# Heuristic Optimisation

## Part 2: Basic concepts

Sándor Zoltán Németh

http://web.mat.bham.ac.uk/S.Z.Nemeth

s.nemeth@bham.ac.uk

University of Birmingham

## Overview

- Representation
- Objective
- Evaluation function
- Optimisation problem definition
- Neighbourhoods and local optima
- The hill-climbing method

## Introduction

*Problem* $\Rightarrow$ *Model* $\Rightarrow$ *Solution*

For any algorithmic approach:

- The representation - the encoding of alternative solutions
- The objective - the purpose (may or may not be minimising one function)
- The evaluation function - how good a solution (given the representation) is

## Representation

SAT problem:
$n$ binary variables $\Rightarrow$ a bit string of length $n$
Search space has size $2^n$, each point corresponds to a feasible solution

TSP:
A permutation of natural numbers $1, 2, \ldots, n$ (each number corresponds to one city)
Search space has size $(n-1)!/2$ for symmetric TSP

NLP:
All real numbers in $n$ dimensions
Infinite search space; for approximate representation with precision of six digits size $10^{7n}$

## Representation cont'd

The size of the search space is determined by the representation and its corresponding interpretation

The problem itself does not determine the search space size

Choosing the right representation is of utmost importance!

## Objective

The mathematical statement of the task

SAT: find the bit vector for which the Boolean statement evaluates to *TRUE*

TSP: minimise the total distance, when visiting each city exactly once and returning to the starting city

$min \sum dist(x, y)$

NLP: minimise or maximise a nonlinear function

## Evaluation function

Generally it is not the same thing as the objective

The evaluation function is a mapping from the space of (feasible) solutions to a set of numbers (reals). The numeric value indicates quality of solution

TSP: tour $\leftarrow$ the sum of distances along the route - exact

Sometimes it is sufficient to know whether one solution is better than the other, without knowing how much better

## How to choose the evaluation function?

The evaluation function is not given with the problem, only the objective is

- A solution that meets the objective should also have the best evaluation
- A solution that fails to meet the objective cannot be evaluated to be better than a solution that meets the objective
- The objective may suggest the evaluation function (TSP, NLP) or may not (SAT)
- The feasible region of the search space must be considered

## Defining the optimisation problem

Search problem = optimisation problem

Given a search space $S$ together with its feasible part $F \subseteq S$, find $x \in F$ such that

$eval(x) \leq eval(y), \forall y \in F$ (minimisation)

$x$ is a global solution

## Reasons for using heuristics

- We can avoid combinatorial explosion
- We rarely need the optimal solution, good approximations are usually sufficient
- The approximations may not be very good in the worst case, but in reality worst cases occur very rarely
- Trying to understand why a heuristic works/does not work leads to a better understanding of the problem

## Neighbourhoods

$dist : S \times S \to \mathbf{R}$
$N(x) = \{y \in S : dist(x, y) < \varepsilon\}$, given $\varepsilon \geq 0$

NLP: Euclidean distance
$dist(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$

SAT: Hamming distance (number of bit positions with different truth assignments)

## Neighbourhood as mapping

Alternatively the neighbourhood can be defined as a mapping

$m : S \rightarrow 2^S$

TSP: 2-swap mapping generates for any potential solution $x$ the set of potential solutions obtained by swapping two cities in $x$

SAT: 1-flip mapping (flipping one bit)

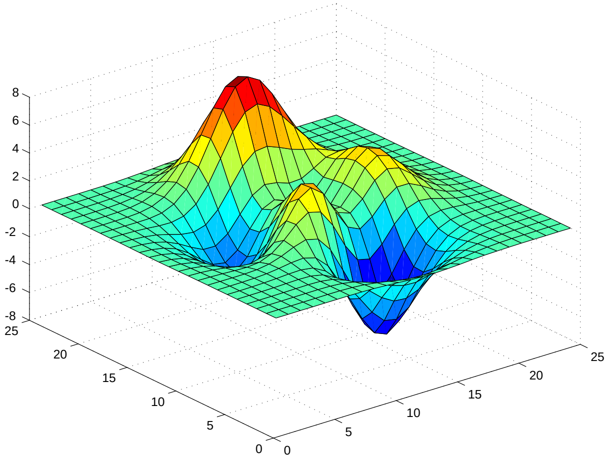
## Local optima

A potential solution $x \in F$ is a local optimum with respect to neighbourhood $N(x)$ iff

$eval(x) \leq eval(y), \forall y \in N(x)$

Local search methods operate on neighbourhoods:
They try to modify the current solution $x$ into
a better one within its neighbourhood $N(x)$

## Hill climbing



## Hill climbing analogy

Orientation and moving on a surface

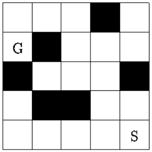
- Height: the quality of a node (the evaluation function)
- Peaks: optimal solutions
- Orientation: evaluating neighbouring positions

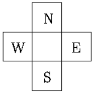
## Basic hill climbing algorithm

1. Evaluate the initial point $x$. If the objective is met, return $x$. Otherwise set $x$ as the current point.
2. Loop until a solution is found:
   (a) Generate all the neighbours of $x$. Select the best one $y$.
   (b) If $y$ is not better than $x$ return $x$
   else set $y$ as the current point.

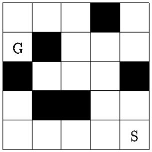
For better efficiency, the algorithm can be restarted a predefined number of times (iterative hill climbing)

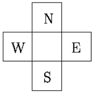
## Example of hill climbing

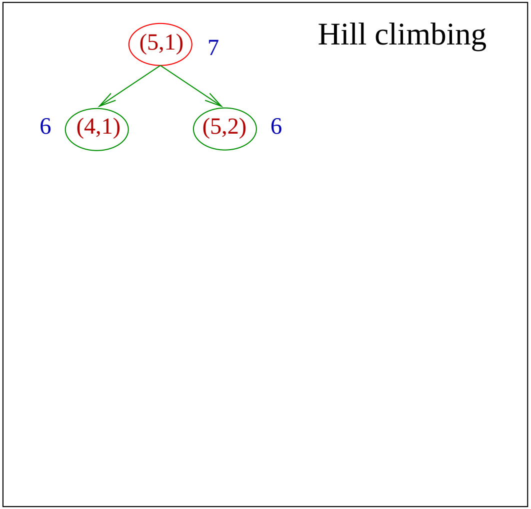




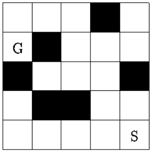
## Example of hill climbing

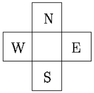


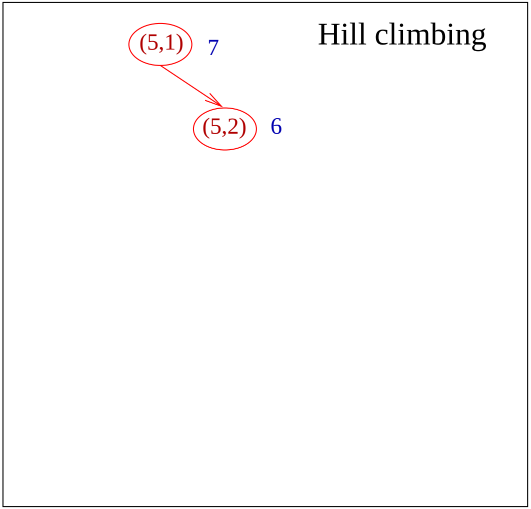




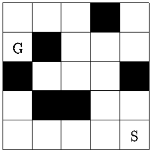
## Example of hill climbing

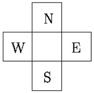


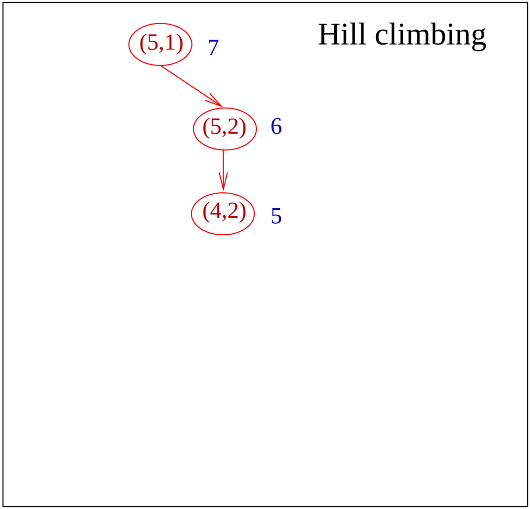




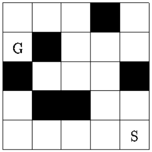
# Example of hill climbing

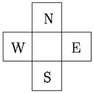


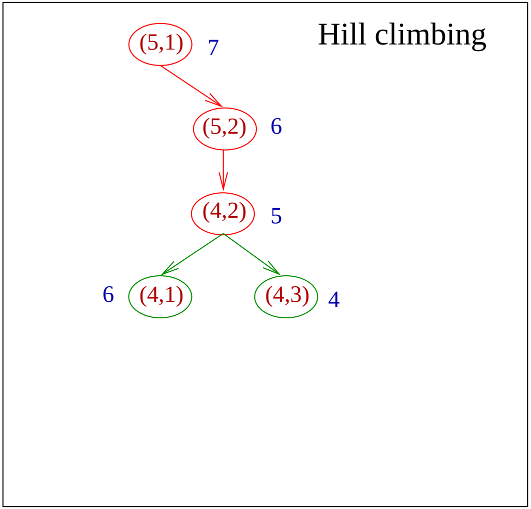




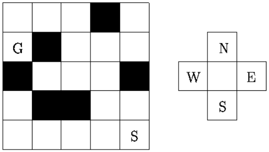
# Example of hill climbing

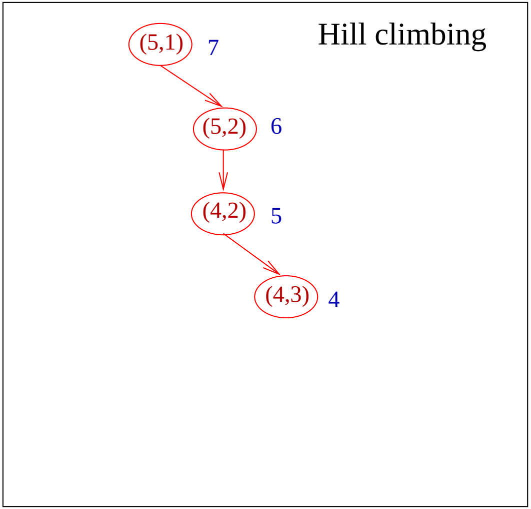




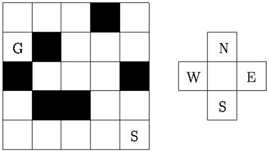


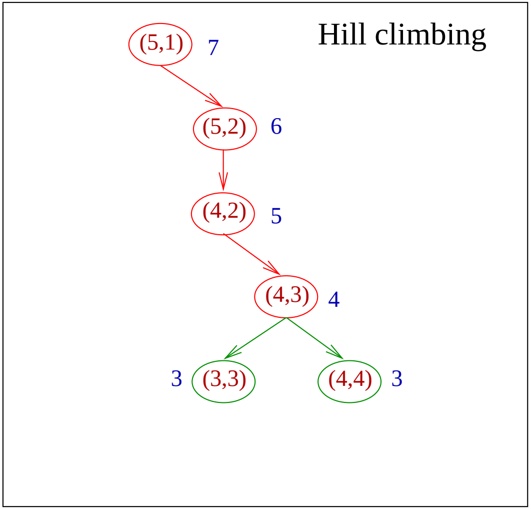
## Example of hill climbing



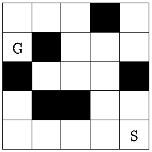


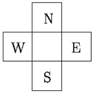
## Example of hill climbing

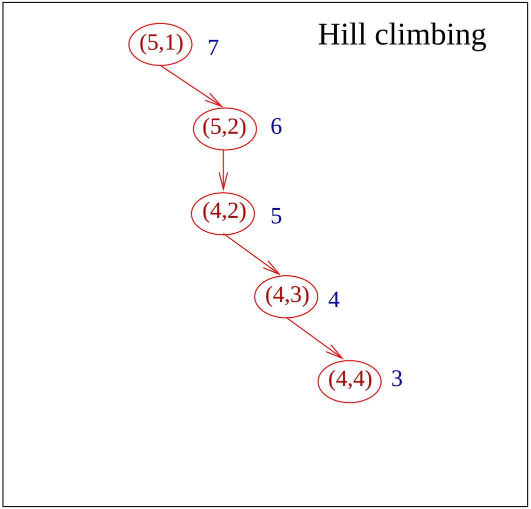




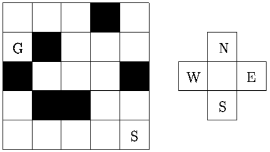
## Example of hill climbing

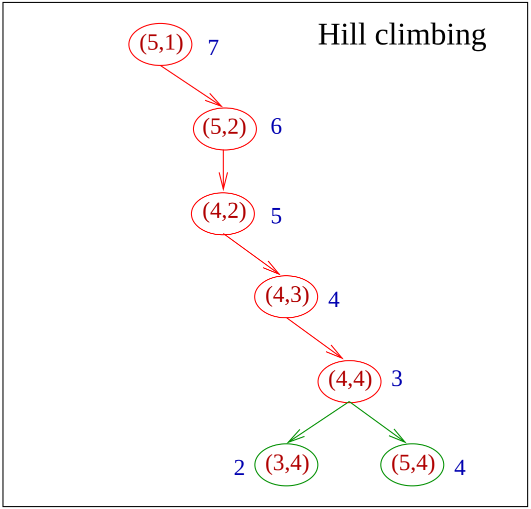




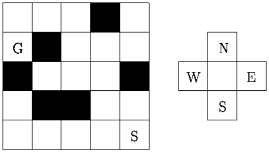


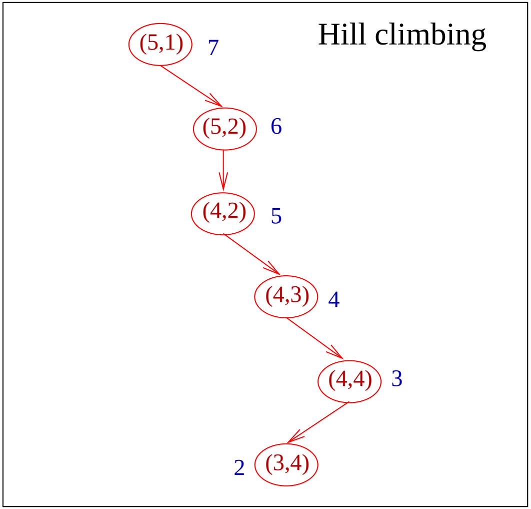
## Example of hill climbing





# Example of hill climbing





## Properties of hill climbing

- Can get stuck in local maxima $\longrightarrow$ backtrack
- Plateaus (areas of search space where the evaluation function is flat) are a problem $\longrightarrow$ big jumps
- There is no information about how far the found local optimum from the global optimum is
- The solution depends on the initial configuration

## Summary

Choosing the right representation is essential for success

The evaluation function must also be carefully chosen

Effective search balances exploitation of the best solutions so far and exploration of the search space

Hill climbing is an easy local search method that is good at exploitation but neglects exploration

Random search is good at exploration, but does no exploitation

## Recommended reading

Z. Michalewicz & D.B. Fogel
How to Solve It: Modern Heuristics

Chapter 2. Basic concepts