Solving CSPs

Solving CSPs involves some combination of:

1. Constraint propagation, to eliminate values that could not be part of any solution
2. Search, to explore valid assignments

Constraint Propagation (aka Arc Consistency)

Arc consistency eliminates values from domain of variable that can never be part of a consistent solution.

$V_i \rightarrow V_j$

Directed arc (V_i, V_j) is arc consistent if
$\forall x \in D_i \; \exists y \in D_j$ such that (x,y) is allowed by the constraint on the arc

Constraint Propagation (aka Arc Consistency)

Arc consistency eliminates values from domain of variable that can never be part of a consistent solution.

$V_i \rightarrow V_j$

Directed arc (V_i, V_j) is arc consistent if
$\forall x \in D_i \; \exists y \in D_j$ such that (x,y) is allowed by constraint

We can achieve consistency on arc by deleting values form D_i (domain of variable at tail of constraint arc) that fail this condition.

Constraint Propagation (aka Arc Consistency)

Arc consistency eliminates values from domain of variable that can never be part of a consistent solution.

$V_i \rightarrow V_j$

Directed arc (V_i, V_j) is arc consistent if
$\forall x \in D_i \; \exists y \in D_j$ such that (x,y) is allowed by constraint

We can achieve consistency on arc by deleting values form D_i (domain of variable at tail of constraint arc) that fail this condition.

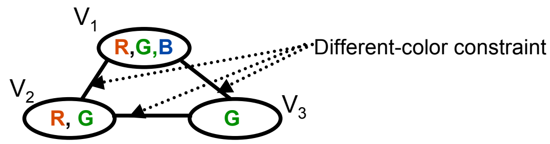
Assume domains are size at most <u>d</u> and there are <u>e</u> binary constraints.

A simple algorithm for arc consistency is $O(ed^3)$ – note that just verifying arc consistency takes $O(d^2)$ for each arc.

Constraint Propagation Example

Graph Coloring

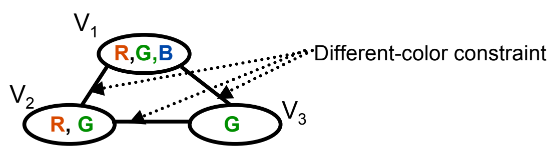
Initial Domains are indicated



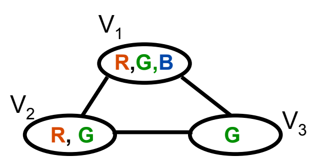
Constraint Propagation Example

Graph Coloring

Initial Domains are indicated



Arc examined	Value deleted

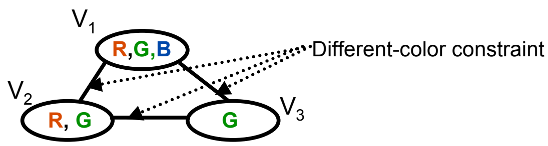


Each undirected constraint arc is really two directed constraint arcs, the effects shown above are from examining BOTH arcs.

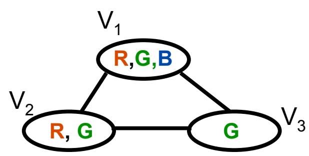
Constraint Propagation Example

Graph Coloring

Initial Domains are indicated



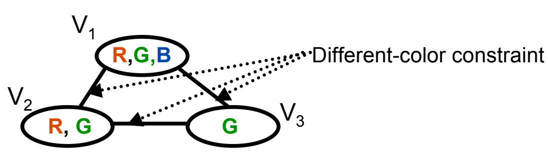
Arc examined	Value deleted
$V_1 - V_2$	none



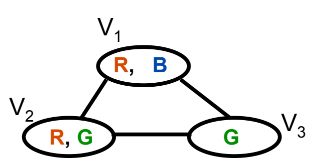
Constraint Propagation Example

Graph Coloring

Initial Domains are indicated



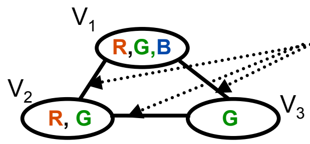
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)



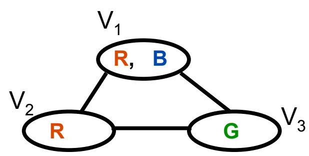
Constraint Propagation Example

Graph Coloring

Initial Domains are indicated



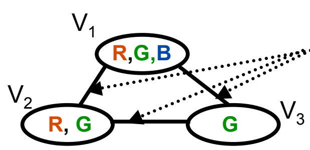
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)



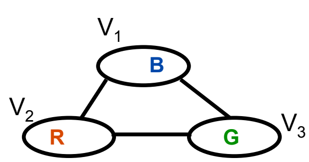
Constraint Propagation Example

Graph Coloring

Initial Domains are indicated



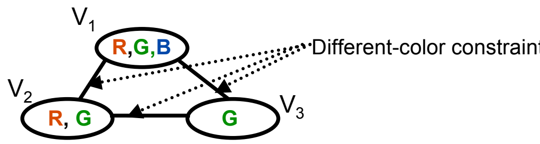
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)
$V_1 - V_2$	V_1(R)



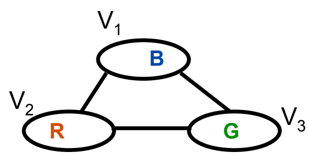
Constraint Propagation Example

Graph Coloring

Initial Domains are indicated



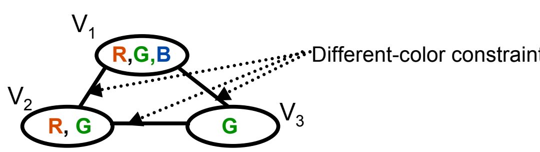
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)
$V_1 - V_2$	V_1(R)
$V_1 - V_3$	none



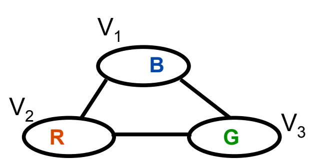
Constraint Propagation Example

Graph Coloring

Initial Domains are indicated

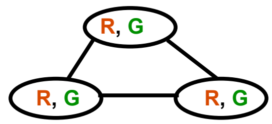


Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)
$V_1 - V_2$	V_1(R)
$V_1 - V_3$	none
$V_2 - V_3$	none



But, arc consistency is not enough in general

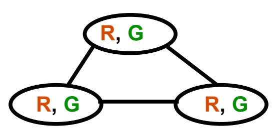
Graph Coloring



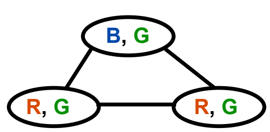
arc consistent but
no solutions

But, arc consistency is not enough in general

Graph Coloring



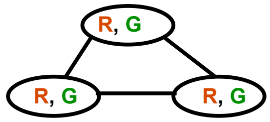
arc consistent but
no solutions



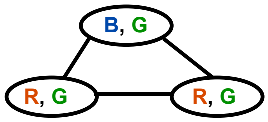
arc consistent but 2 solutions B,R,G ; B,G,R .

But, arc consistency is not enough in general

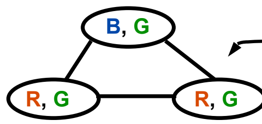
Graph Coloring



arc consistent but
no solutions



arc consistent but 2
solutions B,R,G ;
B,G,R .

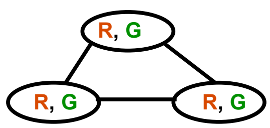


arc consistent but 1
solution

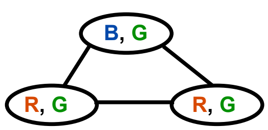
Assume B, R not
allowed

But, arc consistency is not enough in general

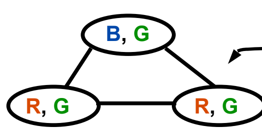
Graph Coloring



arc consistent but
no solutions



arc consistent but 2
solutions B,R,G ;
B,G,R .



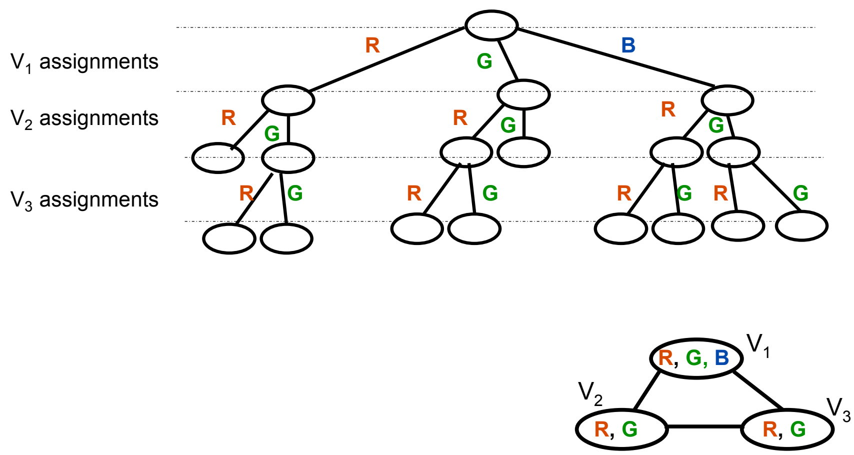
arc consistent but 1
solution

B, R not allowed

Need to do search to find solutions (if any)

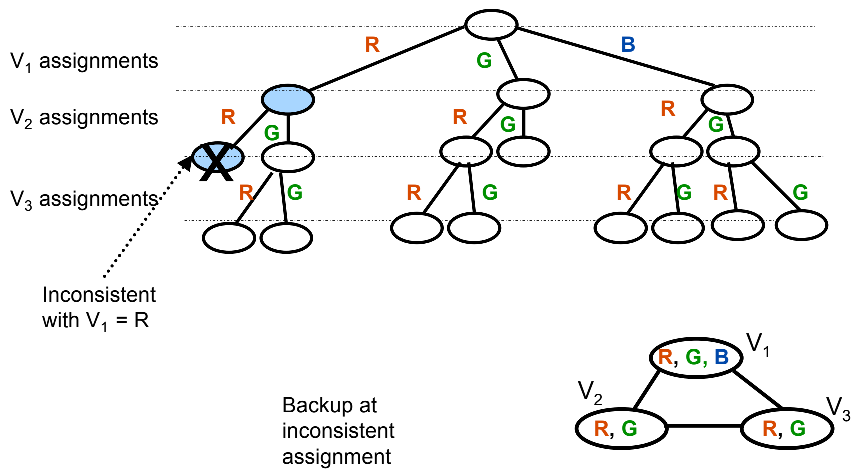
Searching for solutions – backtracking (BT)

When we have too many values in domain (and/or constraints are weak) arc consistency doesn't do much, so we need to search. Simplest approach is pure backtracking (depth-first search).



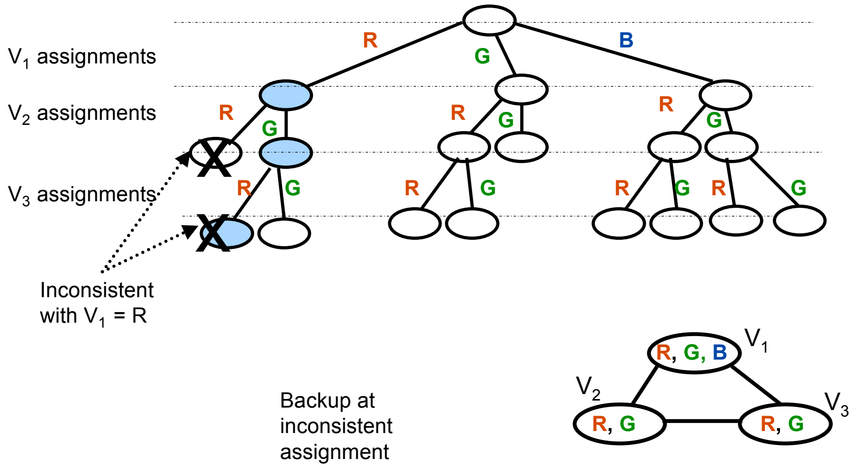
Searching for solutions – backtracking (BT)

When we have too many values in domain (and/or constraints are weak) arc consistency doesn't do much, so we need to search. Simplest approach is pure backtracking (depth-first search).



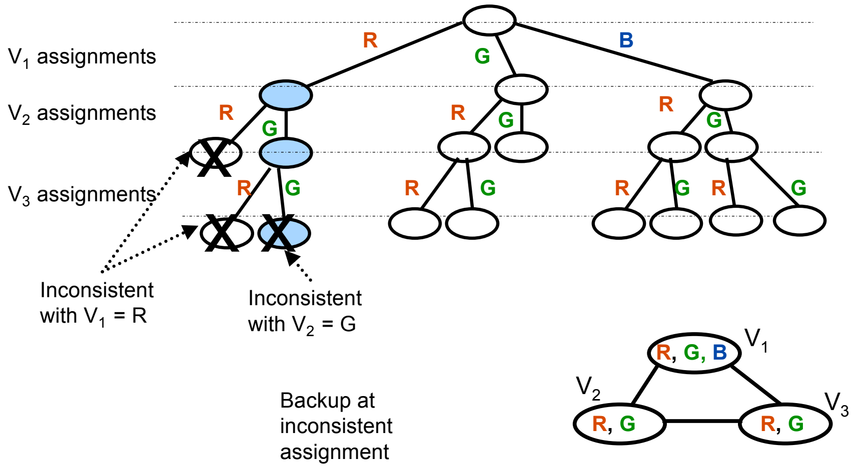
Searching for solutions – backtracking (BT)

When we have too many values in domain (and/or constraints are weak) arc consistency doesn't much, so we need to search. Simplest approach is pure backtracking (depth-first search).



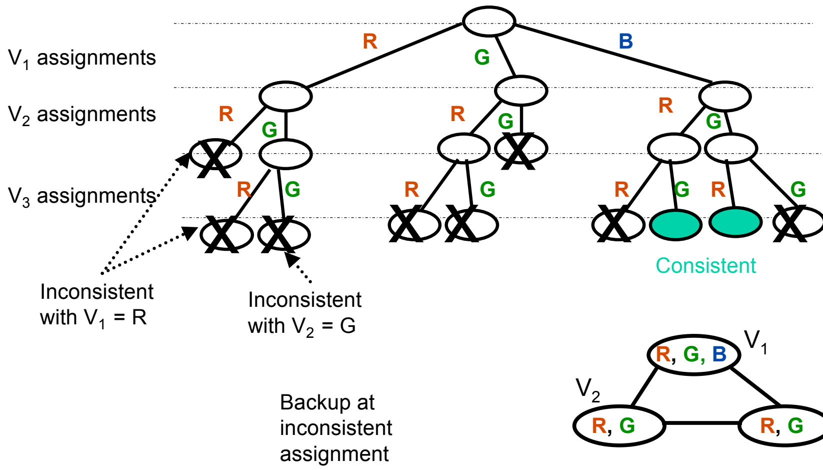
Searching for solutions – backtracking (BT)

When we have too many values in domain (and/or constraints are weak) arc consistency doesn't much, so we need to search. Simplest approach is pure backtracking (depth-first search).



Searching for solutions – backtracking (BT)

When we have too many values in domain (and/or constraints are weak) arc consistency doesn't much, so we need to search. Simplest approach is pure backtracking (depth-first search).



Combine Backtracking & Constraint Propagation

A node in BT tree is partial assignment in which domain of variables has been set (tentatively) to singleton set.

Use constraint propagation (arc-consistency) to propagate effect of this tentative assignment i.e. eliminate values inconsistent with current values.

Combine Backtracking & Constraint Propagation

A node in BT tree is <u>partial</u> assignment in which domain of variables has been set (tentatively) to singleton set.

Use constraint propagation (arc-consistency) to propagate effect of this tentative assignment i.e. eliminate values inconsistent with current values.

Question: How much propagation to do?

Combine Backtracking & Constraint Propagation

A node in BT tree is <u>partial</u> assignment in which domain of variables has been set (tentatively) to singleton set.

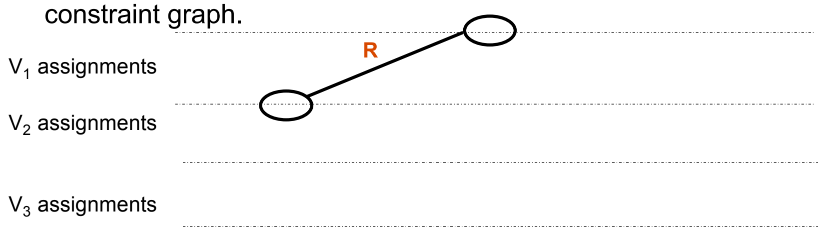
Use constraint propagation (arc-consistency) to propagate effect of this tentative assignment i.e. eliminate values inconsistent with current values.

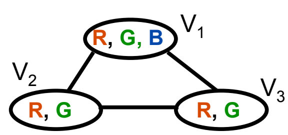
Question: How much propagation to do?

Answer: Not much, just local propagation from domains with unique assignments, which is called forward checking (FC). This conclusion is not necessarily obvious, but it generally holds in practice.

Backtracking with Forward Checking (BT-FC)

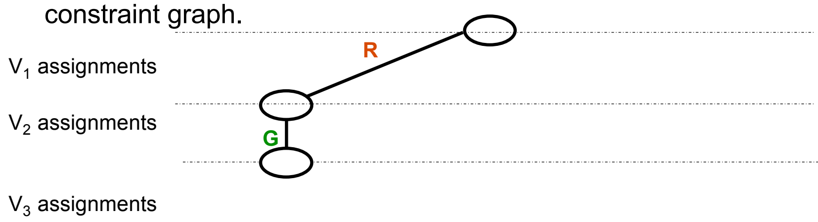
When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

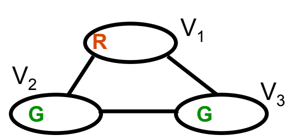




Backtracking with Forward Checking (BT-FC)

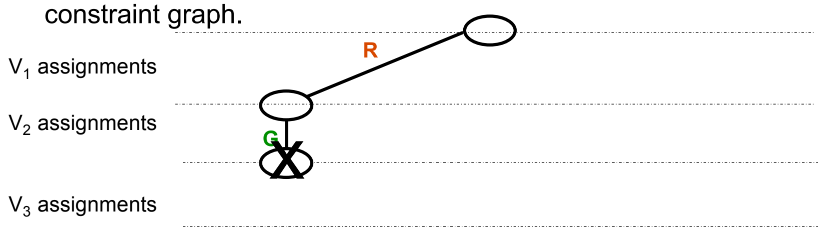
When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



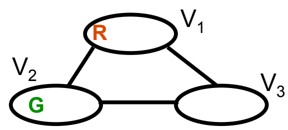


Backtracking with Forward Checking (BT-FC)

When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

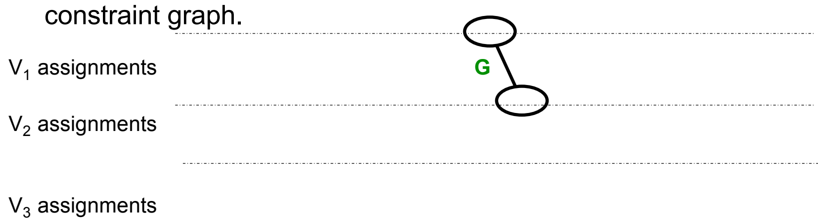


We have a conflict whenever a domain becomes empty.

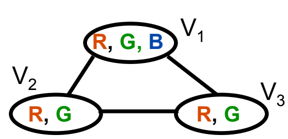


Backtracking with Forward Checking (BT-FC)

When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

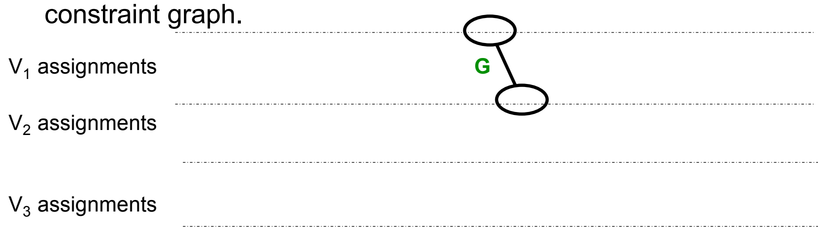


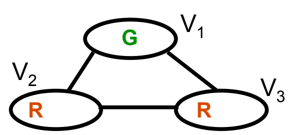
When backing up, need to restore domain values, since deletions were done to reach consistency with tentative assignments considered during search.



Backtracking with Forward Checking (BT-FC)

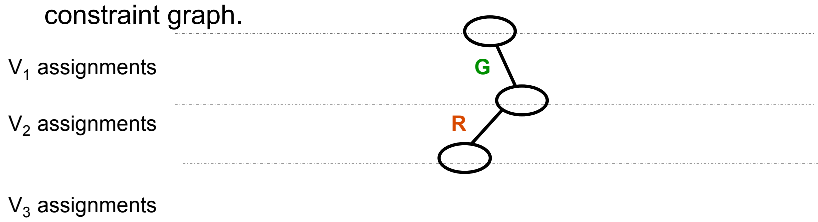
When examining assignment $V_i = d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

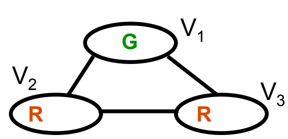




Backtracking with Forward Checking (BT-FC)

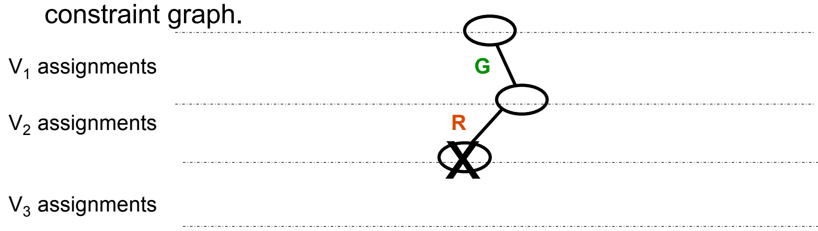
When examining assignment $V_i = d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

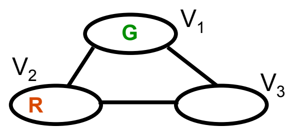




Backtracking with Forward Checking (BT-FC)

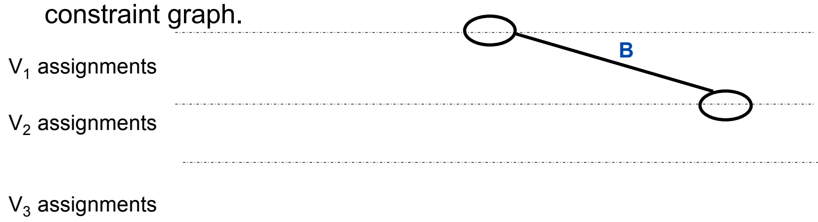
When examining assignment V_i=d_k, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

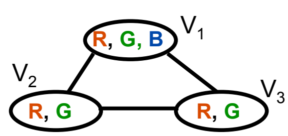




Backtracking with Forward Checking (BT-FC)

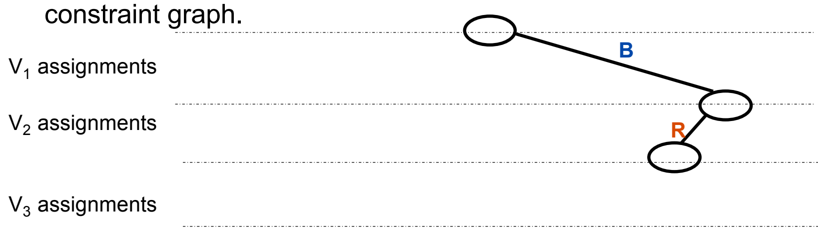
When examining assignment V_i=d_k, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

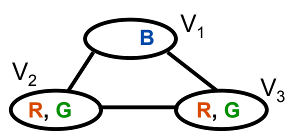




Backtracking with Forward Checking (BT-FC)

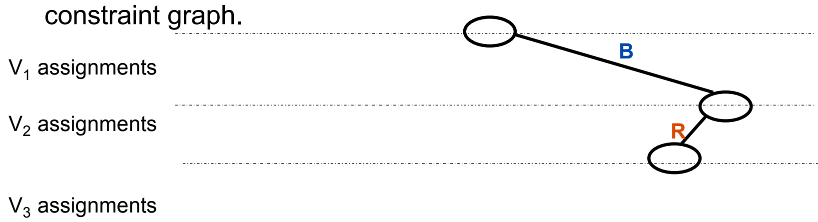
When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

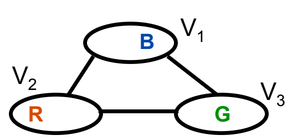




Backtracking with Forward Checking (BT-FC)

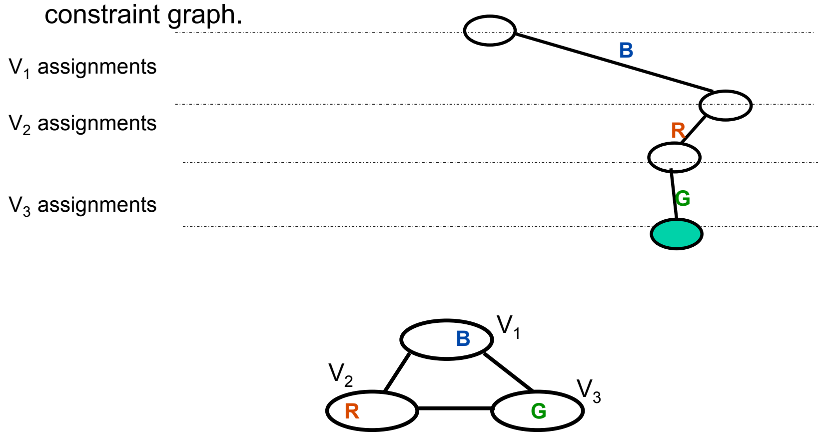
When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.





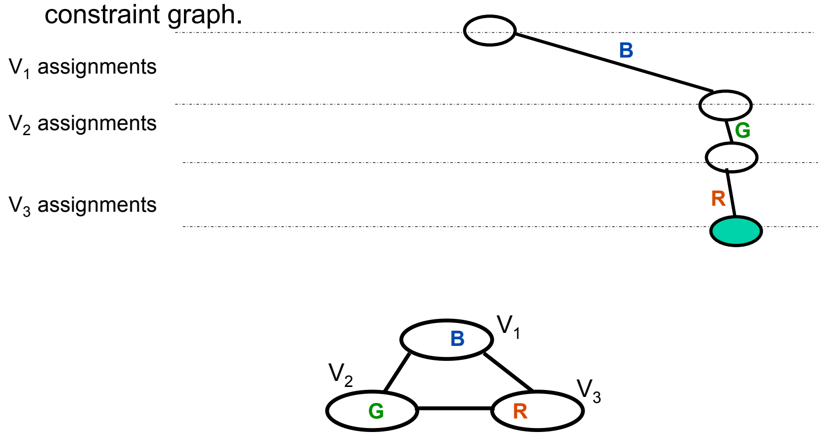
Backtracking with Forward Checking (BT-FC)

When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



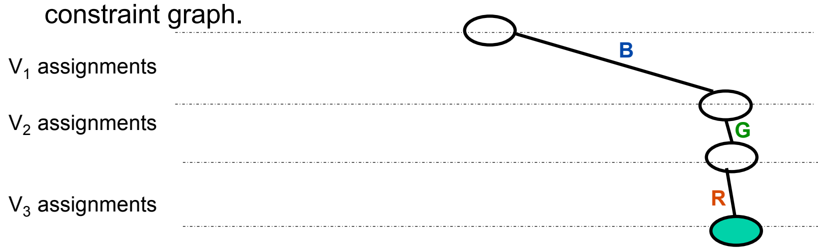
Backtracking with Forward Checking (BT-FC)

When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

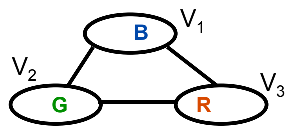


Backtracking with Forward Checking (BT-FC)

When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



No need to check previous assignments



Generally preferable to pure BT

BT-FC with dynamic ordering

Traditional backtracking uses fixed ordering of variables & values, e.g random order or place variables with many constraints first.

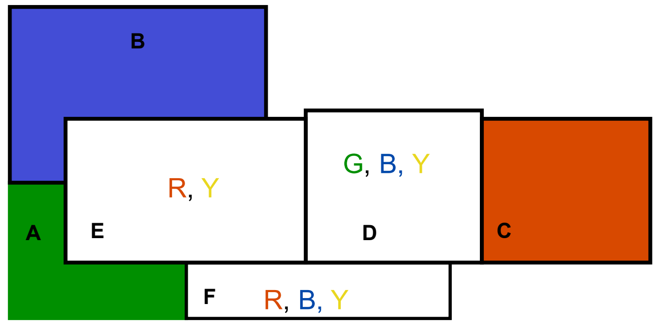
You can usually do better by choosing an order dynamically as the search proceeds.

- Most constrained variable
  when doing forward-checking, pick variable with fewest legal values to assign next (minimizes branching factor)

- Least constraining value
  choose value that rules out the smallest number of values in variables connected to the chosen variable by constraints.

E.g. this combination improves feasible n-queens performance from about n = 30 with just FC to about n = 1000 with FC & ordering.

Colors: R, G, B, Y



Which country should we color next → E most-constrained variable (smallest domain)

What color should we pick for it? → RED least-constraining value (eliminates fewest values from neighboring domains)