LOCAL SEARCH METHODS
APPLICATIONS AND ENGINEERING

Lecture 4

Search Landscape (continued) and 'Simple' Local Search Methods

Marco Chiarandini

Outline

1. Plateaux
2. Barriers and Basins
3. Construction Heuristics and Perturbative Searches for SMTWTP and GCP
4. Iterative Improvement Extensions

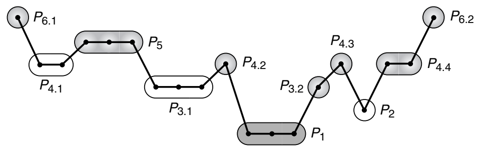
Outline

1. Plateaux
2. Barriers and Basins
3. Construction Heuristics and Perturbative Searches for SMTWTP and GCP
4. Iterative Improvement Extensions

Plateaux

Plateaux, *i.e.*, 'flat' regions in the search landscape, are characteristic for the neutral landscapes obtained for combinatorial problems such as SAT.

Intuition: Plateaux can impede search progress due to lack of guidance by the evaluation function.



Definitions

- **Region:** connected set of search positions.
- **Border of region R:** set of search positions with at least one direct neighbour outside of R (**border positions**).
- **Plateau region:** region in which all positions have the same level, *i.e.*, evaluation function value, l.
- **Plateau:** maximally extended plateau region, *i.e.*, plateau region in which no border position has any direct neighbours at the plateau level l.
- **Solution plateau:** Plateau that consists entirely of solutions of the given problem instance.
- **Exit of plateau region R:** direct neighbour s of a border position of R with lower level than plateau level l.
- **Open / closed plateau:** plateau with / without exits.

Measures of plateau structure:

- *plateau diameter* = diameter of corresponding subgraph of G_N
- *plateau width* = maximal distance of any plateau position to the respective closest border position
- *plateau branching factor* = fraction of neighbours of a plateau position that are also on the plateau.
- *number of exits*, *exit density*
- *distribution of exits within a plateau*, *exit distance distribution* (in particular: avg./max. distance to closest exit)

Some plateau structure results for SAT:

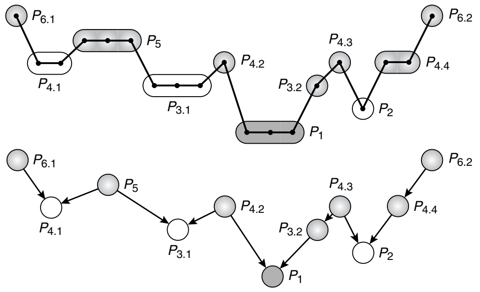
- Plateaux typically don't have an interior, *i.e.*, almost every position is on the border.
- The diameter of plateaux, particularly at higher levels, is comparable to the diameter of search space. (In particular: plateaux tend to span large parts of the search space, but are quite well connected internally.)
- For open plateaux, exits tend to be clustered, but the average exit distance is typically relatively small.

Idea: Obtain abstract view of neutral landscape by collapsing positions on the same plateau into 'macro positions'.

Plateau connection graphs (PCGs):

- *Vertices:* plateaux of given landscape
- *Edges (directed):* connect plateaux that are directly connected by one or more exit.
- Additionally, *edge weights* can be used to indicate the relative numbers of exits from one plateau to its PCG neighbours.

Example: Simple landscape L and plateau connection graph PCG...



Note: The plateaux form a partition of L, *i.e.* every position in L is part of exactly one (possibly degenerate) plateau.

Outline

Barriers and Basins

Observation:

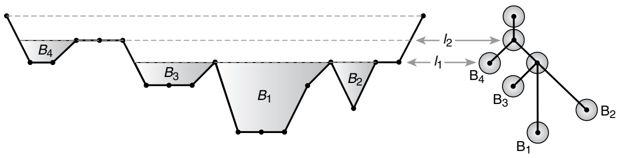
The *difficulty of escaping* from closed plateaux or strict local minima is related to the *height of the barrier*, *i.e.*, the difference in evaluation function, that needs to be overcome in order to reach better search positions:

Higher barriers are typically more difficult to overcome (this holds, *e.g.*, for Probabilistic Iterative Improvement or Simulated Annealing).

Definitions:

- Positions s, s' are *mutually accessible at level* l iff there is a path connecting s' and s in the neighbourhood graph that visits only positions t with $g(t) \leq l$.
- The *barrier level between positions* s, s', $bl(s, s')$ is the lowest level l at which s' and s' are mutually accessible; the difference between the level of s and $bl(s, s')$ is called the *barrier height between* s *and* s'.
- The *depth of a position* s is the minimal barrier height between s and any position s' at a level lower than s, *i.e.*, for which $g(s') < g(s)$.
- **Basins**, *i.e.*, maximal (connected) regions of search positions below a given level, form an important basis for characterising search space structure.

Example: Basins in a simple search landscape and corresponding basin tree

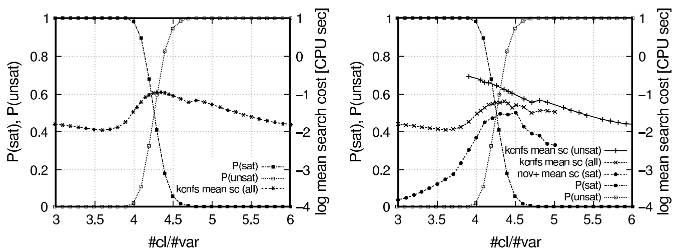


Note: The basin tree only represents basins just below the critical levels at which neighbouring basins are joined (by a *saddle*).

Note:

- Like plateau connection graphs, basin trees can provide much deeper insights into SLS behaviour and problem hardness than global measures of search space structure, such as FDC or ACC.
- **But:** This type of analysis is computationally expensive, since it requires enumeration (or sampling) of large parts of the search space.

Phase Transition for 3-SAT



Outline

Single Machine Total Weighted Tardiness Problem

Construction Heuristics

- Earliest due date
- Modified due date
- Apparent urgency

Iterative Improvement

- Interchange (size $n(n-1)/2$; delta computation in $O(|i-j|)$; complete examination $O(n^3)$)
- Insert (size $(n-1)^2$; delta computation in $O(|i-j|)$; complete examination $O(n^3)$)
- Swap (size $n-1$; delta computation in $O(1)$; complete examination $O(n)$)

Speed-ups:

- Interchange fast branching
- Insert exploration as sequence of swaps

Graph Colouring Problem

Construction Heuristics

- Greedy Algorithm. Vertex order decided:
  - static order (random, largest degree, smallest last degree)
  - dynamic order (largest saturation first)
- Recursive Largest First

Iterative Improvement

Three approaches:

- k fixed, complete (improper) colourings ($g(s) = |E^c|$ or $g(s) = |V^c|$)
  - One exchange
  - Two exchange (but quadratic complexity, hence less used)
- k fixed, partial (proper) colourings ($g(s) = \sum_{v \in C_{imp}} d_{C_{imp}}(v)$)
  - i-swap
- k variable, complete colourings ($g(s) = -\sum_{i=1}^{k} |C_i| + 2 \sum_{i=1}^{k} |C_i||E_i|$)
  - Kempe Chains

Outline

1. Plateaux
2. Barriers and Basins
3. Construction Heuristics and Perturbative Searches for SMTWTP and GCP
4. Iterative Improvement Extensions

Variable Neighborhood Descent

- *Recall:* Local minima are relative to neighborhood relation.
- **Key idea:** To escape from local minimum of given neighbourhood relation, switch to different neighbhourhood relation.
- Use k neighbourhood relations $N_1, \ldots, N_k$, (typically) ordered according to increasing neighbourhood size.
- Always use smallest neighbourhood that facilitates improving steps.
- Upon termination, candidate solution is locally optimal w.r.t. all neighbourhoods

Variable Neighbourhood Descent (VND):

determine initial candidate solution s
$i := 1$
Repeat:
  choose a most improving neighbour s' of s in N_i
  If $g(s') < g(s)$:
    $s := s'$
    $i := 1$
  Else:
    $i := i + 1$
Until $i > k$

Note:

- VND often performs substantially better than simple II or II in large neighbourhoods [Hansen and Mladenović, 1999]
- Many variants exist that switch between neighbhourhoods in different ways.
- More general framework for LS algorithms that switch between multiple neighbourhoods: *Variable Neighbourhood Search (VNS)* [Mladenović and Hansen, 1997].

Variable Depth Search

- **Key idea:** *Complex steps* in large neighbourhoods = variable-length sequences of *simple steps* in small neighbourhood.
- Use various *feasibility restrictions* on selection of simple search steps to limit time complexity of constructing complex steps.
- Perform Iterative Improvement w.r.t. complex steps.

Variable Depth Search (VDS):

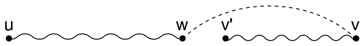
determine initial candidate solution s
$\hat{t} := s$
While s is not locally optimal:
  Repeat:
    select best feasible neighbour t
    If $g(t) < g(\hat{t})$: $\hat{t} := t$
  Until construction of complex step has been completed
  $s := \hat{t}$

Example: The Lin-Kernighan (LK) Algorithm for the TSP (1)

- Complex search steps correspond to sequences of 2-exchange steps and are constructed from sequences of *Hamiltonian paths* (= paths that visit every node in given graph exactly once).
- *δ-path:* Hamiltonian path p + 1 edge connecting one end of p to interior node of p ('lasso' structure):

Basic LK exchange step:

- Start with Hamiltonian path $(u, \ldots, v)$:
- Obtain δ-path by adding an edge (v, w):
- Break cycle by removing edge (w, v'):



- *Note:* Hamiltonian path can be completed into Hamiltonian cycle by adding edge (v', u):

Construction of complex LK steps:

1. start with current candidate solution (Hamiltonian cycle) s; set $t^* := s$; set $p := s$
2. obtain δ-path p' by replacing one edge in p
3. consider Hamiltonian cycle t obtained from p by (uniquely) defined edge exchange
4. if $w(t) < w(t^*)$ then set $t^* := t$; $p := p'$; go to step 2
5. else accept t^* as new current candidate solution s

Note: This can be interpreted as sequence of 1-exchange steps that alternate between δ-paths and Hamiltonian cycles.

Additional mechanisms used by LK algorithm:

- *Tabu restriction:* Any edge that has been added cannot be removed and any edge that has been removed cannot be added in the same LK step.
  Note: This limits the number of simple steps in a complex LK step.
- *Limited form of backtracking* ensures that local minimum found by the algorithm is optimal w.r.t. standard 3-exchange neighbourhood
- (For further details, see text book)

Note:

Variable depth search algorithms have been very successful for other problems, including:

- the Graph Partitioning Problem [Kernigan and Lin, 1970];
- the Unconstrained Binary Quadratic Programming Problem [Merz and Freisleben, 2002];
- the Generalised Assignment Problem [Yagiura *et al.*, 1999].