Metaheuristics
Single State Local Search Methods, part 2

Fernando Lobo

Universidade do Algarve

Variable Depth Search

- Another strategy for escaping a local optimum
- A complex search step is made from a variable-length sequence of simple steps in a small neighbourhood
- Then, the algorithm performs iterative improvement upon these complex search steps
- That is, if the application of a complex search step yields an improvement, we accept the move. Otherwise we reject it.

Variable Depth Search: Magic Square Example

- A simple neighbourhood can be the set of solutions that can be obtained from a current one by swapping the contents of two cells.
- A complex search step can be made out of a sequence of k steps using the simple neighbourhood
  - k can be sampled from a probability distribution each time a complex step is built

Variable Depth Search: TSP Example

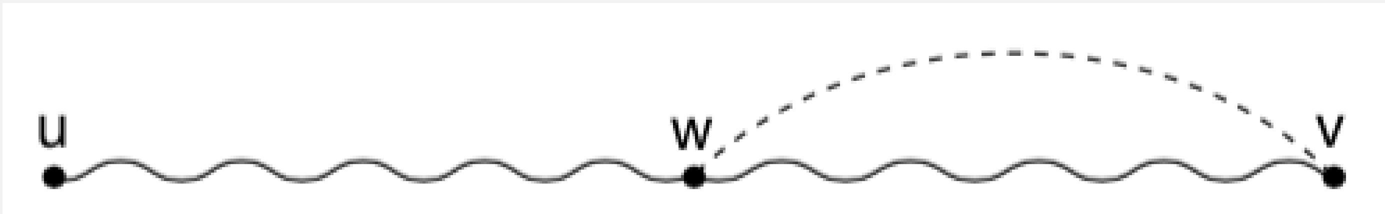
- Complex step can be made from a sequence of 2-exchange steps, as follows:

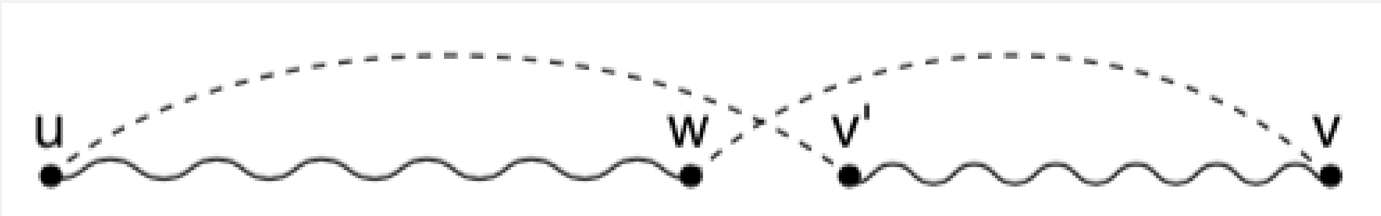
  1. Start from a Hamiltonian path $(u, \ldots, v)$
     - ★ A Hamiltonian path is a path that visits every vertex of the graph exactly once
     - ★ This Hamiltonian path is obtained by removing edge (u, v) from a valid tour
  2. Add an edge (v, w) creating a cycle.
  3. Break the cycle by removing an edge incident to w, call it (w, v'). This yields a new Hamiltonian path.
     - ★ We can get a new Hamiltonian cycle (a new valid tour) by adding edge (v', u)

Variable Depth Search: TSP Example

- Graphically:

1. 

2. 

3. 

Images taken from Hoos and Stützle's book

Variable Depth Search: Lin-Kernighan heuristic for TSP

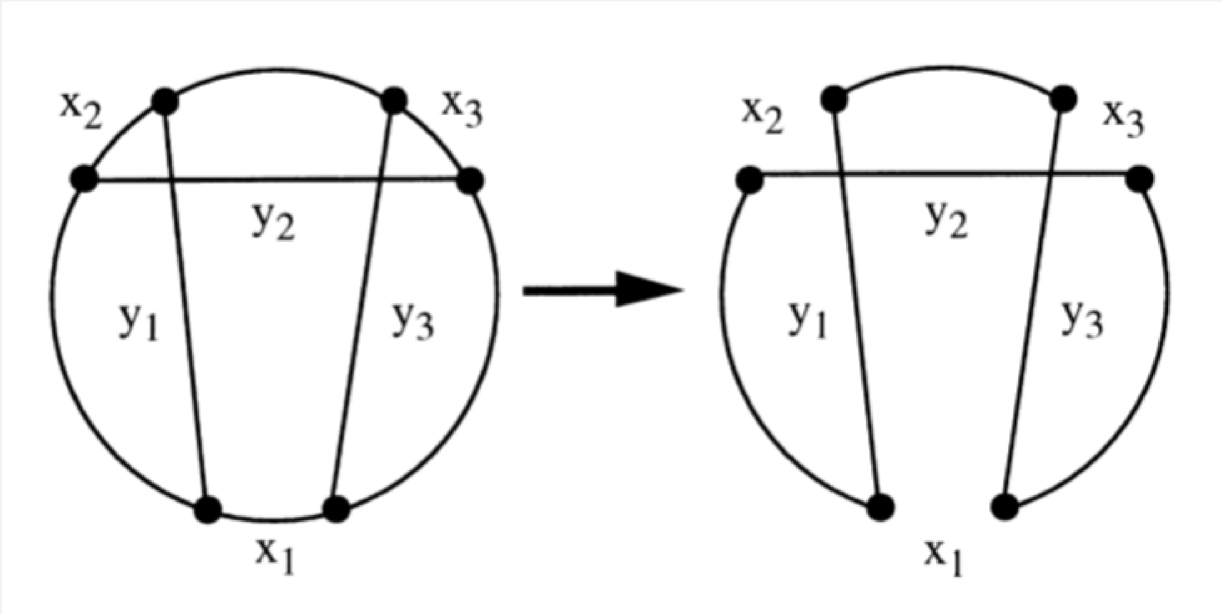
- The Lin-Kernighan heuristic is one of the most effective methods for the TSP
- It is based on the idea illustrated in the previous slides: it repeatedly performs edge exchanges that reduce the length of the tour
- Some restrictions apply on the exchanges:
  - Only sequential exchanges are allowed
  - The gain must be positive
  - A removed edge cannot be added (in the same complex step)
  - An added edge cannot be removed (in the same complex step)

Variable Depth Search: Lin-Kernighan heuristic for TSP

- In a complex step, a set of r edges are removed $X = \{x_1, \ldots, x_r\}$ and a set of r edges are added $Y = \{y_1, \ldots, y_r\}$
- X and Y are built incrementally, both starting from the empty set
- The sequential exchange criterion enforces that x_i and y_i must share an endpoint
- The positive gain criterion enforces that adding edge y_i and removing edge x_i results in a tour with a shorter distance

Variable Depth Search: Lin-Kernighan heuristic for TSP

Example of a sequencial exchange with $r = 3$



(Image taken from Keld Helsgaun, *An effective implementation of the Lin-Kernighan traveling salesman heuristic.* Eur. J. Oper. Res. 126(1): 106-130 (2000))

Variable Depth Search: Lin-Kernighan heuristic for TSP

- There are several other enhancements on top of it
- You may find more details in Hoos and Stützle's book, as well as in the following reference:
  - Keld Helsgaun, *An effective implementation of the Lin-Kernighan traveling salesman heuristic.* Eur. J. Oper. Res. 126(1): 106-130 (2000)

Strategies to escape local optima

- Using restarts (e.g. multi-start hillclimbing)
- Exploring larger neighbourhoods
- What else?

Exploration vs. Exploitation

- Need to find a balance between exploring new regions of the search space and exploiting a region that was already found to be reasonably good
- Hillclimbing represents the extreme case of pure exploitation
- Blind search represents the other extreme of pure exploration
- Some authors also refer to this as the *Diversification vs. Intensification* dilemma

Non-improving moves

- We can allow (sometimes) non-improving moves
- This means accepting a neighbour that is worse or equal than the current solution
- Does not guarantee to always escape from a local optimum

Randomised Iterative Improvement

- At each search step, perform an uninformed random walk step with a fixed probability p. How?
  - generate a random number $r \in [0..1[$
  - if $r < p$ perform the uninformed random walk step. Otherwise do an iterative improvement step
  - An iterative improvement step accepts the better of x and its neighbour
- p is a parameter of the algorithm and must be set by the user
- In theory, can always escape from a local optimum
- In practice, it may take a long time to do so ...

What’s the difference between theory and practice?

“In theory there is no difference between theory and practice. In practice there is.”

– Yogi Berry

Probabilistic Iterative Improvement

- Accept a non-improving move with a probability that depends in the amount of deterioration of the current solution
  - The larger the deterioration, the smaller the probability

Metropolis Probabilistic Iterative Improvement

- A special case of Probabilistic Iterative Improvement
- Given a current solution s, we obtain a neighbour $x \in \mathcal{N}(s)$
- Then we accept x with probability p defined as follows:

$$p = \begin{cases} 1 & \text{if } f(x) \leq f(s) \\ e^{\frac{f(s)-f(x)}{T}} & \text{otherwise} \end{cases}$$

- This is called the *Metropolis condition*, where T is the so-called "temperature" parameter that controls the probability of accepting worsening moves

Metropolis condition

$$p = \begin{cases} 1 & \text{if } f(x) \leq f(s) \\ e^{\frac{f(s)-f(x)}{T}} & \text{otherwise} \end{cases}$$

- Some notes:
    - Formula above is for minimization problems
    - $T > 0$
    - If x is better or equal than s, we always accept it
    - If x is worse than s, then $f(s) - f(x) < 0$, which implies $0 < p < 1$
    - As the difference between $f(s)$ and $f(x)$ increases, p approaches 0

Simulated Annealing

- Proposed by Kirkpatrick, Gelatt and Vecchi (1983)
- The idea is to lower the value of the temperature parameter T as the search takes place
- This is typically done according to a *cooling schedule*
- Inspired by the physical process of annealing metals
  - Annealing steel involves heating it to a specified temperature and then letting it cool down at a very slow and controlled rate
  - Perfect ground states are achieved by lowering the temperature very slowly

Simulated Annealing: cooling schedule

- Start with an initial temperature $T = T_0$
  - T_0 usually depends on the problem instance
- Update T every k iteration steps
  - k is called the *temperature length*
  - k is usually proportional to the neighbourhood size
  - update typically according to an exponential cooling regime:
    - $T = \alpha \cdot T$, with $0 < \alpha < 1$
- Termination criterion:
  - maximum number of iterations (or time)
  - a certain acceptance ratio threshold

Example for TSP

- Taken from Johnson and McGeoch, *The Traveling Salesman Problem: A Case Study in Local Optimization* (1997)
  - 2-exchange neighbourhood
  - T_0 chosen empirically such that only 3% of moves are rejected
  - $k = n \cdot (n - 1)$, with n being the number of cities
  - Cooling schedule: $T = 0.95 \cdot T$
  - Termination: when 5 successive temperature values yield no solution quality improvement and the acceptance ratio is less than 2%

Tabu Search

- Proposed by Fred Glover (1989)
- Also tries to escape from local optima
- Uses additional memory to avoid visiting places that have been visited recently
  - These places become tabu (i.e. forbidden)
  - We can associate tabu status to entire candidate solutions, as well as to solution components
- Unlike Simulated Annealing, only accepts worsening moves when it is stuck at a local optimum
  - SA accepts them probabilistically

Tabu Search

- In its simplest form:
  - Maintain a tabu list of length L of candidate solutions seen so far
  - When a new candidate solution is obtained, it goes to the tabu list
  - If the list is full, we remove the oldest candidate solution
    - that solution is no longer tabu

Tabu Search

- Working at the level of complete candidate solutions is not that effective.
- For large search spaces, we may stay in the same “hill”, even if the tabu list is very large!
- A more common strategy is to use the tabu concept for solution components
- Sometimes we make an exception to things that are in the tabu list (*aspiration criterion*)

Tabu Search: TSP example

- If we obtain a solution by removing a certain edge, we forbid the addition of that edge in future moves for a certain number of time steps
  - This number of time steps is specified by a *tabu tenure* parameter
- Side note: the Lin-Kerninghan heuristic does something similar (before Tabu Search was invented)

Tabu Search: SAT example

- Use a 1-bit Hamming distance neighborhood
- Associate a tabu status (true or false) with each variable of formula F
- A variable is considered tabu if and only if it has been changed in the last tt steps
- We make an exception if flipping a tabu variable yields a solution better than any solution seen so far
- Among all admissible neighbours choose u.a.r. among those that satisfy the most clauses

Tabu Search: SAT example

- Use appropriate data structures to allow an incremental evaluation of the objective function
  - For each variable, keep a list (or pointers) to the clauses that contain that variable
  - For each variable x, store the iteration number when its value was last changed. Let such number be x_t
  - Then, x is tabu if and only if $t - x_t < tt$, where t is the current iteration number

Iterated Local Search (ILS)

- ILS can be seen as a generalization of a restart strategy
- What are the major drawbacks of a pure restart strategy?
  - For many problems, local optima tend to be clustered in the search space
  - Restarting from another candidate solution u. a. r. may be too time consuming
  - We are basically restarting from scratch every single time

Iterated Local Search (ILS)

- A better strategy might be to restart from another solution that is not so far from the current local optimum
- We can do that my doing a *perturbation* of the current solution:
  - that goes beyond the solution's neighbourhood
  - but not so much as being equivalent to a complete restart

Iterated Local Search (ILS)

- ILS has a nice blend of exploration and exploitation
- uses local search to obtain a local optimum (exploitation)
- once there, uses a perturbation operator to escape from it (exploration)
- then, uses local search again to obtain a (perhaps new) local optimum, and the process goes on and on
- ILS has an *acceptance criterion* to decide which of the two local optima should it proceed from

Iterated Local Search (ILS)

- ILS may use aspects of the search history to decide how to do the perturbation
- Common strategy:
  - If the local search keeps reaching the same local optimum, then make a larger perturbation

Iterated Local Search (ILS)

- ILS may also use aspects of the search history to decide which local optimum should it proceed from
- Common strategies:
  - Accept the better of the two local optima
    - performs iterative improvement in the space of local optima (more exploitation)
  - Accept the last local optimum
    - performs a random walk in the space of local optima (more exploration)
  - Do something in between

Iterated Local Search

$s = \mathit{generateInitialSolution}()$
$s = \mathit{localSearch}(s)$
while *not terminate*() **do**
    $t = \mathit{perturb}(s, \mathit{history})$
    $t = \mathit{localSearch}(t)$
    $s = \mathit{accept}(s, t, \mathit{history})$
end

Iterated Local Search (ILS)

- ILS makes trajectories in the space of local optima
- We can build an effective ILS with little code by combining existing local search algorithms

Iterated Local Search (ILS): TSP example

TSP Example (taken from Hoos and Stützle's book):

- *generateInitialSolution*(): greedy heuristic
- *localSearch*(): Lin-Kernighan heuristic
- *perturb*(): double-bridge move
- *accept*(): accept t if t is better or equal than s, otherwise accept s

Double bridge move

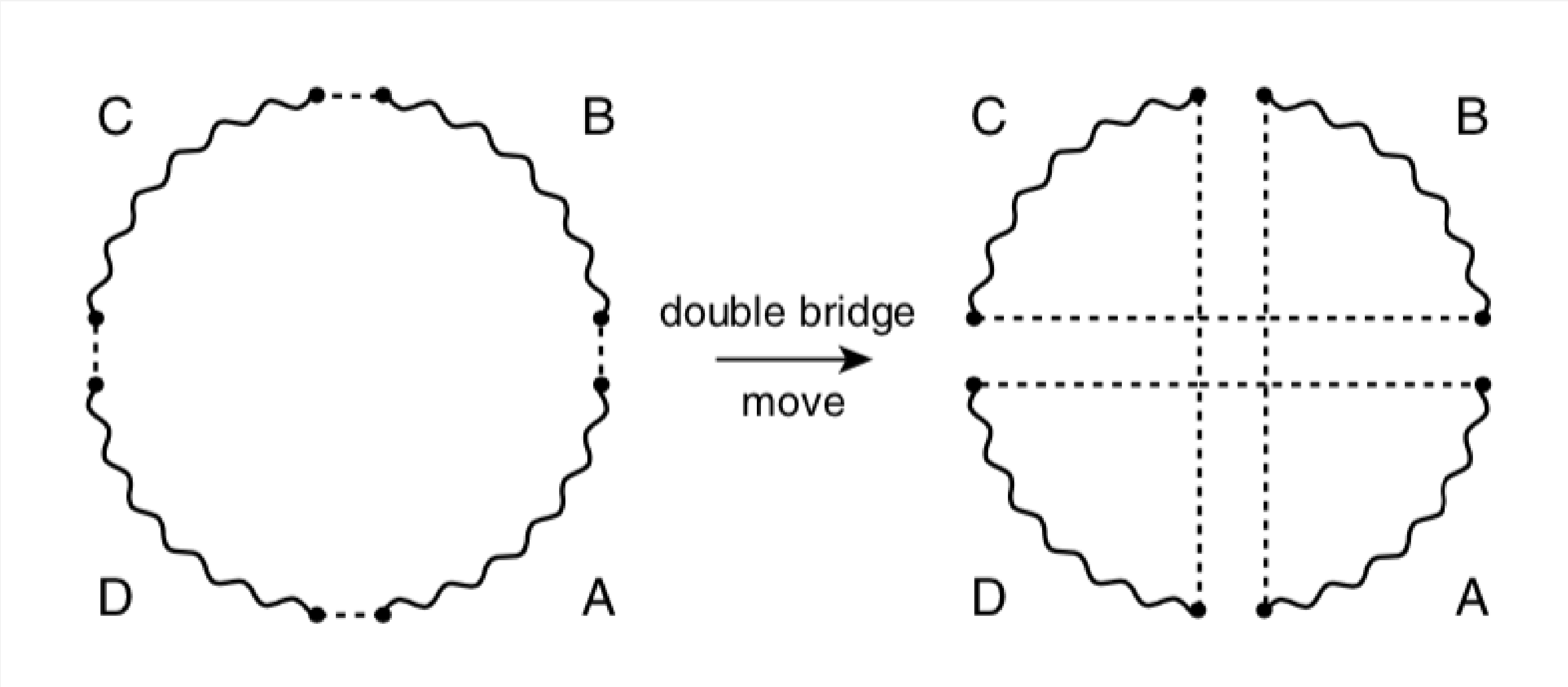


Figure taken from Hoos and Stützle's book

Iterated Local Search (ILS): SAT example

- *generateInitialSolution*(): random initial solution
- *localSearch*(): next-ascent hillclimbing based on 1-flip neighbourhood
- *perturb*(): random k-bit flip move, with $k >> 2$
- *accept*(): accept t if t is better or equal than s, otherwise accept s