TU Wien
Institute for Logic and Computation
Algorithms and Complexity Group

Heuristic Optimization Techniques, WS 2024

Programming Exercise: Assignment 1

v1, 17-10-2024

Work on the problem specified in the separate "Problem Description and General Information" document within your programming exercise group. You are encouraged to split tasks evenly, but everybody has to understand and to be able to explain all the concepts involved, your overall implementation, and the submitted report. Hand in your complete report, submit the solutions for the instances, and upload your source code via TUWEL by **Sunday, 24th November 2024, 23:55**. For questions please contact us via `heuopt@ac.tuwien.ac.at`

The first programming assignment is to develop your *own* construction and local search based metaheuristics for the *Minimum Weighted Crossing with Constraints Problem* (MWCCP). The subtasks for this exercise are:

1. Think about a meaningful real-world application for this problem and briefly describe it.
2. Develop a meaningful deterministic construction heuristic.
3. Derive a randomized construction heuristic to be applied iteratively.
4. Develop or make use of a framework for basic local search which is able to deal with
   - different neighborhood structures
   - different step functions (first-improvement, best-improvement, random)
5. Develop at least three different meaningful neighborhood structures.
6. Develop or make use of a Variable Neighborhood Descent (VND) framework which uses your neighborhood structures.
7. Implement a Greedy Randomized Adaptive Search Procedure (GRASP) using your randomized construction heuristic and an effective neighborhood structure with one step function or (a variant of) your VND. Note that the union of existing neighborhood structures also constitutes a (composite) neighborhood structure.
8. Implement ***one*** of the following metaheuristics:
   - General Variable Neighborhood Search (GVNS) on top of your VND
   - Simulated Annealing (SA)
   - Tabu Search (TS)
9. Use delta-evaluation. Explain which steps in your algorithm use delta-evaluation and describe why delta-evaluation results in better performance in this step. Are there other elements in your algorithm that could have also benefitted from delta-evaluation? If possible analyze the asymptotic runtime of a step using delta-evaluation, and of a step without delta-evaluation. Is it possible to pre-process an instance and use derived information later more efficiently?
10. Perform some experimental manual tuning of relevant algorithmic parameters to find sensible parameter settings for your final experiments. Relevant parameters may be related to the degree of randomization, neighborhood structure sizes, probabilities for the random step function in composite neighborhood structures, the cooling schedule, the tabu list length and its variation, etc. Report the impact of a number of different settings on the solution quality of a selected meaningful subset of instances.

11. Run experiments and compare the following algorithms on the instances of different sizes provided in TUWEL:
    (a) deterministic and randomized construction heuristic and GRASP
    (b) Use the solution of the deterministic construction heuristic to test the other implementations:
        i. Local search for at least three selected (possibly composite) neighborhood structures using each of the three step functions (i.e., at least nine different algorithm variants).
        ii. VND
        iii. GVNS, SA, or TS
    (c) Report on measures such as running time, iterations, average final objective and objective over time. Run each algorithm on each instance multiple times to reduce statistical variance.
    (d) Prepare plots showing the above measures in a concise fashion.
12. Write a report containing the description of your algorithms, the experimental results and what you conclude from them; see the *general information* document for more details.

Some questions to aid your development are:

- How is your solution represented?
- How do you generate different solutions? Which parts of your algorithm can be reasonably randomized and how can you control the degree of randomization? Is it possible to randomize other parts and are your random solutions sufficiently diverse?
- What parameters do you use and which values do you chose for them and why?
- Can subsequent, possibly non-improving, moves in your neighborhood structures reach every solution in the search space?
- Local search: How many iterations does it take on average to reach a local optima? What does this say about your neighborhood structures? Does this change when using different step functions?
- How does delta evaluation work for your neighborhood structures? Can some calculations be simplified using a preprocessing procedure?
- What is the time complexity to fully search one neighborhood of your neighborhood structures?
- VND: Does the order of your neighborhood structures affect the solution quality?

We hope you enjoy these tasks and wish you success!