Vienna University of Technology

192.137 Heuristic Optimization Techniques

TU Wien Algorithmics and Complexity Group

Programming Project 1

Authors:

Tobias Sloviak
01204691

Camilo Tello Fachin
12127084

Supervisors:

Prof. Dr. Günther Raidl

Dipl. Ing. Enrico Iurlano

Dipl. Ing. Laurenz Tomandl

November 30, 2023

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Contents

To Do's

- either make framework work or create new framework
- analyze locks in lecture slides and in paper:
  - Test-and-Set Lock
  - Test-and-Test-and-Set Lock
  - Ticket Lock
  - Array Lock
  - CLH Lock
  - MCS Lock
  - Hemlock (from paper..)
- in the lectureslides from the project it says:
  - Benchmark the following two under various scenarious, meaning low/high contention
  - Benchmark Throughput (probably locks-unlocks performed per time)
  - Benchmark Latency
  - Benchmark for fairness, we have to think of some number to measure fairness, maybe locks/unlocks achieved per thread

1 Introduction

2 Questions

In the end, here will be the answer to the question "Think about a meaningful real-world application for this problem and briefly describe it."
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 a set of meaningful neighborhood structures to address the different aspects of the problem, i.e., related to the quality of the s-plexes, or the assignment of nodes to s-plexes.
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 vari- ant 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. Perform some manual tuning of relevant algorithmic parameters to find sensible parameter settings for the 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. 10. 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?
11. Run experiments and compare all your algorithms on the instances 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
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.

3 Questions to Consider during developtment

• How is your solution represented?
• Ad 4: How do you generate different solutions? Which parts of your algorithm can be reasonably randomized and how can you control the degree of randomization?
• Ad 3 and 4: Does randomization and iterated application improve the generated solutions?
• What parameters do you use and which values do you chose for them?
• Can subsequent – possibly non-improving – moves in your neighborhood structures reach every solution in the search space? Or at least one optimal solution? • Local search: How many iterations does it take on average to reach a local optima? What does this say about your neighborhood structures?
• How does incremental evaluation work for your neighborhood structures?
• 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?

4 Real World application

A general s-plex application that is probably used is the identification of group or friend networks where one can make assumptions about group or friend networks. This is application is not directly derived from our programming assignment, but it is somewhat related since one still has to identify k-plexes in a undirected graph which with the amount of people using social media these days is not trivial. An application that is closer to the given s-plex programming assignment would be telecommunication networks. Where the distance between the antennas, towers or compute node could be correlated to the weight matrix and the s-plex relaxation of the clique's could be used to make crucial system more reduntant (low s-plexes) and other less crucial systems less reduntant (higher s-plexes).

5 Deterministic Construction Heuristic

Our first idea for a construction of solutions is the following: An empty graph (without edges) is always a 1-plex, since a single node without edges is a 1-plex. Removing all the edges gives a valid solution, however a very bad one. Our first idea for an algorithm that finds a good solution was to start with an empty graph and go through all the edges in A_0, check if adding them to A is legal and if so, do that. This algorithm improves the empty graph, however it can only find s-plexes of up to $|S| = s + 1$. The reason is that it will connect the first node v_1 with $\min(s, \deg_{A_0}(v_1))$ many other nodes, then the cluster of v_1 is an s-plex with size at most $s + 1$ and no more nodes can be added. Then, all the other nodes in the cluster might get connected among each other, but the s-plex will not get any bigger. Practically speaking, with this construction we got approximately $n/(s + 1)$ many clusters of size $s + 1$ which were mostly fully connected, so this construction results in small clusters.

deterministic construction heuristic where we explain how the construction heuristic is done. Maybe only do on the problem with 9 vertices and plot some results.

6 Randomized Construction Heuristic

For our random construction we randomly parse the entries of A_0 but do the same as above otherwise.

6.1 Local improvement move operators

After this point we found it difficult to think of the next move, because simply adding one edge in this state will always yield an illegal solution. In trying to understand the problem better, we wrote the adjacency matrices A_0 in files and looked at their structure. What we found was that there is a structure in most of the graphs, more precisely there are 3 types of graphs (see also figure 1):

1. locally strongly connected: 26/60 (Here, nodes that are close in numbering are also strongly connected in the graph)
2. seemingly randomly connected: 23/60
3. tentacle-like connected: 7/60 (Here, the Graphs have a semi-large, strongly connected cluster with ”tentacles” reaching out. Tentacle-shaped clusters are bad for s-plexes, so we will have to seperate these ”lines” into small ”spheres”.)

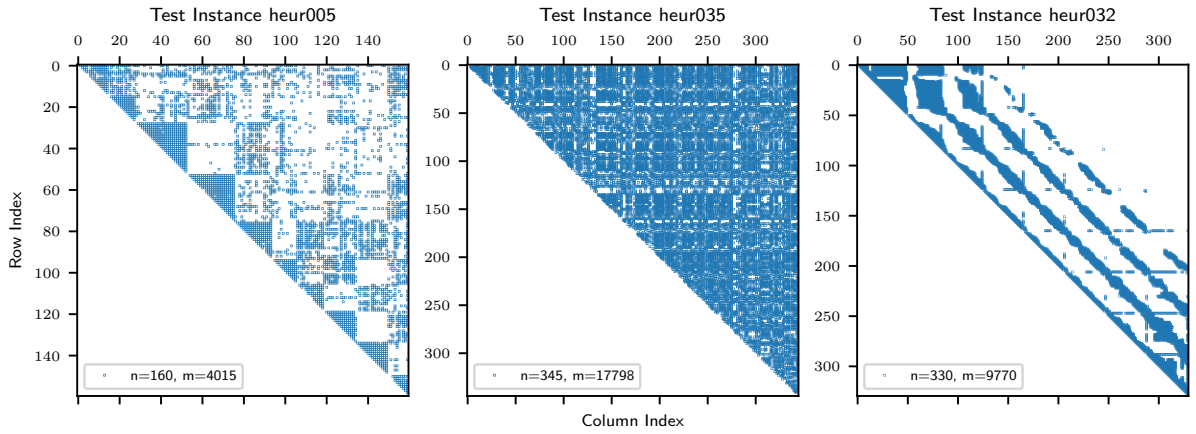

Figure 1: Instances of the different graph types. Left is a locally strongly connected graph, this is part of instance 005. In the middle is a seemingly randomly connected graph, which is part of instance 035. On the right is a tentacle-like connected graph, which is part of instance 032.

We will use this special structure of some instances to derive parts of our algorithms. Some might consider this cheating, but we think, that finding patterns in the instances is an important part of solving problems in real world applications so we find it to be okay. We will see later, that for the locally strongly connected instances we can find reasonably good solutions with very little computational effort. We will also see that the algorithms that were derived with the locally connected graphs in mind also work well on the other graphs.

7 Neighborhood Structures and Step Functions

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 a set of meaningful neighborhood structures to address the different aspects of the problem, i.e., related to the quality of the s-plexes, or the assignment of nodes to s-plexes.

8 Variable Neighborhood Descent (VNS)

text

9 Greedy Randomized Adaptive Search Procedure (GRASP)

10 General Variable Neighborhood Search (GVNS) on top of VND

11 Manual Paramter Tuning

Perform some manual tuning of relevant algorithmic parameters to find sensible parameter settings for the 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.

12 Delta Evaluation

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?

13 Experimental Results

14 Results and Conclusion

Run experiments and compare all your algorithms on the instances 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