Foundations of Artificial Intelligence

21. Combinatorial Optimization: Advanced Techniques

Malte Helmert

University of Basel

April 1, 2019

Foundations of Artificial Intelligence

April 1, 2019 — 21. Combinatorial Optimization: Advanced Techniques

Combinatorial Optimization: Overview

Chapter overview: combinatorial optimization

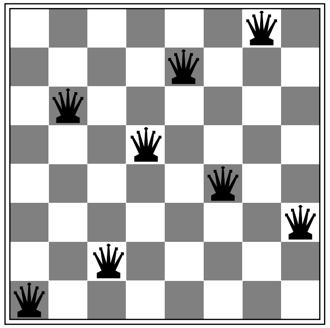
- 20. Introduction and Hill-Climbing
- 21. Advanced Techniques

21.1 Dealing with Local Optima

Example: Local Minimum in the 8 Queens Problem

local minimum:

- candidate has 1 conflict
- all neighbors have at least 2



Weaknesses of Local Search Algorithms

difficult situations for hill climbing:

- local optima: all neighbors worse than current candidate
- plateaus: many neighbors equally good as current candidate; none better

German: lokale Optima, Plateaus

consequence:

- algorithm gets stuck at current candidate

Combating Local Optima

possible remedies to combat local optima:

- allow stagnation (steps without improvement)
- include random aspects in the search neighborhood
- (sometimes) make random steps
- breadth-first search to better candidate
- restarts (with new random initial candidate)

Allowing Stagnation

allowing stagnation:

- do not terminate when no neighbor is an improvement
- limit number of steps to guarantee termination
- at end, return best visited candidate
  - pure search problems: terminate as soon as solution found

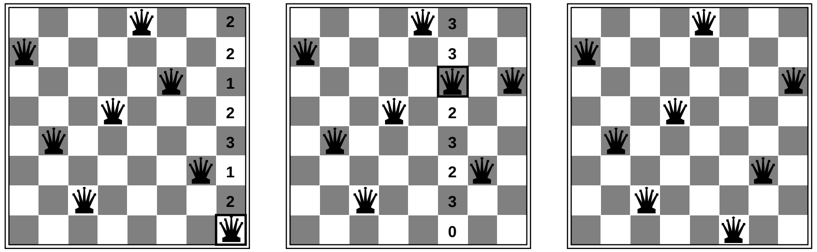
Example 8 queens problem:

- with a bound of 100 steps solution found in 94% of the cases
- on average 21 steps until solution found

⇝ works very well for this problem;
for more difficult problems often not good enough

Random Aspects in the Search Neighborhood

a possible variation of hill climbing for 8 queens:
Randomly select a file; move queen in this file
to square with minimal number of conflicts (null move possible).



⇝ Good local search approaches often combine
randomness (exploration) with heuristic guidance (exploitation).

German: Exploration, Exploitation

21.2 Outlook: Simulated Annealing

Simulated Annealing

Simulated annealing is a local search algorithm that systematically injects noise, beginning with high noise, then lowering it over time.

- walk with fixed number of steps N (variations possible)
- initially it is “hot”, and the walk is mostly random
- over time temperature drops (controlled by a schedule)
- as it gets colder, moves to worse neighbors become less likely

very successful in some applications, e.g., VLSI layout

German: simulierte Abkühlung, Rauschen

Simulated Annealing: Pseudo-Code

Simulated Annealing (for Maximization Problems)

curr := a random candidate
best := **none**
for each $t \in \{1, \dots, N\}$:
    if is_solution(*curr*) **and** (*best* **is none or** $v(curr) > v(best)$):
        best := *curr*
    T := schedule(t)
    next := a random neighbor of *curr*
    $\Delta E := h(next) - h(curr)$
    if $\Delta E \geq 0$ **or** with probability $e^{\frac{\Delta E}{T}}$:
        curr := *next*
return *best*

21.3 Outlook: Genetic Algorithms

Genetic Algorithms

Evolution often finds good solutions.

idea: simulate evolution by selection, crossover and mutation of individuals

ingredients:

- encode each candidate as a string of symbols (genome)
- fitness function: evaluates strength of candidates (= heuristic)
- population of k (e.g. 10–1000) individuals (candidates)

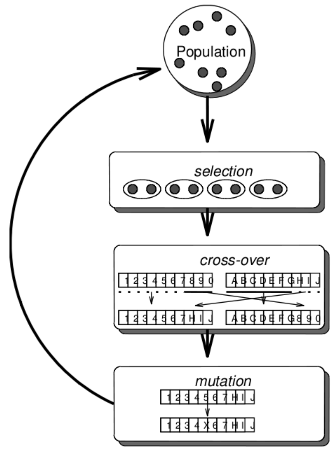
German: Evolution, Selektion, Kreuzung, Mutation, Genom, Fitnessfunktion, Population, Individuen

Genetic Algorithm: Example

example 8 queens problem:

- genome: encode candidate as string of 8 numbers
- fitness: number of non-attacking queen pairs
- use population of 100 candidates

Selection, Mutation and Crossover



many variants:
How to select?
How to perform crossover?
How to mutate?

select according to fitness function, followed by pairing

determine crossover points, then recombine

mutation: randomly modify each string position with a certain probability

21.4 Summary

Summary

- weakness of local search: local optima and plateaus
- remedy: balance exploration against exploitation (e.g., with randomness and restarts)
- simulated annealing and genetic algorithms are more complex search algorithms using the typical ideas of local search (randomization, keeping promising candidates)