**AI – CSP and backtracking**

A picture containing text, screenshot, font, document

Description automatically generatedSometimes constraint propagation is not enough and we have to actively search for a solution to the CSP

But, **how should we choose SELECT-UNASSIGNED-VARIABLE, and what order should its values be tried?**

The most trivial idea is the **MRV heuristic**, which uses a fail-first logic: it picks a variable that is most likely to cause a failure and it’s pruned, thereby selecting the variable with the minimum legal values; this is to cause less backtracking

To reduce the branching factor, we use the **max-degree heuristic**, which selects the variable that is involved in the largest number of constraints on other unassigned variables

To decide the order of examination of values, we use the **least-constraining-value heuristic,** which tries to leave the most flexibility for subsequent variable assignments (fail-last)

***Inference in backtracking – Forward checking and MAC***

We saw AC-3 as inference before searching, to infer during a search, we use **forward checking**

Whenever a variable X is assigned, the forward-checking process establishes arc consistency for it: **for each unassigned variable Y that is connected to X by a constraint, delete from Y’s domain any value that is inconsistent with the value chosen for X**

**MAC – Maintaining Arc Consistency** is an alternative to forward checking:

* After a variable X is assigned a value, we call AC-3 on the arcs (X, Y) ∀ Y that are neighboring X and unassigned
* AC-3 operates as usual, if any variable domain is reduced to {}, the call to AC-3 fails and we backtrack

***Backjumping vs Backtracking***

Backtracking simply *goes back* to the previous variable and tries a different value for it

A picture containing diagram, drawing, clipart

Description automatically generatedHowever, this is inconclusive in some cases, like the one below:

Restarting from T doesn’t make any sense, and we don’t have any legal value left for SA

In these cases, we use **backjumping**, which **backtracks to the most recent assignment in the conflict set** (here, V), the set of assignments that could be causing conflict with some variable (here, SA)

Forward checking and MAC are redundant when using backjumping; indeed, whenever forward checking based on an assignment X =x deletes a value from Y’s domain, it should add X = x to Y’s conflict set

If the last value is deleted from Y’s domain, then the assignments in the conflict set of Y are added to the conflict set of X, as we now know that X = x leads to a contradiction in Y, and a different assignment should be tried for X

***Conflict-directed backjumping (CBJ)***

CBJ is used whenever the cause of failure is a *set* of variables in the conflict set, rather than only one

Let Xj be the current variable, and let conf(Xj) be its conflict set

If every possible value for Xj fails, backjump to the most recent variable Xi in conf(Xj) and recompute the conflict set for Xi as follows:

**conf(Xi) ← conf(Xi)∪conf(Xj)− {Xi}**

Constraint learning is the idea of finding a minimum set of variables from the conflict set that causes the problem/inconsistency (i.e., finding a **no-good** that will add constraints to our CSP)