* LOCAL SEARCH

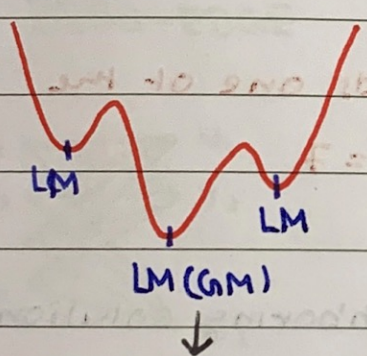
① Neighbor relation:

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)

→ Feasiblity graph:

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

② Analogy to Potential Energy



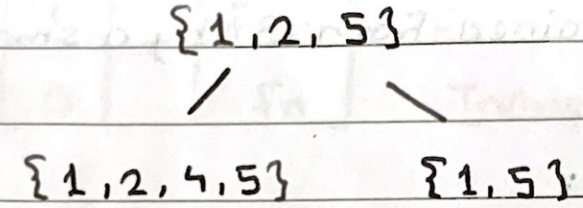
* At every step we tend to move towards a good solution i.e. ↓ PE
* When we are at a LOCAL minima the algorithm terminates.

* Local minima may not be the global minima.

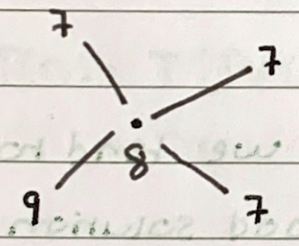
① Vertex Cover:

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



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



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

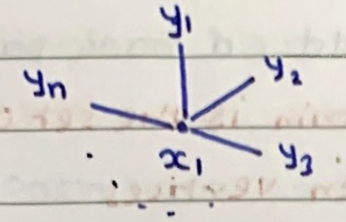
* Each vertex has at most n neighboring solutions where n is the no. of vertices in G.

* ALGO: Initialise set res = { } choose any vertex of the feasiblity graph, consider set S = { all vertices of graph }
  ↓ Vertex of feasiblity graph

→ While S is not empty

  * Choose any vertex v ∈ S, add all u to res ∃ (u,v) ∈ G & remove v & u ∃ (u,v) ∈ G from S

① ★

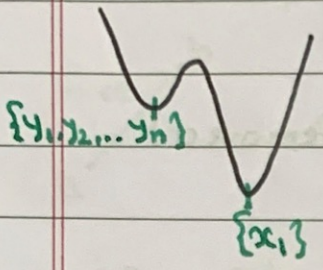


STAR

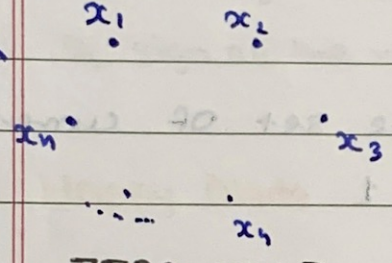
★ Start with the node $\{x_1, y_1, y_2, \ldots, y_n\}$ in feasibility graph

★ If we keep on removing y_i's $\{x_1\}$ as our local optimum.

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



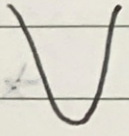
② ★



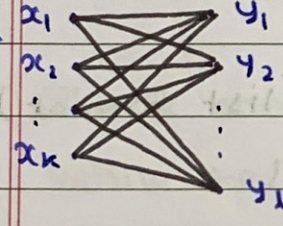
ZERO-EDGE

★ n: vertices , O: edges $\{x_1, x_2, \ldots, x_n\}$

★ ϕ is the only local minima



③ ★



BIPARTITE

★ Complete bipartite graph with $l > k$

→ $\{x_1, x_2, \ldots, x_k\}$ $\{y_1, y_2, \ldots, y_l\}$ are the two local minimas with the former being deeper than the other.

(# no. of vertices)

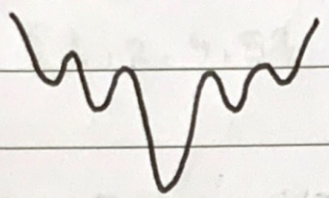
④ ★ 

★ 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 cannot reach the global min.

→ $\{x_2, x_3, x_5, x_6, x_8, x_9, \ldots\}$ is also local min. (1 mod 3 indices removed)



★ At every step no. of vertices in the set of current vertex of Feasibility Graph ↓ 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 $\{x_1, x_2, \ldots, x_n\}$

★ ~~Remove~~ Choose a vertex which is not marked.

Check it's list if any element already removed we cant remove it. (if so mark it else remove it)

★ Hence list of each vertex is traversed only once

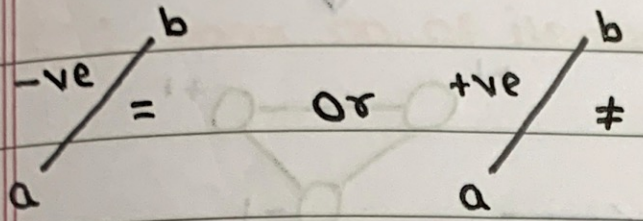
∴ T.C. O(V+E)

② Hopfield Neural Network

Undirected weighted graph

each node assigned +1 or -1

* Good edge

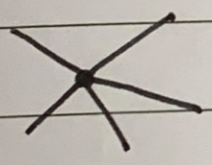


i.e. edge uv is good $\Leftrightarrow$ $w_{uv} s_u s_v < 0$

w_{uv}: weight of the edge

s_i: sign of the vertex

* Happy Node



$$\sum_{\text{good}} |w_e| \geq \sum_{\text{bad}} |w_e|$$

* Stable config: all nodes are happy.

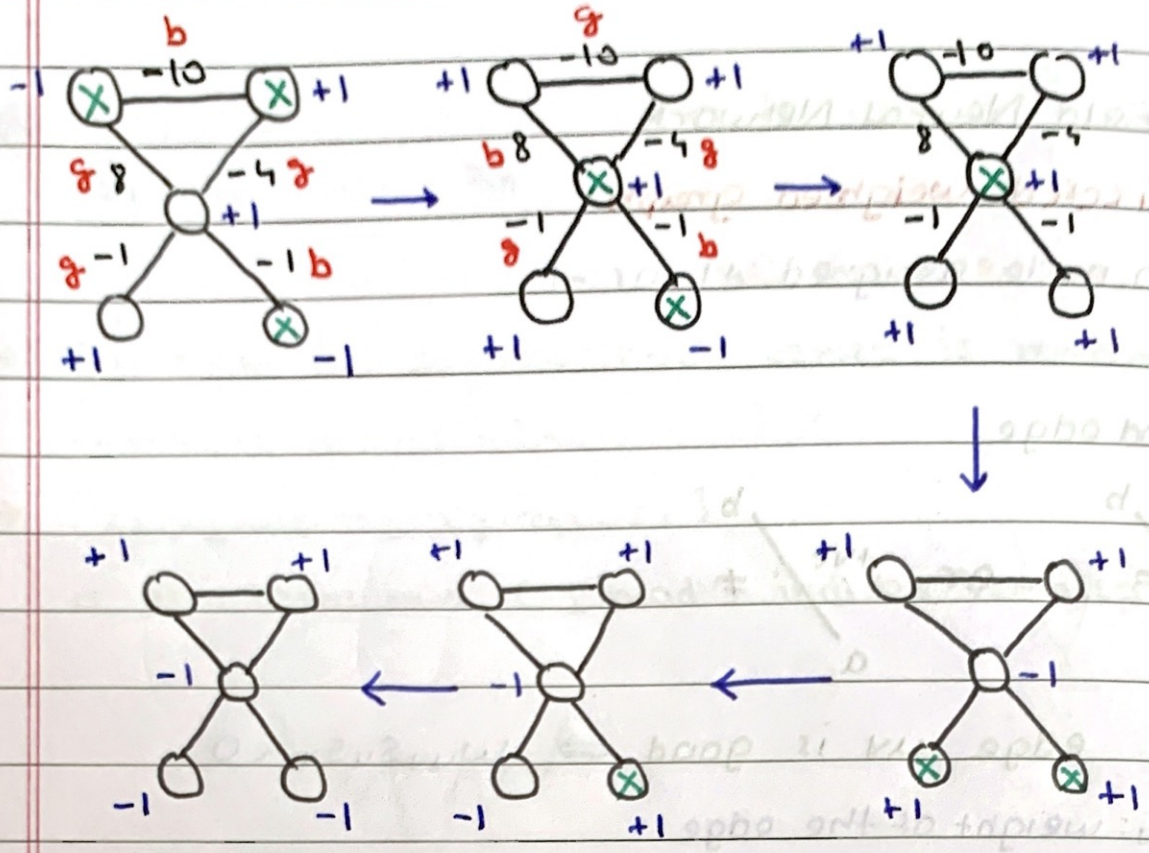
While the current config not stable

There must be an unsatisfied node

Chose that node u

Flip the state (sign) at u

End while.



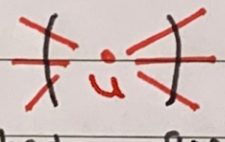
STABLE

* $W = \sum_e |w_e|$

* $\sum_{\text{good } e} |w_e| \leq \sum_e |w_e| = W$

→ At every stage (flip) $\sum_{\text{good } e} |w_e|$ increases

* Let node u be unhappy



$|w_{ug}| < |w_{ub}|$ ... before

after flip

$|w_{ug}| > |w_{ub}|$ ... after

∵ good ↔ bad ∴ sum ↑

classmate
Date
Page

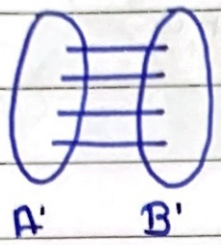
* T.C. $O(W \cdot \text{poly}(m, n))$ (vertices: n, edges: m)

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

→ Such algo "Pseudo Polynomial Time"

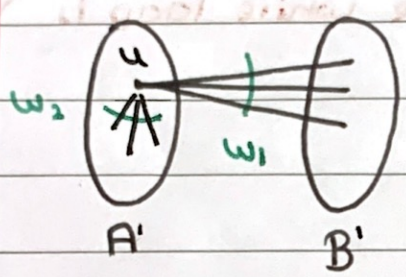
(∵ max no. of iterations of the while loop is W)

③ Max Cut



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

(Cut weight)



If $w_2 > w_1$

Move u from A' to B'

* Neighbor Relation: Change of 1 vertex in partition.

→ At every iteration $W(A',B')$ ↑

* At local optima:

$$\forall u \in A \quad \sum_{v \in A} w(u,v) \leq \sum_{v \in B} w(u,v)$$

$$\forall u \in B \quad \sum_{v \in B} w(u,v) \leq \sum_{v \in A} w(u,v)$$

} n equations

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

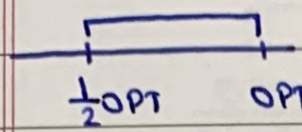
$$\Rightarrow W \leq 2W(A,B)$$

↓

Total weight

But $W(A,B) \leq OPT \leq W$

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



$\therefore$ our algo outputs a soln between $1/2$ OPT & OPT

* This is an approximation algo.

Time: $O(W \cdot poly(m,n))$: Psuedo Polynomial Time algo

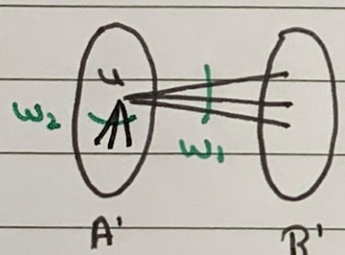
* n : input given

O(n): Psuedo poly

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

O(1): Strongly poly

*



* move u from A' to B'
if $w_2 > w_1(1 + \epsilon/n)$

* At local optima:

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

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

Adding n equations

$$\therefore 2\sum_{u,v\in A} w(u,v) + 2\sum_{u,v\in B} w(u,v) \le 2w(A,B) + \epsilon(w(A,B))$$

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

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

$$\therefore \frac{OPT}{(2+\epsilon/2)} \le w(A,B) \le OPT$$

where $\epsilon > 0$

Initially $w(A',B')$

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

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

⋮

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

$\because (1+1/x)^x \ge 2$ for $x \ge 1$

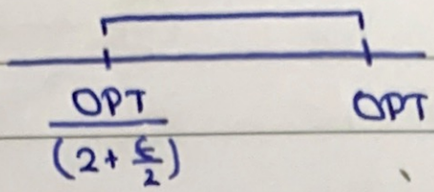
$\therefore (1+\epsilon/n)^{n/\epsilon} \ge 2$

$\therefore$ Objective function ↑ by atleast 2 every n/ϵ flips or transitions.

$\because w(A',B') \le W$

$\therefore$ T.C. is $O\left(\frac{n}{\epsilon}(\log W)\,\text{poly}(n,m)\right)$ $\therefore$ Weakly poly.

★ Polynomial Time approximation Scheme (PTAS)



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