

Artificial Intelligence, Computational Logic

PROBLEM SOLVING AND SEARCH IN ARTIFICIAL INTELLIGENCE

Lecture 5 Tabu Search

Sarah Gaggl



Agenda

- Introduction
- Uninformed Search versus Informed Search (Best First Search, A* Search, Heuristics)
- 3 Local Search, Stochastic Hill Climbing, Simulated Annealing
- Tabu Search
- 5 Answer-set Programming (ASP)
- 6 Constraint Satisfaction (CSP)
- Structural Decomposition Techniques (Tree/Hypertree Decompositions)
- 8 Evolutionary Algorithms/ Genetic Algorithms

Tabu Search

Main Idea

- A memory forces the search to explore new areas of the search space
- Memorize solutions that have been examined recently. They become tabu points in next steps
- Tabu search is deterministic

Tabu Search and SAT

- SAT problem with n = 8 variables
- Initial (random) assignment $\mathbf{x} = (0, 1, 1, 1, 0, 0, 0, 1)$
- Evaluation function: weighted sum of number of satisfied clauses.
 Weights depend on the number of variables in the clause
- Maximize evaluation function (i.e. we're trying to satisfy all clauses)
- Random assignment provides $eval(\mathbf{x}) = 27$
- Neighborhood of x consists of 8 solutions. Evaluate them and select best
- . At this stage, it is the same as hill-climbing
- Suppose flipping 3rd variable generates best evaluation ($eval(\mathbf{x}') = 31$)
- Memory keeps track of actions

Recency-based Memory

- Index of flipped variable + time when it was flipped
- Differentiate between older and more recent flips
- SAT: time stamp for each position of solution vector M (initialized to 0)
- Value of time stamp provides information on recency of flip at position

Memory Vector

$$M(i) = j$$
 (when $j \neq 0$)

j is most recent iteration when i-th bit was flipped

Recency-based Memory

- Index of flipped variable + time when it was flipped
- Differentiate between older and more recent flips
- SAT: time stamp for each position of solution vector *M* (initialized to 0)
- Value of time stamp provides information on recency of flip at position

Memory Vector

$$M(i) = j$$
 (when $j \neq 0$) j is most recent iteration when i -th bit was flipped

Assume information is stored for at most 5 iterations.

Alternative Interpretation

$$M(i) = j$$
 (when $j \neq 0$)
 i -th bit was flipped $5 - j$ iterations ago

Recency-based Memory

- Index of flipped variable + time when it was flipped
- Differentiate between older and more recent flips
- SAT: time stamp for each position of solution vector *M* (initialized to 0)
- Value of time stamp provides information on recency of flip at position

Memory Vector

$$M(i) = j$$
 (when $j \neq 0$) j is most recent iteration when i -th bit was flipped

Assume information is stored for at most 5 iterations.

Alternative Interpretation

$$M(i) = j$$
 (when $j \neq 0$)
i-th bit was flipped $5 - j$ iterations ago

Example

0	0	5	0	0	0	0	0

Memory after one iteration. 3rd bit is **tabu** for next 5 iterations.

Different Interpretations

1st Variant

- Stores iteration number of most recent flip
- Requires a current iteration counter t which is compared with memory values
- If t M(i) > 5 forget
- Only requires updating a single entry, and increase the counter
- Used in most implementations

Different Interpretations

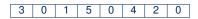
1st Variant

- Stores iteration number of most recent flip
- Requires a current iteration counter t which is compared with memory values
- If t M(i) > 5 forget
- Only requires updating a single entry, and increase the counter
- Used in most implementations

2nd Variant

- Values are interpreted as number of iterations for which a position is not available
- All nonzero entries are decreased by one at every iteration

- Initial assignment $\mathbf{x} = (0, 1, 1, 1, 0, 0, 0, 1)$
- After 4 additional iterations M:



- Most recent flip M(4) = 5
- Current solution: $\mathbf{x} = (1, 1, 0, 0, 0, 1, 1, 1)$ with $eval(\mathbf{x}) = 33$

- Initial assignment $\mathbf{x} = (0, 1, 1, 1, 0, 0, 0, 1)$
- After 4 additional iterations M :

3	0	1	5	0	4	2	0

- Most recent flip M(4) = 5
- Current solution: $\mathbf{x} = (1, 1, 0, 0, 0, 1, 1, 1)$ with $eval(\mathbf{x}) = 33$

Neighborhood of x

$$\mathbf{x}_1 = (0, 1, 0, 0, 0, 1, 1, 1)$$
 $\mathbf{x}_5 = (1, 1, 0, 0, 1, 1, 1, 1)$

$$\mathbf{a}_1 = (0, 1, 0, 0, 0, 1, 1, 1)$$

$$\mathbf{x}_2 = (1, 0, 0, 0, 0, 1, 1, 1)$$
 $\mathbf{x}_6 = (1, 1, 0, 0, 0, 0, 1, 1)$

$$\mathbf{x}_3 = (1, 1, 1, 0, 0, 1, 1, 1)$$
 $\mathbf{x}_7 = (1, 1, 0, 0, 0, 1, 0, 1)$

$$\mathbf{x}_4 = (1, 1, 0, 1, 0, 1, 1, 1)$$

$$\mathbf{x}_{\varepsilon} = (1 \ 1 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1)$$

$$\mathbf{x}_5 = (1, 1, 0, 0, 1, 1, 1, 1)$$

$$a_0 = (1, 1, 0, 0, 0, 0, 1, 1)$$

$$\mathbf{x}_7 = (1, 1, 0, 0, 0, 1, 0, 1)$$

$$\mathbf{x}_8 = (1, 1, 0, 0, 0, 1, 1, 0)$$

- Initial assignment $\mathbf{x} = (0, 1, 1, 1, 0, 0, 0, 1)$
- After 4 additional iterations M :

2	Λ	-1	5	Λ	1	2	Λ
0	U		J	U	-	~	U

- Most recent flip M(4) = 5
- Current solution: $\mathbf{x} = (1, 1, 0, 0, 0, 1, 1, 1)$ with $eval(\mathbf{x}) = 33$

Neighborhood of x

TABU, best evaluation $eval(\mathbf{x}_5) = 32$, **decrease!**

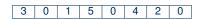
- Current solution: $\mathbf{x} = (1, 1, 0, 0, 0, 1, 1, 1)$ with $eval(\mathbf{x}) = 33$
- New solution: $\mathbf{x}_5 = (1, 1, 0, 0, 1, 1, 1, 1)$ with $eval(\mathbf{x}_5) = 32$

3	0	1	5	0	4	2	0

changes to:



- Current solution: $\mathbf{x} = (1, 1, 0, 0, 0, 1, 1, 1)$ with $eval(\mathbf{x}) = 33$
- New solution: $\mathbf{x}_5 = (1, 1, 0, 0, 1, 1, 1, 1)$ with $eval(\mathbf{x}_5) = 32$



changes to:



Policy might be too restrictive

- What if tabu neighbor x_6 provides excellent evaluation score?
- Make search more flexible: override tabu classification if solution is outstanding
- ⇒ aspiration criterion

Long-term Memory

Question

- 1 What is stored in long-term memory (think of SAT as an example)?
- 2 How can we escape local optima with help of a long-term memory?



Frequency-based Memory

- Operates over a longer horizon
- SAT: vector H serves as long-term memory.
 - Initialized to 0, at any stage of the search

$$H(i) = j$$

interpreted as: during last h (horizon) iterations, the i-th bit was flipped i times

- Usually horizon is large
- After 100 iterations with h = 50, long-term memory H might have the following values

Shows distribution of moves throughout the last 50 iterations

Diversity of Search

Frequency-based memory provides information about which flips have been under-represented or not represented.

⇒ we can diversify the search by exploring these possibilities

Use of Long-term Memory

Special Circumstances

- Situations where all non-tabu moves lead to worse solution
- To make a meaningful decision about which direction to explore next
- Typically: most frequent moves are less attractive
- Value of evaluation score is decreased by some penalty measure that depends on frequency, final score implies the winner

- Assume value of current solution is $eval(\mathbf{x}) = 35$
- Non-tabu flips 2, 3 and 7 have values 30, 33, 31
- None of tabu moves provides value greater than 37 (highest value so far)
 we can't apply aspiration criterion

- Assume value of current solution is $eval(\mathbf{x}) = 35$
- Non-tabu flips 2, 3 and 7 have values 30, 33, 31
- None of tabu moves provides value greater than 37 (highest value so far)
 we can't apply aspiration criterion
- Frequency based-memory and evaluation function for new solution \mathbf{x}' is

$$eval(\mathbf{x}') - penalty(\mathbf{x}')$$

• $penalty(\mathbf{x}') = 0.7 \times H(i)$, where 0.7 coefficient, H(i) value from long-term memory H:

7	for solution created by flipping 2nd bit
11	for solution created by flipping 3nd bit
1	for solution created by flipping 7nd bit

- Assume value of current solution is $eval(\mathbf{x}) = 35$
- Non-tabu flips 2, 3 and 7 have values 30, 33, 31
- None of tabu moves provides value greater than 37 (highest value so far)
 we can't apply aspiration criterion
- ullet Frequency based-memory and evaluation function for new solution x^\prime is

$$eval(\mathbf{x}') - penalty(\mathbf{x}')$$

 penalty(x') = 0.7 × H(i), where 0.7 coefficient, H(i) value from long-term memory H:

7	for solution created by flipping 2nd bit
11	for solution created by flipping 3nd bit
1	for solution created by flipping 7nd bit

New scores are:

$$30 - 0.7 \times 7 = 25.1$$
 2nd bit $33 - 0.7 \times 11 = 25.3$ 3nd bit $31 - 0.7 \times 1 = 30.3$ 7th bit

ullet Frequency based-memory and evaluation function for new solution \mathbf{x}' is

$$eval(\mathbf{x}') - penalty(\mathbf{x}')$$

• $penalty(\mathbf{x}') = 0.7 \times H(i)$, where 0.7 coefficient, H(i) value from long-term memory H:

7	for solution	created	by	flipping	2nd	bit
11	for solution	created	by	flipping	3nd	bit
1	for solution	created	by	flipping	7nd	bit

New scores are:

$$30 - 0.7 \times 7 = 25.1$$
 2nd bit $33 - 0.7 \times 11 = 25.3$ 3nd bit $31 - 0.7 \times 1 = 30.3$ 7th bit

Diversify Search

Including frequency values in a penalty measure for evaluating solutions.

Further Options to Diversify Search

We migth add additional rules:

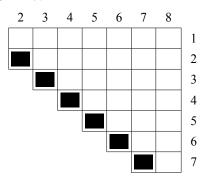
- Aspiration by default: select the oldest of all considered
- Aspiration by search direction: memorize whether or not the performed moves generated any improvement
- Aspiration by influence: measures the degree of change of the new solution
 - a) in terms of the distance between old and new solution
 - b) change in solution's feasibility, if we deal with a constraint problem
 - Intuition: particular move has a larger influence if a larger step was made from old to new solution

Tabu Search and the TSP

- Move: swap two cities in a particular solution
- Current solution: (2, 4, 7, 5, 1, 8, 3, 6)
- 28 neighbors $\binom{8}{2} = \frac{7 \cdot 8}{2} = 28$
- Recency-based memory: swap of cities i and j in i-th row and j-th column (for i < j)
- Maintain number of remaining iterations for which swap stays on tabu list
- Frequency-based memory: same structure; indicate totals of all swaps within horizon h = 50

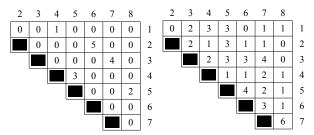
Tabu Search and the TSP

- Move: swap two cities in a particular solution
- Current solution: (2, 4, 7, 5, 1, 8, 3, 6)
- 28 neighbors $\binom{8}{2} = \frac{7 \cdot 8}{2} = 28$
- Recency-based memory: swap of cities i and j in i-th row and j-th column (for i < j)
- Maintain number of remaining iterations for which swap stays on tabu list
- Frequency-based memory: same structure; indicate totals of all swaps within horizon h = 50



Tabu Search and the TSP ctd.

- Assume both memories initialized to zero and 500 iterations have been completed
- Current solution: (7, 3, 5, 6, 1, 2, 4, 8) with length: 173, best solution so far 171



left: recency-based memory; right: frequency-based memory

Summary

- Simulated annealing and tabu search are both design to escape local optima
- Tabu search makes uphill moves only when it is stuck in local optima
- Simulated annealing can make uphill moves at any time
- Simulated annealing is stochastic, tabu search is deterministic
- Compared to classic algorithms, both work on complete solutions. One can halt them at any iteration and obtain a possible solution
- Both have many parameters to worry about

References



Zbigniew Michalewicz and David B. Fogel.

How to Solve It: Modern Heuristics, volume 2. Springer, 2004.



Knox, J.

Tabu Search Performance on the Symmetric Traveling Salesman Problem, Computer Operations Research, Vol.21, No.8, pp.867–876, 1994.