CSP and Back tracking

Artificial Intelligence

CSE3007

Introduction

- If we want to say a problem is solved using **constraint satisfaction problem**, or CSP when each variable that remains in the solution consists of a value that satisfies all the constraints on that particular variable.

- CSP search algorithms used to solve complex problems.

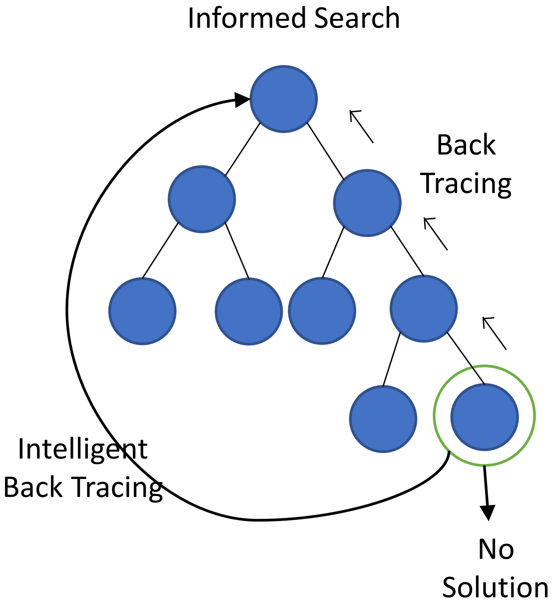
- The main idea is to eliminate large portions of the search space all at once by identifying variable/value combinations that violate the constraints.

Introduction to Constraint satisfaction problem (CSP)

- ❑ In Uniformed and informed search, we used state space to represent a problem. Ex: Graph, Trees

- ❑ CSP uses different representation.

- ❑ Examples: SUDOKU and MAP or graph Coloring.



Intro to CSP

- CSP consists of three components V, D, C.
- V is set of variables v1, v2,…..vn. i.e. V = {v1, v2,…. vn}
- D is set of domains: D = {D1, D2,…. Dn}.
- C is set of Constraints that specify allowable combination of values. (Scope, Relation).
- Where **Scope** is set of Variables that participate in Constraint and **Relation** define the values that variable can take.

Intro to CSP

- Suppose, v1 -> A(domain), v2 -> B.
- Then the constraint : values v1 and v2 should not be same which is represented as:

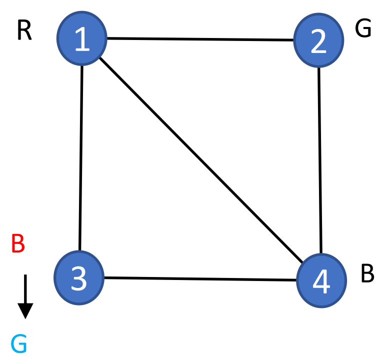
  C1 = ((v1, v2), (v1 != v2))

  - Example: A =(1, 2) B =(2, 4)

    Then C = ((v1, v2), (1, 2), (1, 4), (2, 4))

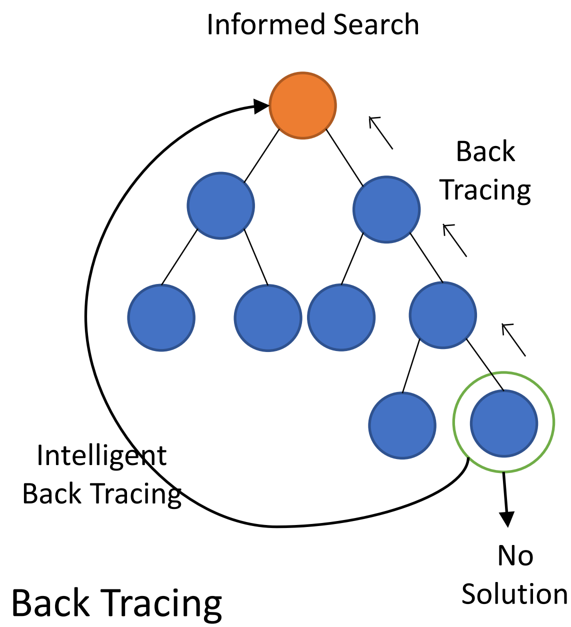
- Constraints are the set of rules which help to reduce the search space of the problem.
- Example: Sudoku, D = 1 to 9, C = No row, column or box should contain same number more than once.

Example 1: CSP map colouring problem:



$V = \{1, 2, 3, 4\}$
$D = \{\text{Red, Green , Blue}\}$
$C = \{1 \neq 2, 1 \neq 3, 1 \neq 4, 2 \neq 4, 3 \neq 4\}$

Contstraint: No neighbor nodes contain same color



Intelligent Back Tracing

	1	2	3	4
Initial Dom.	R, G, B	R, G, B	R, G, B	R, G, B
1 = R	R	GB	GB	GB
2 = G	R	G	GB	B
3 = B	R	G	B	B

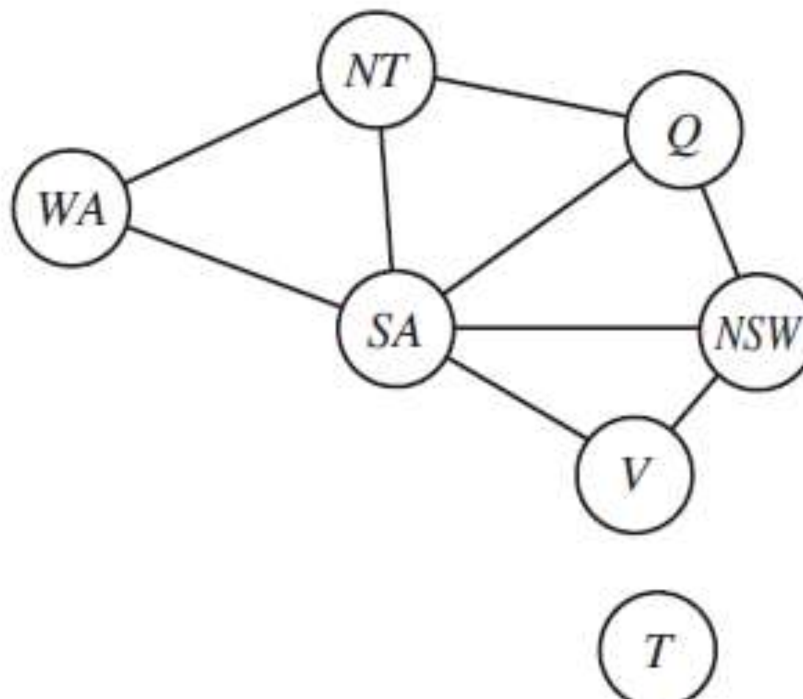
Empty
No solution

	1	2	3	4
Initial Dom.	R, G, B	R, G, B	R, G, B	R, G, B
1 = R	R	GB	GB	GB
2 = G	R	G	GB	B
3 = G	R	G	G	B

Example problem 2: Map coloring

- The goal is to assign colors to each region so that no neighboring regions have the same color.

(a)



(b)

- The variables to be the regions, X = {WA, NT, Q, NSW, V, SA, T} .
- The domain of each variable is the set Di = {red , green, blue}.

- The constraints require neighboring regions to have distinct colors. Since there are nine places where regions border, there are nine constraints:

C = {SA != WA, SA != NT, SA != Q, SA != NSW, SA != V, WA != NT, NT != Q, Q != NSW, NSW != V } .

- SA != WA can be fully enumerated in turn as

{(red , green), (red , blue), (green, red), (green, blue), (blue, red), (blue, green)} .

{SA $\neq$ WA, SA $\neq$ NT, SA $\neq$ Q, SA $\neq$ NSW, SA $\neq$ V, WA $\neq$ NT, NT $\neq$ Q, Q $\neq$ NSW, NSW $\neq$ V }

One Solution: {*WA* = *red*, *NT* = *green*, *Q* = *red*, *NSW* = *green*, *V* = *red*, *SA* = *blue*, *T* = *green*}

There are many possible solutions to this problem, such as {WA=red ,NT =green, Q=red , NSW =green, V =red , SA=blue, T =red }.

- CSPs yield a natural representation for a wide variety of problems.
- CSP solvers can be faster than state-space searchers because the CSP solver can quickly eliminate large portion of the search space.
- Many problems that are intractable for regular state-space search can be solved quickly when formulated as a CSP

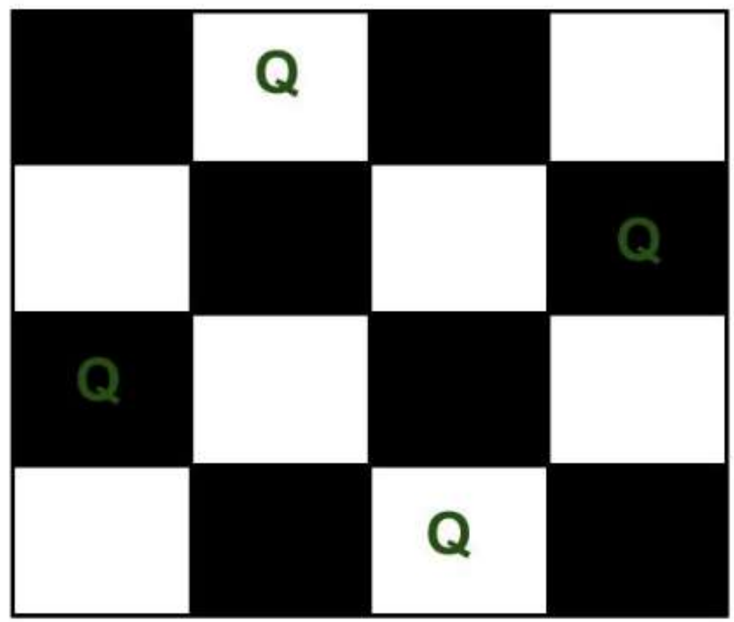
Backtracking search

Backtracking search

- Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time.
- For example, consider the SudoKo solving Problem, we try filling digits one by one. Whenever we find that current digit cannot lead to a solution, we remove it (backtrack) and try next digit.

N-queen problem

- The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other and should not be on same row and column.



- The idea is to place queens one by one in different columns, starting from the leftmost column.
- When we place a queen in a column, we check for clashes with already placed queens.
- In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution.
- If we do not find such a row due to clashes then we backtrack and return false.

4-Q

1st - 16 option