INFO-H-413 - Heuristic Optimization

École Polytechnique de Bruxelles

Notes by Charles Hamesse

June 2017

Contents

1 Introduction **3**
1.1 Combinatorial Problems . . . 3
1.2 Two Prototypical Combinatorial Problems . . . 3
1.3 Computational Complexity . . . 3
1.3.1 Time Complexity . . . 3
1.3.2 Theory of NP-completeness . . . 3
1.4 Approximation algorithms . . . 3
1.5 Search Paradigms . . . 3
1.6 Stochastic Local Search . . . 3

2 SLS Methods **3**
2.1 Constructive Heuristics (Revisited) . . . 3
2.2 Iterative Improvement (Revisited) . . . 3
2.2.1 Variable Neighbourhood Descent (VND) . . . 3
2.2.2 Very Large Scale Neighborhood (VLSN) . . . 3
2.2.3 Variable Depth Search (VDS) . . . 3
2.2.4 Dynasearch . . . 4
2.3 Simple SLS Methods . . . 4
2.3.1 Randomised Iterative Improvement (RII) . . . 4
2.3.2 Probabilistic Iterative Improvement (PII) . . . 4
2.3.3 Simulated Annealing (SA) . . . 4
2.3.4 Tabu Search (TS) . . . 5
2.3.5 Dynamic Local Search (DLS) . . . 5
2.4 Hybrid SLS Methods . . . 5
2.4.1 Iterated Local Search (ILS) . . . 5
2.4.2 Variable Neighbourhood Search (VNS) . . . 5
2.4.3 Variable Neighborhood Decomposition Search (VNDS) . . . 6
2.4.4 Greedy Randomised Adaptive Search Procedures (GRASP) . . . 6
2.4.5 Iterated Greedy (IG) . . . 6
2.5 Population-Based SLS Methods . . . 6
2.5.1 Ant Colony Optimisation (ACO) . . . 6
2.5.2 Max-Min Ant System (MMAS) . . . 6
2.5.3 Ant Colony System (ACS) . . . 7
2.5.4 Evolutionary Algorithms (EA) . . . 7
2.5.5 Memetic Algorithms (MA) or Genetic Local Search (GLS) . . . 7

3 Generalised Local Search Machines **8**

3.1 The Basic GLSM Model . . . 8
3.2 State, Transition and Machine Types . . . 8
3.3 Modelling SLS Methods Using GLSMs . . . 8

4 Empirical Analysis of SLS Algorithms 8
4.1 Run-Time Distributions . . . 8
4.1.1 Run-Time Distribution (RTD) . . . 8
4.1.2 Qualified Run-Time Distribution (QRTD) . . . 8
4.1.3 Solution Quality Distribution (SQD) . . . 8
4.1.4 Solution Quality Distribution over Time (SQT) . . . 8
4.2 Summary descriptive statistics . . . 8
4.3 Analyses on Instance Ensembles and Comparison of Algorithms . . . 8
4.4 Benchmark sets . . . 8

5 Automatic Algorithm Configuration 8
5.1 Design Choices and Parameters . . . 8
5.1.1 ACO . . . 8
5.2 Offline and Online Configuration . . . 8
5.3 Approaches to Configuration . . . 8
5.3.1 The Racing Approach . . . 8
5.3.2 The F-Race Algorithm . . . 9
5.3.3 Mixed Integer Programming (MIP) Solvers . . . 9
5.3.4 Automatic Design of Hybrid SLS Algorithms . . . 9

6 Search Space Structure and SLS Performance 10
6.1 Fundamental Search Space Properties . . . 10
6.2 Search Landscapes . . . 10
6.3 Local Minima . . . 10
6.4 Fitness-Distance Correlation (FDC) . . . 10
6.5 Ruggedness . . . 10
6.6 Plateaux . . . 10
6.7 Barriers and Basins . . . 10

7 Heuristic Algorithms for Multiobjective Combinatorial Problems (MOCP) 10
7.1 MCOPs and Solution Methods . . . 10
7.2 SLS Algorithms . . . 10
7.3 Multiobjective Local Search (MOLS) . . . 10
7.4 Performance Assessment . . . 10

1 Introduction

1.1 Combinatorial Problems

1.2 Two Prototypical Combinatorial Problems

1.3 Computational Complexity

1.3.1 Time Complexity

1.3.2 Theory of NP-completeness

1.4 Approximation algorithms

1.5 Search Paradigms

1.6 Stochastic Local Search

2 SLS Methods

2.1 Constructive Heuristics (Revisited)

2.2 Iterative Improvement (Revisited)

2.2.1 Variable Neighbourhood Descent (VND)

Recall: Local minima are relative to neighbourhood relation. Key idea: To escape from local minimum of given neighbourhood relation, switch to different neighbhourhood relation. I usekneighbourhoodrelationsN1,...,Nk,(typically) ordered according to increasing neighbourhood size. I always use smallest neighbourhood that facilitates improving steps. I upon termination, candidate solution is locally optimal w.r.t. all neighbourhoods

Piped VND I differentiterativeimprovementalgorithmsII1...IIk available I key idea: build a chain of iterative improvement algorithms I dierent orders of algorithms often reasonable, typically same as would be done in standard VND I substantial performance improvements possible without modifying code of existing iterative improvement algorithms

2.2.2 Very Large Scale Neighborhood (VLSN)

VLSN algorithms are iterative improvement algorithms that make use of very large neighborhoods, often exponentially-sized ones I very large scale neighborhoods require e cient neighborhood search algorithms, which is facilitated through special-purpose neighborhood structures I two main classes I explore heuristically very large scale neighborhoods example: variable depth search I define special neighborhood structures that allow for e cient search (often in polynomial time) example: Dynasearch, cyclic exchange neighbourhoods

2.2.3 Variable Depth Search (VDS)

I Key idea: Complex steps in large neighbourhoods = variable-length sequences of simple steps in small neighbourhood. I the number of solution components that is exchanged in the complex step is variable and changes from one complex step to another. I Use various feasibility restrictions on

selection of simple search steps to limit time complexity of constructing complex steps. I Perform Iterative Improvement w.r.t. complex steps.

Example: The Lin-Kernighan (LK) Algorithm for the TSP (1) I Complex search steps correspond to sequences of 1-exchange steps and are constructed from sequences of Hamiltonian paths I -path: Hamiltonian path p + 1 edge connecting one end of p to interior node of p ('lasso' structure):

Construction of complex LK steps: 1. start with current candidate solution (Hamiltonian cycle) s; sett :=s;setp:=s 2. obtain -path p0 by replacing one edge in p 3. consider Hamiltonian cycle t obtained from p by (uniquely) defined edge exchange 4. ifw(t)¡w(t)thensett :=t;p:=p0 5. if termination criteria of LK step construction not met, go to step 2 6. accept t as new current candidate solution s if w(t) ¡ w(s) dfNote: The construction can be interpreted as sequence of 1-exchange steps that alternate between -paths and Hamiltonian cycles.

2.2.4 Dynasearch

I Key idea: E ciently find optimal combination of mutually independent simple search steps using Dynamic Programming. I Successful applications to various combinatorial optimisation problems, including: I the TSP and the Linear Ordering Problem [Congram, 2000] I the Single Machine Total Weighted Tardiness Problem (scheduling) [Congram et al., 2002]

2.3 Simple SLS Methods

Goal: Effectively escape from local minima of given evaluation function.

General approach: For fixed neighbourhood, use step function that permits worsening search steps.

2.3.1 Randomised Iterative Improvement (RII)

Key idea: In each search step, with a fixed probability perform an uninformed random walk step instead of an iterative improvement step.

Note: I No need to terminate search when local minimum is encountered Instead: Bound number of search steps or CPU time from beginning of search or after last improvement. I Probabilistic mechanism permits arbitrary long sequences of random walk steps Therefore: When run su ciently long, RII is guaranteed to find (optimal) solution to any problem instance with arbitrarily high probability. I A variant of RII has successfully been applied to SAT (GWSAT algorithm), but generally, RII is often outperformed by more complex SLS methods.

Note: I A variant of GUWSAT, GWSAT [Selman et al., 1994], was at some point state-of-the-art for SAT I Generally, RII is often outperformed by more complex SLS methods I Very easy to implement I Very few parameters

2.3.2 Probabilistic Iterative Improvement (PII)

Key idea: Accept worsening steps with probability that depends on respective deterioration in evaluation function value: bigger deterioration = smaller probability Realisation: I Function p(g, s): determines probability distribution over neighbours of s based on their values under evaluation function g. I Let step(s)(s0) := p(g,s)(s0). Note: I Behaviour of PII crucially depends on choice of p. I II and RII are special cases of PII.

2.3.3 Simulated Annealing (SA)

Key idea: Vary temperature parameter, i.e., probability of accepting worsening moves, in Probabilistic Iterative Improvement according to annealing schedule (aka cooling schedule). Inspired by a simulation of the physical annealing process: I candidate solutions = states of physical system I evaluation function

= thermodynamic energy I globally optimal solutions = ground states I parameter T = physical temperature Note: In physical process (e.g., annealing of metals), perfect ground states are achieved by very slow lowering of temperature.

2.3.4 Tabu Search (TS)

Key idea: Use aspects of search history (memory) to escape from local minima.

Note: I Non-tabu search positions in N(s) are called admissible neighbours of s. I After a search step, the current search position or the solution components just added/removed from it are declared tabu for a fixed number of subsequent search steps (tabu tenure). I Often, an additional aspiration criterion is used: this specifies conditions under which tabu status may be overridden (e.g., if considered step leads to improvement in incumbent solution).

Note: Performance of Tabu Search depends crucially on setting of tabu tenure tt:

- tt too low: search stagnates due to inability to escape from local minima;
- tt too high: search becomes ineffective due to overly restricted search path (admissible neighbourhoods too small)

2.3.5 Dynamic Local Search (DLS)

I Key Idea: Modify the evaluation function whenever a local optimum is encountered in such a way that further improvement steps become possible. I Associate penalty weights (penalties) with solution components; these determine impact of components on evaluation function value. I Perform Iterative Improvement; when in local minimum, increase penalties of some solution components until improving steps become available.

2.4 Hybrid SLS Methods

Combination of 'simple' SLS methods often yields substantial performance improvements. Simple examples:

- Commonly used restart mechanisms can be seen as hybridisations with Uninformed Random Picking
- Iterative Improvement + Uninformed Random Walk = Randomised Iterative Improvement

2.4.1 Iterated Local Search (ILS)

Key Idea: Use two types of SLS steps: I subsidiary local search steps for reaching local optima as e ciently as possible (intensification) I perturbation steps for eectively escaping from local optima (diversification). Also: Use acceptance criterion to control diversification vs intensification behaviour.

Note: I Subsidiary local search results in a local minimum. I ILS trajectories can be seen as walks in the space of local minima of the given evaluation function. I Perturbation phase and acceptance criterion may use aspects of search history (i.e., limited memory). I In a high-performance ILS algorithm, subsidiary local search, perturbation mechanism and acceptance criterion need to complement each other well.

2.4.2 Variable Neighbourhood Search (VNS)

Variable Neighbourhood Search (VNS) Key Idea: systematically change neighbourhoods during search Motivation: I recall: changing neighbourhoods can help escape local optima I a global optimum is locally optimal w.r.t. all neighbourhood structures I principle of VNS: change the neighbourhood during the search

2.4.3 Variable Neighborhood Decomposition Search (VNDS)

I central idea I generate subproblems by keeping all but k solution components fixed I apply local search only to the k "free" components

relationship between ILS and VNS I the two SLS methods are based on dierent underlying "philosophies" I they are similar in many respects I ILS apears to be in literature more flexible w.r.t. optimization of the interaction of modules I VNS gives place to approaches like VND for obtaining more powerful local search approaches

2.4.4 Greedy Randomised Adaptive Search Procedures (GRASP)

Key Idea: Combine randomised constructive search with subsequent perturbative local search. Motivation: I Candidate solutions obtained from construction heuristics can often be substantially improved by perturbative local search. I Perturbative local search methods typically often require substantially fewer steps to reach high-quality solutions when initialised using greedy constructive search rather than random picking. I By iterating cycles of constructive + perturbative search, further performance improvements can be achieved.

2.4.5 Iterated Greedy (IG)

Key Idea: iterate over greedy construction heuristics through destruction and construction phases Motivation: I start solution construction from partial solutions to avoid reconstruction from scratch I keep features of the best solutions to improve solution quality I if few construction steps are to be executed, greedy heuristics are fast I adding a subsidiary local search phase may further improve performance

2.5 Population-Based SLS Methods

SLS methods discussed so far manipulate one candidate solution of given problem instance in each search step. Straightforward extension: Use population (i.e., set) of candidate solutions instead.

The use of populations provides a generic way to achieve search diversification.

Population-based SLS methods fit into the general definition from Chapter 1 by treating sets of candidate solutions as search positions.

2.5.1 Ant Colony Optimisation (ACO)

Key idea: Can be seen as population-based constructive approach where a population of agents – (artificial) ants – communicate via common memory – (simulated) pheromone trails

2.5.2 Max-Min Ant System (MMAS)

extension of Ant System with stronger exploitation of best solutions and additional mechanism to avoid search stagnation

exploitation: only the iteration-best or best-so-far ant deposit pheromone

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}^{\text{best}} \tag{1}$$

frequently, a schedule for choosing between iteration-best and best-so-far update is used

2.5.3 Ant Colony System (ACS)

Key idea pseudo-random proportional action choice rule I with probability q0 an ant k located at city i chooses successor city j with maximal ij (t) · [ij] (exploitation) I with probability 1 q0 an ant k chooses the successor city j according to action choice rule used in Ant System (biased exploration)

2.5.4 Evolutionary Algorithms (EA)

Key idea: Iteratively apply genetic operators mutation, recombination, selection to a population of candidate solutions.

2.5.5 Memetic Algorithms (MA) or Genetic Local Search (GLS)

Problem: Pure evolutionary algorithms often lack capability of su cient search intensification. Solution: Apply subsidiary local search after initialisation, mutation and recombination.

3 Generalised Local Search Machines

3.1 The Basic GLSM Model

3.2 State, Transition and Machine Types

3.3 Modelling SLS Methods Using GLSMs

4 Empirical Analysis of SLS Algorithms

4.1 Run-Time Distributions

4.1.1 Run-Time Distribution (RTD)

4.1.2 Qualified Run-Time Distribution (QRTD)

4.1.3 Solution Quality Distribution (SQD)

4.1.4 Solution Quality Distribution over Time (SQT)

4.2 Summary descriptive statistics

4.3 Analyses on Instance Ensembles and Comparison of Algorithms

4.4 Benchmark sets

5 Automatic Algorithm Configuration

5.1 Design Choices and Parameters

5.1.1 ACO

5.2 Offline and Online Configuration

5.3 Approaches to Configuration

5.3.1 The Racing Approach

The racing approach

1. Start with a set of initial candidates
2. Consider a stream of instances
3. Sequentially evaluate candidates
4. Discard inferior candidates
5. ...Repeat until a winner is selected or computation time expires

5.3.2 The F-Race Algorithm

Statistical testing 1. family-wise tests for dierences among configurations I Friedman two-way analysis of variance by ranks 2. if Friedman rejects H0, perform pairwise comparisons to best configuration I apply Friedman post-test

Iterated race Racing is a method for the selection of the best configuration and independent of the way the set of configurations is sampled Iterated racing sample configurations from initial distribution While not terminate() apply race modify sampling distribution sample configurations

The irace Package Manuel Lo pez-Ib an ez, J er emie Dubois-Lacoste, Thomas Stu tzle, and Mauro Birattari. The irace package, Iterated Race for Automatic Algorithm Configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Universit e Libre de Bruxelles, Belgium, 2011. http://iridia.ulb.ac.be/irace I implementation of Iterated Racing in R Goal 1: flexible Goal 2: easy to use I but no knowledge of R necessary I parallel evaluation (MPI, multi-cores, grid engine ..) I initial candidates irace has shown to be eective for configuration tasks with several hundred of variables

Other tools: ParamILS, SMAC ParamILS I iterated local search in configuration space I requires discretization of numerical parameters I http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/ SMAC I surrogate model assisted search process I encouraging results for large configuration spaces I http://www.cs.ubc.ca/labs/b capping: eective speed-up technique for configuration target run-time

5.3.3 Mixed Integer Programming (MIP) Solvers

Mixed integer programming (MIP) solvers I powerful commercial (e.g. CPLEX) and non-commercial (e.g. SCIP) solvers available I large number of parameters (tens to hundreds) I default configurations not necessarily best for specific problems

5.3.4 Automatic Design of Hybrid SLS Algorithms

Automatic design of hybrid SLS algorithms Approach I decompose single-point SLS methods into components I derive generalized metaheuristic structure I component-wise implementation of metaheuristic part Implementation I present possible algorithm compositions by a grammar I instantiate grammer using a parametric representation I allows use of standard automatic configuration tools I shows good performance when compared to, e.g., grammatical evolution [Mascia, Lo pes-Ib an ez, Dubois-Lacoste, Stu tzle, 2014

6 Search Space Structure and SLS Performance

6.1 Fundamental Search Space Properties

6.2 Search Landscapes

6.3 Local Minima

6.4 Fitness-Distance Correlation (FDC)

6.5 Ruggedness

6.6 Plateaux

6.7 Barriers and Basins

7 Heuristic Algorithms for Multiobjective Combinatorial Problems (MOCP)

7.1 MCOPs and Solution Methods

7.2 SLS Algorithms

7.3 Multiobjective Local Search (MOLS)

7.4 Performance Assessment