Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering
IAMP Institut für Angewandte
Mathematik und Physik

Lecture 3: Exercises

MSE Algorithms - Metaheuristics

Exercises

*** Relevant for Exam**

*Task 1: Manual Local Searches

An integer function [–7, 7] x [-6, 7] → [–10, 754] is explicitly given in the following table. The moves consist of modifying a single variable by +/- one unit. The moves are ordered as follows: (+1, 0), (0, +1), (–1, 0) and (0, –1). The goal is, to minimize the function value by means of a local search using

a) First Improving Move strategy, starting from the upper right corner (x=7, y=-6).

b) Best Improving Move strategy, starting form upper left corner (x=–7, y=–6).

y\x	–7	–6	–5	–4	–3	–2	–1	0	1	2	3	4	5	6	7
-6	248	216	210	222	230	234	256	304	336	372	428	495	585	650	754
-5	193	175	157	166	174	181	215	249	295	329	382	454	539	597	707
-4	138	144	126	116	124	150	184	194	250	305	361	425	480	566	646
-3	123	89	85	97	105	109	129	179	209	246	302	368	458	525	627
-2	92	58	70	70	78	94	98	148	168	223	282	339	413	510	582
-1	68	34	46	46	54	70	74	124	144	199	258	315	388	486	558
0	51	17	14	25	33	38	57	107	136	174	230	296	386	454	555
1	18	25	5	-4	3	29	65	74	131	185	240	305	361	445	527
2	27	6	-10	0	8	13	46	83	126	160	213	284	371	429	539
3	33	0	-3	7	15	20	39	89	118	156	212	278	368	436	537
4	33	12	-4	6	14	19	52	89	132	166	219	290	377	435	545
5	30	37	17	7	15	41	77	86	143	197	252	317	373	457	539
6	69	35	32	43	51	56	75	125	154	192	248	314	404	472	573
7	92	58	70	70	78	94	98	148	168	223	282	339	412	510	582

Exercises

***Task 2: 2-opt and 3-opt moves for TSP**

For a Travelling Salesperson Problem instance with 8 cities, consider the tour:

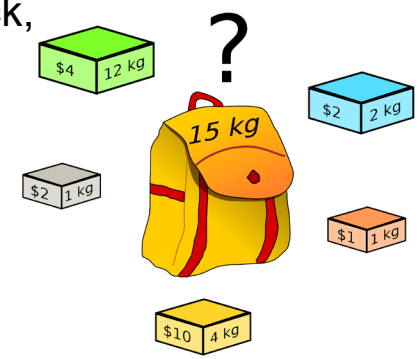
c1-c2-c3-c4-c5-c6-c7-c8-c1.

a) Give the resulting tour of applying the 3-opt move, where the edges c2-c3 and c4-c5 and c6-c7 are replaced, such that no sub trail is reversed.
b) Same as in a), but this time the two shortest sub trails are reversed.
c) Give the resulting tour of applying the 2-opt move, where the edges c2-c3 and c6-c7 are replaced, such that the sub trail containing c1 is not reversed.

Exercises

Task 3: Greedy for Knapsack Problem

The Knapsack Problem is defined as follows: Given a set of n items numbered from 1 up to n, each with a weight w_i and a value v_i, along with a maximum weight capacity W of the knapsack,



$$\text{maximize} \sum_{i=1}^{n} v_i x_i \qquad \text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0, 1\}.$$

a) Consider a "Naïve Greedy" algorithm, which always takes the item with maximum value that still fits into the knapsack. Show that this algorithm can create arbitrarily bad solutions.

b) Algorithm "Smarter Greedy" works as follows: compute v_i/w_i for each item, and sort all items in descending order. Then take items in this order as long as they fit into the knapsack.
This algorithm can create good solutions, but also bad ones. Show that there exists an example with two items where the solution is arbitrarily bad.

c) What are reasonable neighbourhoods for the Knapsack Problem?

Exercises

Note that sample data and implementation for knapsack heuristics is provided on Moodle.

Task 4: Knapsack Problem with Simulated Annealing

Note:

- A ready to use Simulated Annealing heuristic for the knapsack problem is provided in the Python framework on Moodle (run `demo_knapsack_simulated_annealing.py`).
- The neighbourhood used in this heuristic is “Flip a single item” (check the method `move()` in `knapsack_simulated_annealing.py` in the subfolder `Python -> heuristics`.
- The test instance shown on the following slide is provided on Moodle as well.

a) Run the algorithm for different values of start and end temperature and the number of iterations (check the call `ks.set_schedule()` in `demo_knapsack_simulated_annealing.py`). Give reasonable values for the parameters which lead to good solutions at good running times.

b) Implement a different reasonable neighbourhood. Compare the results and running times achieved with your neighbourhood to those found in a).

Knapsack Instance

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
IAMP Institut für Angewandte Mathematik und Physik

N = 24

W = 6404180

Weights:

382745
799601
909247
729069
467902
44328
34610
698150
823460
903959
853665
551830
610856
670702

Values:

825594
1677009
1676628
1523970
943972
97426

488960
951111
323046
446298
931161
31385
496951
264724
224916
169684

69666
1296457
1679693
1902996
1844992
1049289
1252836
1319836
953277
2067538
675367
853655
1826027
65731
901489
577243
466257
369261

Source: http://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/knapsack_01.html

Exercise

Note that sample data and implementation for TSP heuristics is provided on Moodle.

Task 5: Simulated Annealing for TSP

Note:

- A ready to use Simulated Annealing heuristic for the TSP problem is provided in the Python framework on Moodle (run `demo_tsp_simulated_annealing.py`).
- The neighbourhood used in this heuristic is "Swap two cities" (check the method `move()` in `tsp_simulated_annealing.py` in the subfolder `Python -> heuristics`.
- There are various test instances of the TSP problem provided on Moodle as well.

a) Run the algorithm for different values of start and end temperature and number of iterations (check the call `tsp.set_schedule()` in `demo_tsp_simulated_annealing.py`). Note that the optimal values of these parameters vary, when solving different instances.

b) Implement a different neighbourhood (e.g. 2-opt or 3-opt). Document your best results for the provided instances of TSP achieved with Simulated Annealing at https://docs.google.com/forms/d/e/1FAIpQLSckpYlWYSNPc24MDi3SYxeLs1hwmKEFY1BLuSfRIGJ2SGoesw/viewform?usp=sf_link