TECHNISCHE UNIVERSITÄT DRESDEN

Artificial Intelligence, Computational Logic

PROBLEM SOLVING AND SEARCH IN ARTIFICIAL INTELLIGENCE

Lecture 2 Uninformed Search vs. Informed Search

Sarah Gaggl

Dresden

Agenda

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

Traditional Methods

- There are many classic algorithms to search spaces for an optimal solution.
- Broadly, they fall into two disjoint classes:
  - Algorithms that only evaluate complete solutions (exhaustive search, local search, . . .).
  - Algorithms that require the evaluation of partially constructed or approximate solutions.
- Algorithms that treat complete solutions can be stopped any time, and give at least one potential answer.
- If you interrupt an algorithm that works on partial solutions, the results might be useless.

Complete Solutions

- All decision variables are specified.
- For example, binary strings of length n constitute complete solutions for any n-variable SAT.
- Permutations of n cities constitute complete solutions for a TSP.
- We can compare two complete solutions using an evaluation function.
- Many algorithms rely on such comparisons, manipulating one single complete solution at a time.
- When a new solution has a better evaluation than the previous best solution, it replaces that prior solution.
- Exhaustive search, local search, hill climbing as well as modern heuristic methods such as simulated annealing, tabu search and evolutionary algorithms fall into this category.

Partial Solutions

There are two forms:

1. incomplete solution to the problem originally posed, and
2. complete solution to a reduced (i.e. simpler) problem.

- Incomplete solutions reside in a subset of the original problem's search space.
  - In an SAT, consider all of the binary strings where the first two variables were assigned the value 1 (i.e. TRUE).
  - In a TSP, consider every permutation of cities that contains the sequence $7 - 11 - 2 - 16$.
  - We fix the attention on a subset of the search space that has a partial property.
  - Hopefully, that property is also shared by the real solution!

Partial Solutions ctd.

- Decompose original problem into a set of smaller and simpler problems.
  - Hope: solving each of the easier problems and combine the partial solutions, results in an answer for the original problem.
  - In a TSP, consider only k out of n cities and try to establish the shortest path from city i to j that passes through all k of these cities.
  - Reduce the size of the search space significantly and search for a complete solution within the restricted domain.
  - Such partial solutions can serve as building blocks for the solution to the original problem.

Partial Solutions ctd.

- Decompose original problem into a set of smaller and simpler problems.
  - Hope: solving each of the easier problems and combine the partial solutions, results in an answer for the original problem.
  - In a TSP, consider only k out of n cities and try to establish the shortest path from city i to j that passes through all k of these cities.
  - Reduce the size of the search space significantly and search for a complete solution within the restricted domain.
  - Such partial solutions can serve as building blocks for the solution to the original problem.
- But, algorithms that work on partial solutions pose additional difficulties. One needs to
  - devise a way to organize the sub-spaces so that they can be searched efficiently, and
  - create a new evaluation function that can assess the quality of partial solutions.

Exhaustive Search

- Checks every solution in the search space until the best global solution has been found.
- Can be used only for small instances of problems.
- Exhaustive (enumerative) algorithms are simple.
- Search space can be reduced by backtracking.
- Some optimization methods, e.g., branch and bound and A* are based on an exhaustive search.

Exhaustive Search

- Checks every solution in the search space until the best global solution has been found.
- Can be used only for small instances of problems.
- Exhaustive (enumerative) algorithms are simple.
- Search space can be reduced by backtracking.
- Some optimization methods, e.g., branch and bound and A* are based on an exhaustive search.
- How can we generate a sequence of every possible solution to the problem?
  - The order in which the solutions are generated and evaluated is irrelevant (because we evaluate all of them).
  - The answer for the question depends on the selected representation.

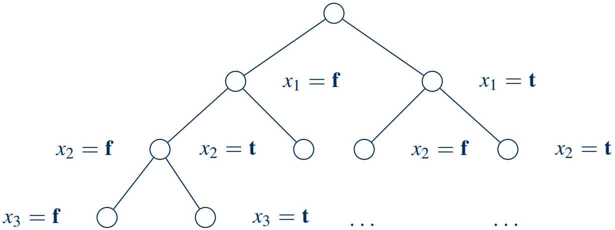
Enumerating the SAT

- We have to generate every possible binary string of length n.
- All solutions correspond to whole numbers in a one-to-one mapping.
- Generate all non-negative integers from 0 to $2^n - 1$ and convert each of these integers into the matching binary string of length n.

0000	0	0100	4	1000	8	1100	12
0001	1	0101	5	1001	9	1101	13
0010	2	0110	6	1010	10	1110	14
0011	3	0111	7	1011	11	1111	15

- Bits of the string are the truth assignments of the decision variables.
- Organize the search space, for example partition into two disjoint sub-spaces. First contains all the vectors where $x_1 = \mathbf{f}$ (FALSE), and the second contains all vectors where $x_1 = \mathbf{t}$ (TRUE).

Enumerating the SAT ctd.



Binary search tree for SAT

Search Strategies

A strategy is defined by picking the order of node expansion.
Strategies are evaluated along the following dimensions:

- Completeness - does it always find a solution if one exists?
- Time complexity - number of nodes generated/expanded.
- Space complexity - maximum number of nodes in memory.
- Optimality - does it always find a least-cost solution?

Time and space complexity are measured in terms of

- b - maximum branching factor of the search tree;
- d - depth of the least-cost solution;
- m - maximum depth of the state space (may be ∞).

Self-study

OPAL

The material for self-studying is available in OPAL:
`https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/23291363328/CourseNode/101501861304634`

After going through the chapters you should be able to answer the following properties for the discussed search methods.

- Completeness
- Time complexity
- Space complexity
- Optimality

References

Zbigniew Michalewicz and David B. Fogel.
How to Solve It: Modern Heuristics, volume 2. Springer, 2004.

Stuart J. Russell and Peter Norvig.
Artificial Intelligence - A Modern Approach (3. edition). Pearson Education, 2010.