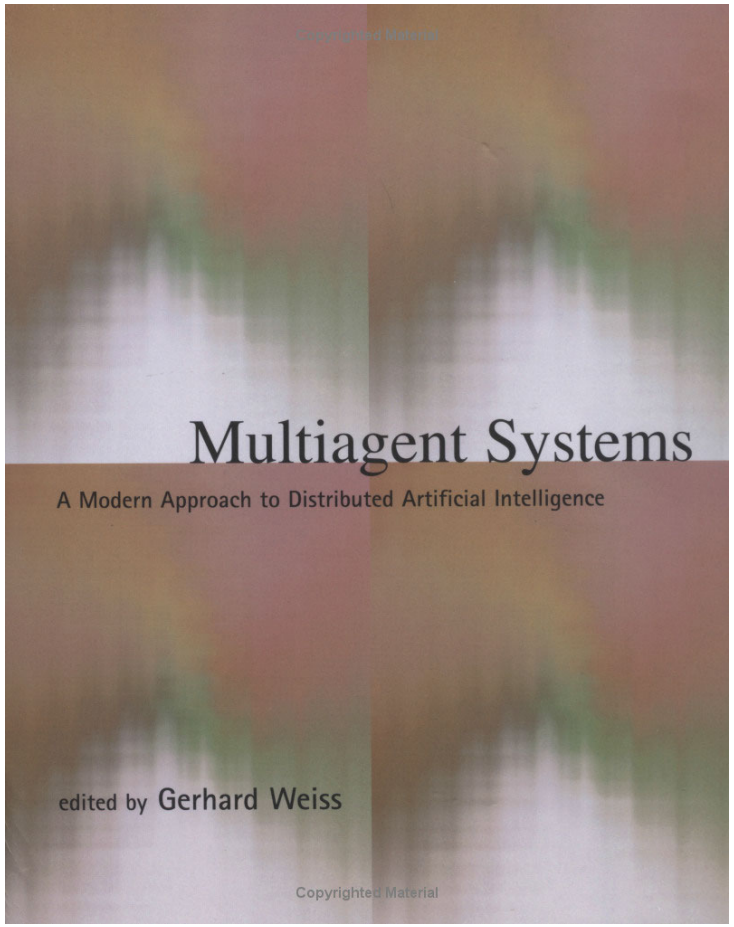
THRESH messages

سیستم‌های چندعاملی

ارضای قید توزیع‌شده

منابع

منبع اصلی



Gerhard Weiss (ed.),
Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence,
MIT Press, 1999.
Chapter 4

4 Search Algorithms for Agents

Makoto Yokoo and Toru Ishida

4.1 Introduction

In this chapter, we introduce several search algorithms that are useful for problem solving by multiple agents. Search is an umbrella term for various problem solving techniques in AI. In search problems, the sequence of actions required for solving a problem cannot be known *a priori* but must be determined by a trial-and-error exploration of alternatives. Since virtually all AI problems require some sort of search, search has a long and distinguished history in AI.

The problems that have been addressed by search algorithms can be divided into three classes: path-finding problems, constraint satisfaction problems, and two-player games.

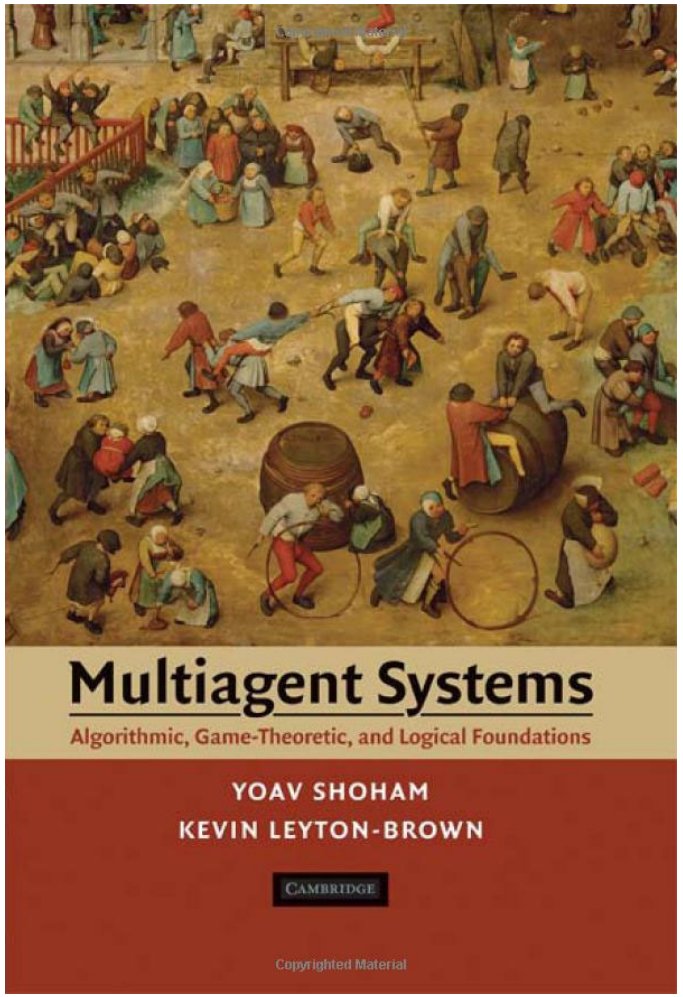
A typical example of the first class, i.e., path-finding problems, is a puzzle called the *n-puzzle*. Figure 4.1 shows the 8-puzzle, which consists of eight numbered tiles arranged on a 3×3 board (in a generalized case, there are $n = k^2 - 1$ tiles on a $k \times k$ board). The allowed moves are to slide any tile that is horizontally or vertically adjacent to the empty square into the position of the empty square. The objective is to transform the given initial configuration to the goal configuration by making allowed moves. Such a problem is called a path-finding problem, since the objective is to find a path (a sequence of moves) from the initial configuration to the goal configuration.

A constraint satisfaction problem (CSP) involves finding a goal configuration rather than finding a path to the goal configuration. A typical example of a CSP is a puzzle called 8-queens. The objective is to place eight queens on a chess board (8×8 squares) so that these queens will not threaten each other. This problem is called a constraint satisfaction problem since the objective is to find a configuration that satisfies the given conditions (constraints).

Another important class of search problems is two-player games, such as chess. Since two-player games deal with situations in which two *competitive* agents exist, it is obvious that these studies have a very close relation with DAI/multiagent systems where agents are competitive.

On the other hand, most algorithms for the other two classes (constraint satisfaction and path-finding) were originally developed for single-agent problem solving.

منبع کمکی



Yoav Shoham and Kevin Leyton-brown,
Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations,
Cambridge University Press, 2009.
Chapter 1

1 *Distributed Constraint Satisfaction*

sensor network

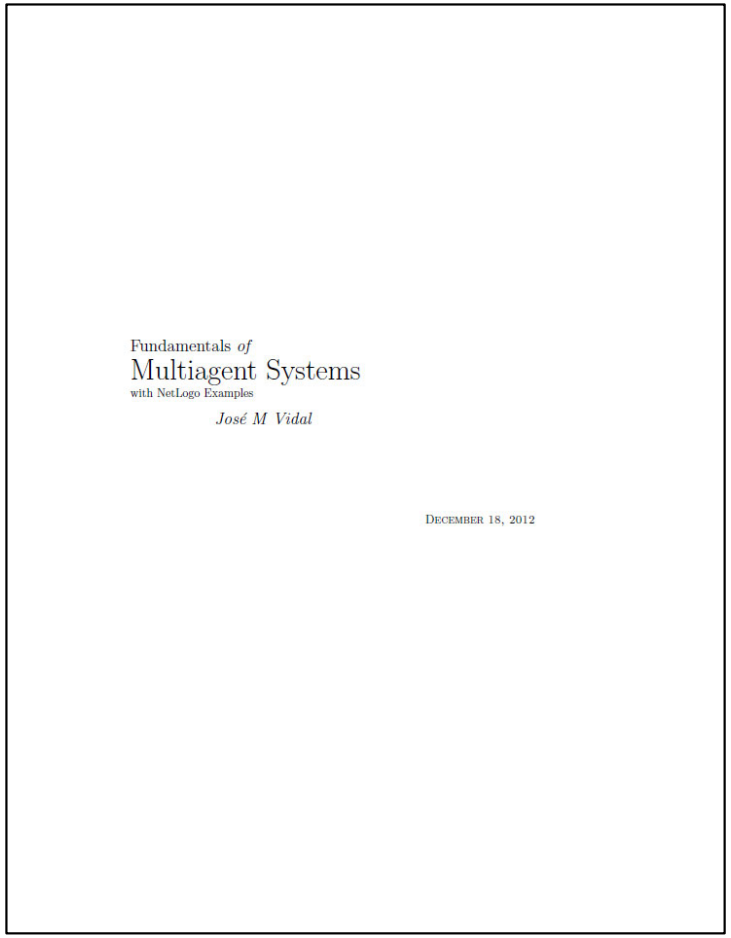
In this chapter and the next we discuss cooperative situations in which agents collaborate to achieve a common goal. This goal can be viewed as shared between the agents or, alternatively, as the goal of a central designer who is designing the various agents. Of course, if such a designer exists, a natural question is why it matters that there are multiple agents; they can be viewed merely as end sensors and effectors for executing the plan devised by the designer. However, there exist situations in which a problem needs to be solved in a distributed fashion, either because a central controller is not feasible or because one wants to make good use of the distributed resources. A good example is provided by *sensor networks*. Such networks consist of multiple processing units, each with local sensor capabilities, limited processing power, limited power supply, and limited communication bandwidth. Despite these limitations, these networks aim to provide some global service. Figure 1.1 shows an example of a fielded sensor network used for monitoring environmental quantities like humidity, temperature and pressure in an office environment. Each sensor can monitor only its local area and, similarly, can communicate only with other sensors in its local vicinity. The question is what algorithm the individual sensors should run so that the center can still piece together a reliable global picture.

Distributed algorithms have been widely studied in computer science. We concentrate on distributed problem-solving algorithms of the sort studied in artificial intelligence. We divide the discussion into two parts. In this chapter we cover distributed constraint satisfaction, where agents attempt in a distributed fashion to find a feasible solution to a problem with global constraints. In the next chapter we look at agents who try not only to satisfy constraints, but also to optimize some objective function subject to these constraints.

Later in this book we will encounter additional examples of distributed problem solving. Each of them requires specific background, however, which is why they are not discussed here. Two of them stand out in particular.

- In Chapter 7 we encounter a family of techniques that involve learning, some of them targeted at purely cooperative situations. In these situations the agents learn through repeated interactions how to coordinate a choice of action. This material requires some discussion of noncooperative game theory (discussed in

منبع کمکی


Fundamentals *of*
Multiagent Systems
with NetLogo Examples

José M Vidal

December 18, 2012

José M Vidal,
Fundamentals of Multiagent Systems with NetLogo Examples,
Unpublished, 2012.
Chapter 2


Chapter 2

Distributed Constraints

Most multiagent systems are characterized by a set of autonomous agents each with local information and ability to perform an action when the set of actions of all must be coordinated so as to achieve a desired global behavior. In all these cases you own all the agents in question and can program them to do whatever you want. Thus, there is no need to properly incentivise them as there would be in an open system, which we study in later Chapters. However, there is still the problem of coordination. Since each agent only has local information it might be hard for it to decide what to do.

In this chapter we look at some algorithms for performing distributed search in cooperative multiagent systems where each agent has some local information and where the goal is to get all the agents to set themselves to a state such that the set of states in the system is optimal. For example, imagine a group of small sensors in a field. Each sensor can communicate only with those that are near him and has to decide which of its modalities to use (sense temperature, point radar North, point radar South, etc.) so that the group of sensors gets a complete view of the important events in the field. Or, imagine a group of kids who have been told to stand in a circle, each one must move find a spot to stand on but the position of that spot depends on everyone else's location. In these examples the agents have local information, can take any one of their available actions at any time, but the utility of their actions depends on the actions of others. We find that many multiagent problems can be reduced to a distributed constraints problem. Thus, the algorithms we present in this chapter have many different applications.

2.1 Distributed Constraint Satisfaction

We start by formally describing the problem. In a **constraint satisfaction problem** (CSP) we are given a set of variables, each with its own domain along with a set of constraints. The goal is to set every variable to a value from its domain such that no constraints are violated. Formally,

CONSTRAINT SATISFACTION PROBLEM

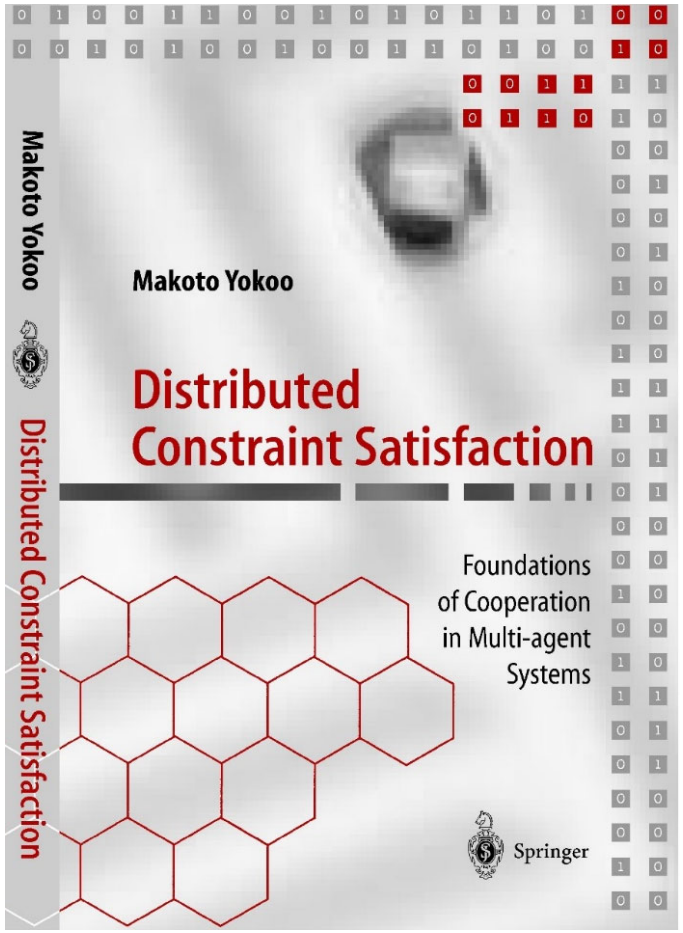
Definition 2.1 (Constraint Satisfaction Problem). *Given a set of variables $x_1, x_2, \ldots, x_n$ with domains $D_1, D_2, \ldots, D_n$ and a set of boolean constraints P of the form $p_k(x_{k1}, x_{k2}, \ldots, x_{kj}) \rightarrow \{0,1\}$, find assignments for all the variables such that no constraints are violated.*

The most widely studied instance of a constraint satisfaction problem is the graph coloring problem, shown in figure 2.1. In this problem we are given a graph and a set of colors. The problem is to find out if there is a way to color each node with one of the given colors such that no two nodes that are connected by an edge have the same color. We can easily map the graph coloring problem to the formal definition of a CSP by considering each node to be a variable, the domains to be the set of colors, and each edge becomes a constraint between its two nodes that is true only if their values are different. Notice that, in graph coloring constraints are only over two variables instead of over any set of variables as we find in the general constraint satisfaction problem.

The constraint satisfaction problem is NP-complete. As such, me use search algorithms to find a solution and hope that the actual running time will be less than

19

منبع کمکی



Makoto Yokoo,
Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems,
Springer, 2001