DM811

Heuristics for Combinatorial Optimization

Lecture 8

Local Search: Further Analysis

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Single Machine Total Weighted Tardiness

Given: a set of n jobs $\{J_1, \ldots, J_n\}$ to be processed on a single machine and for each job J_i a processing time p_i, a weight w_i and a due date d_i.

Task: Find a schedule that minimizes
the total weighted tardiness $\sum_{i=1}^{n} w_i \cdot T_i$
where $T_i = \max\{C_i - d_i, 0\}$ (C_i completion time of job J_i)

Example:

Job	J_1	J_2	J_3	J_4	J_5	J_6
Processing Time	3	2	2	3	4	3
Due date	6	13	4	9	7	17
Weight	2	3	1	5	1	2

Sequence $\phi = J_3, J_1, J_5, J_4, J_1, J_6$

Job	J_3	J_1	J_5	J_4	J_1	J_6
C_i	2	5	9	12	14	17
T_i	0	0	2	3	1	0
$w_i \cdot T_i$	0	0	2	15	3	0

The Max Independent Set Problem

Also called "stable set problem" or "vertex packing problem".
Given: an undirected graph $G(V, E)$ and a non-negative weight function ω on V ($\omega : V \rightarrow \mathbf{R}$)

Task: A largest weight independent set of vertices, i.e., a subset $V' \subseteq V$ such that no two vertices in V' are joined by an edge in E.

Escaping Local Optima

Possibilities:

- Enlarge the neighborhood
- Restart: re-initialize search whenever a local optimum is encountered.
  (Often rather ineffective due to cost of initialization.)
- Non-improving steps: in local optima, allow selection of candidate solutions with equal or worse evaluation function value, *e.g.*, using minimally worsening steps.
  (Can lead to long walks in *plateaus*, *i.e.*, regions of search positions with identical evaluation function.)
  This is what Metaheuristics do.

Note: None of these mechanisms is guaranteed to always escape effectively from local optima.

Diversification *vs* Intensification

- Goal-directed and randomized components of LS strategy need to be balanced carefully.
- Intensification: aims at greedily increasing solution quality, *e.g.*, by exploiting the evaluation function.
- Diversification: aims at preventing search stagnation, that is, the search process getting trapped in confined regions.

Examples:

- Iterative Improvement (II): *intensification* strategy.
- Uninformed Random Walk/Picking (URW/P): *diversification* strategy.

Balanced combination of intensification and diversification mechanisms forms the basis for advanced LS methods.

Scientific Knowledge on LS

- Performance analysis
  - probabilistic analysis: aims to determine average-case perfomance for a given probability distribution of the instances
  - worst-case analysis: over all possible instances
  - empirical analysis
- Time complexity
  e.g. # of iterations required to reach local optima $\rightsquigarrow$ general theory of time complexity of LS
- Asymptotic convergence when a probabilistic iteration mechanism is applied

Computational Complexity of LS

For a local search algorithm to be effective, search initialization and individual search steps should be efficiently computable.

Complexity class $\mathcal{PLS}$: class of problems for which a local search algorithm exists with polynomial time complexity for:

- search initialization
- any single search step, including computation of evaluation function value

For any problem in $\mathcal{PLS}$ ...

- local optimality can be verified in polynomial time
- improving search steps can be computed in polynomial time
- **but:** finding local optima may require super-polynomial time

Computational Complexity of LS

$\mathcal{PLS}$-complete: Among the most difficult problems in $\mathcal{PLS}$;
if for any of these problems local optima can be found
in polynomial time, the same would hold for all problems in $\mathcal{PLS}$.

Some complexity results:

- TSP with k-exchange neighborhood with $k > 3$ is $\mathcal{PLS}$-complete.
- TSP with 2- or 3-exchange neighborhood is in $\mathcal{PLS}$, but $\mathcal{PLS}$-completeness is unknown.

Outline

Learning goals of this section

- Review basic theoretical concepts
- Fix terminology
- Develop intuition on features of local search that may in guiding the design of LS algorithms

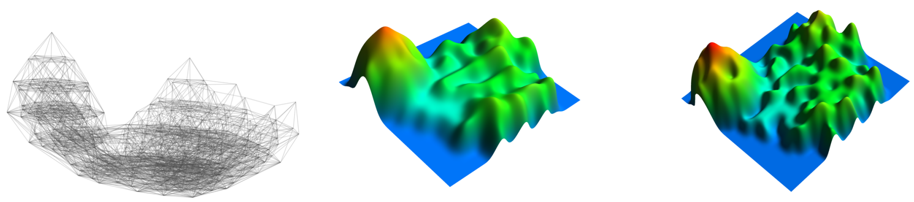
Definitions

- Search space S
- Neighborhood function $\mathcal{N} : S \subseteq 2^S$
- Evaluation function $f(\pi) : S \mapsto \mathbf{R}$
- Problem instance π

Definition:

The **search landscape** L is the vertex-labeled neighborhood graph given by the triplet $\mathcal{L} = (S(\pi), N(\pi), f(\pi))$.

Search Landscape

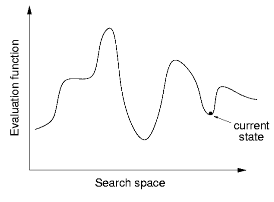


Transition Graph of Iterative Improvement
Given $\mathcal{L} = (S(\pi), N(\pi), f(\pi))$, the transition graph of iterative improvement is a directed acyclic subgraphs obtained from $\mathcal{L}$ by deleting all arcs (i, j) for which it holds that the cost of solution j is worse than or equal to the cost of solution i.

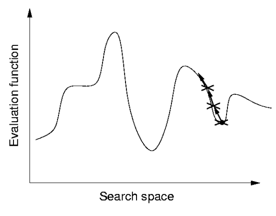
It can be defined for other algorithms as well and it plays a central role in the theoretical analysis of proofs of convergence.

Simplified representation

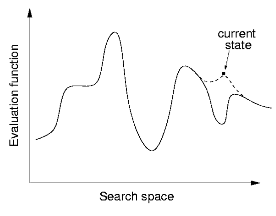
- Landscape in one-dimension



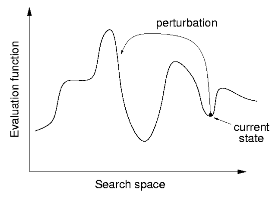
- Tabu Search



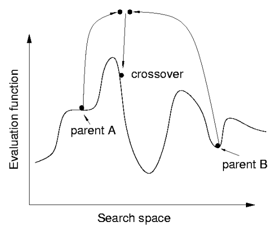
- Guided Local Search



- Iterated Local Search



- Evolutionary Alg.



Fundamental Properties

The behavior and performance of an LS algorithm on a given problem instance crucially depends on properties of the respective search landscape.

Simple properties:

- search space size $|S|$
- reachability: solution j is reachable from solution i if neighborhood graph has a path from i to j.
  - strongly connected neighborhood graph
  - weakly optimally connected neighborhood graph
- distance between solutions
- neighborhood size (ie, degree of vertices in neigh. graph)
- cost of fully examining the neighborhood

Neighborhood Operator

Goal: providing a formal description of neighborhood functions for the three main solution representations:

- Permutation
  - linear permutation: Single Machine Total Weighted Tardiness Problem
  - circular permutation: Traveling Salesman Problem
- Assignment: Graph Coloring Problem, SAT, CSP
- Set, Partition: Knapsack, Max Independent Set

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

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

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 $\pi(i) = \pi_i$

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 $\pi^{-1} \cdot \pi = \iota$

$$\Delta_N \subset \Pi$$

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

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

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

($\equiv$ set of all transpositions)

Insert operator

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

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

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)

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

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

Assignments

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

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

One-exchange operator

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

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

Two-exchange operator

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

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

Partitioning

An assignment can be represented as a partition of objects selected and not selected $s : \{X\} \to \{C, \overline{C}\}$
(it can also be represented by a bit string)

One-addition operator

$$\Delta_{1E} = \{\delta^v_{1E} | v \in \overline{C}\}$$

$$\delta^v_{1E}(s) = \{s : C' = C \cup v \text{ and } \overline{C}' = \overline{C} \setminus v\}$$

One-deletion operator

$$\Delta_{1E} = \{\delta^v_{1E} | v \in C\}$$

$$\delta^v_{1E}(s) = \{s : C' = C \setminus v \text{ and } \overline{C}' = \overline{C} \cup v\}$$

Swap operator

$$\Delta_{1E} = \{\delta^v_{1E} | v \in C, u \in \overline{C}\}$$

$$\delta^v_{1E}(s) = \{s : C' = C \cup u \setminus v \text{ and } \overline{C}' = \overline{C} \cup v \setminus u\}$$

Distances

Set of paths in $\mathcal{L}$ with $s, s' \in S$:

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

If $\phi = (s_1, \ldots, s_h) \in \Phi(s, s')$ 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 $\mathcal{L}$:

$$d_{\mathcal{N}}(s, s') = \min_{\phi \in \Phi(s,s')} |\Phi|$$

$\mathtt{diam}(\mathcal{L}) = \max\{d_{\mathcal{N}}(s, s') \mid s, s' \in S\}$ (= maximal distance between any two candidate solutions)
(= worst-case lower bound for number of search steps required for reaching (optimal) solutions)

Note: with permutations it is easy to see that:

$$d_{\mathcal{N}}(\pi, \pi') = d_{\mathcal{N}}(\pi^{-1} \cdot \pi', \iota)$$

Distances for Linear Permutation Representations

- Swap neighborhood operator
  computable in $O(n^2)$ by the precedence based distance metric:
  $d_S(\pi, \pi') = \#\{\langle i, j\rangle | 1 \le i < j \le n, pos_{\pi'}(\pi_j) < pos_{\pi'}(\pi_i)\}$.
  $\mathtt{diam}(G_{\mathcal{N}}) = n(n-1)/2$

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

- 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.
  $\mathtt{diam}(G_{\mathcal{N}_I}) = n - 1$

Distances for Circular Permutation Representations

- Reversal neighborhood operator
  sorting by reversal is known to be NP-hard
  surrogate in TSP: bond distance

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

Distances for Assignment Representations

- Hamming Distance

Distances for Partitioning Problems

given a set of elements $E = \{1, 2 \dots |E|\}$ find a partition of the set into a number of subsets $\mathcal{P} = \{C_1, C_2, \dots, C_k\}$, $C_i \subseteq E$ and $C_i \cap C_j = \emptyset$ for all $i \neq j$, with each of the subsets having to meet the same requirements. (Exhibit intrinsic symmetry)

- 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 $|C_i \cap C'_j|$ with $C_i \in \mathcal{P}$ and $C'_j \in \mathcal{P}'$ and defined $A(\mathcal{P}, \mathcal{P}')$ the assignment of maximal sum then it is $d_{1E}(\mathcal{P}, \mathcal{P}') = n - A(\mathcal{P}, \mathcal{P}')$

Example: Search space size and diameter for the TSP

- Search space size = $(n-1)!/2$
- Insert neighborhood
  size = $(n-3)n$
  diameter = $n-2$
- 2-exchange neighborhood
  size = $\binom{n}{2} = n \cdot (n-1)/2$
  diameter in $[n/2, n-2]$
- 3-exchange neighborhood
  size = $\binom{n}{3} = n \cdot (n-1) \cdot (n-2)/6$
  diameter in $[n/3, n-1]$

Example: Search space size and diameter for SAT

SAT instance with n variables, 1-flip neighborhood:
$G_{\mathcal{N}}$ = n-dimensional hypercube; diameter of $G_{\mathcal{N}} = n$.

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two different neighborhood functions for the same instance (S, f, π) of a combinatorial optimization problem.

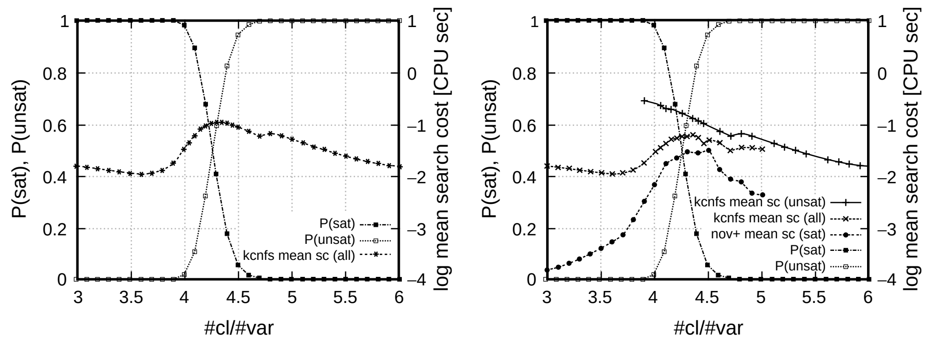
If for all solutions $s \in S$ we have $N_1(s) \subseteq N_2(s')$ then we say that $\mathcal{N}_2$ dominates $\mathcal{N}_1$

Example:

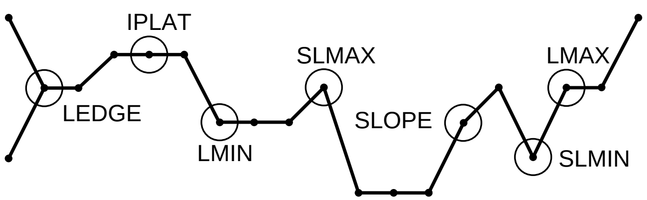
In TSP, 1-insert is dominated by 3-exchange.
(1-insert corresponds to 3-exchange and there are 3-exchanges that are not 1-insert)

Phase Transition for 3-SAT

Random instances $\Rightarrow$ m clauses of n uniformly chosen variables



Classification of search positions



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

Ruggedness

Idea: Rugged search landscapes, *i.e.*, landscapes with high variability in evaluation function value between neighboring 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.

$\rightsquigarrow$ NK model [Kauffman, The origin of Order, 1993] to study evolution (used also in econmics)

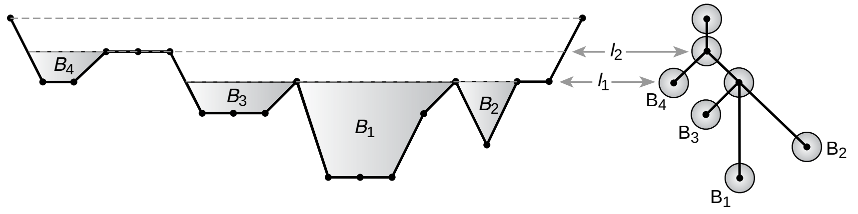
- N loci ie. genes in a genotype
- 2 alleles
- K epistatic interactions (dependencies among genes in the contribution to fitness)

Barriers and Basins

Definitions:

- Positions s, s' are *mutually accessible at level* l iff there is a path connecting s' and s in the neighborhood 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'.
- **Basins**, *i.e.*, maximal (connected) regions of search positions below a given level, form an important basis for characterizing 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 neighboring basins are joined (by a *saddle*).

Outline

Example: Scheduling in Parallel Machines

Total Weighted Completion Time on Unrelated Parallel Machines Problem

Input: A set of jobs J to be processed on a set of parallel machines M. Each job $j \in J$ has a weight w_j and processing time p_{ij} that depends on the machine $i \in M$ on which it is processed.

Task: Find a schedule of the jobs on the machines such that the sum of weighted completion time of the jobs is minimal.

Example: Steiner Tree

Steiner Tree Problem

Input: A graph $G = (V, E)$, a weight function $\omega : E \mapsto \mathbf{N}$, and a subset $U \subseteq V$.

Task: Find a Steiner tree, that is, a subtree $T = (V_T, E_T)$ of G that includes all the vertices of U and such that the sum of the weights of the edges in the subtree is minimal.

Vertices in U are the special vertices and vertices in $S = V \setminus U$ are Steiner vertices.