COMP30030: Introduction to Artificial Intelligence

Neil Hurley

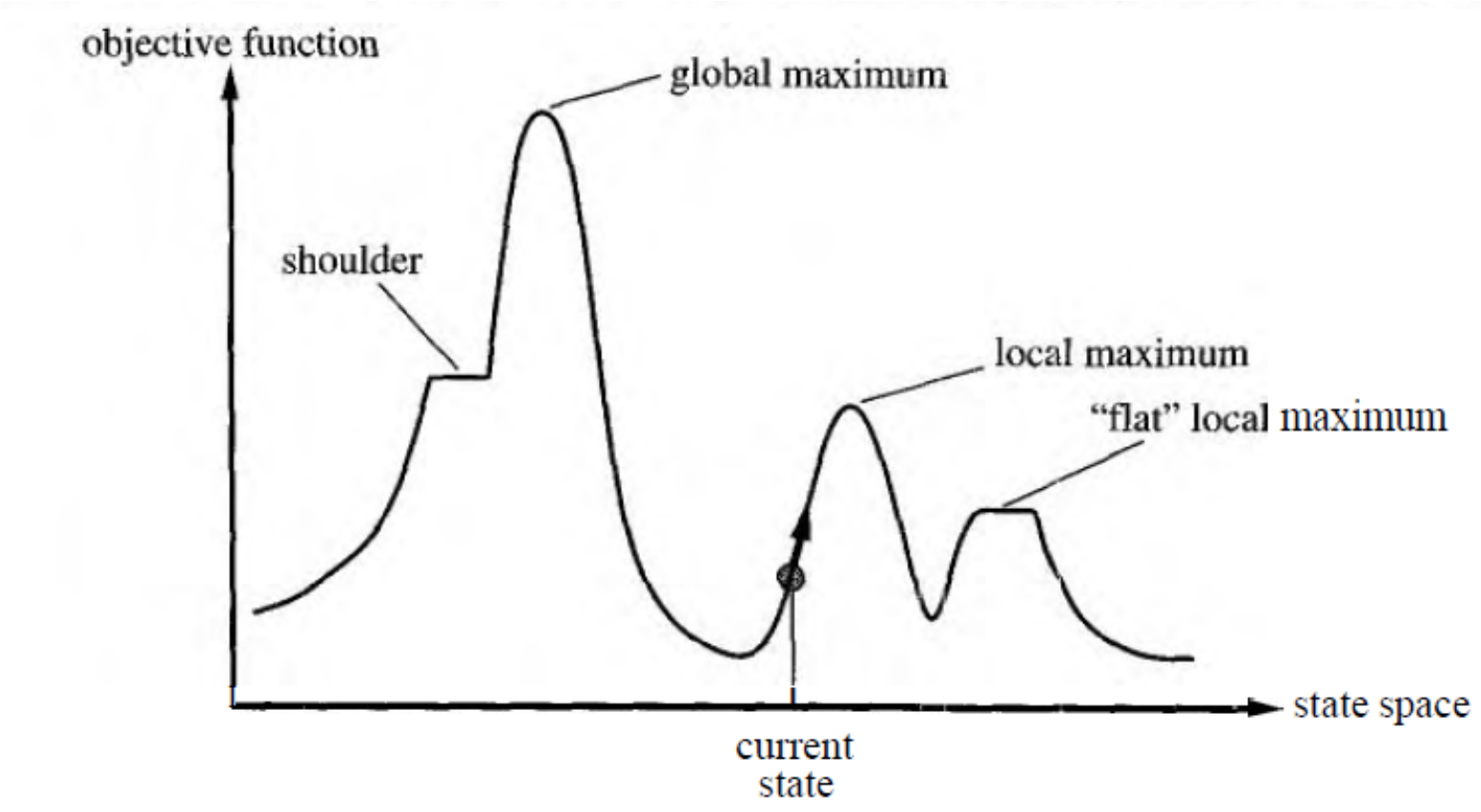
School of Computer Science
University College Dublin
neil.hurley@ucd.ie

October 16, 2018

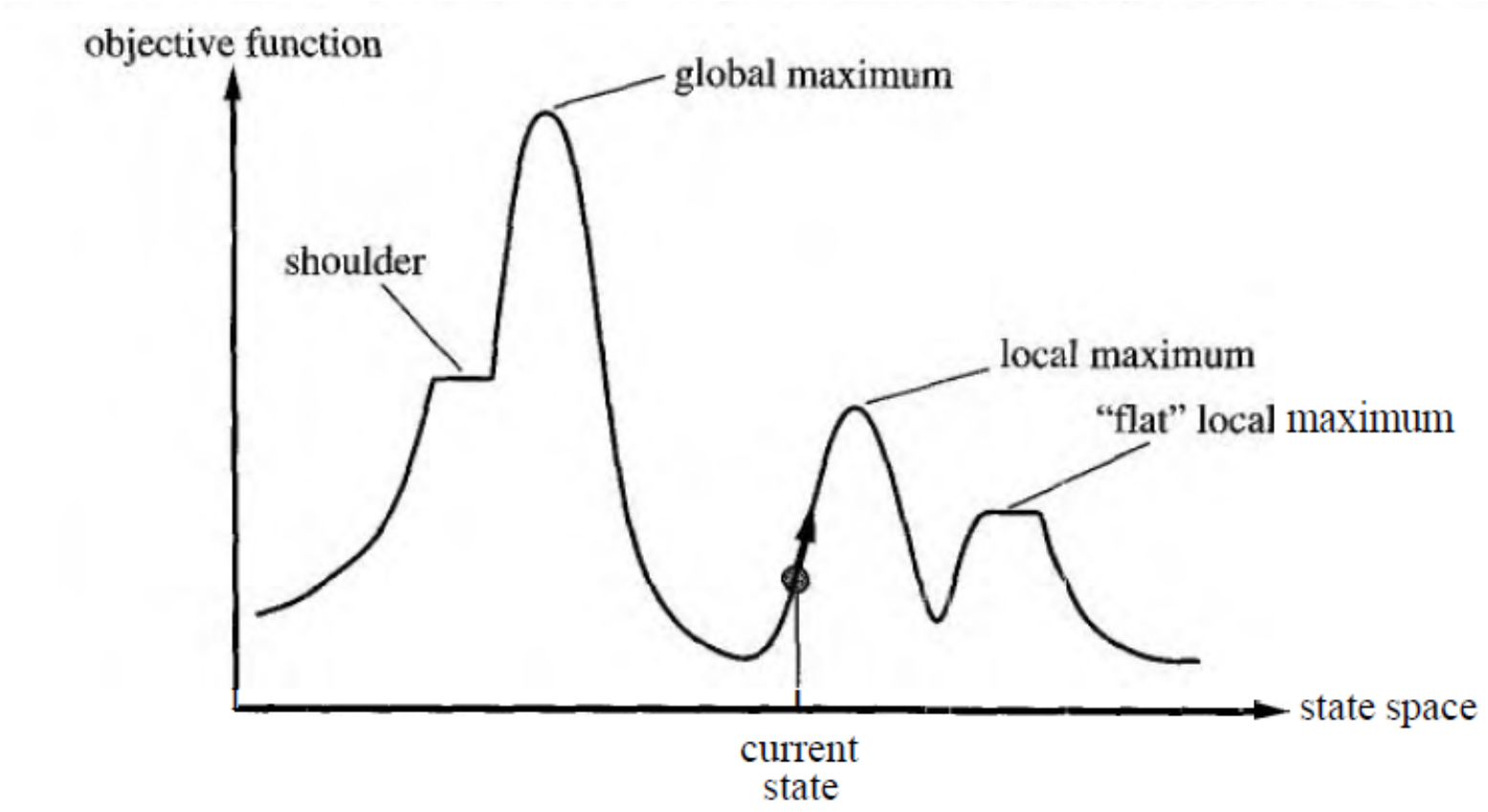
Optimisation

- **An optimisation problem is one in which we do not know a solution, but we have an objective function to weigh states.**
- **The aim of this optimisation is to find the state with the lowest weight, or at least one with a suitably low weight.**
- **Local search techniques can be used for optimisation as well as standard search.**

Optimisation and landscapes



Optimisation and landscapes



- **If this looks simple, think of it in 10,000 dimensions.**

Hill climbing search

- **Only keep one state.**
- **Only go uphill (or downhill) as much as you can**

- **That is: one moves from a state by generating all its neighbours, and picking the best one.**

- **Tiny memory use, smashingly simple.**

Hill climbing problems

- **Local minima (and plateaux, and ridges)!**
- **Hill climbing goes uphill no matter what. It may happen to be on an uphill path to a peak which isn't the tallest one.**

- **Whether it succeeds or not will depend on the landscape.**
- **"Many real problems have a landscape that looks more like a family of porcupines on a flat floor, with miniature porcupines living on the tip of each porcupine needle, *ad infinitum*" (Russel&Norvig)**

Hill climbing variations

- Random restart: if you get stuck, start again from a different state.
- If your chances of succeeding in one attempt are p, then you expect it to take you approximately 1/p starts on average to get a solution.
- Greatly increases the chances of solution, though:
  - if p is tiny, it might take a while
  - it multiplies the complexity

Hill climbing variations (2)

- **Stochastic hill climbing: pick a successor at random among the uphill ones.**
- **First-choice hill climbing: generate random successors until you find one that is uphill.**

- **Both mitigate the local maxima problem.**
- **First-choice makes it possible to apply hill climbing to infinite-dimensional problems.**

Local beam search

- **Keep track of k searches instead of just one. At a given step, generate all (or many) neighbours of *all* states, and keep only the k best in the full list.**
- **Different from doing k hill climbing searches because there is interplay.**
- **Tricky issues:**
  - **who is k?**
  - **how to prevent diversity from quickly disappearing?**

Stochastic beam search

- **Generate list of successors of all k states.**
- **Pick k out of the list *at random* but with a probability that depends on their quality.**

- **E.g. divide a segment into M intervals where M is the successor number, and interval i is exp(quality_i/T) wide, then pick k points (uniformly) randomly distributed over the interval and pick the corresponding states.**

Stochastic beam search (2)

- **It resembles (asexual) natural selection.**
- **An individual has a better *chance* of surviving if it is fitter than the others**
- **This doesn't guarantee it will survive and the less fit individuals won't.**

- **Less promising solutions are kept around, and their successors still have a chance of looking more promising.**

Stochastic beam search (3)

- **This alleviates many of the problems of standard beam search, because it makes it less likely for the k states to converge quickly.**

- **However, it adds a tricky free parameter (T).**
- **Too high and your search is random.**
- **Too low and it becomes a very expensive hill climbing.**

Metaheuristic Algorithms

- We'll focus in more detail on two other approaches, namely simulated annealing and genetic algorithms.
- There are sometimes called metaheuristics as they are general problem solving strategies, that can be applied to many different problems.

Combinatorial Optimisation Problems I

- A combinatorial optimisation problem is a minimisation or maximisation problem that is specified by a set of problem instances.
- An instance of a combinatorial optimisation problem is a pair (S, f) where the solution space or state space S denotes the finite set of all possible solutions and the cost function f is a mapping defined as

$$f : S \rightarrow \mathbb{R}$$

- In the case of minimisation, the problem is to find a solution $i_{\text{opt}} \in S$ which satisfies

$$f(i_{\text{opt}}) \leq f(i) \forall i \in S$$

or, in the case of maximisation, the problem is to find a solution $i_{\text{opt}} \in S$ which satisfies

$$f(i_{\text{opt}}) \geq f(i) \forall i \in S$$

Combinatorial Optimisation Problems II

- Let f_{opt} denote the optimal cost and S_{opt} the set of optimal solutions.

The Travelling Salesman Problem (TSP)

- Given n cities and an $n \times n$ matrix d_{pq} whose elements denote the distance between each pair p and q of the n cities. A tour is defined as a closed path visiting each city exactly once. The problem is to find a tour of minimal length.
- A state for this problem is given by a cyclic permutation

$$\pi = (\pi(1) \dots \pi(n))$$

where $\pi(k)$ denotes the successor of city k, with

$$\begin{aligned} \pi^l(k) &\neq k \ \forall l \in \{1, \dots, n-1\} \\ \pi^n(k) &= k \end{aligned} \tag{1}$$

- The cost function is defined as

$$f(\pi) = \sum_{i=1}^{n} d_{i\pi(i)}$$

TSP – Simple Algorithm I

```
INPUT:    An integer n≥3 and an n × n distance matrix D of
          non-negative integers
OUTPUT: The shortest tour of n cities
begin
for all cyclic permutations of pi of (1,2,...,n) {
    calculate cost
    if (cost<min)
       min := cost
       besttour := pi
    }
}
output besttour
end
```

Simple Algorithm

- But ... how many cyclic permutations are there?

$$(n-1)!$$

n	$n!$
6	120
10	362880
50	6.08×10^{62}

Note that this number includes tours that can be obtained from each other by reversing the direction along each link. If we take such tours as equivalent, the number of tours reduces to $(n-1)!/2$.

Complexity Theory

- An algorithm is said to be polynomial if its worst case complexity is bounded by a polynomial of n. It is exponential if its worst-case complexity is bounded by c^n for some $c > 1$ where n is the size of the input.
- We can classify a problem as 'hard' or 'easy' depending on whether it can be solved by an algorithm with polynomial time complexity.
- Decision Problem : requires only yes or no answer.

TSP Decision Problem

- Given an integer $n \geq 3$ and an $n \times n$ matrix $D = d_{ij}$ and an integer $b >= 0$, is there a cyclic permutation π of the integers $\{1 \ldots n\}$ such that

$$\sum_{i=1}^{n} d_{i\pi(i)} \leq b$$

- If there is a polynomial-time algorithm for TSP, then there must be one for the TSP decision problem.

Non-deterministic Polynomial Algorithms I

- A non-deterministic algorithm is one which is equipped with the (powerful) instruction

  go to both label1, label 2

- Executing such a statement divides the problem into two parallel processes. By encountering many such statements, the computation will branch like a tree into a number of parallel computations that potentially can grow as an exponential function of the time elapsed. If any branch reports 'yes', then the non-deterministic algorithm has answered 'yes'. If none of the branches ever reports 'yes', then the answer is 'no'.
- The class of decision problems which can be solved in polynomial time using a non-deterministic algorithm are referred to as non-deterministic polynomial (NP) algorithms.

$NP \neq P$?

Open problem:

Prove that $NP \neq P$

- TSP is an NP-complete problem i.e. one to which all problems in NP can be polynomially transformed.

Local Search

Definition

Let (S, f) be a combinatorial optimisation problem. Then a **neighbourhood structure** is a mapping

$$N : S \to 2^S$$

which defines for $i \in S$ a set $S_i \subset S$ of solutions that are 'close' to i in some sense. We assume that

$$j \in S_i \leftrightarrow i \in S_j$$

Neighbourhood for the TSP

A k-change or k-opt defines for each state i a neighbourhood S_i consisting of the set of states that can be obtained from the given state i by removing k edges from the tour corresponding to state i and replacing them with k other edges such that again a tour is obtained.

Neighbourhood for the TSP

- The $N_2(p, q)$ change can be viewed as a reversal of the order in which the cities are traversed between p and q.
- The neighbourhood of a state i given by the application of the 2-change function is

$$S_i = \{j \in S | \pi_j \text{ is obtained from } \pi_i \text{ by a } 2 - \text{change}\}$$

The size of S_i is given by $|S_i| = (n \times (n-3))/2$

Local Search Algorithm I

- A state i is a local minimum if $f(j) \geq f(i) \forall j \in S_i$

```
LOCAL_SEARCH()
{
  INITIALISE(i_start);
  i = i_start;
  do while (i is not a local minimum){
     GENERATE(j from S_i);
     if (f(j) < f(i) )
         i = j;
     }
  } else end;
```

- So local search algorithms terminate in a local minimum and there is generally no guarantee as to 'how far' this local minimum is from a global maximum.

What can we do? I

- Execute local search algorithms for a large number of initial conditions.
- Introduce a more complex neighbourhood structure in order to be able to search a larger portion of the state space.
- Accept in a limited way transitions corresponding to an increase in the value of the cost function.

Definition

A metaheuristic is a high-level algorithmic framework or approach that can be specialised to solve optimisation problems.

Examples of metaheuristics are **simulated annealing** and **genetic algorithms**.

For the remainder of this optimisation section, we will focus on minimisation problems.