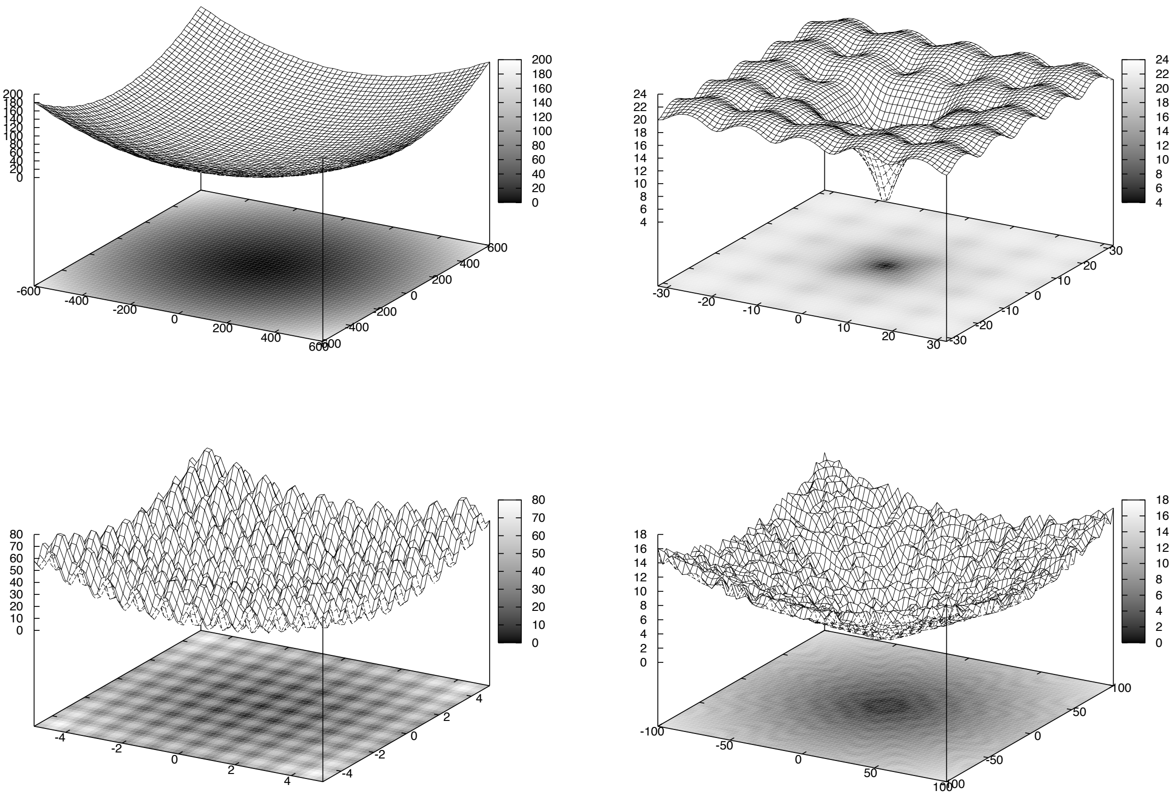
HEURISTIC OPTIMIZATION

Search Space Analysis

slightly adapted from slides for SLS:FA, Chapter 5



Fundamental Search Space Properties

The behaviour and performance of an SLS algorithm on a given problem instance crucially depends on properties of the respective search space.

Simple properties of search space S:

- search space size $\#S$
- search space diameter $diam(G_N)$
  (= maximal distance between any two candidate solutions)

Note: The diameter of a given search space depends on the *neighbourhood size*.

Example: Search space size and diameter for the TSP

- **Given:** Symmetric TSP instance with n vertices.
- Candidate solutions = permutations of vertices
- Search space size $= (n-1)!/2$
- Size of 2-exchange neighbourhood
  $= \binom{n}{2} = n \cdot (n-1)/2$
- Size of 3-exchange neighbourhood
  $= \binom{n}{3} = n \cdot (n-1) \cdot (n-2)/6$
- Diameter of neighbourhood graphs: Exact values unknown.
  - Bounds for 2-exchange neighbourhood $= [n/2, n-1]$
  - Bounds for 3-exchange neighbourhood $= [n/3, n-1]$

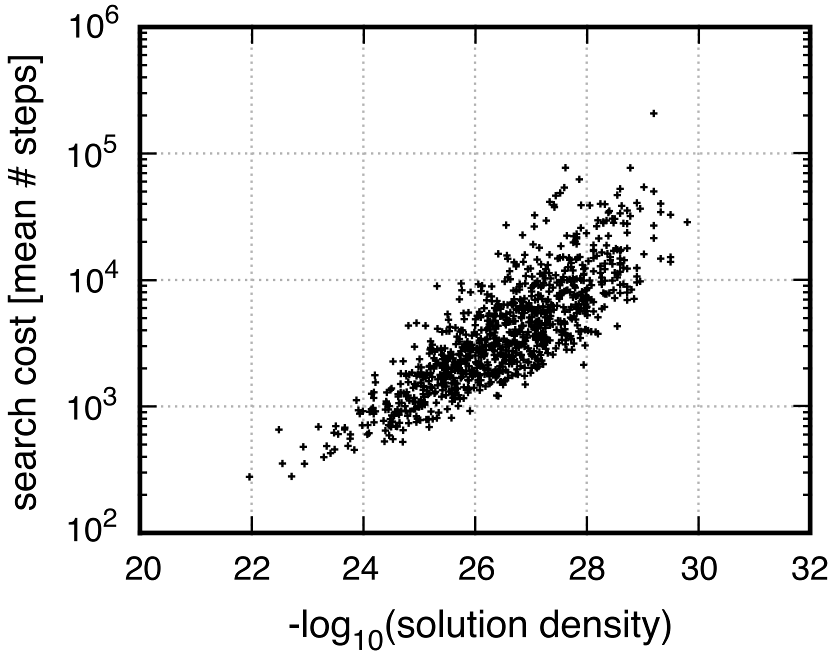
Simple properties of search space S (continued):

- number of (optimal) solutions $\#S'$, *solution density* $\#S'/\#S$
- distribution of solutions within the neighbourhood graph

Note:

- Solution densities and distributions can generally be determined by:
  - exhaustive enumeration;
  - sampling methods;
  - counting algorithms (often variants of complete algorithms).
- In many cases, (optimal) solutions tend to be clustered; this is reflected in uneven distributions of pairwise distances between solutions.

Example: Correlation between solution density and search cost for GWSAT over set of hard Random-3-SAT instances:



Search Landscapes

The behaviour of all but the simplest SLS algorithms depends on an *evaluation function* that guides the search process.

Definition:

Given an SLS algorithm A and a problem instance π with associated

- search space $S(\pi)$,
- neighbourhood relation $N(\pi)$,
- evaluation function $g(\pi) : S \mapsto \mathbb{R}$

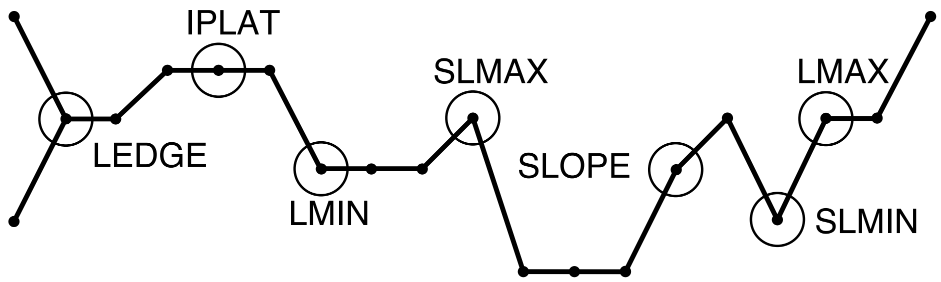
the *search landscape of* π, $L(\pi)$, is defined as $L(\pi) := (S(\pi), N(\pi), g(\pi))$.

Classification of search positions (according to evaluation function values of direct neighbours):

position type	$>$	$=$	$<$
SLMIN (strict local min)	+	–	–
LMIN (local min)	+	+	–
IPLAT (interior plateau)	–	+	–
SLOPE	+	–	+
LEDGE	+	+	+
LMAX (local max)	–	+	+
SLMAX (strict local max)	–	–	+

"+" = present, "–" absent; table entries refer to neighbours with larger (">") , equal ("="), and smaller ("<") evaluation function values

Example for various types of search positions:



Example: Complete distribution of position types for hard Random-3-SAT instances

instance	*avg sc*	SLMIN	LMIN	IPLAT
`uf20-91/easy`	13.05	0%	0.11%	0%
`uf20-91/medium`	83.25	$< 0.01\%$	0.13%	0%
`uf20-91/hard`	563.94	$< 0.01\%$	0.16%	0%

instance	SLOPE	LEDGE	LMAX	SLMAX
`uf20-91/easy`	0.59%	99.27%	0.04%	$< 0.01\%$
`uf20-91/medium`	0.31%	99.40%	0.06%	$< 0.01\%$
`uf20-91/hard`	0.56%	99.23%	0.05%	$< 0.01\%$

(based on exhaustive enumeration of search space;
sc refers to search cost for GWSAT)

Example: Sampled distribution of position types for hard Random-3-SAT instances

instance	*avg sc*	SLMIN	LMIN	IPLAT
`uf50-218/medium`	615.25	0%	47.29%	0%
`uf100-430/medium`	3 410.45	0%	43.89%	0%
`uf150-645/medium`	10 231.89	0%	41.95%	0%

instance	SLOPE	LEDGE	LMAX	SLMAX
`uf50-218/medium`	$< 0.01\%$	52.71%	0%	0%
`uf100-430/medium`	0%	56.11%	0%	0%
`uf150-645/medium`	0%	58.05%	0%	0%

(based on sampling along GWSAT trajectories;
sc refers to search cost for GWSAT)

Local Minima

Note: Local minima impede local search progress.

Simple measures related to local minima:

- number of local minima $\#lmin$, *local minima density* $\#lmin/\#S$
- distribution of local minima within the neighbourhood graph

Problem: Determining these measures typically requires exhaustive enumeration of search space.

Solution: Approximation based on sampling or estimation from other measures (such as autocorrelation measures, see below).

Example: Distribution of local minima for the TSP

Goal: Empirical analysis of distribution of local minima for Euclidean TSP instances.

Experimental approach:

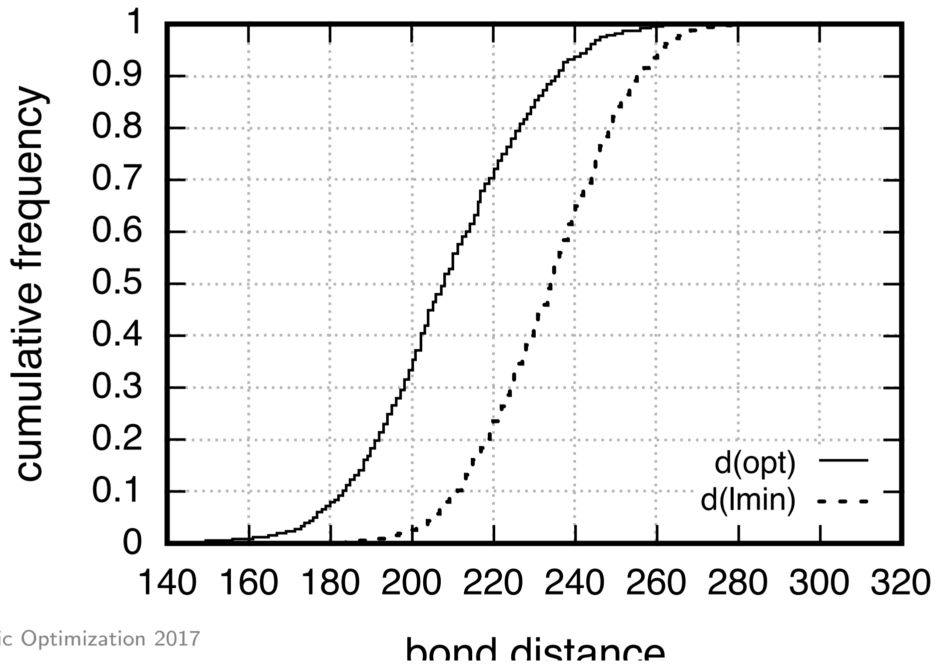
- Sample sets of local optima of three TSPLIB instances using multiple independent runs of two TSP algorithms (3-opt, ILS).
- Measure pairwise distances between local minima (using *bond distance* = number of edges in which two given tours differ).
- Sample set of purportedly globally optimal tours using multiple independent runs of high-performance TSP algorithm.
- Measure minimal pairwise distances between local minima and respective closest optimal tour (using bond distance).

Empirical results:

Instance	avg sq [%]	avg d_{lmin}	avg d_{opt}
	Results for 3-opt		
rat783	3.45	197.8	185.9
pr1002	3.58	242.0	208.6
pcb1173	4.81	274.6	246.0
	Results for ILS algorithm		
rat783	0.92	142.2	123.1
pr1002	0.85	177.2	143.2
pcb1173	1.05	177.4	151.8

(based on local minima collected from 1 000/200 runs of 3-opt/ILS)

Distribution of distances between local optima and to closest global optimum:



Interpretation:

- Average distance between local minima is small compared to maximal possible bond distance, n.

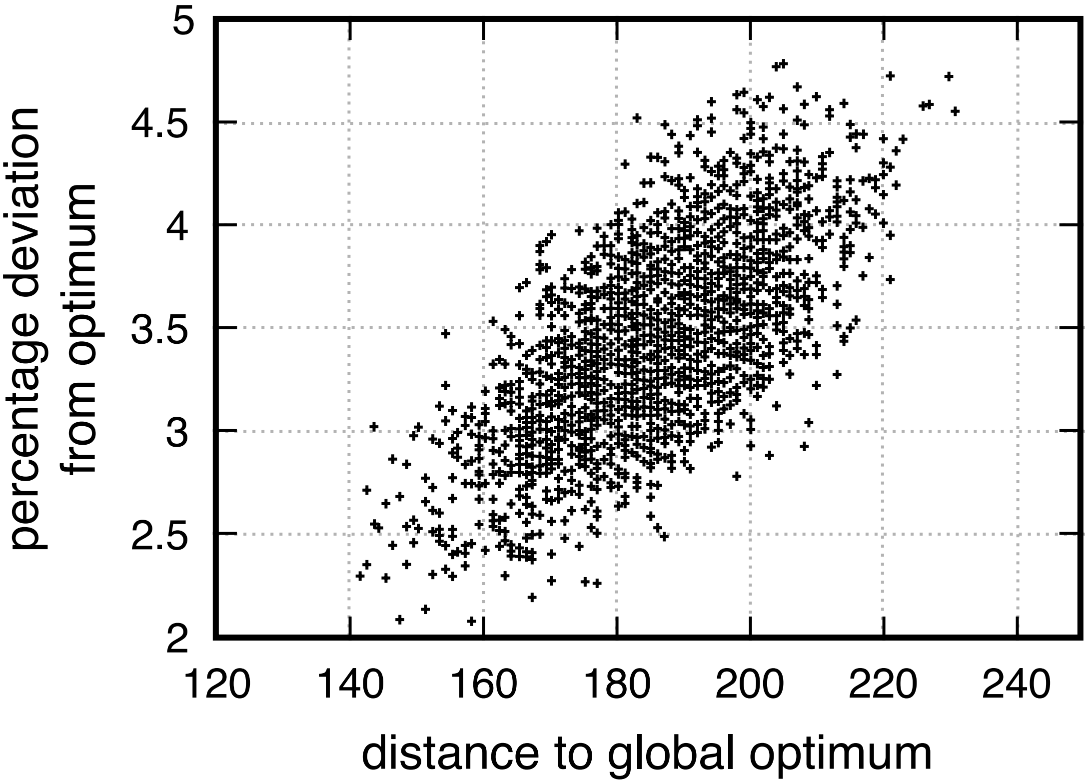
  $\Rightarrow$ *Local minima are concentrated in a relatively small region of the search space.*

- Average distance between local minima is slightly larger than distance to closest global optimum.

  $\Rightarrow$ *Optimal solutions are located centrally in region of high local minima density.*

- Higher-quality local minima found by ILS tend to be closer to each other and the closest global optima compared to those determined by 3-opt.

  $\Rightarrow$ *Higher-quality local minima tend to be concentrated in smaller regions of the search space.*



Fitness-Distance Correlation (FDC)

Idea: Analyse correlation between solution quality (fitness) g of candidate solutions and distance d to (closest) optimal solution.

Measure for FDC: *empirical correlation coefficient*

$$r_{fdc} := \frac{\widehat{Cov}(g, d)}{\widehat{\sigma}(g) \cdot \widehat{\sigma}(d)},$$

where

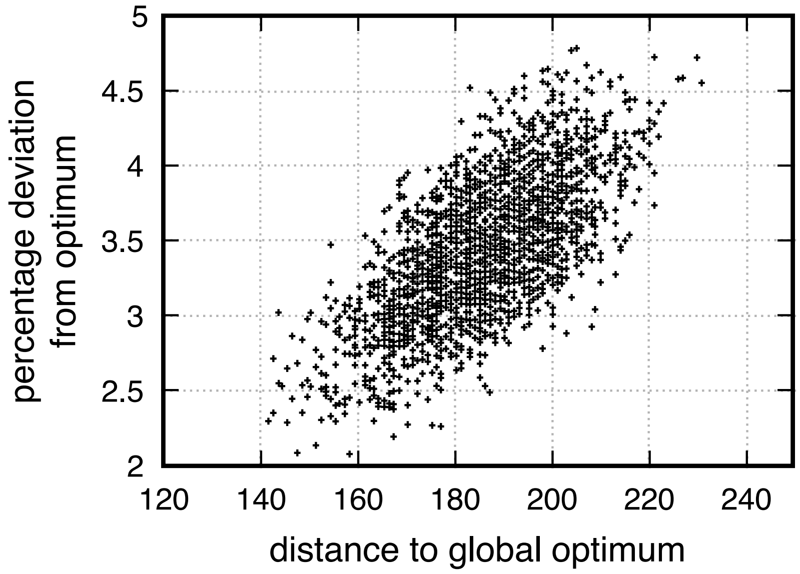
$$\widehat{Cov}(g, d) := \frac{1}{m-1} \sum_{i=1}^{m} (g_i - \bar{g})(d_i - \bar{d}),$$

$$\widehat{\sigma}(g) := \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (g_i - \bar{g})^2}, \quad \widehat{\sigma}(d) := \sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (d_i - \bar{d})^2}$$

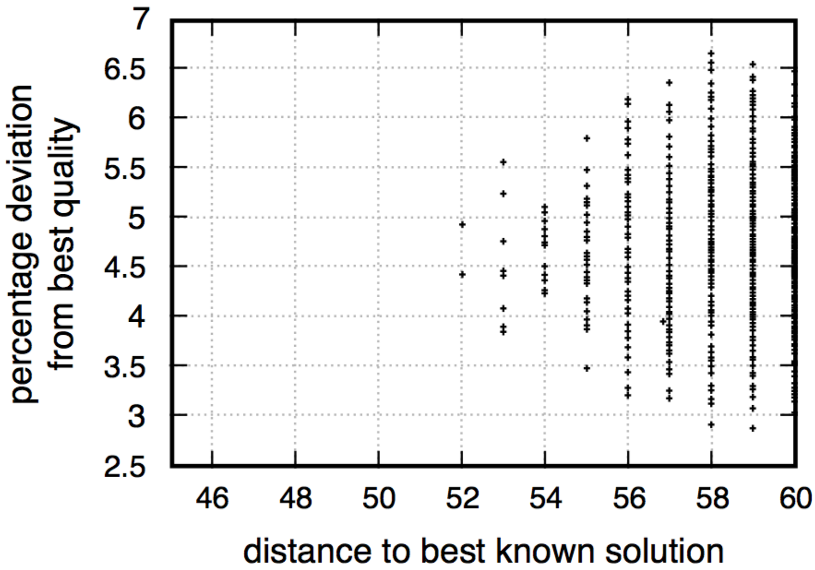
Note:

- The FDC coefficient, r_{fdc} depends on the given neighbourhood relation.
- r_{fdc} is calculated based on a sample of m candidate solutions (typically: set of local optima found over multiple runs of an iterative improvement algorithm).
- *Fitness-distance plots*, *i.e.*, scatter plots of the (g_i, d_i) pairs underlying an estimate of r_{fdc}, are often useful to graphically illustrate fitness distance correlations.

Example: FDC plot for TSPLIB instance `rat783`, based on 2 500 local optima obtained from a 3-opt algorithm



Example: FDC plot for QAPLIB instance `tai60a`, based on 1 000 local optima obtained from a 2-opt algorithm



High FDC (r_{fdc}:

- 'Big valley' structure of landscape provides guidance for local search;
- search initialisation: high-quality candidate solutions provide good starting points;
- search diversification: (weak) perturbation is better than restart;
- typical, *e.g.*, for TSP.

Low FDC (r_{fdc}:

- global structure of landscape does not provide guidance for local search;
- typical for very hard combinatorial problems, such as certain types of QAP (Quadratic Assignment Problem) instances.

Applications of fitness-distance analysis:

- algorithm design: use of strong intensification (including initialisation) and relatively weak diversification mechanisms;
- comparison of effectiveness of neighbourhood relations;
- analysis of problem and problem instance difficulty.

Limitations and short-comings:

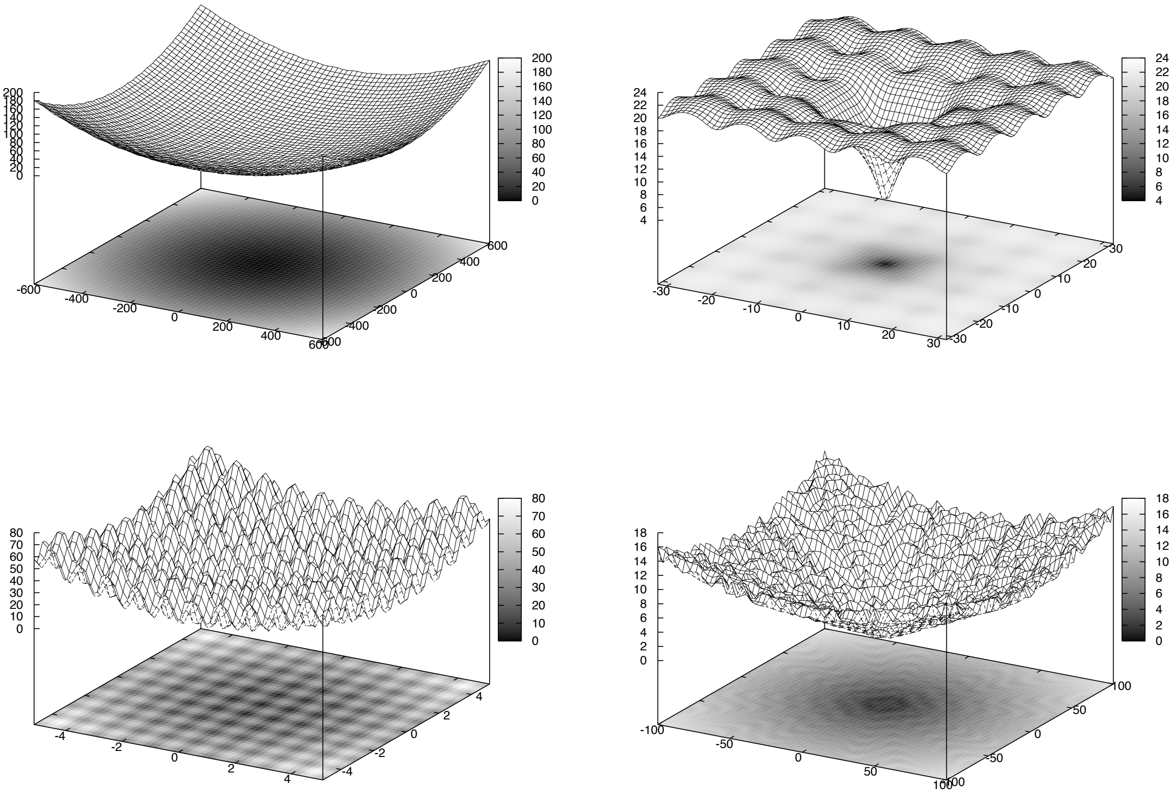
- *a posteriori* method, requires set of (optimal) solutions, **but:** results often generalise to larger instance classes;
- optimal solutions are often not known, using best known solutions can lead to erroneous results;
- can give misleading results when used as the sole basis for assessing problem or instance difficulty.

Ruggedness

Idea: Rugged search landscapes, *i.e.*, landscapes with high variability in evaluation function value between neighbouring search positions, are hard to search.

Example: Smooth *vs* rugged search landscape

Note: Landscape ruggedness is closely related to local minima density: rugged landscapes tend to have many local minima.



The ruggedness of a landscape L can be measured by means of the *empirical autocorrelation function $r(i)$*:

$$r(i) := \frac{1/(m-i) \cdot \sum_{k=1}^{m-i} (g_k - \bar{g}) \cdot (g_{k+i} - \bar{g})}{1/m \cdot \sum_{k=1}^{m} (g_k - \bar{g})^2}$$

where $g_1, \ldots g_m$ are evaluation function values sampled along an uninformed random walk in L.

In many cases empirical autocorrelation function shows decay in form of $r(i) = e^{-i/l}$, where l is called empirical correlation length.

This can be transformed to

$$l := 1/(\ln(|r(1)|)$$

Note: $r(i)$ depends on the given neighbourhood relation and instance size.
Note: l is therefore often rescaled by diameter of neighborhood graph to $l' := l/diam(G_N(\pi)$

Large l or l':

- "smooth" landscape;
- evaluation function values for neighbouring candidate solutions are close on average;
- low local minima density;
- problem typically relatively easy for local search.

Small l or l':

- very rugged landscape;
- evaluation function values for neighbouring candidate solutions are almost uncorrelated;
- high local minima density;
- problem typically relatively hard for local search.

Note:

- Empirical autocorrelation analysis is computationally cheap compared to, *e.g.*, fitness-distance analysis.
- (Bounds on) correlation length / $r(1)$ can be theoretically derived in many cases, *e.g.*, the TSP with the 2-exchange neighbourhood.
- There are other measures of ruggedness, such as *(empirical) correlation length*.

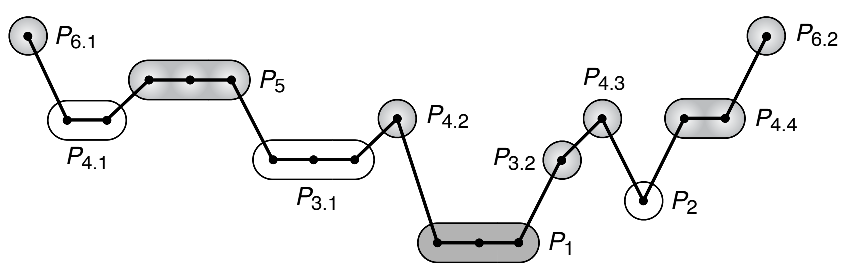
Note:

- Measures of ruggedness are often insufficient for distinguishing between the hardness of individual
problem instances;
- but they can be useful for
  - analysing differences between neighbourhood relations for a given problem,
  - studying the impact of parameter settings of a given SLS algorithm on its behaviour,
  - classifying the diffculty of combinatorial problems.

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.



Definition

- *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.

Definition

- *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:

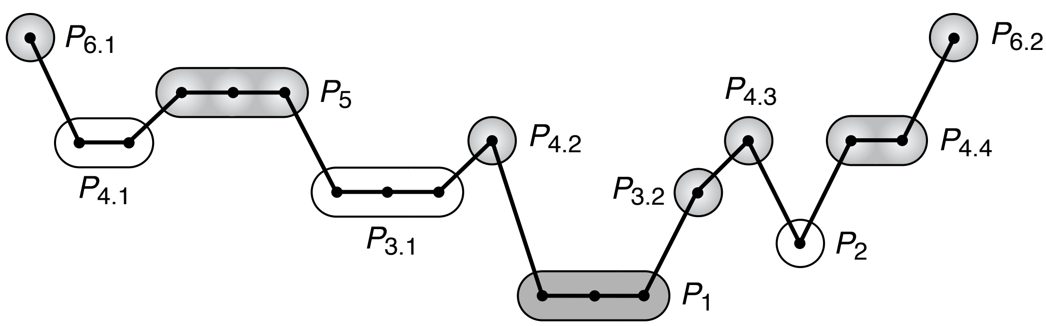
- Plateaux typically don't have an interior, *i.e.*, almost every position is on the border.
- The diameter of plateaux, particularly at higher levels, is comparable to the diameter of search space. (In particular: plateaux tend to span large parts of the search space, but are quite well connected internally.)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

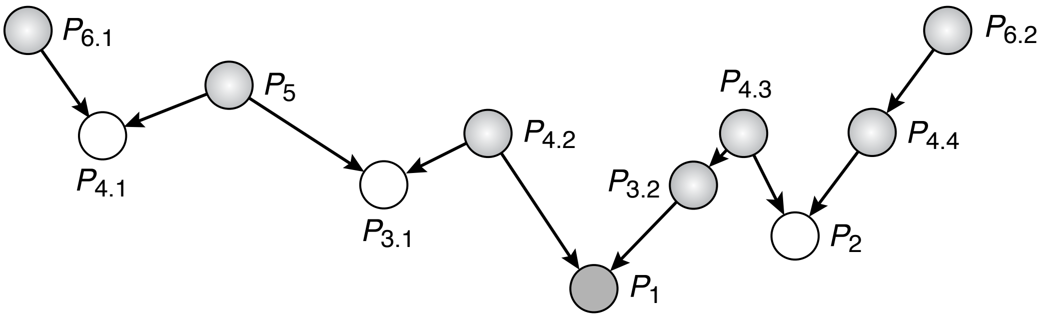
- *Vertices:* plateaux of given landscape
- *Edges (directed):* connect plateaux that are directly connected by one or more exit.
- Additionally, *edge weights* can be used to indicate the relative numbers of exits from one plateau to its PCG neighbours.

Example: Simple neutral search landscape L ...

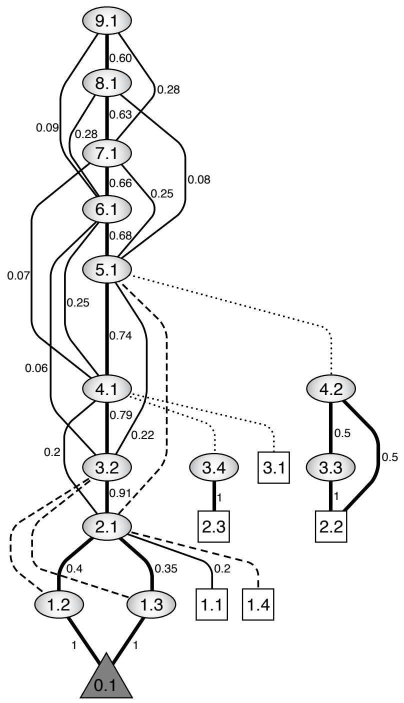


Note: The plateaux form a partition of L, *i.e.* every position in L is part of exactly one (possibly degenerate) plateau.

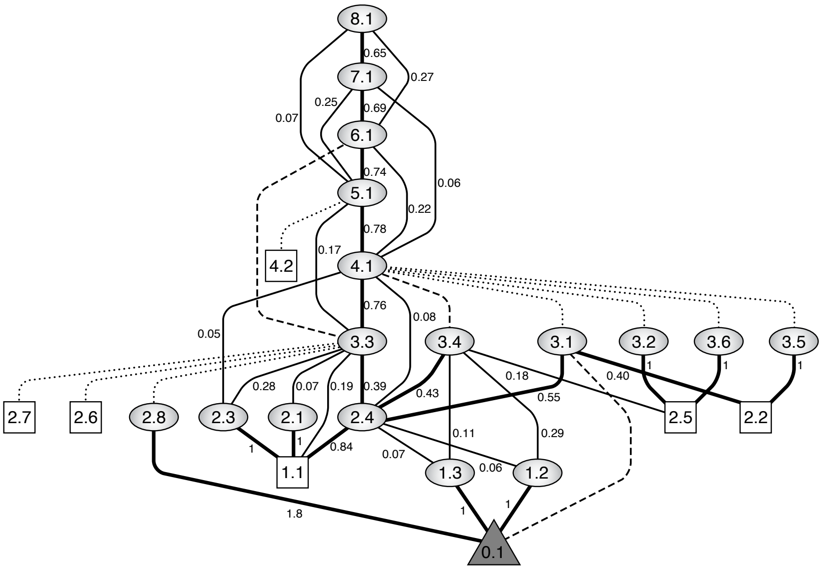
Example: . . . and the respective plateau connection graph



Example: PCG of *easy* Random 3-SAT instance



Example: PCG of *hard* Random 3-SAT instance



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).

Definition:

- 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;

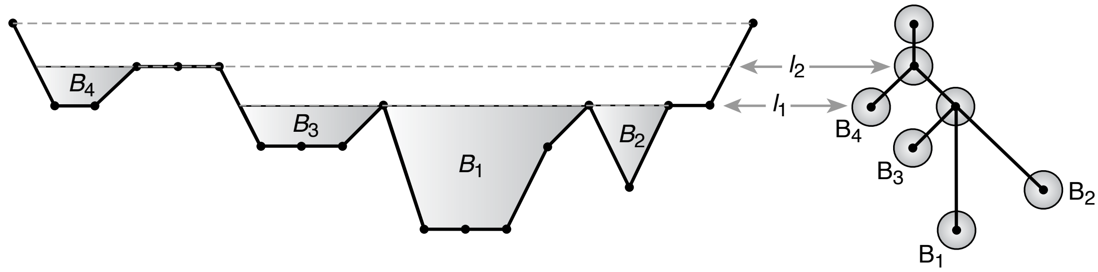
  the difference between the level of s and $bl(s, s')$ is called the *barrier height between s and s'*.
- The *depth of a position s* is the minimal barrier height between s and any position s' at a level lower than s, *i.e.*, for which $g(s') < g(s)$.

Basins, *i.e.*, maximal (connected) regions of search positions below a given level, form an important basis for characterising search space structure.

Note:

- Basins of a given landscape form a *hierarchy*, *i.e.*, two basins are either disjoint, or one is contained in the other.
- Basin hierarchies can be formally represented as *basin trees*.

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.