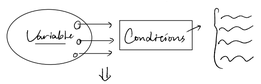
CSP: Constraint Satisfaction Problem



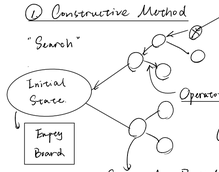
A "Solution": an assignment (variables ⟷ satisfies ⟷ conditions)

8-Queens puzzle

no two Queens in { Same row / Same col / Same diagonal } → translate to a CSP Problem

- Variable: position of queen in Col 1 to Col 8
  - Denoted: $q_1, q_2, q_3 \cdots q_8$.
- Domain: $q_i \in \{1, \ldots 8\}$.
- Constraint:
  1. $q_i - q_j \neq 0$ for $i \neq j$
  2. $|q_i - q_{i+k}| \neq k$ for $k = 1, 2, \ldots, 7$; $i = 1, \ldots 8-k$
     - $|q_1 - q_{1+1}| \neq 1$ for $i = 1, 2, \ldots, 7$
     - $|q_1 - q_{1+2}| \neq 2$ for $i = 2, 3, \cdots, 7$
     - ⋮

① Constructive Method — "Search"



Specify all "Conditions"

Operator: Assign a new value to a variable

Case: 0 ⇒ "depth-first" Search

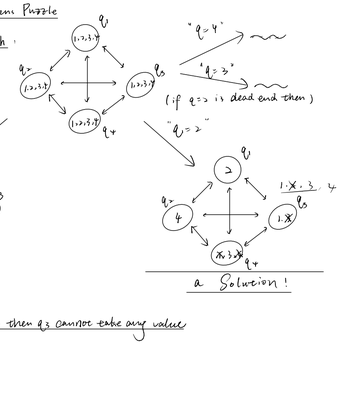
States: Any Partial Assignment

Search Space / Time is too Large!

② Constraint Propagation ⇒ to Prune the Search Space

Example. 4-Queens Puzzle

Constraint Graph:



Satisfiability (SAT)

Clauses: $C = l_1 \vee l_2 \cdots \vee l_n$ — "literal"

CNF: Conjunctive Normal Form

$C_1 \wedge C_2 \wedge C_3 \cdots$

SAT: decide if a CNF is satisfiable

3SAT: each clause have no more than 3 literal.

DPLL Procedure

α: a set of clauses $C_1 \wedge C_2 \wedge C_3 \cdots$

l: a literal

```
α(l):
    Let l be true.
    for C ∈ α.
        if C mention l:
            eliminate C in α
        else if C mention ¬l:
            delete ¬l from C.
```

```
Procedure DPLL(α):
    if α is empty return yes.   # no constraints ⇒ always satisfiable.
    else if there is a empty clause in α: return no.   # if we have delete all literal from a clause while still not satisfiable, then the set is unsatisfiable.
    else if pure literal C in α. return DPLL(α(C))
    else if a unit clause {l} in α. then return DPLL(α(l))
    else:
        Select a variable p in α.
        if DPLL(α(p)) return yes. then return yes.
        else return DPLL(α(¬p))
```

actually is trying both { DPLL(α(p)) / DPLL(α(¬p)) }

Modern DPLL

Heuristic Repair

"Sound & complete procedure"

Complete methods: e.g. DPLL: if satisfiable then "yes" if unsatisfiable then "no" but slow

GSAT: incomplete, i.e. Output: no (might) Satisfiable/unsatisfiable.

Beat Base — Output: yes (must) satisfiable. (Sound Procedure)

```
Procedure GSAT(α, max-restore: mr, max-climb: mc)
    for i=1 to mr:
        get a random generated truth assignment v.
        for j=1 to mc:
            if v satisfies α. then return yes,
            else (or v be a random choice of one best Successors of v
    return failure
```

- Constructive method: Start with empty assignment, build up solution gradually (assigning values 1 by 1)
- Heuristic Repair: Start with a proposed solution (randomly generated), repair it until satisfiable.

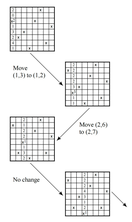
"Min-conflicts repair":

Randomly initialize a state then do:

① Select a "conflict" col.

② Move queens in this col to the row with smallest number of conflicts.

③ repeat until no conflict



Clustering as CSP

Unsupervised Learning: no "labeled" Data.

k-means Clustering as CSP

- Variables: $c_1, c_2 \cdots c_n$ (c_i: cluster which x_i (i-th data point) belongs to.) ; $\mu_1, \mu_2, \ldots, \mu_k$ ← centroids → 1
- Domains: $c_i \in \{1, \ldots, k\}$. (i.e. belongs to which cluster) ; $\mu \in$ feature space. (e.g. 2 features ⇒ $\mathbb{R}^2$)
- Constraint: minimize: $\sum_{i=1}^{n} \| \phi(x_i) - \mu_{c_i} \|^2$.

Insights: if we know one of μ, c. We can easily compute the another one

$$\begin{cases} c_i = \operatorname{argmin}_i (\phi(x_i) - \mu_i)^2 & \text{if known } \mu_i \\ \mu_k = \operatorname{argmin}_\alpha \sum_{c_i = k} (\phi(x_i) - \alpha)^2 \end{cases}$$

"CSP" Solution

① Initialize $\mu_1 \cdots \mu_k$.

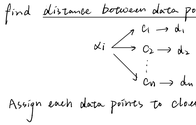
② Compute best assignment $c_1 \cdots c_k$ for the given μ

③ Compute best centroids $\mu_1 \cdots \mu_k$ for the given c.

iterate ② ③ until termination condition → Output

K-means Clustering:

1. Choose K random data points to be initial centroids
2. find distance between data points with each Centroids.



3. Assign each data points to closet centroids.
4. Re-compute Centroids
5. if convergence criterion not MET. repeat 2 ~ 4

Hill-Climbing Search

"Search as Function Maximization"

e.g. 8-queens problem: goal: find a state which none of 8 queens are attacked ("Constraints").

Value(x) = $\frac{1}{e^{h(n)}}$, where h(n) = # queens attacked

"8-queens problem ⇔ Maximize Value(x)"!

Hill-Climbing Search

"At each step, move to the next state with highest value"

No "Search tree", No "Backtrack"

```
Hill-Climbing (Problem)
    current ← make-nodes (initial state (problem))
    loop do
        next ← a highest-value successor of current
        if value[next] < value[current]
            return current.
        current ← next
```

May return local maximum!

"爬山演算法"

Simulated Annealing

When choosing successor, instead of "Choosing the best" do:

① randomly select a successor

② if Value(successor) > Value(current):
Accept!

else:
- Accept with probability p (p becomes smaller as the algorithm progresses)
- Remain "current" with probability (1-p)

Sound Procedure vs Complete Procedure

Sound	Yes	No
Satisfiable	✓	○
unsatisfiable	X	○

Complete	Yes	No
Satisfiable	○	X
unsatisfiable	○	✓