The role of Operations Research

Solving Decisional Models for Environmental Management
The search process

Control Variables

Simulation Model

$\mathbf{x}(t+1) = f(\mathbf{x}(t), u(t))$

Search algorithm

Modification of control variable $u(t)$

Simulation Results

Performance Criterion

$J = J(\mathbf{x}, u)$

Time series

Maps

$J_{\text{max}}$?

Search algorithm

Stop

yes

Adapted from Seppelt, 2003
What is ‘search’

• To search means to explore possible values for the control variable $u$, to obtain the value of the state $x$ caused by $u$, and to evaluate the performance $J$. 
The policy

\[ P \triangleq \{ M_t (\cdot) ; \ t = 0, 1, \ldots \} \]

- The policy defines the value of \( u \) to be chosen in front of the value of \( x \) at each time step \( t \) and in each spatial location \( z \).
Finding the policy

- The problem is hard
- effects are uncertain
- decisions are dynamic and recursive
The methodology

- Scenario Analysis
- Optimisation
Scenario analysis

• Scenario definition requires knowledge on the modelled system, which might not be available, due to complexity (Jakeman and Letcher, 2003)

• In some (most) cases Scenarios are too limited

• Too much weight on past experience
Optimisation

- Given the goal, the computer automatically generates scenarios, by combination of parameters, controls, exogenous inputs
- A numerical optimisation algorithm is in charge of evaluating the performance and directing exploration in the most promising direction
Simulation explained
The simulation model

\[ x_{t+\Delta t}(z) = M_{\Delta t}(x_t(z), u_t(z), \theta(z), z) \]

- It is a **constraint** for the optimisation problem
- all model variables may vary in space
- boundary and initial conditions are given
The performance criterion

\[ J(x_t(z), u_t(z)) \]

• state \( x \) and control \( u \) depend on time and location

• the performance criterion maps a trajectory from state and policy space to a real number
The problem

Model: \( x_{t+\Delta t}(z) = M_{\Delta t}(x_t(z), u_t(z), \theta(z), z) \)

Performance criterion: \( J(x_t(z), u_t(z)) \)

Problem: find \( u^* \) so that

\[
\begin{align*}
x(t) &= M_{\Delta t}(x, u^*, \theta) \\
J(x, u^*) &\geq J(x, u)
\end{align*}
\]

for all \( t \) in \([0, T]\) and \( u \) in \( U \)
Solving the problem

• Exact methods
• Approximation algorithms
• Heuristic and metaheuristic algorithms
Exact methods

- Exact algorithms by definition always find a solution, which is an optimal one.
- For many real-world problems, they may take too long to find an optimal solution, even on the fastest computers available.
Exact methods

• The algorithm depends on the problem formulation
• Linear programming, simplex
• Integer programming, branch & bound
• Dynamic programming
Approximation algorithms

- Find a sub-optimal solution
- It is possible to say how far from the optimum (e.g. 95%)
- Formally proven
- Often is very specific to the problem at hand
- E.g: neuro-dynamic programming
Heuristics

- Apply rules-of-thumb to the solution of the problem
- No guarantee to find an optimum
- No guarantee on the quality of the solution
- Yet, they are fast and generic
Metaheuristics

• set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems.

• general algorithmic frameworks which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem
Metaheuristics are...

- Simulated annealing
- Evolutionary computation
- Tabu search
- Ant Colony Systems
Ant Colony Systems
Ants

Leaf cutters, fungi growers

Breeding other Insects

Weaver ants
Insects, social insects, and ants

- $10^{18}$ living insects (estimate)
- ~2% of all insects are social
- Examples of social insects:
  - All ants
  - All termites
  - Some bees
  - Some wasps
- 50% of all social insects are ants
- Average body weight of an ant: 1 - 5 mg
- Total mass of ants ~ total mass of humans
- Ants have colonized earth for 100 million yrs; Homo sapiens sapiens for 50’000 years
Ant colony society

- Size of a colony: from a few (30) to millions of worker ants

- Labour division:
  - Queen: reproduction
  - Soldiers: defense
  - Specialised workers: food gathering, offspring tending, nest grooming, nest building and maintenance
How do ants coordinate their activities?

- Basic principle: **stigmergy**

- **stigmergy** is a kind of indirect communication, used by social insects to coordinate their activities.
Stigmergy

Stimulation of working ants due to the results they have reached. Grassé P., 1959
What are ant-based algorithms?

- Ant-based algorithms are multi-agent systems using artificial stigmergy
- To coordinate artificial ants to solve computational problems
“Artificial” Stigmergy

- Indirect communication by changes on the environment accessible only locally to communicating agents (Dorigo and Di Caro, 1999)

- Features of artificial stigmergy:
  - Indirect communication
  - Local accessibility
The bridge experiment

Goss et al., 1989, Deneubourg et al., 1990
Pheromone trail

Ants and termites follow pheromone trails
Asymmetric bridge experiment

Goss et al., 1989
Ant System for TSP

Travelling Salesman Problem
Dorigo, Maniezzo, Colorni, 1991
Gambardella & Dorigo, 1995

Pheromone trail deposition

Probabilistic rule to choose the path
Pheromone, euristics and memory to visit the next city

\[ p_{ij}^k(t) = f(\tau_{ij}(t), \eta_{ij}(t)) \]

Memory of visited cities
Pheromone trail deposition

\[ \tau_{ij}^k(t + 1) \leftarrow (1 - \rho) \cdot \tau_{ij}^k(t) + \Delta \tau_{ij}^k(t) \]

where \((i,j)\) are the links visited by ant \(k\), and

\[ \Delta \tau_{ij}^k(t) = \text{quality}^k \]

where \(\text{quality}^k\) is inversely proportional to the length of the solution found by ant \(k\).
The algorithm

- **Ants** depart from the depot choosing the next visit in the list of customers.
- **Ants** follow a probabilistic route as a function of:
  1. some *artificial pheromone values* and
  2. local *euristic values*,
- **Ants** memorise the current tour and the current travel time, taking into account the problem constraints (e.g. capacity, time windows).
- Once they have completed a tour, they update the global pheromone trail, in order to distribute the information gathered on the new solution.

*AntSystem* is distributed and not synchronised.
Why does it work?

• 3 main components:
  • TIME: a short route gets pheromone more quickly
  • QUALITY: a short route gets more pheromone
  • COMBINATORICS: a short path gets pheromone more frequently since it (usually) has a smaller number of decision points
Some results on the Travelling Salesman Problem

<table>
<thead>
<tr>
<th>Problem name</th>
<th>ACS</th>
<th>GA</th>
<th>EP</th>
<th>SA</th>
<th>Optimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eil50 (50-city problem)</td>
<td>425 (427.96) [1,830]</td>
<td>428 (N/A) [25,000]</td>
<td>426 (427.86) [100,000]</td>
<td>443 (N/A) [68,512]</td>
<td>425 (N/A)</td>
</tr>
<tr>
<td>Eil75 (75-city problem)</td>
<td>535 (542.37) [3,480]</td>
<td>545 (N/A) [80,000]</td>
<td>542 (549.18) [325,000]</td>
<td>580 (N/A) [173,250]</td>
<td>535 (N/A)</td>
</tr>
<tr>
<td>KroA100 (100-city problem)</td>
<td>21,282 (21,285.44) [4,820]</td>
<td>21,761 (N/A) [103,000]</td>
<td>N/A</td>
<td>N/A</td>
<td>21,282 (N/A)</td>
</tr>
</tbody>
</table>
A constructive heuristics with local search

• The best strategy to approximate the solution of a combinatorial optimisation problem is to couple
  • a constructive heuristics and
  • local search

• It is hard to find the best coupling:
  • Ant System (and similar algorithms) appear to have found it
The ACO Metaheuristics

• Ant System has been extended, to be applied to any problem formulated as a shortest path problem

• The extension has been named “Optimisation metaheuristics based on Ant Colonies (ACO)”

• The main application:
  • NP complex combinatorial optimisation problems
The ACO-metaheuristic procedure

procedure ACO-metaheuristic()
  while (not-termination-criterion)
    planning subroutines
      generate-and-manage-ants()
      evaporate-pheromone()
    execute-daemon-actions() {opzionale}
  end planning subroutines
  end while
end procedure
ACO: Applications

- Sequential ordering in a production line
- Vehicle Routing of trucks goods distributions
- Job-shop scheduling
- Project scheduling
- Water distribution problems
Genetic Algorithms
Evolutionary computing

- EC (Evolutionary computing) =
  - GA (Genetic Algorithms - Holland, 1975)
  - ES (Evolution Strategies - Rechenberg 1973)
  - EP (Evolutionary Programming - Fogel, Owens, Walsh, 1966)
  - GP (Genetic Programming - Koza, 1992)
Biological basis

• Evolution operates on chromosomes, which ‘encode’ the structure of a living being

• Natural selection favours reproduction of the most efficient chromosomes

• Mutations (new chromosomes) and recombination (mixing chromosomes) occur during reproduction
Use

- GA are very well suited when the structure of the search space is not well known.
- In other words, if the model of our system is rough, we can use GA to find the best policy.
How it works

- A genetic algorithm maintains a population of candidate solutions for the problem at hand, and makes it evolve by iteratively applying a set of stochastic operators.
The skeleton of the algorithm

- Generate initial population $P(0)$
- $t := 0$
- while not converging do
  - evaluate $P(t)$
  - $P'(t) \leftarrow$ select best individuals from $P(t)$
  - $P''(t) \leftarrow$ apply reproduction on $P'(t)$
  - $P(t+1) \leftarrow$ replace individuals $(P(t), P''(t))$
- end while
Components of a GA

- Encoding principles (gene, chromosome)
- Initialization procedure (creation)
- Selection of parents (reproduction)
- Genetic operators (mutation, recombination)
- Evaluation function (environment)
- Termination condition
Representation (encoding)

- Possible individual’s encoding
  - Bit strings (0101 ... 1100)
  - Real numbers (43.2 -33.1 ... 0.0 89.2)
  - Permutations of element (E11 E3 E7 ... E1 E15)
  - Lists of rules (R1 R2 R3 ... R22 R23)
  - Program elements (genetic programming)
  - ... any data structure ...
Choosing the encoding

- Use a data structure as close as possible to the natural representation
- Write appropriate genetic operators as needed
- If possible, ensure that all genotypes correspond to feasible solutions
- If possible, ensure that genetic operators preserve feasibility
Initialization

• Start with
  • a random population
  • a previously saved population
  • a set of solutions provided by a human expert
  • a set of solutions provided by another heuristic
Selection

- **Purpose:** to focus the search in promising regions of the space
- **Inspiration:** Darwin’s “survival of the fittest”
- **Trade-off between exploration and exploitation of the search space**
Linear Ranking Selection

Based on sorting of individuals by decreasing fitness. The probability to be extracted for the $i$th individual in the ranking is defined as

$$p(i) = \frac{1}{n} \left[ \beta - 2(\beta - 1) \frac{i - 1}{n - 1} \right], 1 \leq \beta \leq 2$$

where $\beta$ can be interpreted as the expected sampling rate of the best individual
Local Tournament Selection

- Extracts k individuals from the population with uniform probability (without re-insertion) and makes them play a “tournament”,

- where the probability for an individual to win is generally proportional to its fitness

- Selection pressure is directly proportional to the number k of participants
Recombination (Crossover)

- Enables the evolutionary process to move toward promising regions of the search space
- Matches good parents’ sub-solutions to construct better offspring
Mutation

• Purpose: to simulate the effect of errors that happen with low probability during duplication

• Results:
  • Movement in the search space
  • Restoration of lost information to the population
Evaluation (fitness function)

• Solution is only as good as the evaluation function; choosing a good one is often the hardest part

• Similar-encoded solutions should have a similar fitness
Termination condition

• Examples:
  • A pre-determined number of generations or time has elapsed
  • A satisfactory solution has been achieved
  • No improvement in solution quality has taken place for a pre-determined number of generations
Acknowledgements

• Part of the material extracted from ‘Introduction to Genetic Algorithms’ by Assaf Zaritsky, Ben Gurion University,
Simulated Annealing
The origin

- It is the oldest metaheuristic
- Originated in statistical mechanics (Metropolis Monte Carlo algorithm)
- First presented to solve combinatorial problems by Kirkpatrick et al. 1983
The idea

• It searches in directions which result in solutions that are of worse quality than the current solution

• It allows to escape local minima

• The probability of the exploration of these ‘unpromising’ directions is decremented during the search
The algorithm

- \( s \leftarrow \text{GenerateInitialSolution}() \)
- \( \text{Temp} := \text{InitialTemp} \)
- while not converging loop
  - \( s' \leftarrow \text{PickAtRandomFromNeighbor}(s) \)
  - if \( J(s') < J(s) \)
    - \( s \leftarrow s' \)
    - end if
  - else
    - Accept \( s' \) as new solution with prob \( p(T,s',s) \)
    - Update(\( T \))
  - end else
- end while
Boltzmann probability

• if s’ is worse than s, then s’ might still be chosen as the new solution

• the probability depends on d=f(s’)-f(s) and on temperature T

• the higher d the lower the probability

• the higher T the higher the probability
Boltzmann probability

\[ P(s \rightarrow s') \sim e^{-\frac{f(s')-f(s)}{T}} \]

- It determines the equilibrium distribution of a system in various energy states at a given temperature
- Temperature \( T \) decreases during the search process (similarity with annealing process of metal and glass)
Pros and Cons

• Pros
  • proven to converge to the optimum
  • easy to implement

• Cons
  • converges in infinite time
  • the cooling process must be slow
End of Part II