Class Discussion on Local Search (Sections 7.1 & 7.2)
Presentation by Ryan Kinworthy on February 24

Scribe: Nurzhan Ustemirov

1 Overview

The topic of the class discussion was alternative search techniques for solving CSPs. The traditional systematic search we have learned earlier doesn't efficiently work on large-scale problems. The alternative Local Search implements several approximation techniques, saving space and time for solving CSPs.

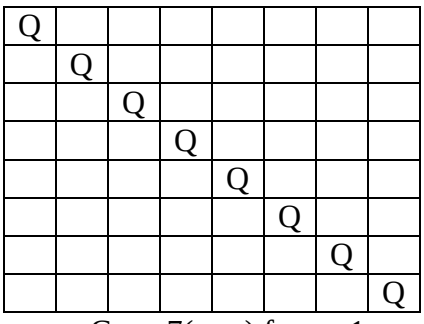
2 Greedy Local Search (7.1)

A general advantage of a greedy algorithm is its efficiency in terms of both space and time. Usually a solution is generated in linear time, which is particularly attractive for large problems. However, unless have very special conditions (see greedy algorithms of CSCE 310), it is not in general a complete, sound, or optimal strategy.

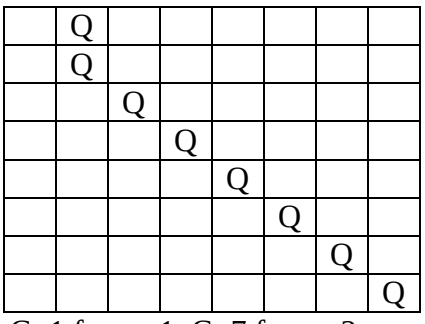
How greedy *local* search works

- The algorithm starts with random instantiation of all variables.
- Calculates the cost (# of violated constraints)
- Calculates the improvement (cost) if any variable is changed. In other words by how much can we reduce the total cost if a (Single) variable was changed.
- Selects one with best improvement (max reduction) and changes it.
- Stops either when the cost reaches 0 (problem is solved) or when it reaches stage when no improvement could be made, then algorithm is restarted with new random assignment.

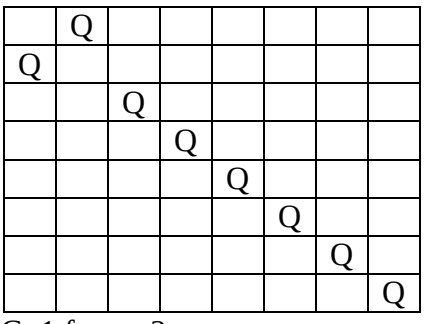
For example, consider the pseudo-random instantiation of the 8-queen problem below



Cost=7(cost) for var1



C=1 for var1, C=7 for var2



C=1 for var2

Thus, moving each queen to place with minimum conflicts the algorithm moves toward the solution (hopefully). Unfortunately, the process does not guarantee finding a solution and may get stuck in a local minimum.

- **Local minima** is a situation where the algorithm reached a point that is not a solution and from where no improvement could be made to current assignment.

One of the first successful applications of greedy local search is the Traveling Salesperson Problem (TSP). The TSP is an optimization problem to find the shortest route to visit every city only once. The algorithm starts with a random sequence of the cities. Then, making best swaps of the cities the cost is reduced. If no swap improves the cost than the algorithm either had reached the solution or is stuck in a local minimum.

Stochastic Local Search (SLS):

Dechter gives SLS algorithm on page 199.

Input: a constraint Network R, # of MAX_TRIES, a cost function
Output: A solution iff the problem is consistent, "false" otherwise
Algorithm,

- Repeat random initialization MAX_TRIES times,
  - Randomly assign all variables
  - Repeat until no improvements could be made
    - if the cost is global minimum (C=0) then the assignment is a solution (probably won't happen at the first try)
    - calculate improvement for each VVP if it was changed.
    - choose one that makes maximum improvement of the cost, and change the value

SLS is not perfect and most probably will get stuck in a local minimum.

- The technique saw vast popularity in solving SAT problem, and the term **"value flipping"** was introduced in the chapter. Since the domain of variables in SAT problems have only two values the "value change" was defined as "value flipping".

Example 7.1.1 (Dechter)
Given the formula $\varphi = \{(\neg C)(\neg A \vee \neg B \vee C)(\neg A \vee D \vee E)(\neg B \vee \neg C)\}$.

- Initially all variable are assigned "1" (some sort of randomness).
- The first and the last clauses are violated, thus the cost C=2.
- Flipping A, E or D does not improve the cost, Where is flipping C or B reduces the total cost by 1.
- Given no preference on choosing certain literal C has been flipped. The first and the last clauses are satisfied but the second one is violated. Now, the cost C=1.
- In the next step B is flipped and the cost is reduced to "0" (global minima). And the solution has been reached.

3 **Improving SLS (7.1.1)**

The SLS by itself can easily get stuck in a local minimum. To improve SLS the following methods were suggested:

- Improve the selection of the initial assignment (instead of random assignment apply heuristics)
- Improve the nature of the local changes considered (heuristics of algorithms)
- Try to escape local minima
  - applying heuristics
  - or using different combinations of heuristics

Suggested heuristics: Most of heuristics were briefly summarized, except for Plateau search, which lead to mass misunderstanding (including myself). After discussion ate the end of the lecture, the heuristic became clear. Later I double checked with a paper by Gent and Walsh. They implemented the search on GSAT. It is very good paper for those who are interested. You can find the link for the paper at the bottom of scribe.

Plateau search: Local optima reached a flat area could called plateau. When an algorithm reaches a plateau, the cost may not be improved. Instead of reinitializing the whole assignment, continuing with non-improving "sidewalks" actually could lead to improvements (hopefully). The better local minima could be nearby the plateau and limited number of trials (sidewalks) could lead us to find that local minima (if one exists). (JAIR / Gent & Walsh).

The rest of the heuristics are from Ryan's slides.

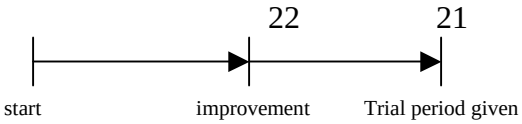
Constraint Weighting: Where cost function is a weighted sum of the violated constraints $F(\bar{a}) = \sum_i w_i * C_i(\bar{a})$ where w_i is the current weight of constraint C_i and if a violates constraint C_i then $C_i(\bar{a}) = 1$, otherwise it is equal to 0. At each step the algorithm selects a variable-value pair such that its change leads to largest change of overall cost function. And at a local minimum the weights of the violated constraint is increased by one.

Tabu search: The search was covered in detail last semester. The idea is that a list of last *n* variable-value assignments is kept. And that list is forbidden for next new assignments.

The-Breaking Rules: This heuristics is to resolve the case when several best improvements have the same change over the cost. And tie-breaking suggests to pick the one that was least recently modified.

Value Propagation: Another heuristic to escape a local minimum is to apply value propagation techniques (e.g., arc-consistency or unit resolution) over violated constraints.

Automating Max-Flips: How many flips to do? Or how to decide the value of MAX_TRIES?
The simplest approach for the first – continue as long as there is a progress. For the second one the bigger the better.
More sophisticated approach for max-flips would be to record the time of improvement and give the algorithm equivalent duration to try.



Random Walk Strategy (7.2)

Dr. Choueiry: - When we have many solutions "Randomness" is a friend.

The greedy choice in the local search could be replaced with random walks (steps) so the selection is not based on the best choice but on a random pick. The random step is a random selection frm the neighboring states. It avoids getting into the local minimum.

The strategy was first developed for SAT problems.
WalkSAT algorithm: (slide 15[1] or page 202 of [2])

> Input: constraint network R, MAX_TRIES, MAX_FLIPS, and the probability *p*
> Output: True if the problem is consistent, false otherwise.

The algorithm initializes all variables randomly and randomly picks a violated constraint. Then with probability *p* it chooses a random variable-value pair and with probability *1-p* chooses the best neighbor (greedy choice). So, the algorithm is actually the combination of randomness and greedy choice. It is moving toward the least cost and simultaneously trying to avoid getting into a local minimum by allowing randomness (to some extent).

Simulated Annealing

Was covered in the next lecture.

Reference:

[1] Ryan Kinworthy, *Lecture Slides*, February 24, 2003.
[2] Rina Dechter, *Constraint Processing*, October 3, 2002.
[3] Ian P. Gent and Toby Walsh. *An Empirical Analysis of Search in GSAT*. Journal of Artificial Intelligence Research 1 (1993) 47-59
http://www.cs.washington.edu/research/jair/abstracts/gent93a.html