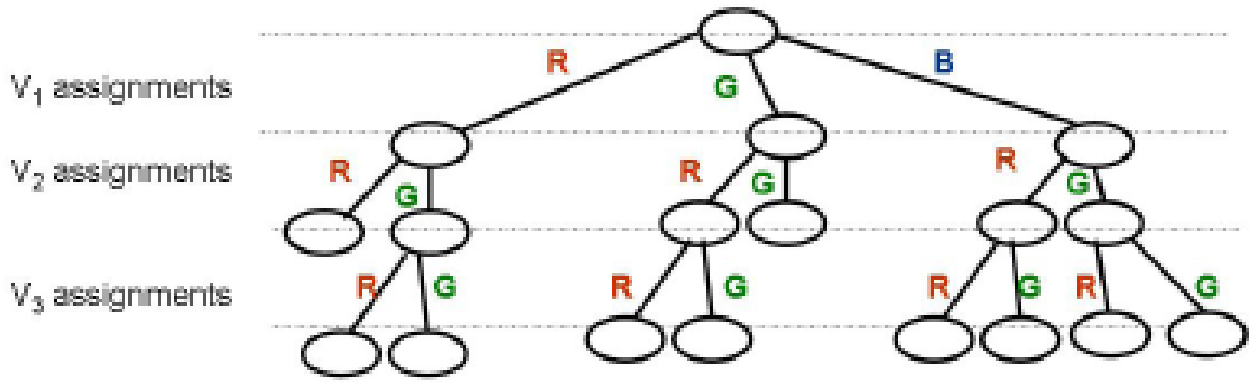
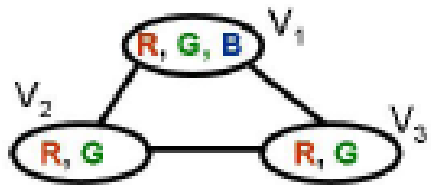
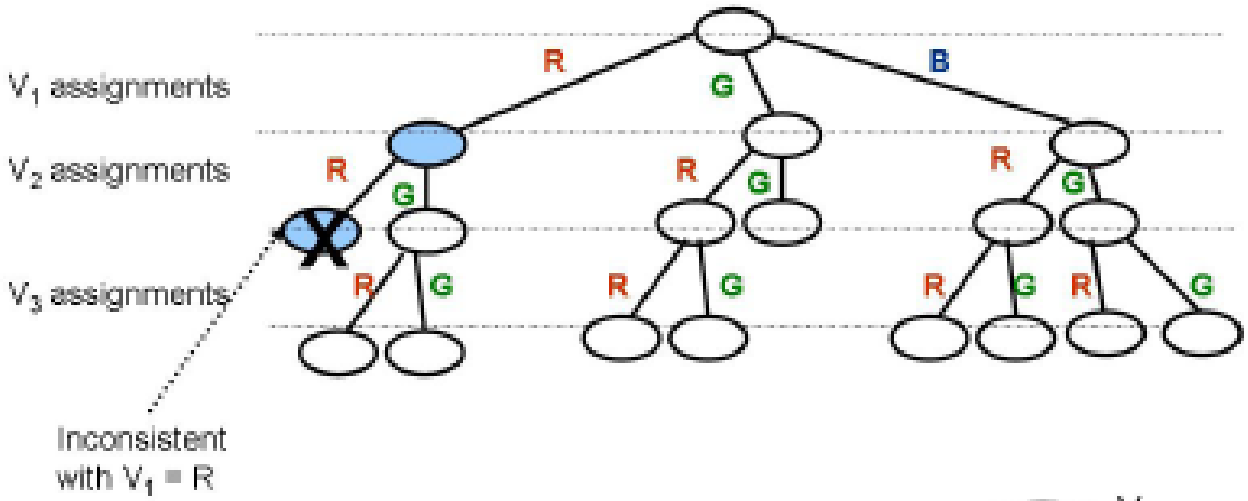
Backtracking

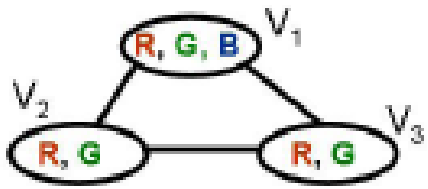




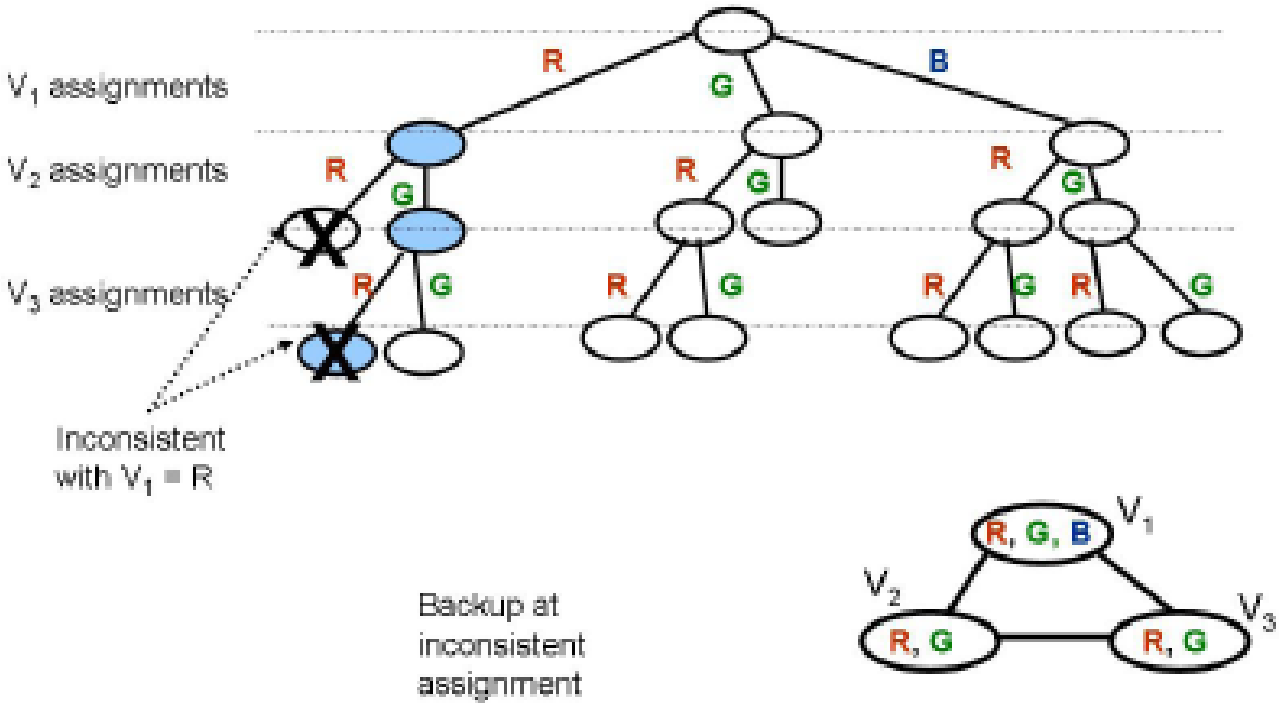
Backtracking



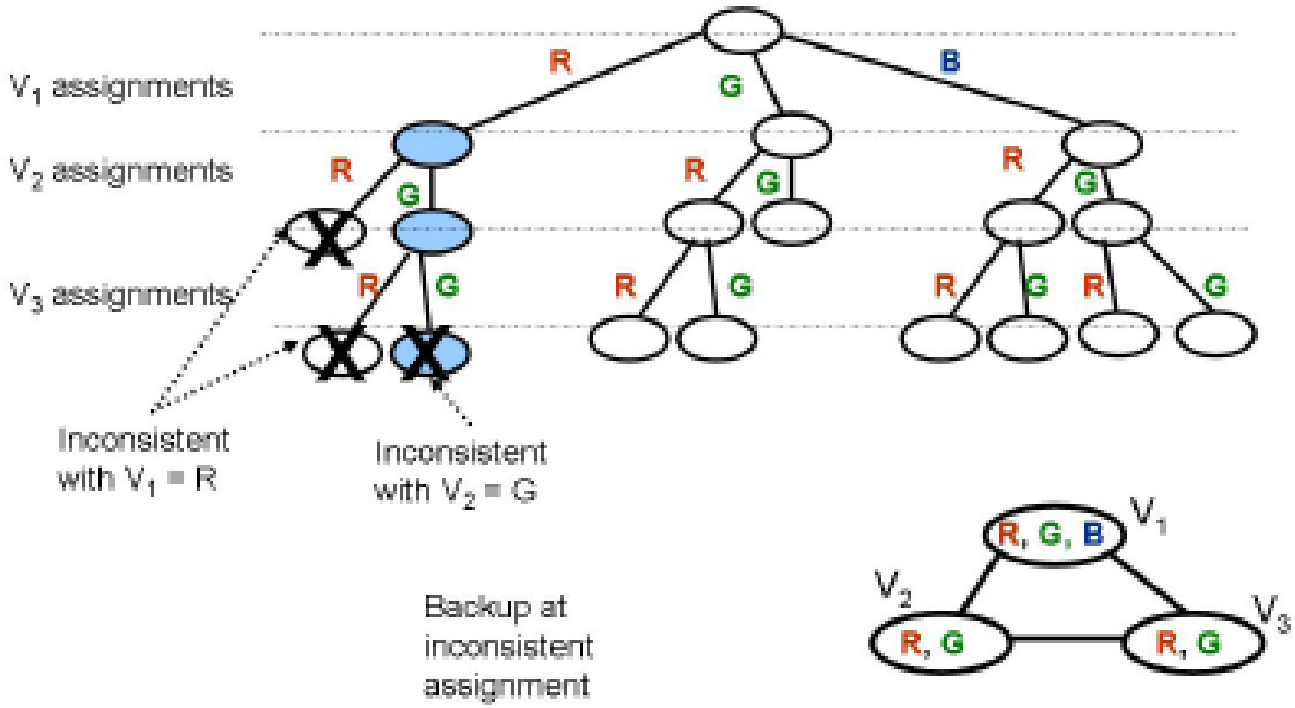
Backup at
inconsistent
assignment



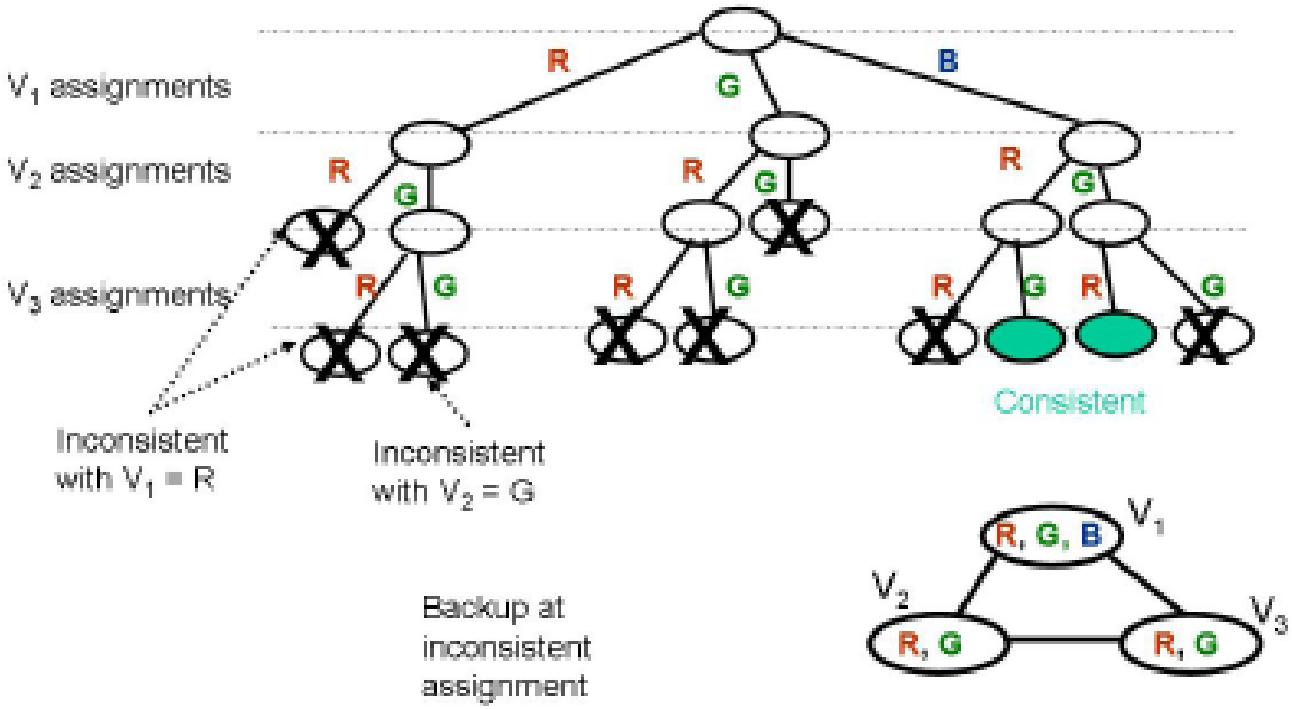
Backtracking



Backtracking



Backtracking



Dynamic Reordering

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

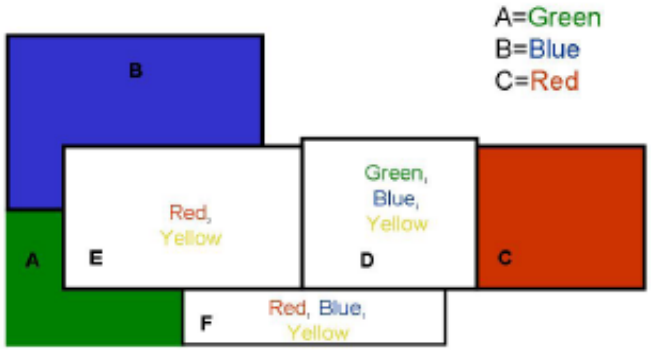
You can usually do better by choosing an order dynamically as the search proceeds.

- 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 fewest values from neighboring domains

Dynamic Reordering

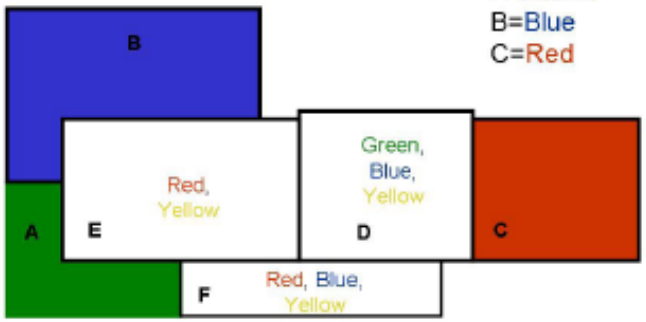
Colors: R, G, B, Y



Which country should we color next →

What color should we pick for it? →

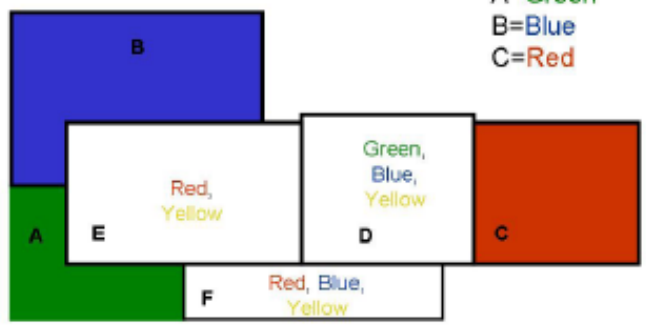
Colors: R, G, B, Y



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

What color should we pick for it? →

Colors: R, G, B, Y

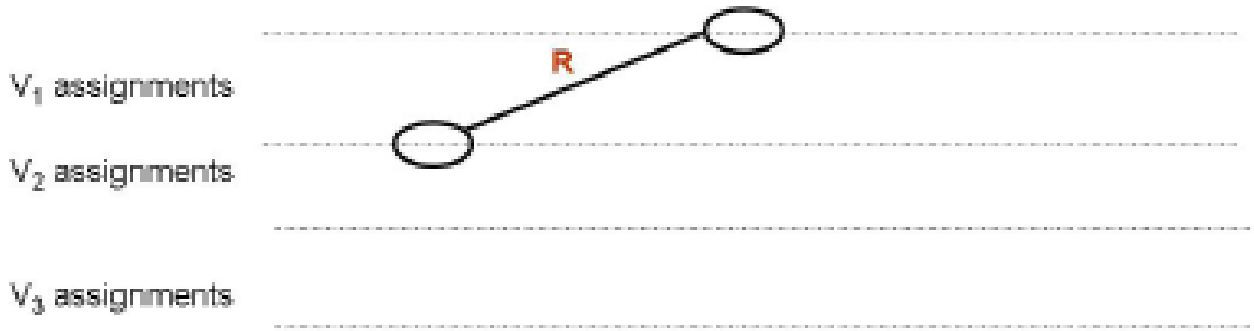


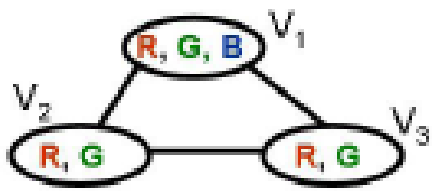
Which country should we color next → E most-constrained variable (smallest domain)

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

Backtracking with Forward Checking (BT-FC)

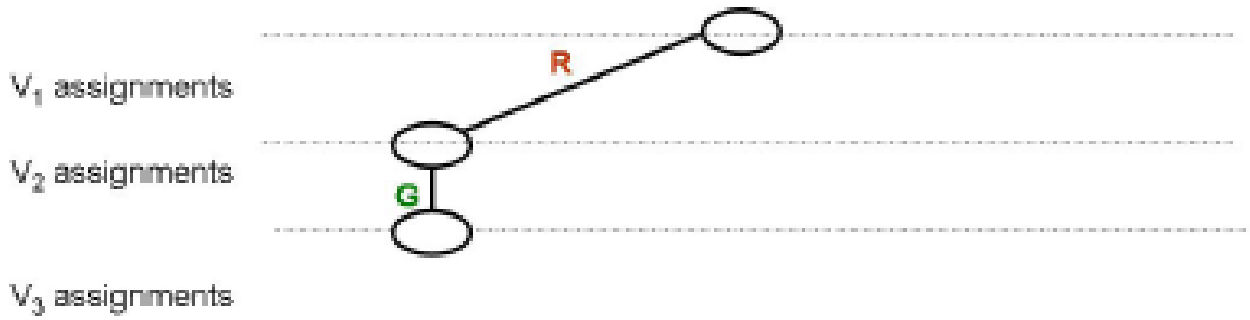
When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

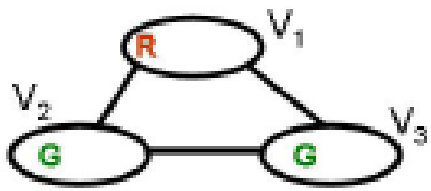




Backtracking with Forward Checking (BT-FC)

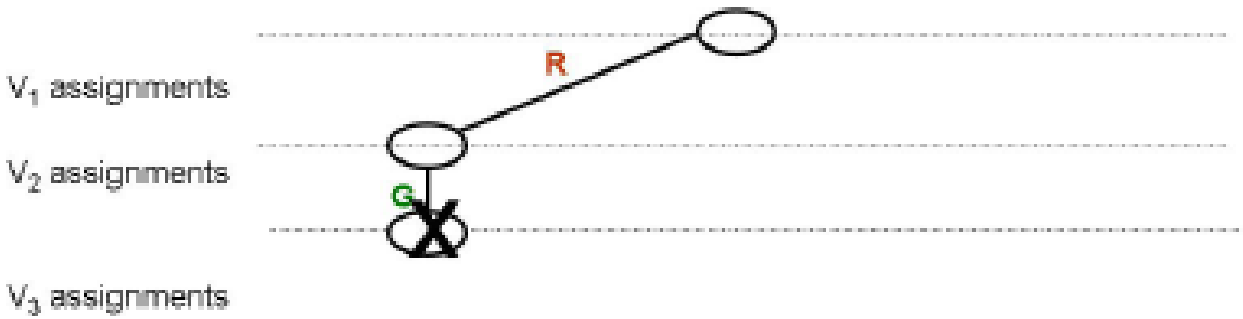
When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.



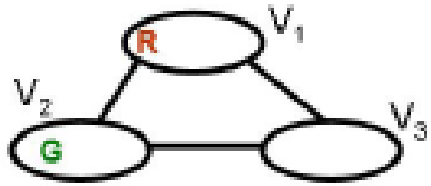


Backtracking with Forward Checking (BT-FC)

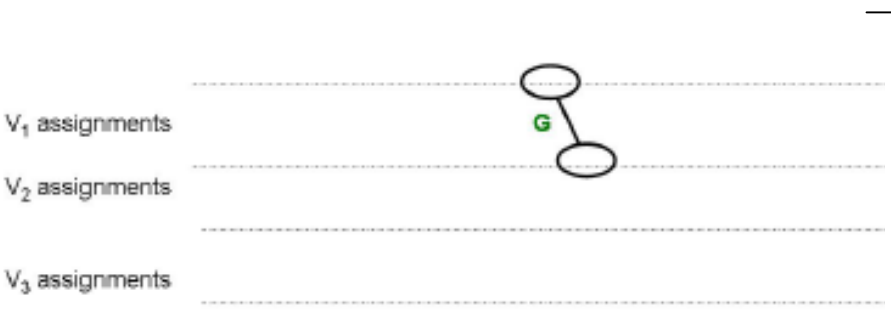
When examining assignment $V_i=d_k$, remove any values inconsistent with that assignment from neighboring domains in constraint graph.

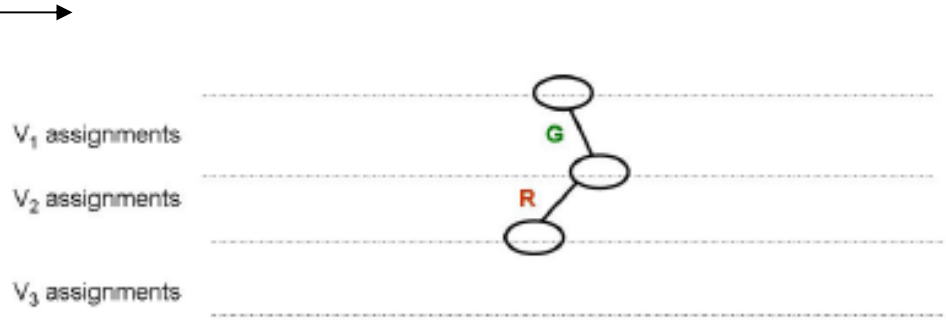


We have a conflict whenever a domain becomes empty.

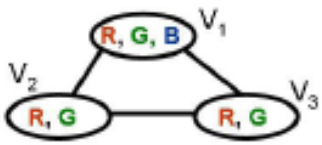


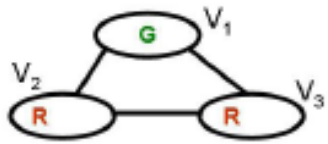
Backtracking with Forward Checking (BT-FC)

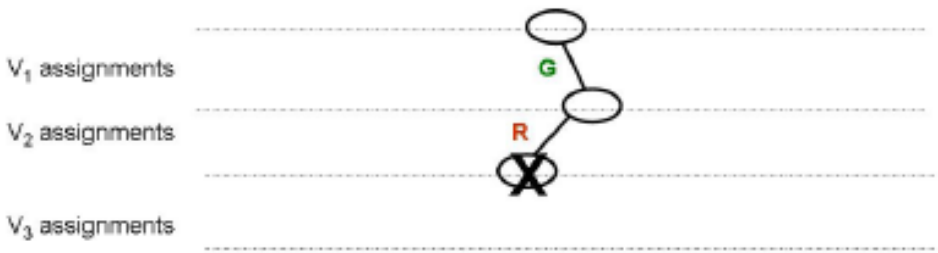


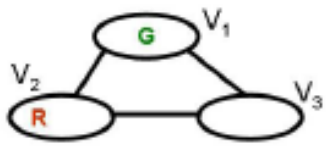


When backing up, need to restore domain values, since deletions were done to reach consistency with tentative assignments considered during search.

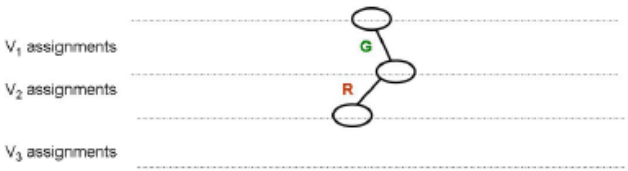


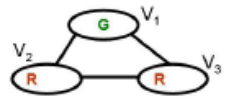


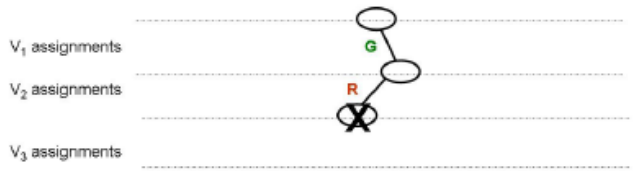


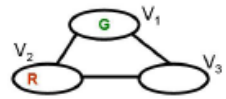


Backtracking with Forward Checking (BT-FC)

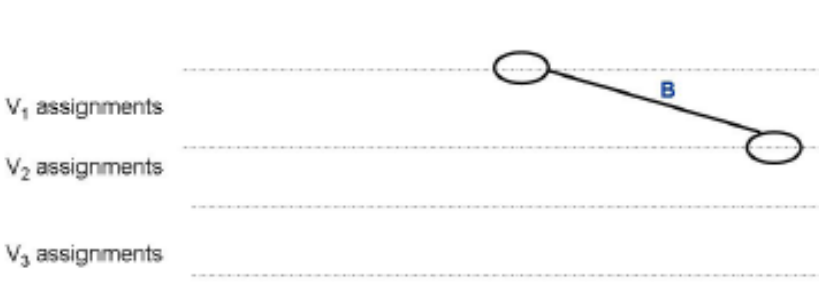


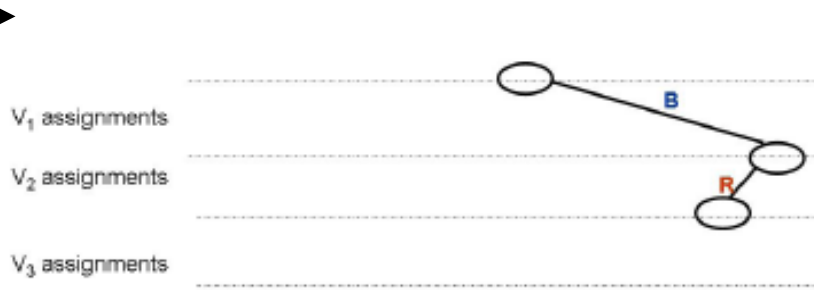


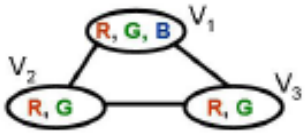


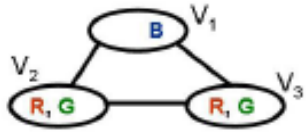


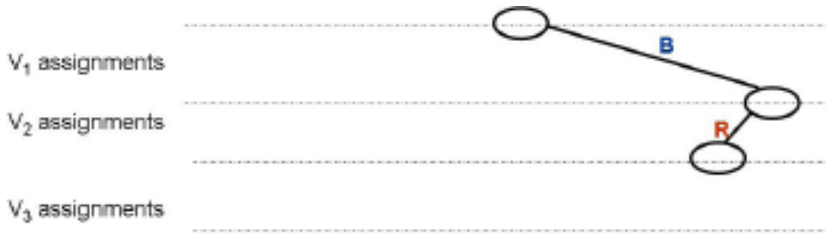
Backtracking with Forward Checking (BT-FC)

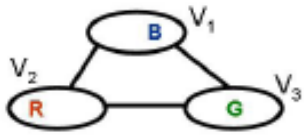


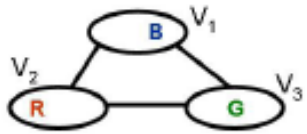












Constraint Propagation (aka Arc Consistency)

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

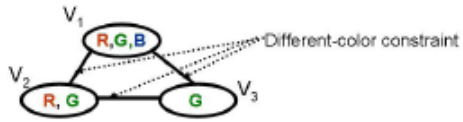
Directed arc (V_i, V_j) is arc consistent if
$\forall x \in D_i \; \exists y \in D_j$ such that (x,y) is allowed by the constraint on the arc

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

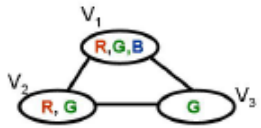
Constraint Propagation Example

Graph Coloring

Initial Domains are indicated

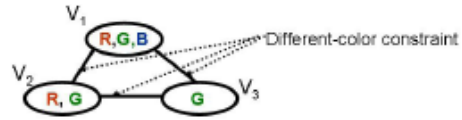


Arc examined	Value deleted

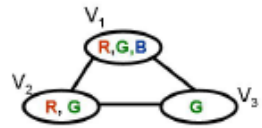


Graph Coloring

Initial Domains are indicated

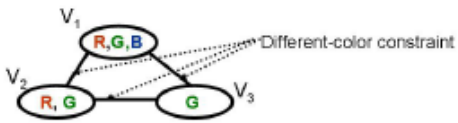


Arc examined	Value deleted
$V_1 - V_2$	none

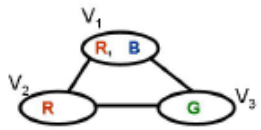


Graph Coloring

Initial Domains are indicated

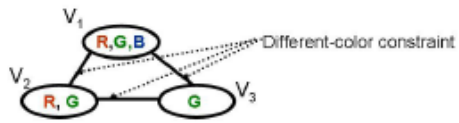


Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)

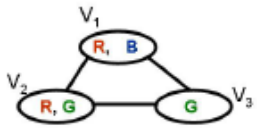


Graph Coloring

Initial Domains are indicated

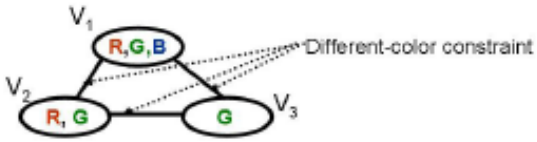


Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)

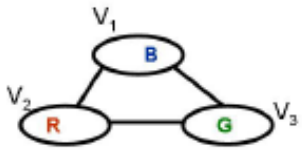


Constraint Propagation Example

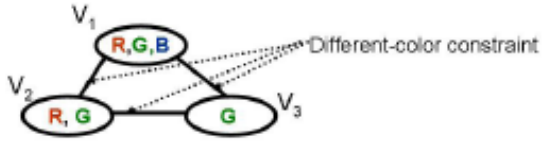
Graph Coloring
Initial Domains are indicated



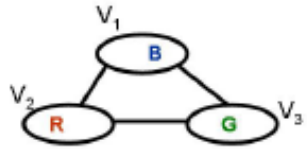
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)
$V_1 - V_2$	V_1(R)



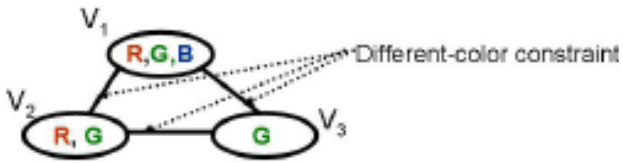
Graph Coloring
Initial Domains are indicated



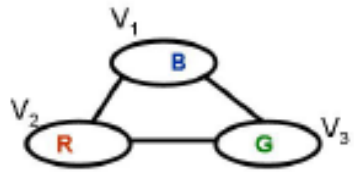
Arc examined	Value deleted
$V_1 - V_2$	none
$V_1 - V_3$	V_1(G)
$V_2 - V_3$	V_2(G)
$V_1 - V_2$	V_1(R)
$V_1 - V_3$	none



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



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

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 d and there are e 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.