LOCAL SEARCH METHODS
APPLICATIONS AND ENGINEERING

Lecture 3

Perturbative Search and Search Landscape

Marco Chiarandini

Outline

Solution Representations and Neighborhoods

Three different types of solution representations:

- Single Machine Total Weighted Tardiness Problem: *linear permutation*
- Traveling Salesman Problem: *circular permutation*
- Graph Coloring Problem: *assignment*

A neighborhood function $N : S \rightarrow S \times S$ is also defined through an operator. An operator Δ is a collection of operator functions $\delta : S \rightarrow S$ such that

$$s' \in N(S) \quad \Longleftrightarrow \quad \exists \delta \in \Delta | \delta(s) = s'$$

Permutations

$\Pi(n)$ indicates the set all permutations of the numbers $\{1, 2, \ldots, n\}$

$(1, 2 \ldots, n)$ is the identity permutation ι.

If $\pi \in \Pi(n)$ and $1 \leq i \leq n$ then:

- π_i is the element at position i
- $pos_\pi(i)$ is the position of element i

Alternatively, a permutation is a bijective function $\pi(i) = \pi_i$, then the permutation product $\pi \cdot \pi'$ is the composition $(\pi \cdot \pi')_i = \pi'(\pi(i))$
For each π there exists a permutation such that $\pi^{-1} \cdot \pi = \iota$

Neighborhood Operators for Linear Permutations

Swap operator

$$\Delta_S = \{\delta_S^i | 1 \leq i \leq n\}$$

$$\delta_S^i(\pi_1 \ldots \pi_i \pi_{i+1} \ldots \pi_n) = (\pi_1 \ldots \pi_{i+1} \pi_i \ldots \pi_n)$$

Interchange operator

$$\Delta_X = \{\delta_X^{ij} | 1 \leq i < j \leq n\}$$

$$\delta_X^{ij}(\pi) = (\pi_1 \ldots \pi_{i-1} \pi_j \pi_{i+1} \ldots \pi_{j-1} \pi_i \pi_{j+1} \ldots \pi_n)$$

Insert operator

$$\Delta_I = \{\delta_I^{ij} | 1 \leq i \leq n, 1 \leq j \leq n, j \neq i\}$$

$$\delta_I^{ij}(\pi) = \begin{cases} (\pi_1 \ldots \pi_{i-1} \pi_{i+1} \ldots \pi_j \pi_i \pi_{j+1} \ldots \pi_n) & i < j \\ (\pi_1 \ldots \pi_j \pi_i \pi_{j+1} \ldots \pi_{i-1} \pi_{i+1} \ldots \pi_n) & i > j \end{cases}$$

Neighborhood Operators for Circular Permutations

Reversal (2-edge-exchange)

$$\Delta_R = \{\delta_R^{ij} | 1 \leq i < j \leq n\}$$

$$\delta_R^{ij}(\pi) = (\pi_1 \ldots \pi_{i-1} \pi_j \ldots \pi_i \pi_{j+1} \ldots \pi_n)$$

Block moves (3-edge-exchange)

$$\Delta_B = \{\delta_B^{ijk} | 1 \leq i < j < k \leq n\}$$

$$\delta_B^{ij}(\pi) = (\pi_1 \ldots \pi_{i-1} \pi_j \ldots \pi_k \pi_i \ldots \pi_{j-1} \pi_{k+1} \ldots \pi_n)$$

Short block move (Or-edge-exchange)

$$\Delta_{SB} = \{\delta_{SB}^{ij} | 1 \leq i < j \leq n\}$$

$$\delta_{SB}^{ij}(\pi) = (\pi_1 \ldots \pi_{i-1} \pi_j \pi_{j+1} \pi_{j+2} \pi_i \ldots \pi_{j-1} \pi_{j+3} \ldots \pi_n)$$

Neighborhood Operators for Assignments

An assignment can be represented as a mapping
$\sigma : \{X_1 \ldots X_n\} \rightarrow \{v : v \in D, |D| = k\}$:

$$\sigma = \{X_i = v_i, X_j = v_j, \ldots\}$$

One exchange operator

$$\Delta_{1E} = \{\delta_{1E}^{il} | 1 \leq i \leq n, 1 \leq l \leq k\}$$

$$\delta_{1E}^{il}(\sigma) = \left\{\sigma : \sigma'(X_i) = v_l \text{ and } \sigma'(X_j) = \sigma(X_j) \ \forall j \neq i\right\}$$

Two exchange operator

$$\Delta_{2E} = \{\delta_{2E}^{ij} | 1 \leq i < j \leq n\}$$

$$\delta_{2E}^{ij}\{\sigma : \sigma'(X_i) = \sigma(X_j),\ \sigma'(X_j) = \sigma(X_i) \text{ and } \sigma'(X_l) = \sigma(X_l) \ \forall l \neq i, j\}$$

Perturbative Search on the Traveling Salesman Problem

- k-exchange heuristics
  - 2-opt
  - 2.5-opt
  - Or-opt
  - 3-opt
- complex neighborhoods
  - Lin-Kernighan
  - Helsgaun's Lin-Kernighan
  - Dynasearch
  - ejection chains approach

Implementations exploit speed-up techniques

- neighborhood pruning: fixed radius nearest neighborhood search
- neighborhood lists: restrict exchanges to most interesting candidates
- don't look bits: focus perturbative search to “interesting” part
- sophisticated data structures

Iterative Improvement (2 OPT)

```
procedure TSP-2opt-first(s)
   input: an initial candidate tour s ∈ S(∈)
   output: a local optimum s ∈ S(π)
   Δ = 0;
   for i = 1 to n − 2 do
   if i = 1 then n′ = n − 1 elsen′ = n
      for j = i + 2 to n′ do
         Δ_ij = d(c_i, c_j) + d(c_{i+1}, c_{j+1}) − d(c_i, c_{i+1}) − d(c_j, c_{j+1})
         if Δ_ij < 0 then
            UpdateTour(s,i,j)
      end
   end
end TSP-2opt-first
```

In total $\binom{n}{2}$ possible moves

Concepts that Define the Search Landscape

- Neighborhood structure $N : S \rightarrow S \times S$
  Often defined through an operator
- Neighborhood Graph $G_N = (S, E_N)$
- Distance d_N: length of shortest path between two solutions s and s' in G_N
- Evaluation Function $g(\pi) : S \mapsto \mathbb{R}$

Definition:
The **Search Landscape** $L(\pi)$ of π The triple $L(\pi) := \langle S(\pi), N(\pi), g(\pi)\rangle$

Distances

Set of paths in G_N with $s, s' \in S$:

$$\Phi(s, s') = \{(s_1, \ldots, s_h) | s_1 = s, s_h = s' \; \forall i : 1 \leq i \leq h-1, \langle s_i, s_{i+1}\rangle \in E_N\}$$

If $\phi = (s_1, \ldots, s_h) \in \Psi(s, s')$ let $|\phi| = h$ be the length of the path; then the **distance** between any two solutions s, s' is the length of shortest path between s and s' in G_N:

$$d_N(s, s') = \min_{\phi \in \Phi(s,s')} |\Phi|$$

Note: with permutations it is easy to see that:

$$d_N(\pi, \pi') = d_N(\pi^{-1} \cdot \pi', \iota)$$

Distances for Linear Permutation Representations

Swap neighborhood operator
computable in $O(n^2)$ by the precendence distance metric:
$\#\{\langle i, j\rangle | 1 \leq i < j \leq n, pos_{\pi'}(\pi_j) < pos_{\pi'}(\pi_i)\}$.
$diam(G_N) = n(n-1)/2$

Interchange neighborhood operator
Computable in $O(n) + O(n)$ since
$d_X(\pi, \pi') = d_X(\pi^{-1} \cdot \pi', \iota) = n - c(\pi^{-1} \cdot \pi')$ where $c(\pi)$ is the number of disjoint cycles that decompose a permutation.
$diam(G_{N_X}) = n - 1$

Insert neighborhood operator
Computable in $O(n) + O(n\,log(n))$ since
$d_I(\pi, \pi') = d_I(\pi^{-1} \cdot \pi', \iota) = n - |lis(\pi^{-1} \cdot \pi')|$ where $lis(\pi)$ denotes the length of the longest increasing subsequence.
$diam(G_{N_I}) = n - 1$

Distances for Circular Permutation Representations

Reversal neighborhood operator
sorting by reversal is known to be NP-hard

Block moves neighborhood operator
unknown whether it is NP-hard but there does not exist a proven polynomial-time algorithm

Distances for Assignment Representations

An assignment can be seen as a partition of n in k mutually exclusive non-empty subsets

One-exchange neighborhood operator
The *partition-distance* $d_{1E}(\mathcal{P}, \mathcal{P}')$ between two partitions $\mathcal{P}$ and $\mathcal{P}'$ is the minimum number of elements that must be moved between subsets in $\mathcal{P}$ so that the resulting partition equals $\mathcal{P}'$.

The partition-distance can be computed in polynomial time by solving an assignment problem. Given the assignment matrix M where in each cell (i, j) it is $|S_i \cup S'_j|$ with $S_i \in \mathcal{P}$ and $S'_j \in \mathcal{P}'$ and defined $A(\mathcal{P}, \mathcal{P}')$ the assignment of maximal sum then it is $d_{1E}(\mathcal{P}, \mathcal{P}') = n - A(\mathcal{P}, \mathcal{P}')$

Fundamental Search Space Properties

The behavior 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 *neighborhood size*.
- number of (optimal) solutions $|S'|$, *solution density* $|S'|/|S|$
- distribution of solutions within the neighborhood graph

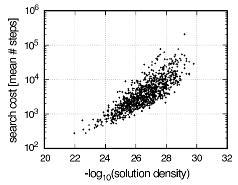
Solution densities and distributions can generally be determined by:

- exhaustive enumeration;
- sampling methods;
- counting algorithms (often variants of complete algorithms).

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 neighborhood
  $= \binom{n}{2} = n \cdot (n-1)/2$
- Size of 3-exchange neighborhood
  $= \binom{n}{3} = n \cdot (n-1) \cdot (n-2)/6$
- Diameter of neighborhood graphs: Exact values unknown.
  - Bounds for 2-exchange neighborhood = $[n/2, n-1]$
  - Bounds for 3-exchange neighborhood = $[n/3, n-1]$

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



A landscape $L := (S, N, g)$ **is ...**

- *invertible* (or *non-degenerate*), iff
  $\forall s, s' \in S : [g(s) = g(s') \Longrightarrow s = s']$;

- *locally invertible*, iff
  $\forall r \in S : \forall s, s' \in N(r) \cup \{r\} : [g(s) = g(s') \Longrightarrow s = s']$;

  Note: Every invertible landscape is also locally invertible
  (but not necessarily vice versa).

- *non-neutral*, iff
  $\forall s \in S : \forall s' \in N(s) : [g(s) = g(s') \Longrightarrow s = s']$.

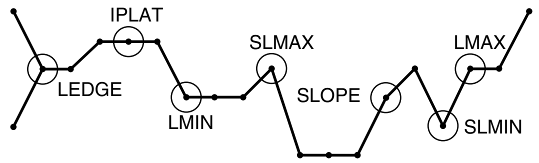
  Note: Every locally invertible landscape is also non-neutral
  (but not necessarily vice versa).

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

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 neighbors 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 properties of local minima:

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

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

$\Rightarrow$ 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)

Interpretation:

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

  ⇒ *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.

  ⇒ *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.

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

Note: These results are fairly typical for many types of TSP instances and instances of other combinatorial problems.
In many cases, local optima tend to be clustered; this is reflected in multi-modal distributions of pairwise distances between local minima.

Fitness-Distance Correlation (FDC)

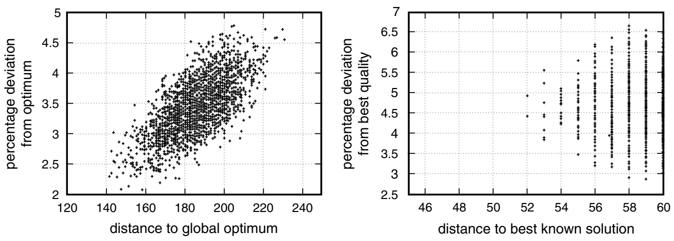
Idea: Analyse correlation between solution quality (fitness) g of candidate solutions and distance d to (closest) optimal solution.

Measure for FDC: *empirical correlation coefficient* r_{fdc}.

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.

- 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).

Example: FDC plot for TSPLIB instance `rat783`, based on 2500 local optima obtained from a 3-opt algorithm



High FDC (r_{fdc} close to one):

- '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} close to zero):

- 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.

Note: $r(i)$ depends on the given neighbourhood relation.

- Empirical autocorrelation analysis is computationally cheap compared to, *e.g.*, fitness-distance analysis.
- (Bounds on) AC 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 autocorrelation coefficient* and *(empirical) correlation length*.

High AC (close to one):

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

Low AC (close to zero):

- 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:

- Measures of ruggedness, such as AC, 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.

Appendix

Quadratic Assignment Problem

Input: n objects with a matrix B of flows between them and n locations with a matrix A of distances
Task: Find the assignment φ of objects to locations that minimise the sum of product between flows and distances, ie,

$$f(\varphi) = \sum b_{ij} a_{\varphi(i)\varphi(j)}$$