

TABU SEARCH

Glover: Tabu Search: A Tutorial
 Glover, Taillard, de Werra: A User's Guide to Tabu Search
 Glover: Tabu Search, Part I ORSA JOC 1(3), 1989, pp 190-206

Like simulated annealing, this method is designed to not get trapped in local minima. In addition, it is also designed to avoid cycling, by making certain moves "tabu", i.e., forbidden.

Want to solve the problem

$$\begin{aligned} \min \quad & c(x), \\ \text{st.} \quad & x \in X \subseteq \mathbb{R}^n. \end{aligned}$$

Eg: X could be the set of all binary points satisfying some linear constraints.
 $c(x)$ could be a linear objective function, ~~but it might be~~

Have current point x . Move to a neighbour ~~x'~~ x' .

Let $N(x)$ be the set of neighbours of x .

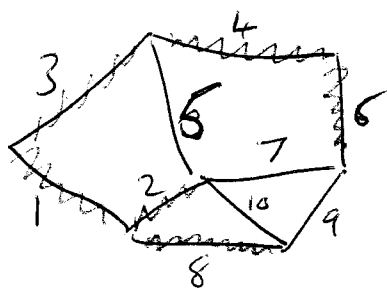
Let T be the set of ~~for~~ tabu neighbours of x :
 we are not allowed to move to these neighbours from x .
~~##~~ Moving ~~into~~ to these neighbours may lead to cycling.

Simple tabu search:

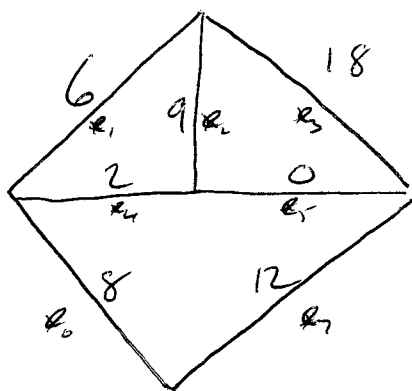
The neighbour x' to which we move is the best point in $N(x)$ which is not tabu.

Example: Find a minimum ^{weight} spanning tree in a graph, subject to some side constraints.

(Without the side constraints, we can use a greedy algorithm to solve this:
 → pick the edge with smallest weight which does not create a cycle
 → repeat until no edges left.



This is an example of a matroid problem. Matroid problems can always be solved by greedy algorithms.



Constraints:
 $x_1 + x_2 + x_6 \leq 1$ (at most one of the corresponding edges)
 $x_1 \leq x_3$ (can only include e_1 if also include e_3 .)

Use penalty function:

Cost of solution = cost of tree + ~~50~~ penalty of 50 for each violated constraint.

Neighbours:

Two trees are neighbours if we can get from one to the other by deleting one edge and adding one edge.

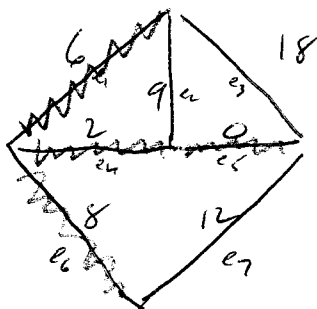


TABU SEARCH EXAMPLE

Spanning tree with side constraints

Tabu restriction: Forbid dropping the most recently added edges.

Initial solution:



$$\text{Cost} = (6+2+0+8) + 50 + 50 = 116$$

If add e_2 : dropping e_1 gives cost
 Create cycle e_1, e_2, e_4
 $9+2+0+8 + 50 = 69$

dropping e_4 gives cost
 $6+9+0+8 + 50 + 50 = 123$

~~dropping e_5 gives cost
 $6+9+2+8 + 50 + 50 = 125$~~

Side constraints:

$$x_1 \leq x_3$$

$$x_1 + x_2 + x_6 \leq 1$$

If add e_3 :
 Create cycle e_1, e_3, e_4, e_5
 dropping e_1 gives cost
 $18+2+0+8 = 28$

dropping e_4 gives cost
 $18+6+0+8 + 50 = 82$

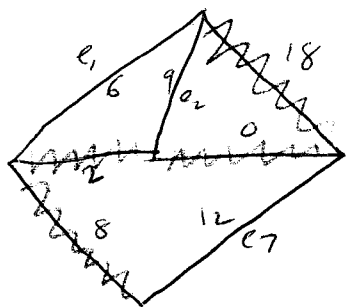
dropping e_5 gives cost
 $18+2+6+8 + 50 = 84$

If add e_7 :
 Create cycle e_4, e_5, e_6, e_7

~~dropping e_4 gives cost~~
 All choices have cost ≥ 50 since $x_1 \leq x_3$ violated

So best move is to add e_3 and drop e_1

$$\text{Cost} = 28$$



If add e_1 , cost ≥ 50 since violated $x_1 + x_2 + x_6 \leq 1$
 (NB: can not drop e_3)

If add e_2 , cost ≥ 50 since violated $x_1 + x_2 + x_6 \leq 1$
 (NB: can not drop e_3)

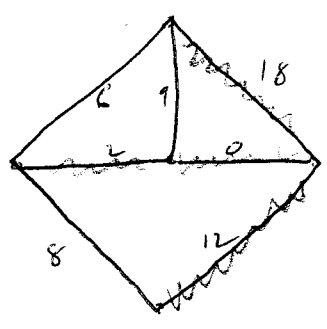
If add e_7 :
 Create cycle e_4, e_5, e_6, e_7

Drop e_4 : Cost = $18+0+12+8 = 38$

Drop e_5 : Cost = $18+2+12+8 = 40$

Drop e_6 : Cost = $18+2+0+12 = 32$

So best move is to add e_7 and drop e_6 . Note that this hurts the objective.



Cost = 32.

If add e_1 :
create cycle $e_1 e_3 e_4 e_5$

Drop e_3 : Cost = $6+2+0+12+50 = 70$ TABU
 Drop e_4 : Cost = $6+18+0+12 = 36$
 Drop e_5 : Cost = $6+18+2+12 = 38$

If add e_2 :
create cycle $e_2 e_3 e_4$

Drop e_3 : Cost = $9+2+0+12 = 23$ TABU
 Drop e_4 : Cost = $9+18+2+12 = 41$

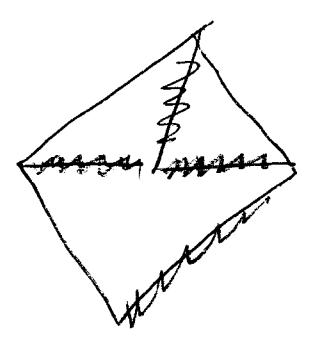
If add e_6 :
create cycle $e_6 e_5 e_4 e_7$

Drop e_4 : Cost = $18+0+8+12 = 38$
 Drop e_5 : Cost = $18+2+8+12 = 40$
 Drop e_7 : Cost = $18+2+8+0 = 28$ TABU

Notice that if we add e_6 , the best move is to drop e_7 , which returns our previous solution, so this is TABU.

The best point in the neighbourhood is to add e_2 and drop e_3 , but this is TABU. However, the objective function value in this case is 23, which is less than the best value seen so far, so we make the move anyway.
 This is an example of an ASPIRATIONAL CRITERION:

New solution:



Value = 23.

This is optimal.

Aspiration criteria:

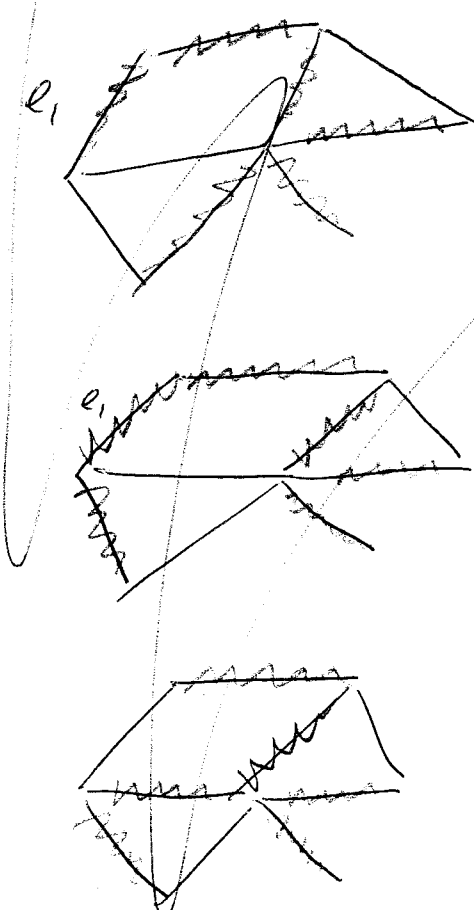
Simplest: If a tabu move results in a better solution than we've seen to date, make it.

More complicated: If we get to the current solution from a point worse than the tabu neighbour, move to the neighbour anyway.

Even more complicated: If every time we get to be as good as we are now, we came from somewhere worse than the tabu neighbour, move to the neighbour.

Eg: Say edge e_1 is tabu: can not drop it. We know a soln of value 20

See exampl
on next page,
755 a.



Value 24
Got by adding e_1

Value 26

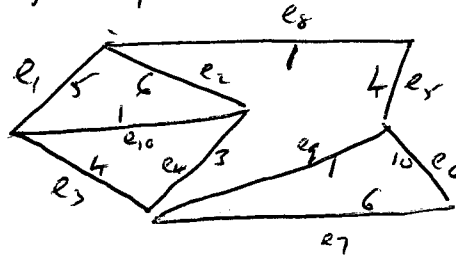
Value 22

Make this move because it gives a better solution than the first picture.

An example of using an aspiration criterion in TABU SEARCH.

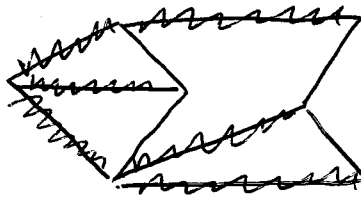
TABU to drop an edge for k iterations.

MST with side constraints:

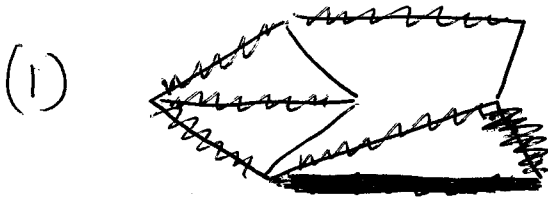


$x_1 + x_5 \leq 1$
 $x_3 + x_5 \leq 1$
 $x_4 \leq x_2$
 Say: penalty of 50 for violating constraint.

Say we earlier found the feasible solution with value 18:



We've moved away from this solution. Our current point is:



Value = 22. Say e_6 is TABU.

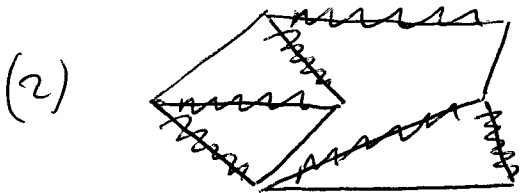
Possible moves:

Add e_5 : will violate $x_1 + x_5 \leq 1$ or $x_3 + x_5 \leq 1$, so cost ≈ 50 .

Add e_4 : violate $x_4 \leq x_2$, so cost ≈ 50 .

Add e_7 : remove either e_6 (TABU) or e_9 with cost 5.

BEST → Add e_2 : remove e_1 with cost 1



Value = 23. e_6, e_2 TABU.

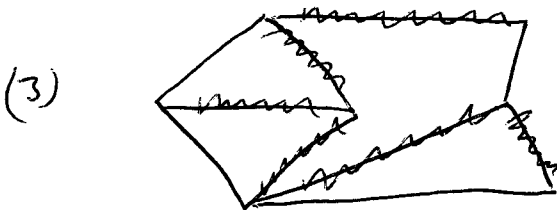
Possible moves:

Add e_1 : can't remove e_2 (TABU), so remove e_{10} : cost = 4

BEST → Add e_4 : Remove e_3 : BENEFIT = 1

Add e_5 : Remove e_3 , net cost = 0

Add e_7 : Can't remove e_6 (TABU), removing e_9 costs 5



Value = 22. e_6, e_2, e_7 TABU.

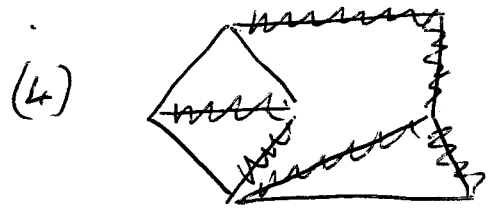
Add e_1 or e_3 : either TABU or costs.

Add e_7 : removing e_6 only brings soln to 18, so don't override TABU.

→ Add e_5 : value = 20 if drop e_2 — better than in (2), immediately before we added e_2

So override TABU restriction because meet aspiration criterion.

(over.)

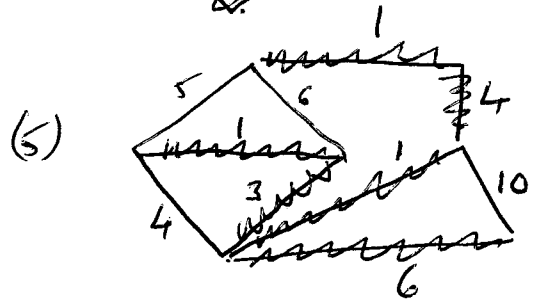
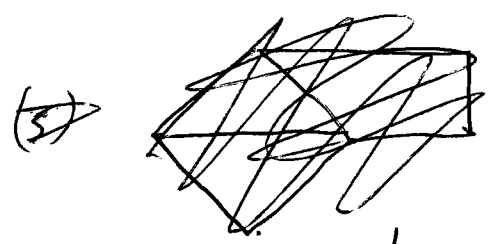


Value = 20. e_6, e_2, e_7, e_5

If add e_7 :

If drop e_6 , value drops to 16 —
better than best known solution.

So: override TABU restriction on e_6 because we meet aspiration criterion.



Value = 16

Optimal.

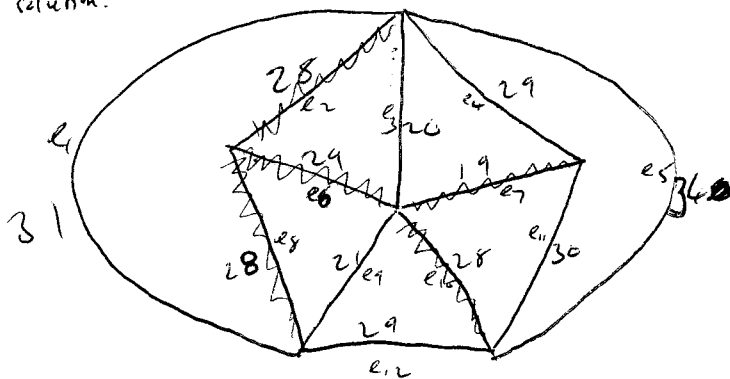
(Although we can't verify the optimality from the tabu search procedure.)

Diversification Techniques

Tabu restrictions are a short-term memory structure. It is also useful to use longer term memory to push the process into areas it has not yet visited.

Eg: Maximum Spanning Tree with constraints:

Initial solution:



Constraints:

$$x_9 \leq x_7, \quad \cancel{x_5 + x_7 \leq 2x_3}$$

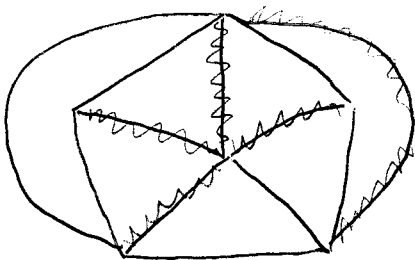
$$x_3 \leq x_5, \quad \cancel{x_7 \leq x_3}$$

$$\text{Value} = \cancel{135} - \cancel{215} = 132$$

penalty of 50 for each violated constraint.

Tabu restriction: added edges can not leave for two iterations.

Optimal solution:



$$\text{Value} = 123.$$

But tabu search never adds in edge e_5 , because that increases the cost too much, i.e., more than adding some other edge in a non-tabu way. Also, edge e_3 is never added because we will always violate the constraint $x_3 \leq x_5$, until e_5 is added.

We could put a tabu restriction on dropped edges:
 if an edge is dropped, it can not reenter for at least k
 iterations. Here, we'd need k at least 5 to ensure that e_5
 gets added eventually. The problem with this is that it
 makes other good moves hard to find - the process becomes forced
 to make a particular move. In addition, if the graph was larger,
~~that~~ we would need k to be correspondingly larger.

Remedy: diversification:

If process has not used a particular edge for a long time, and if
 it seems to be stuck around a local minimum, force it to use the
 missing edge.

There are other ways to enforce diversification:

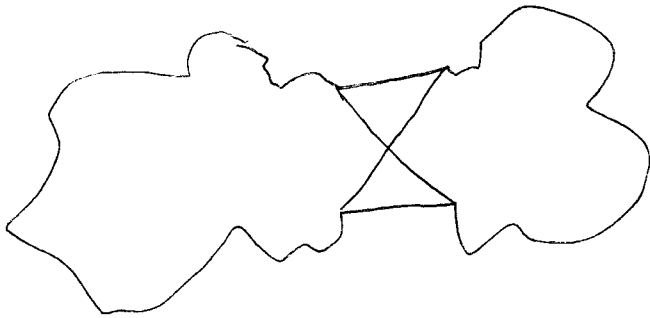
- eg, reduce penalties for violating constraints,
- reduce weights on missing edges
- allowing two edges to be added at once
- ⋮

Could also ~~choose~~ ^{choose} moves probabilistically at a k simulated annealing.
 The advantage of a methodical diversification procedure is that it is still trying
 to choose good moves and it is exploring the feasible region in an ^{organized} ~~systematic~~ manner.

Travelling Salesman Problem

Have graph $G=(V,E)$. Looking for a tour through the cities, of minimum length.

Use 2-change heuristic (Lin-Heuristic) to get local minimum:



Take two edges, and recombine their endpoints in a different way.



This leads to reasonable tours; a considerable improvement is possible by using 3-change - recombine the endpoints of three edges.

Can use tabu search in conjunction with 2-opt moves to solve TSPs.

~~When leaving a local minimum, tabu~~

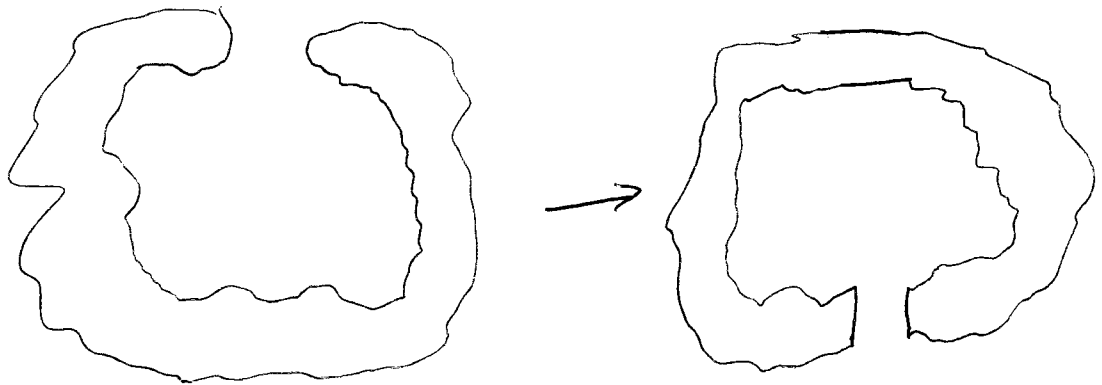
Tabu restriction: don't let recently added edges leave
dropped enter.

The list of tabu dropped edges is longer than tabu added edges, because there are more potential new edges than edges which can leave.

Process reaches a local minimum, and then it moves out using ~~tabu~~ ^{moves} 2-change moves. The tabu restrictions stop it returning to the local minimum.

Problem: Tend to introduce crosses with 2-changes. If $\#$ tabu list is long, these crosses don't go away. If tabu list is too short, return to local minimum.

Solution: a diversification move:



Break the tour in two places.

This is the equivalent of three 2-changes, one of which may have high cost.

So, after getting a local minimum, look at various diversification moves.

