

* LOCAL SEARCH

① Neighbor relation:

Current solution $S \in C$ (Set of all possible solutions)

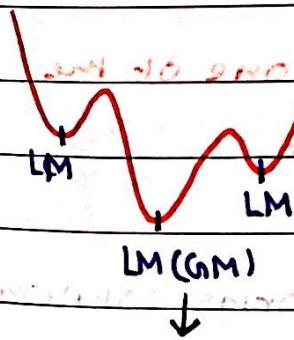
In a given step it chooses a neighbor S' of S , then S' is current solution.

(S' can be obtained from S by a small modification)

→ Feasibility graph:

Graph on set of all possible solutions with edges joining neighboring pairs of solutions.

② Analogy to Potential Energy



* At every step we tend to move towards a good solution i.e. $\downarrow PE$

* When we are at a LOCAL minima the algorithm terminates.

Global Minima

* Local minima may not be the global minima.

① Vertex Cover:

Objective function:

Min. no of vertices such that all edges are covered

Neighbor relation:

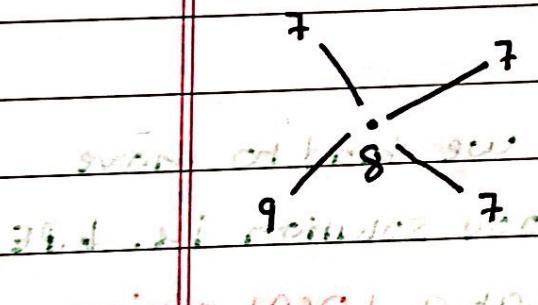
v_1 is neighbor of v_2 if diff of only 1 vertex is there in 2 solutions

$\{1, 2, 5\}$

$\{1, 2, 4, 5\}$

$\{1, 5\}$

* Here the cost function is the no. of vertices in the set at each vertex & we want to minimize it.



* We move towards one of the vertex with $|V| = 7$

* Each vertex has at most n neighboring solutions where n is the no. of vertices in G .

* ALGO: Initialise set $res = \{\}$ choose any vertex of the feasibility graph, consider set $S = \{ \text{all vertices of graph} \}$

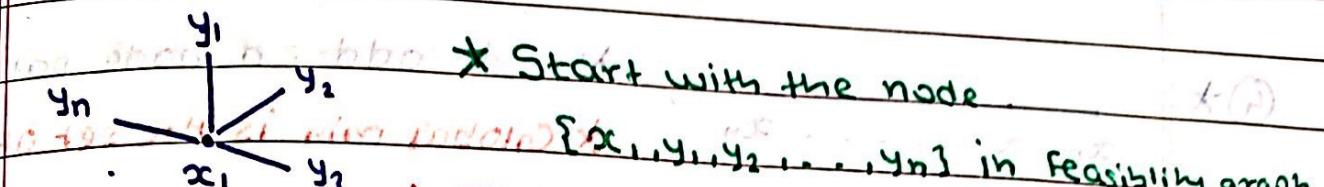
→ While S is not empty

↓
vertex of feasibility graph

* Choose any vertex $v \in S$, add all u to res

$\exists (u, v) \in G$ & remove v & u & $(u, v) \in G$ from S

①*



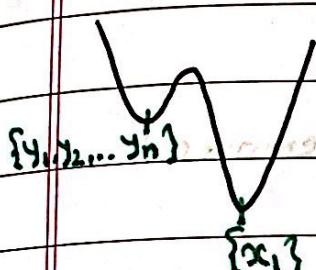
\star Start with the node.

\star If we keep on removing y_1 in feasible graph $\{x_1, y_1, y_2, \dots, y_n\}$

STAR

\star If we keep on removing y_1 in feasible graph $\{x_1, y_1, y_2, \dots, y_n\}$ in feasible graph

\star Remove x_1 we have $\{y_1, y_2, \dots, y_n\}$ as local minimum.



②*

 x_1 x_2

\star n: vertices, 0: edges

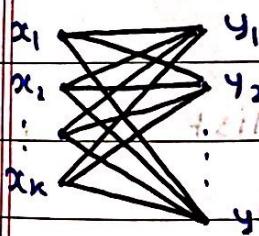
x_1 x_2 x_3

\star $\{x_1, x_2, \dots, x_n\}$

\star Φ is the only local minima

ZERO-EDGE

③*



\star Complete bipartite graph with $l > k$

$\rightarrow \{x_1, x_2, \dots, x_k\}$ are the two

$\{y_1, y_2, \dots, y_l\}$ are the two

BIPARTITE

local minimas with the former

being deeper than the latter (# no. of vertices)

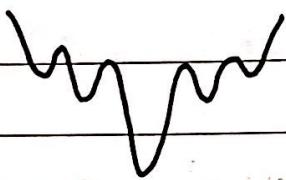
(# no. of vertices)

(4)*

* n is odd, n node path

* Global min is the set of all even vertices

- But even if any one even vertex is removed we can't reach the global min.
- $\{x_2, x_3, x_5, x_6, x_8, x_9, \dots\}$ is also local min. ($1 \bmod 3$ indices removed)



* At every step no. of vertices in the set of current vertex of Feasibility graph \downarrow by 1

→ ∴ there are almost $O(n)$ iterations

(Achieved in case of zero-edge graph)

* Time complexity : $O(V+E)$ (adjacency list format)

Feasible solution initially $\{x_1, x_2, \dots, x_n\}$

* Choose Remove a vertex which is not marked.

Check it's list if any element already removed

We can't remove it: (if so mark it else remove it)

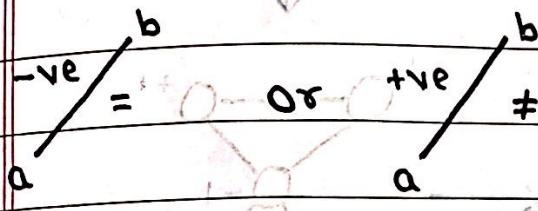
* Hence list of each vertex is traversed only once
 $\therefore T.C. O(V+E)$

② Hopfield Neural Network

Undirected weighted graph

each node assigned +1 or -1

* Good edge



i.e. edge uv is good $\Leftrightarrow w_{uv} s_u s_v < 0$

w_{uv} : weight of the edge

s_i : sign of the vertex

* Happy Node



$$\sum_{\text{good}} |w_{ei}| \geq \sum_{\text{bad}} |w_{ei}| \Rightarrow \text{Total } \Sigma |w_{ei}| > 0$$

* Stable config: all nodes are happy.

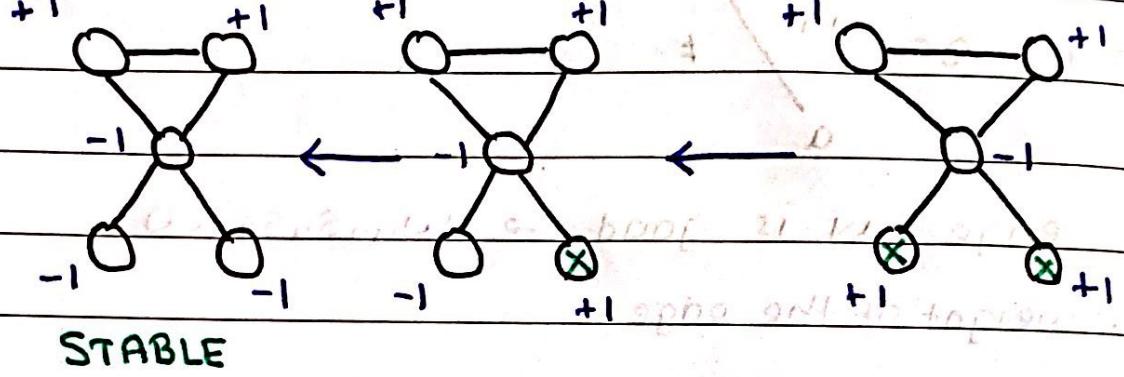
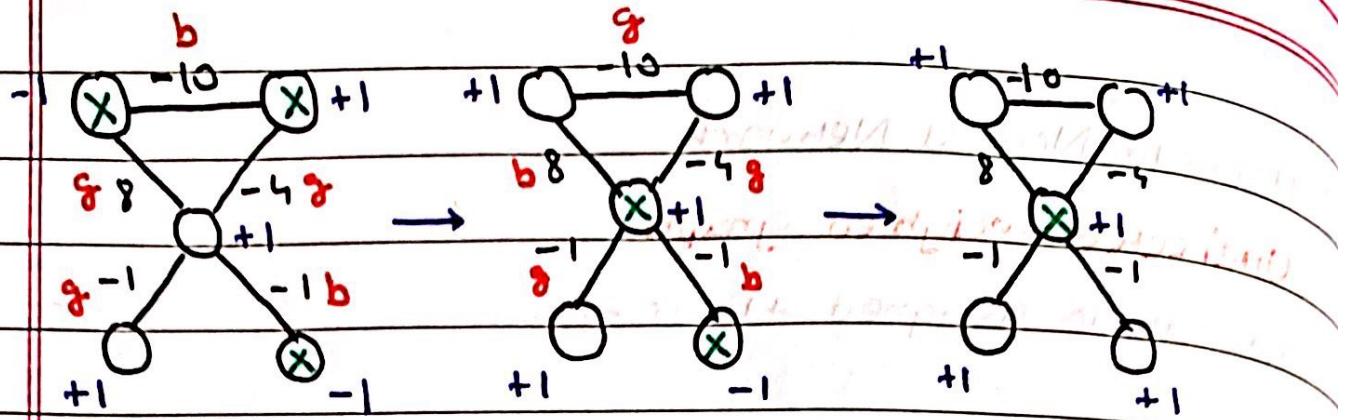
While the current config not stable

There must be an unsatisfied node

Chose that node u

Flip the state (sign) of u

End while.

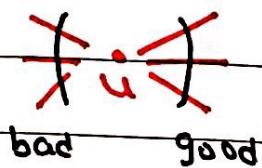


* $W = \sum_e l_{we}$

* $\sum_{\text{good}} l_{we} \leq \sum_{\text{bad}} l_{we} = W < \text{fault}$

→ At every stage (flip) $\sum_{\text{good}} l_{we}$ increases

* Let node u be unhappy



$|l_{wug}| < |l_{wub}| \dots \text{before}$

after flip

$|l_{wug}| > |l_{wub}| \dots \text{after}$

$\because \text{good} \leftrightarrow \text{bad} \therefore \text{sum } 1$

* T.C. $O(CW^{\text{poly}(m,n)})$

\downarrow
 \downarrow
vertices
edges

$2^{\log w} = 2^{\text{no. of bits to encode input } w}$

Time complexity = $O(2^{\log w} \cdot \text{Time for dynamic programming})$

→ Such algo "Pseudo Polynomial Time"

(C: max no. of iterations of the while loop is w)

Time complexity = $O(w \cdot \text{Time for dynamic programming})$

No. of P matrix operations = $O(w^2)$

No. of P matrix multiplications = $O(w^3)$

No. of P matrix additions = $O(w^2)$

No. of P matrix assignments = $O(w^2)$

No. of P matrix comparisons = $O(w^2)$

No. of P matrix assignments = $O(w^2)$

No. of P matrix comparisons = $O(w^2)$

No. of P matrix assignments = $O(w^2)$

No. of P matrix comparisons = $O(w^2)$

No. of P matrix assignments = $O(w^2)$

No. of P matrix comparisons = $O(w^2)$

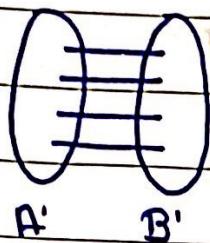
No. of P matrix assignments = $O(w^2)$

No. of P matrix comparisons = $O(w^2)$

No. of P matrix assignments = $O(w^2)$

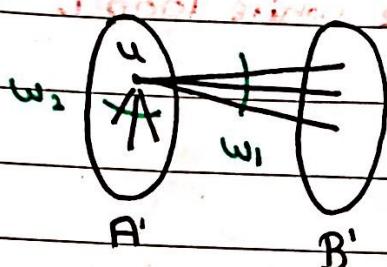
No. of P matrix comparisons = $O(w^2)$

③ Max Cut



$$\text{maximise } W(A', B') = \sum_{\substack{u \in A' \\ v \in B'}} w_{u,v}$$

(Cut weight)



If $w_2 > w_1$,

Move u from A' to B'

- * Neighbor Relation: Change of 1 vertex in partition.
- At every iteration $W(A', B') \uparrow$

- * At local optima:

$$\left. \begin{array}{l} \forall u \in A \quad \sum_{v \in A} w_{u,v} \leq \sum_{v \in B} w_{u,v} \\ \forall u \in B \quad \sum_{v \in B} w_{u,v} \leq \sum_{v \in A} w_{u,v} \end{array} \right\} n \text{ equations}$$

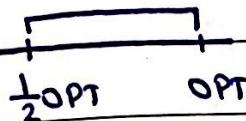
$$\Rightarrow 2 \sum_{u, v \in A} w_{u,v} + 2 \sum_{u, v \in B} w_{u,v} \leq 2W(A, B)$$

$$\Rightarrow W \leq 2W(A, B)$$

↓
Total weight

$$\text{But } W(A, B) \leq OPT \leq W$$

$$\frac{1}{2} \text{OPT} \leq W(A, B) \leq \text{OPT}$$



∴ our algo outputs a soln between
 $\frac{1}{2} \text{OPT}$ and OPT

$$(0.01)(\epsilon^2 + \epsilon) = \text{small}$$

* This is an approximation algo.

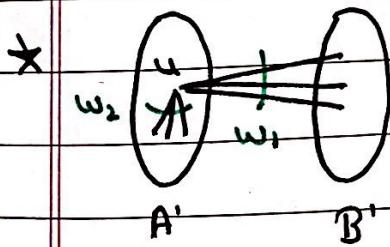
Time: $O(W \text{poly}(cm, n))$: Pseudo Polynomial Time algo

* n : input given

$O(n)$: Pseudo poly

$O(\log n)$: Weakly poly (# gcd)

$O(1)$: Strongly poly



* move u from A' to B'

$$\text{if } w_2 > w_1 (1 + \epsilon/n)^{\epsilon^2 + \epsilon}$$

* At local optima:

$$\forall u \in A \quad \sum_{v \in B} w(u, v) \leq \sum_{v \in B} w(u, v) + \frac{\epsilon}{n} W(A, B)$$

$$\forall u \in B \quad \sum_{v \in A} w(u, v) \leq \sum_{v \in A} w(u, v) + \frac{\epsilon}{n} W(A, B)$$

Adding n equations: $2 \sum_{U,V \in A} w(U,V) + 2 \sum_{U,V \in B} w(U,V) \leq 2w(A,B) + \epsilon w(A,B)$

$$\Rightarrow W \leq 2w(A,B) + \frac{\epsilon}{2} w(A,B)$$

$$\Rightarrow W \leq (2 + \epsilon/2) w(A,B)$$

$$\therefore \underline{OPT} \leq w(A,B) \leq \overline{OPT}$$

$$(2 + \epsilon/2)$$

where $\epsilon > 0$

Initially $w(A',B')$

$$(1 + \epsilon/n) w(A',B')$$

$$(1 + \epsilon/n)^2 w(A',B')$$

:

:

$$(1 + \epsilon/n)^{n/\epsilon} w(A',B')$$

$$\because (1 + 1/b)^x \geq 2 \text{ for } x \geq 1$$

$$\therefore (1 + \epsilon/n)^{n/\epsilon} \geq 2$$

\therefore Objective function \uparrow by at least 2 every $n/6$ flip or transition.

$$\therefore w(A',B') \leq w$$

$$\therefore \text{T.C. is } O\left(\frac{n}{\epsilon} (\log w) \text{poly}(nm)\right)$$

* Polynomial Time approximation Scheme (PTAS)

$$\frac{\text{OPT}}{(2 + \frac{\epsilon}{2})} \quad \text{OPT}$$

* As $\epsilon \uparrow$, T.C. \downarrow , approx range \uparrow