Stochastic Local Search
Introduction & Engineering

Thomas Stützle

IRIDIA
Université Libre de Bruxelles
Brussels, Belgium
stuetzle@iridia.ulb.ac.be
iridia.ulb.ac.be/~stuetzle

outline

combinatorial optimisation problems

examples

- finding minimum cost schedule to deliver goods
- finding optimal sequence of jobs in production line
- finding best allocation of flight crews to airplanes
- finding a best routing for Internet data packets
- . . . and many more

features

- arise in many real-world applications
- many have high computational complexity ($\mathcal{NP}$-hard)
- in research, often abstract versions of real-world problems are treated

search paradigms

systematic search

- traverse search space of instances in systematic manner
- *complete:* guaranteed to find optimal solution in finite amount of time (plus proof of its optimality)

local search

- start at some initial solution
- iteratively move from search position to neighbouring one
- *incomplete:* not guaranteed to find optimal solutions

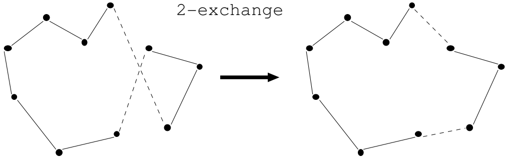
search paradigms (2)

constructive search

- search space = partial candidate solutions
- search step = extension with solution components
- example: nearest neighbour heuristic for TSP

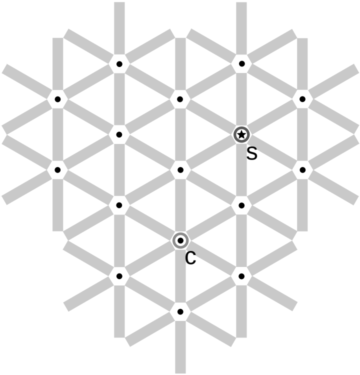
perturbative search

- search space = complete candidate solutions
- search step = modification of solution components
- example:

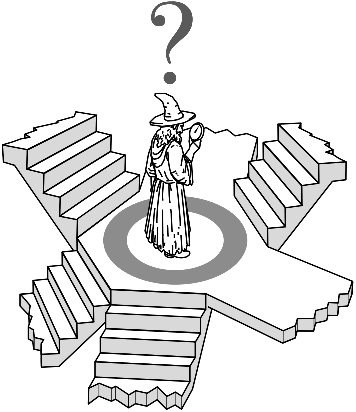


stochastic local search — global view

- *vertices*: candidate solutions (search positions)
- *edges*: connect neighbouring positions
- *s*: (optimal) solution
- *c*: current search position



stochastic local search — local view



- next search position is selected from local neighbourhood based on local information, e.g., heuristic values.

Definition: **Stochastic Local Search Algorithm** (1)

For given problem instance π:

- *search space* $S(\pi)$
  (*e.g.*, for TSP: all round trips visiting each city at most once)
- *solution set* $S'(\pi) \subseteq S(\pi)$
  (*e.g.*, for TSP: shortest round-trips)
- *neighbourhood relation* $N(\pi) \subseteq S(\pi) \times S(\pi)$
  (*e.g.*, for TSP: neighbouring round-trips differ in at most k edges)

Definition: **Stochastic Local Search Algorithm** (2)

- *set of memory states* $M(\pi)$
  (may consist of a single state, for SLS algorithms that do not use memory)
- *initialisation function* $init : \emptyset \mapsto \mathcal{D}(S(\pi) \times M(\pi))$
  (specifies probability distribution over initial search positions and memory states)
- *step function* $step : S(\pi) \times M(\pi) \mapsto \mathcal{D}(S(\pi) \times M(\pi))$
  (maps each search position and memory state onto probability distribution over subsequent, neighbouring search positions and memory states)
- *termination predicate*
  $terminate : S(\pi) \times M(\pi) \mapsto \mathcal{D}(\{\top, \bot\})$
  (determines the termination probability for each search position and memory state)

a simple SLS algorithm

iterative improvement

- start from some initial solution
- iteratively move from the current solution to an improving neighbouring one as long as such one exists

main problem

- getting stuck in local optima

solution

- general–purpose SLS methods (aka metaheuristcs) that direct the search and allow escapes from local optima

SLS methods

modify neighbourhoods

- variable neighbourhood search

accept occasionally worse neighbours

- simulated annealing
- tabu search

modify evaluation function

- dynamic local search

generate new (starting) solutions (for local search)

- EAs / memetic algorithms
- ant colony optimization
- iterated local search

SLS methods

enormous research efforts

- hundreds / thousands of publications
- conference series (MIC); 190 submissions in 2005
- sub-areas become established fields (evolutionary algorithms, swarm intelligence)

significant successes and developments

- excellent results in many application areas
- new algorithmic ideas (MAs, ACO, VLSN, etc.)
- sophisticated data structures
- sufficient computational power nowadays available

SLS methods

current deficiencies

- no general guidelines of how to design efficient SLS algorithms; application often considered an art
- high development times and expert knowledge required
- relationship between problem / instance characteristics and performance not well understood
- shortcomings in experimental methodology
- gap between theory and practice
- limited usage in related areas like multi-objective, dynamic, or stochastic problems (or more fancy things ..)

SLS methods

insights from Metaheuristics Network

- success with SLS algorithms due to
  - level of expertise of developer and implementer
  - time invested in designing and tuning the SLS algorithm
  - creative use of insights into algorithm behaviour and interplay with problem characteristics
- fundamental are issues like choice of underlying neighbourhoods, efficient data structures, creative use of algorithm components; to a less extent the strict attainment to the rules of a specific SLS method

SLS algorithm engineering

Algorithm engineering (AE)

- process of designing, analyzing, implementing, tuning, and experimentally evaluating algorithms *[Demetrescu et al. 2003]*
- is conceived as an extension of traditional (rather theoretical) research in algorithmics

SLS algorithm engineering

- analogous high-level process to AE
- but much more difficult because
  - problems tackled are highly complex ($\mathcal{NP}$-hard)
  - stochasticity of algorithms makes analysis harder
  - many degrees of freedom

SLS algorithm engineering (2)

Main GOAL: develop an engineering methodology for the implementation of effective stochastic local search algorithms that guides researchers and practitioners in their development of such algorithms for solving challenging optimization problems

- devise *systematic procedure* that leads to high performing SLS algorithms
- tentative step-wise engineering procedure
  - get insight into the problem being tackled
  - implement basic constructive and local search procedures
  - starting from these add complexity (simple SLS methods)
  - add advanced concepts like perturbations, population
  - if needed: *iterate* through these steps

bottom-up approach: add complexity step-by-step

SLS algorithm engineering (3)

Tools

- tools are need to assist development process
- examples
  - R, LEDA, TefoA, software libraries, etc.
- missing: integration into an SLS design process

Practical GOAL: make available a complete set of procedures to assist the design process of SLS algorithms

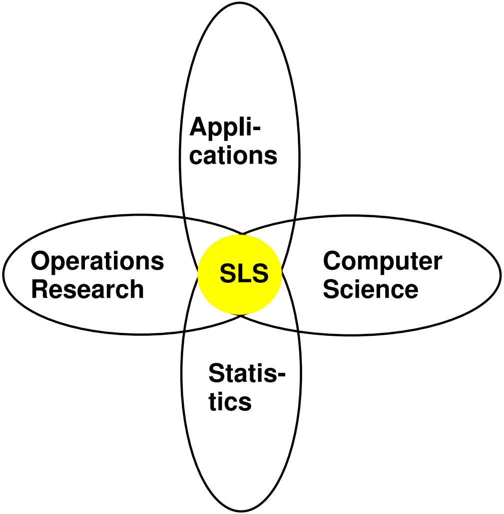
SLS algorithm engineering (4)

Knowledge

- awareness about important knowledge in SLS algorithms and problems
  - general-purpose SLS methods as well as basic things (constructive heuristics, iterative improvement)
  - problems, their features and characteristics and classical solution techniques
  - computer science basics (especially algorithmics and AI)
  - statistical methodologies
  - relationship between algorithm performance and problem features

Pedagogical GOAL: define a curriculum for SLS; give tutorials and summerschools, provide complete case-studies of SLS algorithm development

SLS algorithm engineering (5)



SLS algorithm engineering (6)

Scientific issues

- close gap between theory and practice
- understand the relationship between performance, instance features and SLS algorithm components

Scientific GOAL: enhance our abilities to answer these issues

SLS algorithm engineering – cross-sectorial aspect

- SLS algorithms are widely applicable (from bioinformatics over telecommunications and engineering to business administration)
- advancements of methodological aspects have the high potential to have strong repercussion in many application fields

SLS algorithm engineering – structuring aspects

- research in SLS very much scattered into different directions; not clear how they go together
- SLS engineering offers orientation by defining important areas
  - methodological developments
  - systematic, in-depth experimental studies
  - development of new tools (F-races, LEDA, etc)
  - development of new algorithmic techniques (large-scale neighbourhoods, ACO, ..)
  - research in related scientific questions
  - theoretical advances
  - etc.

Future

SLS engineering

- definition of goals, framework, environment etc.
- steps to do
  - extract set of procedures followed in literature
  - test initial engineering procedure using challenging problems
  - adapt / extend / fine-tune set of engineering rules
  - further development of existing tools plus development of new ones
  - raise scientific questions regarding the understanding of the relationship between SLS algorithms, problem features, and performance.

Future (2)

other things

- workshop on SLS engineering (tentative date spring 2007)
- more publicity type of things (tutorials, summerschool)
- consider different types of problems (multi-objective, dynamic, stochastic)
- get in contact with industrial partners for possible projects

Future (2)

other things

- workshop on SLS engineering (tentative date spring 2007)
- more publicity type of things (tutorials, summerschool)
- consider different types of problems (multi-objective, dynamic, stochastic)
- get in contact with industrial partners for possible projects
- convince robotics people that they should add at some point some optimization tasks into their work

Future (2)

other things

- workshop on SLS engineering (tentative date spring 2007)
- more publicity type of things (tutorials, summerschool)
- consider different types of problems (multi-objective, dynamic, stochastic)
- get in contact with industrial partners for possible projects
- convince robotics people that they should add at some point some optimization tasks into their work
- apply for more funding to have between 34 to 67 PhD students and 11 PostDocs