Meta-heuristics for Combinatorial Optimization

Una Benlic

TESCO

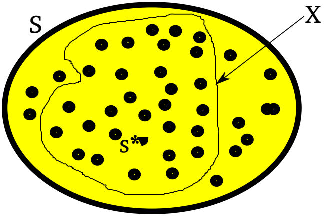
Plan

Combinatorial problem - definition

Given a couple (S, f) where:

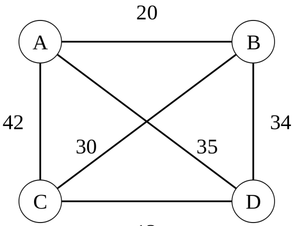
- S is a finite set of solutions or configurations (search space)
- $f : S \rightarrow R$ a cost function (or objective)

In case of a minimization prb., find $s^* \in X \subseteq S$ such that $f(s^*) \leq f(s)$ for each $s \leq X$ (feasible space).



Example: Traveling Salesman Problem (TSP)

- Find the shortest route between a set of locations that must be visited.
- Problem representation: Undirected complete weighted graph $G = (V, E)$ where V and E are the set of vertices and edges respectively.
- Solution representation: An ordered sequence of vertices in V.
- Search space size: $|V|!$. For $V = 10 \rightarrow 3628800$; $V = 100 \rightarrow 9.332621544 * 10^{157}$



Tesco's applications of TSP and its variants

- Hive and Bumblebee - routing of vehicles from depot to stores (stores to clients) so as to minimize a cost (travel time/distance, fuel, etc);
- Picking optimization - seeks to optimize a picker's routes through the store;

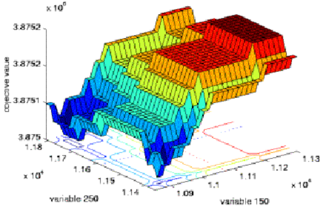
- Exact: Guarantee optimality, but generally require an exponential computing time with respect to the problem size. E.g., Constraint Programming (CP) and Mathematical Programming (Linear/Non-linear Programming, Integer Programming, etc.)
- Approximate: Search for high quality solutions, but not necessarily optimal, with reasonable computing efforts.
  - α-approximation algorithms: polynomial time algorthms that produce a solution whose objective value is within a factor of α of the value of an optimal solution.
  - Heuristics/metaheustics: polynomial time algorithms without any knowledge on the quality of attained solutions.

Four main (meta)heuristic approaches

1. Construction: Step-by-step instantiation of variables according to a static or dynamic order (greedy methods, etc.)
2. Local search: iterative improvement of a complete solution by local modifications e.g.,:
    - decent/hill climbing;
    - iterated local search;
    - simulated annealing;
    - tabu search;
    - variable neighborhood search.
3. Population-based algorithms: improvement (evolution) of a population of solutions e.g.,:
    - genetic algorithms;
    - scatter search.
4. Hybrids: combination of different approaches e.g.,:
    - memetic algorithms = local search + genetic algorithm;
    - math-heuristics = mathematical programming + heuristic;
    - hyper-heuristics = combination of lower level heuristics.

Intensification and diversification

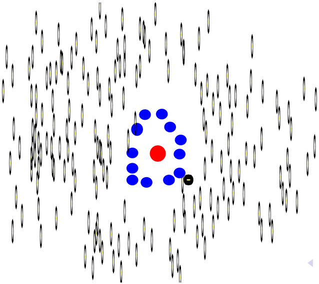
- Important to balance between intensification and diversification;
- *Intensification* - exploitation of a limited region of the search space;
- *Diversification* - exploration of new search space regions.



Local search - basic elements

Neighborhood

- Function $N : S \rightarrow 2^S, \forall s \in S, N(s) \subset S$,
  i.e., this function associates to each solution $s \in S$ a subset of S
- Defined by a move operator which performs local changes on the current solution
- $s \in S$ is a local optimum with respect to N if $\forall s' \in N(s)$,
  $f(s) \leq f(s')$ (for min. problem)



TSP: Neighborhood move operators

Figure: Insertion move operator

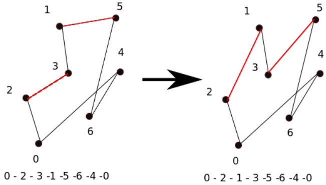
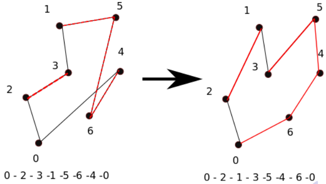


Figure: 2-opt move operator

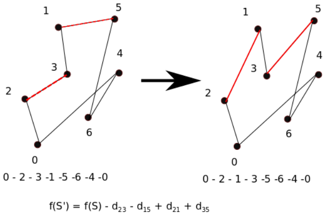


Local search - basic elements continued...

Evaluation function

- Function $f : S \rightarrow R, \forall s \in S$,
  i.e., this function evaluation the quality of solutions in S.

Figure: TSP: Insert move evaluation function



Move strategy

- Rules governing transition from the current solution to a neighboring solution.

Local search - general procedure

Step 1 (initialization)

- **a** choose an initial solution $s \in S$
- **b** $s^* \leftarrow s$ (i.e., record the best solution found so far)

Step 2 (choice and termination)

- **a** choose $s' \in N(s)$
- **b** $s \leftarrow s'$ (i.e., replace s by s')
- **c** terminate and return the best solution found if the termination criterion is verified.

Step 3 (update)

- **a** $s^* \leftarrow s$ if $f(s) < f(s^*)$
- **b** go to step 2

Remark: Meta-heuristics differ depending on the strategy used at step 2.

Local Search: Pure descent

step 2 (choice & termination)

(a) choose $s' \in N(s)$ such that $\forall s' \in N(S), f(s') < f(s)$

(b) $s \leftarrow s'$ (i.e., replace s by s')

(c) terminate if $\forall s' \in N(s), f(s') \geq f(s)$

Remark:

- Decisions to be taken
  - First improvement or best improvement
  - How to effectively and rapidly evaluate neighbors at each iteration (use of special data structures)
- Local optimum & remedy
  - Stop once a local optimum is found;
  - Random re-run;
  - Acceptance of non-improving neighbors;

Local Search: Simulated annealing

step 2 (choice & termination)

(a) choose randomly $s' \in N(s)$

(b) if $f(s') \leq f(s)$ then accept s, otherwise accept s with probability $p(\Delta, T)$

(c) terminate if stop condition is verified (eg., max nb of iterations)

Remark:

- Decisions to be taken
  - How to determine the probability $p(\Delta, T)$
  - How to effectively and rapidly evaluate neighbors at each iteration (use of special data structures)
- A search method based (partially) on randomness (exploration > exploitation)

Local Search: Tabu search

step 2 (choice & termination)

(a) choose the best neighbor $s' \in N(s)$ such that s' is not prohibited by the tabu list

(b) $s \leftarrow s'$ even if $f(s') > f(s)$

(c) terminate if stop condition is verified (eg., max nb of iterations)

Remark:

- Decisions to be taken
  - What to record in tabu list
  - How to determine the length (tabu tenure) of tabu list (dynamic or static)
  - How to evaluate rapidly the neighbors (move values) at each iteration
- Randomness is not essential (exploration < exploitation)

Other local search methods

- Variable Neighborhood Search (VNS): a set of (nested) neighborhood relations are alternatively used during the search process;
- Greedy randomized adaptive search procedure (GRASP): a hybrid method combining construction and local search;
- Iterated local search (ILS): alternates between an exploitation and exploration phase.

Local search - summary

- **Descent**: Choose an improving neighbor $s' \in N(s)$, i.e., $f(s') < f(s)$ fast but stops at the first local optimum.
- **Simulated annealing:** Choose randomly $s' \in N(s)$; if $(s') \leq f(s)$ then accept s', otherwise accept s' with probability $p(\Delta f, T)$
- **Tabu serach:** Choose the best neighbor $s' \in N(s)$, accept s' even if $f(s') > f(s)$ (use tabu list to prevent the search from cycling)

Remark: Simulated annealing and tabu search don't stop at the first local optimum encountered.

Local search - performance

- Convergance to a global optimum is not guaranteed;
- High quality experimental results for numerous hard problems;
- Adaptation is necessary:
  - problem encoding (configuration and search space);
  - neighborhood relations;
  - constraint handling;
  - data structures.
- Improvement with hybridization (local search + genetic algorithms, local search + construction approaches).

Evolutionary Approach

Basic concepts

- evolution of a set of configurations (notion of population)
- evolution operators (selection, recombination and mutation)

General procedure

step 1 : *(initialization)* choose a set of initial configurations (population)

step 2 : *(evolution)* application of recombination and mutation operators

step 3 : *(update)* re-organization of the population (e.g., elimination of bad configurations from the population)

Remarks:

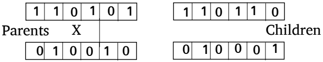
- different schools: genetic algorithms, evolutionary strategies, evolutionary programming;
- a general and powerful framework for algorithm design.

Simple genetic algorithms (John H. Holland 75)

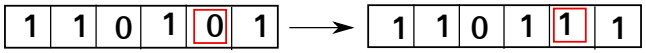
Main features:

- universal problem representation based on binary encoding (binary strings)
- random genetic operators (mutation and crossover)

Crossover: exchange of sub-strings between two individuals (monopoint, bi-points, uniform)



Mutation: random modification of bit values of a new configuration



Evolutionary approach: In practice

Specialization:

- Specialized encoding *adapted* to each problem (e.g., permutation for TSP)
- Specialized evolution operators based on the specialized encoding

Hybrid:

- with contruction approaches
- with local search

Evolutionary approach: performance

- Convergance towards global optimum is not guaranteed
- Weak results for combinatorial optimization with simple GA (blind mutation and crossover)
- Competitive results with *specialized* GA
  - problem specific encoding
  - problem specific evolution operators integrating problem knowledge
- Very competitive results with hybrid GA
  - hybrid with construction methods
  - hybrid with local search (memetic algorithms)

Hybrid genetic algorithms

Basic idea: Combine 2 complementary methods - *global* search and *local* search

step 1 (initialization)

- **a)** generate a population of configurations P
- **b)** **apply a local search to each configuration of the population P**

step 2 (evolution and termination)

- **a)** choose p_1 and p_2 in P
- **b)** generate a configuration e by a recombination of p_1 and p_2
- **c)** **improve e by local search**
- **d)** insert the improved e in the population
- **e)** terminate and return the best solution found when stop condition is verified

step 3 (update)

- **a)** re-organization of the population (elimination of bad configuarions from the population)

Adaptation of metaheuristics

Problem solving with meta-heuristics

- Problem modeling
- choice of a meta-heuristic according to
  - the solution quality required
  - the availability of problem knowledge
  - the know-how...
- adaptation of the chosen meta-heuristic to the problem
  - configuration (search space)
  - neighborhood and evaluation function
  - search operators and constraint handling
  - data structures ...

Performance evaluation (benchmarking whenever possible)

- The quality of the best solution found
- Search profile (time vs. quality plot)
- efficieny, i.e., efforts (computing time, number of iterations) necessary to reach the best solutions
- robustness

Conclusion

Strong points

- General and applicable to a large class of problems
- Possibility of time-quality compromise
- Preferred application domains: large combinatorial optimization problems that are not highly constrained.

Weak points

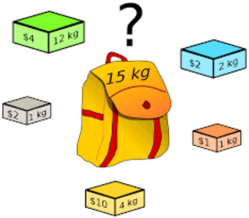
- Only local optimum
- Adaptation indispensible
- Difficult to predict the performance (quality and time)

Performance

- Theory: No proof of convergence towards an optimal solution
- Practice: Depends on each adaptation (problem encoding, integration of problem knowledge, constraint handling, data structures...)

Knapsack problem

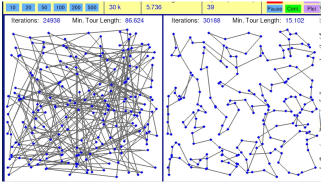
Given a knapsack of a limited capacity k and a set of items $I = \{(w_1, v_1), (w_2, v2), ..., (w_n, v_2)\}$, where w_i and v_i are weight and value of $i \in I$, select a subset of items to place in the knapsack.



Apply any local search framework of your choice and decide on the main algorithm elements: the solution and problem representation, neighborhood operator(s), move evaluation and objective function.

Genetic algorithm for TSP

Propose a design of a genetic algorithm for TSP (crossover operator, mutation operator, population acceptance strategy).



Crossovers for TSP

Figure: Uniform crossover

Parent 1	0	3	1	2	5	7	4	6	8	0
Parent 2	0	1	3	2	6	7	4	5	8	0
Offspring	0	5	6	2	3	7	4	1	8	0

Figure: Single-point crossover

Parent 1	0	3	1	2	5	7	4	6	8	0
Parent 2	0	1	3	2	6	7	4	5	8	0
Offspring	0	3	1	2	6	7	4	5	8	0

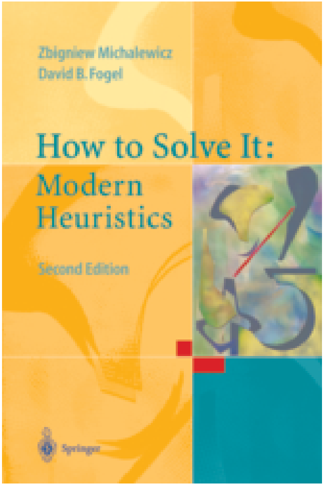
School timetabling problem

Let C be a set of courses, P the set of periods in a week ($\{day, timeSlot\}$ pairs), R the set of rooms. Create a timetable such that the following hard and soft constraints are satisfied:

- $H1$: ensure that a course $c \in C$ takes place at least c_n times per weak;
- $H2$: ensure that no course takes place in the same room at the same time;
- $H3$: ensure that a teacher is available to teach at given time period $p \in P$;
- $H4$: ensure that no two courses of the same curriculum take place at the same time;
- $S1$: the number of students attending a course needs to be less than or equal to the number of seats in the room;
- $S2$: minimize the break between courses of the same curriculum.

Book for beginners

How to Solve It: Modern Heuristics
Authors: Michalewicz, Zbigniew, Fogel, David B.



Thank you :)