Infrastructure Optimization for Autonomous Vehicles

Candidate numbers: 10062, 10019, 10041

March 30 2021

1 Introduction

Autonomax AB is a company intending to implement autonomous vehicle transportation between cities in Sweden. The vehicles are dependent on a communication infrastructure made up by cables along the routes where the vehicles are supposed to operate, forming a connected network. Our task is to design an optimization model which can be used as decision support, to suggest how the network should be constructed to minimize the cost of the infrastructure. Construction of the network is restricted by some given requirements for network structure, and the bandwidth demand related to each city.

1.1 Network structure requirements

Communication in the network is transmitted from a control center located in one of the cities, which is to be decided by the model. It is required that a predetermined number of cities should be directly or indirectly connected to the control center using grafen cables, forming what is defined as a core network. Grafen cables have an expensive per-km price, but infinite bandwidth capacity, making the bandwidth demand for the cities part of the core network unrelated to the cost. The core network can either be constructed as a cycle or as a path, which should be a predetermined parameter. The remaining cities must be directly or indirectly connected to the core network using regular cables, spanning out from the core network. These structures are defined as sub networks. Regular cables have a cost function dependent on both the length and the bandwidth requirement of the cable. Sub networks can be viewed as independent spanning trees, connected to the core network through a root node. The bandwidth required for each edge is accumulated upwards in the tree according to the demand of the cities further down in the tree (i.e. further away from the root).

1.1.1 Cost functions

The cost function of grafen cables is described by the following formula

$$Cost_{grafen} = 10D_{i,j},$$

where $D_{i,j}$ is the distance between city i and j in kilometers.

The cost function of regular cables is described by

$$Cost_{regular} = (0.1 * D_{i,j})^{1.5} b_{i,j},$$

where $D_{i,j}$ is the distance between city i and j and $b_{i,j}$ is the bandwidth requirement for the cable.

1.1.2 Assumptions

In this project we have assumed that when the core net is a cycle, it can only contain exactly one cycle. When the problem description states that the control center needs to be directly connected to the core net, we assume this means that the control center needs to be a part of the core net.

2 Mathematical formulation

In the mathematical formulation and implementation we refer to cities as nodes, grafen cables as core edges and regular cables as sub edges.

Sets
N - Set of all nodes.
Indices
i,j - To index specific node or cable between edges $i \in N$ and $j \in N$
Parameters
$D_{i,j}$ - Distance from node i to node j.
B_i^W - Bandwidth demand of node i
C^N - Number of nodes in core net
Variables

$$x_{i,j} = \begin{cases} 1, & \text{if edge from i to j is a core edge} \\ 0, & \text{otherwise} \end{cases}$$

$$y_{i,j} = \begin{cases} 1, & \text{if edge from i to j is a sub edge} \\ 0, & \text{otherwise} \end{cases}$$

$b_{i,j}$ - Bandwidth requirement of edge from node i to node j

$$w_i = \begin{cases} 1, & \text{if node } i \text{ is in the core net} \\ 0, & \text{otherwise} \end{cases}$$

$$s_i = \begin{cases} 1, & \text{if control center is located in node i} \\ 0, & \text{otherwise} \end{cases}$$

Objective function
The objective function consists of the cost of all core edges and all sub edges in the network with respect to the cost functions defined in section 1.1.1.

$$\min z = \sum_i \sum_j (10 x_{i,j} D_{i,j} + 10 y_{i,j} + (0.1 D_{i,j})^{1.5} b_{i,j}) \tag{1}$$

s.t.

Constraints

$$x_{i,j} + y_{i,j} \leq 1 \quad \forall i, j \in N \tag{2}$$

$$M(1 - \sum_j y_{j,i}) + \sum_j b_{j,i} \geq B_i^W + \sum_j b_{i,j} \quad \forall i \in N \tag{3}$$

$$w_i \leq \sum_j x_{j,i} + \sum_j x_{i,j} \quad \forall i \in N \tag{4}$$

$$\sum_j y_{j,i} = 1 - w_i \quad \forall i \in N \tag{5}$$

$$\frac{1}{2} \sum_j x_{j,i} + \frac{1}{2} \sum_j x_{i,j} \leq w_i \quad \forall i \in N \tag{6}$$

$$\sum_i \sum_j x_{i,i} = C^N - (1 - Z) \tag{7}$$

$$\sum_i \sum_j x_{i,j} \leq |S| - 1, \quad \forall S \subset N : 2 \leq |S| \leq C^N - 1 \tag{8}$$

$$\sum_i w_i = C^N \tag{9}$$

$$\sum_i s_i = 1 \tag{10}$$

$$s_i \leq w_i \quad \forall i \in N \tag{11}$$

$$\sum_j x_{j,i} + \sum_j y_{j,i} + s_i \geq 1 \quad \forall i \in N \tag{12}$$

$$b_{i,j} \leq M y_{i,j} \quad \forall i, j \in N \tag{13}$$

$$w_i, s_i \in \{0, 1\} \quad \forall i \in N \tag{14}$$

$$x_{i,j}, y_{i,j} \in \{0, 1\} \quad \forall i, j \in N \tag{15}$$

$$b_{i,j} \geq 0 \quad \forall i, j \in N \tag{16}$$

Constraint (2) ensures there is only one sub or core edge along any possible edge. Constraint (3) handles the bandwidth of the sub net, ensuring that if there is a sub edge into a node, the bandwidth into the node is larger than the demand of the node plus the bandwidth out of the node. Constraint (4) ensures that all core nodes have at least one core edge in or out. Constraint (5) ensures that no core edges have incoming sub edges, and that all sub nodes have exactly one incoming sub edge, forcing the tree structure of the sub nets. Constraint (6) forces nodes with incoming or outgoing core edges to be assigned core nodes, which together with constraint (4) ensures that all core nodes are connected to the core net. However, sub tour elimination is needed to detect multiple cycles. Constraint (7) ensures that the number of core edges is correct in reference to the structure, which ensures the core structure. Constraint (8) is the sub-tour elimination, making sure that there are no cycles in the core net not connected to the main core net (i.e. no sub-tours/sub-cycles). Constraint (9) ensures the correct number of core nodes, while constraints (10) and (11) enforces that there is only one control center node, and that it is a core node, respectively. Constraint (12) ensures that all nodes except the control center node has a parent node, which ensures a flow from the control center node. Finally, constraint (13) ensures that only sub edges are assigned bandwidth.

3 Formulation analysis

3.1 Structures

The model can, aside from a few complicating constraints, be split into three parts. The first part is choosing which cities to choose as core nodes, and finding the best core edges between them. The second part is building the sub nets and assigning sufficient bandwidth to the sub net edges. A third, fairly insignificant part is deciding a control center node.

The core-part of the problem draws similarities to the traveling salesman problem (TSP), with the difference being that one must also choose what nodes to include in the TSP. The process of selecting core nodes could be compared to a binary knapsack problem, since we must choose from a set of items and only have room for a given number of them. However, we don't have a value of each item (city), so this would have to be approximated based on parameters like bandwidth demand or neighbours' demand. Note that we have formulated the problem as a directed graph, something that is not required as of the problem description, and changing this in the formulation will turn the subproblem into a symmetric TSP instead of the current asymmetric TSP.

The second part of the problem has many similarities with the network design problem (NDP). Instead of a minimum flow on edges, our formulation has a demand on each node, and a constraint (3) on flow to meet the demand in succeeding nodes. Instead of having a limited set of possible edges, our formulation has all possible edges between nodes. Aside from this and the objective function, the formulation is close to the NDP, and if the core net is decided, the rest of the problem might be easier to solve taking this into account.

The final part of the formulation, not necessarily chronologically, is deciding a control center node. This was added to help in other constraints, but is not necessary to solve the problem, as any core node can just be assigned as the control center node afterwards. Removing this would make for fewer variables and thus make it easier to solve and to decompose the constraints, and make the formulation more similar to the aforementioned models. This would include removing constraints (10) and (11), and reformulate constraint (12).

Such structures can be exploited by various solution methods, for example by a construction heuristic. First choosing core nodes as a 0-1 knapsack problem with the aforementioned value approximations, then solving the following TSP through branch-and-bound would yield a core network. Then we could solve the sub network part as a NDP, with the differences discussed above. Finally, a control center node would be chosen, and a solution is reached. This could be paired with a local search heuristic to guide a solution method, which we will discuss in section 5.

A structure to consider is the prevalence of binary variables, and the fact that the number of non-zero binary variables of each kind is known. This could be exploited by branch-and-bound algorithms by branching on for example sets of core nodes rather than a random selection of variables, which forms a limited branch-and-bound method. It would let the algorithm quickly search through a larger number of feasible solutions, since less solutions breaking constraint (9) would be proposed. Consequently, smaller trees would be created, reducing the memory usage.

Another structure to consider is the number of constraints, indicating that column generation might be exploited in solution methods. With a large number of binary variables, where a majority of them are zero, a restricted master problem (RMP) can be formulated encompassing fewer variables, and alternating between solving this together with the subproblems outlined above would likely provide a quicker solution method.

3.2 Dantzig-Wolfe reformulation

While identifying these different parts might help solution methods, the different subproblems are difficult to separate due to complicating constraints, like constraints (2) and (12). Therefore, a Dantzig-Wolfe reformulation could yield a RMP. The subproblems will be the three parts outlined in section 3.1: Subproblem 1 will consist of constraints (3) and (13), 2 will consist of constraints (4) and (6)-(9), while 3 only includes constraint (10). This leaves the master problem to consist of constraints (2), (5), (11) and (12). The separation produces a primal block angular structure to the problem illustrated in Figure 1.

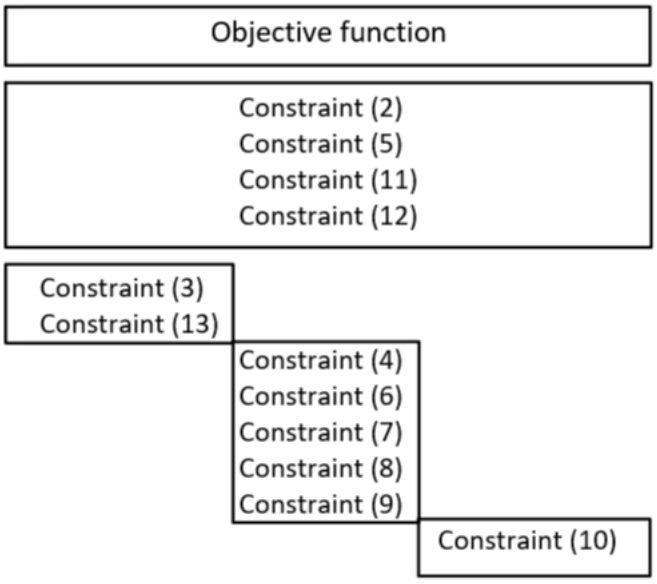

Figure 1: Possible block angular structure of the problem.

The extreme points from subproblem 1 are weighted by the variable λ_l^1, where l is used to index the extreme points. Similarly, the extreme points from subproblem 2 and 3 are weighted by the variables λ_k^2 (indexed by k) and λ_m^3 (indexed by m) respectively. The master problem can thus be formulated as:

$$\min z = \sum_i \sum_j (10D_{i,j}(\sum_k x_{i,j}^k \lambda_k^2) + 10(\sum_l y_{i,j}^l \lambda_l^1) + (0.1D_{i,j})^{1.5}(\sum_l b_{i,j}^l \lambda_l^1)) \tag{17}$$

s.t.

$$\sum_k x_{i,j}^k \lambda_k^2 + \sum_l y_{i,j}^l \lambda_l^1 \leq 1 \quad \forall i,j \in N \tag{18}$$

$$\sum_j (\sum_l y_{j,i}^l \lambda_l^1) = 1 - \sum_k w_i^k \lambda_k^2 \quad \forall i \in N \tag{19}$$

$$\sum_m s_i^m \lambda_m^3 \leq \sum_k w_i^k \lambda_k^2 \quad \forall i \in N \tag{20}$$

$$\sum_j (\sum_k x_{j,i}^k \lambda_k^2) + \sum_j (\sum_l y_{j,i}^l \lambda_l^1) + \sum_m s_i^m \lambda_m^3 \geq 1 \quad \forall i \in N \tag{21}$$

$$\sum_l \lambda_l^1 = 1 \tag{22}$$

$$\sum_k \lambda_k^2 = 1 \tag{23}$$

$$\sum_m \lambda_m^3 = 1 \tag{24}$$

$$\lambda_l^1 \geq 0 \quad \forall l \tag{25}$$

$$\lambda_k^2 \in \{0,1\} \quad \forall k \tag{26}$$

$$\lambda_m^3 \in \{0,1\}; \quad \forall m \tag{27}$$

The objective function (17) and constraint (18)-(21) are the same as in the original model, but the original variables are substituted by the weighted sum of the extreme points from their respective subproblems. Constraint (22)-(24) can be considered as convexity constraints. The lambdas from subproblem 2 and 3 are binary (constraint (26)-(27)), because all of the variables in these subproblems are binary so there cannot exist a feasible integer point that is not an extreme point. This is not the case for subproblem 1, so the corresponding lambda is not binary (constraint (25)).

To simplify the model, we will define a couple of parameters on a more aggregated level. This is done by re-arranging the order of summation to obtain the following parameters:

Cost of core net k.

$$C_k^C = \sum_i \sum_j 10D_{i,j} x_{i,j}^k \tag{28}$$

Cost of sub net configuration l.

$$C_l^S = \sum_i \sum_j (10y_{i,j}^l + (0.1D_{i,j})^{1.5} b_{i,j}^l) \tag{29}$$

Number of incoming sub net edges for city i in sub net configuration l.

$$I_{i,l}^S = \sum_j y_{j,i}^l \tag{30}$$

Number of incoming core net edges for city i in core net k.

$$I^C_{i,k} = \sum_j x^k_{j,i} \tag{31}$$

This leads to the simplified master problem:

$$\min z = \sum_k C^C_k \lambda^2_k + \sum_l C^S_l \lambda^1_l \tag{32}$$

s.t.

$$\sum_k x^k_{i,j} \lambda^2_k + \sum_l y^l_{i,j} \lambda^1_l \leq 1 \quad \forall i,j \in N \tag{33}$$

$$\sum_l I^S_{i,l} \lambda^1_l = 1 - \sum_k w^k_i \lambda^2_k \quad \forall i \in N \tag{34}$$

$$\sum_m s^m_i \lambda^3_m \leq \sum_k w^k_i \lambda^2_k \quad \forall i \in N \tag{35}$$

$$\sum_k I^C_{i,k} \lambda^2_k + \sum_l I^S_{i,l} \lambda^1_l + \sum_m s^m_i \lambda^3_m \geq 1 \quad \forall i \in N \tag{36}$$

In addition to constraints (22)-(27) from the non-simplified master problem.

This Dantzig-Wolfe formulation might not be very appropriate, as it has master problem with 4 constraints and one subproblem with only one constraint. To alleviate this, we could instead include constraints (10) and (12) in subproblem 2. This would yield a smaller master problem of 3 constraints ((2), (5) and (11)), and 2 subproblems. This would lead to the new primal block angular structure illustrated in Figure 2.

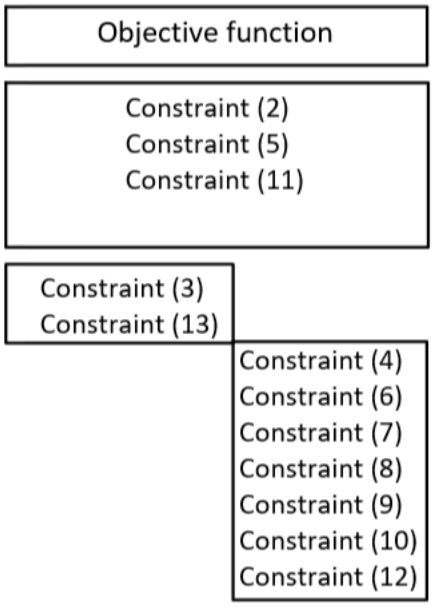

Figure 2: Another possible primal block angular structure.

With the master problem (MP) formulated, column generation should produce a solution faster than a normal branch-and-bound method. However, this would require solving the subproblems efficiently, and the

structures of the subproblems allow for several techniques to be utilized. Subproblem 2 consists of choosing core nodes, which can be seen as a binary knapsack problem, and finding the shortest path between them, which is similar to a TSP. The knapsack part can be solved through dynamic programming. By modeling as a shortest-path problem with resource constraints and produce labels as one either picks a node as a core node or leaves it as a sub node, one can iterate through all nodes to find the optimal set of core nodes. Like described previously in this section, this would require assigning a value to core nodes, based on demand, since this is included in the labels. The TSP part of subproblem 2 should be easily solved by a traditional B&B, as the number of core nodes is very small. Subproblem 1 consists of choosing sub edges and assigning sufficient bandwidth, similar to a NDP. This can be solved dynamically by creating labels as nodes are added to sub-trees, and choosing the dominating labels. This process would be easier to solve by adding a constraint on the number of sub edges, as this is predetermined. Subproblem 3 is trivial to solve, as a control center node can just be chosen from the core nodes.

Finally, with a structure allowing for column generation and solution methods for each subproblem, we have laid the foundation for a branch-and-price solution method. By branching on the binary variables, solving the MP and using it's duals to solve the subproblems, we generate a new column and add it to the RMP. This should quickly produce a solution, as the number of non-zero variables is small compared to the 5002 variables.

4 Implementation and preliminary testing

4.1 Implementation

The mathematical formulation is implemented in Mosel, with two supplementary Python scripts to create data and plot the solution after a run. *autonomax.mos* first runs the Python script *subset_script.py*. The script is used for the subtour elimination constraint, to generate all possible subsets of nodes based on whether or not the core net should be a cycle and the number of cities in the core net. Whether or not the core net should be a cycle is relevant, because only subsets of size 2 through $\left\lfloor \frac{C^N}{2} \right\rfloor$, where C^N is the number of cities in the core net, need to be generated if the core net is cyclic. This significantly decreases the number of constraints in the implementation. The subsets are represented as a binary $n \times m$ matrix where n is the number of subsets and m is the overall number of cities. Row n will have a 1 at column m if city m is a part of subset n, otherwise 0. This is saved to *subsets.txt*, which is in turn used to initialize the 2-dimensional subset array that is used in the subtour elimination constraints (constraint (8)). The remaining data is initialized from *AutonomaxData.txt*, and the problem is solved. The solution, regardless of whether the problem is solved to optimality or not (i.e. max time is reached), will then be written to the file *out.txt* and printed to the console. Finally, *plot.py* uses the output from the output file to generate and display a network plot of the solution.

Note that the Xpress-package *python3* does not work for all computers and operating systems, and that the python scripts require you to install some dependencies. Please refer to *README.md* or *README.pdf* in order to fix this.

4.2 Preliminary testing

When using the parameter settings $Scenario = 1$, $Z = 1$ and $NC = 3$, the optimizer stops before the maximum solution time of 600 seconds is reached. The $MIPRELSTOP$ parameter is set to it's default value of 0.0001, which means that the optimizer can guarantee that the solution (objective value), which in this case is 15476.1224, is within 0.01% of the optimal solution. The number of columns (variables) before presolve is 5002, and the number of rows (constraints) is 3611. After presolve, the number of columns (variables) is 5002, and the number of rows (constraints) is 3529. That is, the number of variables remains the same but Xpress is able to reduce the number of constraints by 82. If this number is significantly large, it can be an indication that there are a lot of unnecessary constraints. However, this does not seem to be the case here, as it is as an implementation issue regarding the declaration of the constraints (i.e. constraints are declared for all combinations of two cities, but never initialized when the two cities are the same). Studying Figure 5, which is a graphical representation of the solution, the results seem to be as expected given our

interpretation of the problem. There are only 3 cities in the core net, the core net is cyclic and all of the sub nets have a tree structure and are connected to the core net through a single edge.

With the parameter settings $Scenario = 1$, $Z = 1$ and $NC = 4$, the problem is again solved to optimality before the maximum time is reached. The optimal objective value in this case is 15895.04888, and the original problem has 5002 columns (variables) and 4431 rows (constrains). The number of variables is the same as for $NC = 3$, which is to expected, whereas the original problem has 820 more constrains than $NC = 3$. This is also to be expected, because there are no subtour elimination constraints with 3 cities in the core net and 820 is the number of subsets of size 2. The number of columns and rows after presolve is 5002 and 4349, respectively. Again, Xpress was able to reduce the number of constraints by 82. The results seem to be as expected. There are only 4 cities in the core net, the core net is cyclic and all of the sub nets have a tree structure and are connected to the core net through a single edge (cf. Figure 6).

4.2.1 LP-relaxation and proposed improvements

An LP-relaxation of a mixed integer problem forms a best bound for the problem, and can be used as a indicator to measure the tightness of a model, and compare different model formulations.

Solving the LP-relaxation of the model with parameter settings $Scenario = 1$, $Z = 1$ and $NC = 3$ to optimality yields an objective value of 1720.896. The LP-relaxation with the same settings besides $NC = 4$ yields an optimal objective value of 2071.984. It is clear that the optimal solution of the LP-relaxation shows a significant difference from the optimal solution of the MIP problem. Since the LP-relaxation has no integer restrictions on the variables, the logic from the majority of the constraints which defines the feasible solutions collapses, as many of them are based on graph theory concerning number of nodes and edges in the network. As a consequence, the number of feasible solutions in the LP-relaxation is higher than that of the MIP-problem.

An immediate weakness of the model can be related to the control center variable. Since the placement of the control center has no impact on the problem other than that one of the core cities needs to be chosen, which adds 41 redundant variables and 83 extra constraints to the problem. Still, from a practical perspective, it allows us to define the direction of transmission (i.e. "flow") out from the control center, which could be useful in the practical problem. Excluding the control center variable from the problem formulation gives no improvement on the lower bound generated from the LP-relaxation.

Another weakness of the formulation is that constraint (2) is unnecessary. Even though it restricts the number of feasible solutions, since the problem is a minimization problem, it will never be an optimal solution where there is both grafen and regular cables between two cities. Removing this constraint reduces the number of constraints by $41 * 40 = 1640$. By experimentation one can see that even though the removal has no effect on the optimal LP-relaxation, the first integer feasible solution found by the solver is lowered, indicating that it has a positive impact on the heuristic search in the solver. Note that this was tested only for NC=3 and NC=4 with Z=1, but it is a reasonable assumption that this would be the case for other cases as well.

A third weakness, which is connected to the implementation rather than the mathematical formulation, is the choice of M in constraints (3) and (13). In our initial implementation, we chose M to be the maximum of the total demand in each scenario, which is 125 (Scenario 3). Knowing that choosing an appropriate M has an impact on the tightness of a model, an improvement could be to adjust the M according to which scenario is running. For example, scenario 1 has a total demand of 112,4. For this scenario, M could be set to 113. When implementing this adjustment, the optimal objective value of the LP-relaxation, with parameters $NC = 3$ and $Z = 1$, is increased by 4% creating a tighter bound for the problem.

5 Heuristics

An operator $N(x)$ should output a set of neighbour solutions with similar characteristics solution x. How the neighbourhood is defined will have an impact on the goodness of the heuristic. While studying results generated from our problem implementation, it was discovered a pattern that the core net tends to be located in areas with high density of cities with high bandwidth demand. This is reasonable as the cost function related to grafen cables is independent of the bandwidth demand, in contrast to regular cables where the cost

varies with the bandwidth requirement of the cable. From this observation we suggest two neighbourhood operators. Pseudo code for how each operator generates solution-neighbourhoods is provided in section 7.1.

Operator 1 can be considered as a variant of 1-swap neighbourhood, modified to enforce feasibility in the neighbouring solutions. It defines a neighbour as a solution where one of the core nodes are swapped with a sub node which is directly connected to the core net. In that way we exploit the pattern that the core net usually is located in an areas with high density of cities and high bandwidth demand, assuming the initial solution is located in such an area. By that reasoning, exploring different combinations of such swaps increases the probability of finding an improving solution. Figure 3 exemplifies a neighbour generated from operator 1.

Operator 2 is intended to construct neighbours with improved sub net structures. We do so by swapping nodes in a sub net tree according to an evaluation function which guarantees an improvement in a neighbouring solution. We define it as the DemandDistanceScore, which is derived from the cost function for regular cables. Since cost of a sub net is related to the bandwidth demand of the sub net nodes included, and how they are structured, it is reasonable to assume that swapping sub nodes such that the nodes with the highest DemandDistanceScore is moved closer to the core net, will generate an improved solution. The gain from swapping two sub edge nodes i and j will be $0.1^{1.5}(DemandDistanceScore(i) - DemandDistanceScore(j))$. Figure 4 demonstrates a possible transformation using operator 2. A disadvantage with this approach, is that the operator is restricted to only produce improving neighbours, which makes it possible for the operator to return no neighbours, trapping the search at a local optimum.

Intuitively is possible to argue that the two operators will complement each other in a combined heuristic as they operate on different parts of the network structure. The first neighbourhood operator will explore the solution space with respect to different core net structures, while the second focuses on improving existing sub net trees. In an iterative local search, the first operator would produce new sub net structures which could expand the neighbourhood space for the second operator, and consequently guide the search towards improved solutions in total. Whether operator 2 should be considered as a neighbour operator or a meta heuristic itself, is debatable as the two terms some times is intertwined, but we regard it as an operator for our specific heuristic when it is combined with operator 1.

6 Economic analysis

6.1 Cyclic core net with increased number of cities in core net

Figure 5-10 in the appendix shows the results of running the model with 3 to 8 core net cities for the main scenario solved to optimality.

Our first observation is that the optimal core net cities tend to be located in the same area in the demographical center of the country, which coincides with our previous observations in the heuristic section. From an economical point of view, this can be explained as a trade-off between low distance between the cities in the core net, which is more cost efficient for the core net, and shorter sub net trees, which impacts the sub net costs. When increasing the number of cities in the core net, naturally the core net cost increases, but the structures are created such that the core net covers a larger area, which reduces the sub net costs, as the sub net trees are reduced in size. This can be directly observed in the plots for 3 to 5 core net nodes, which are the ones we have solved for optimally. We expect the same pattern if the remaining models had been solved optimally as well.

6.2 Scenarios impact on infrastructure

In Figure 12-15 the optimal infrastructure for the scenario two and third for a cyclic core net with three and four core cities is depicted. The different scenarios represents different demand patterns for the cities. One can observe that for both scenario two and three the core net is shifted southwards when compared to scenario 1, creating longer sub nets in the northern part of the country. By looking at the demand difference for each scenario as shown in Figure 11, we see that the for both second and third scenario, the demand for bandwidth tends to increase in the southernmost cities, compared to scenario 1. Consequently, it is evident that it will be cost-reducing to locate the core network further south, as the cost of having longer sub nets in the north would be reduced. We also observe that the total cost in scenario 2 is lower than in scenario 1

even though the total demand is higher, confirming our assumption that locating the core net in high density areas with high demand is more economically beneficial. As the southern cities are located closer to each other, and in the .

6.3 Characteristics comparison of path versus cyclic core net

In Figure 16 and 17 the infrastructure for 3 and 4 core nodes with path structure is shown. We see that the core nets for $NC = 3$ and $NC = 4$ with path structure are located in approximately the same area as the corresponding ones with cyclic core nets(Figure 5 and 6), which is as expected due to the same trade-off reasoning as described earlier. A difference is that the path core net have a longer vertical (from north to south) reach than the cycle core net, causing a reduction in length of the longer sub net trees. This, combined with having one less of the expensive grafen cables is probably the reason why using a path structure in the core net gives a significant cost reduction, with a total cost of 15476.12 for a cycle net and 13451.23 for a path, both for 3 core nodes in scenario 1.

6.4 Cost of Östersund as control center

In Table 1 we show the cost difference from model runs when forcing Östersund as the control center.

As we see, placing the control center in Östersund induces extra costs of between (approximately) three to five thousand for the different configurations. The cost is due to Östersund being forced as a city in the core net, even though it is not in any of the optimal solutions. Considering that Östersund is placed at sparse area of cities, it is from an economical point of view not a suitable location to place the control center.

Configuration	**Optimal cost**	**Cost with control center in ÖsterSund**	**Difference**
Cyclic with three core net cities	15476	20515	-5039
Cyclic with four core net cities	15895	20400	-4505
Path with three core net cities	13461	16405	-2944

Table 1: Cost change of restricting Östersund to be the control center node in different configurations

7 Appendix

7.1 Heuristics

Pseudo code for operator 1

```
Neighbourhood = {}
for each corenode i connected to a sub edge:
    for sub edge (i,j):
        neighbour = OriginalSolution
        Swap i and j so that j becomes a core net node
        and i becomes a sub net node
        For each core edge a into node i:
            remove core edge (a,i)
            add core edge (a,j)
        For each core edge (i,b):
            remove core edge (i,b)
            add core edge (j,b)
        Remove sub_edge (i,j)
        Add sub edge (j,i)
        Set j as control center node if i is the control center
        Update demand variable
        b_ij = 0
        b_ji = sum_n b_in + B_i^W

        add neighbour to Neighbourhood set
```

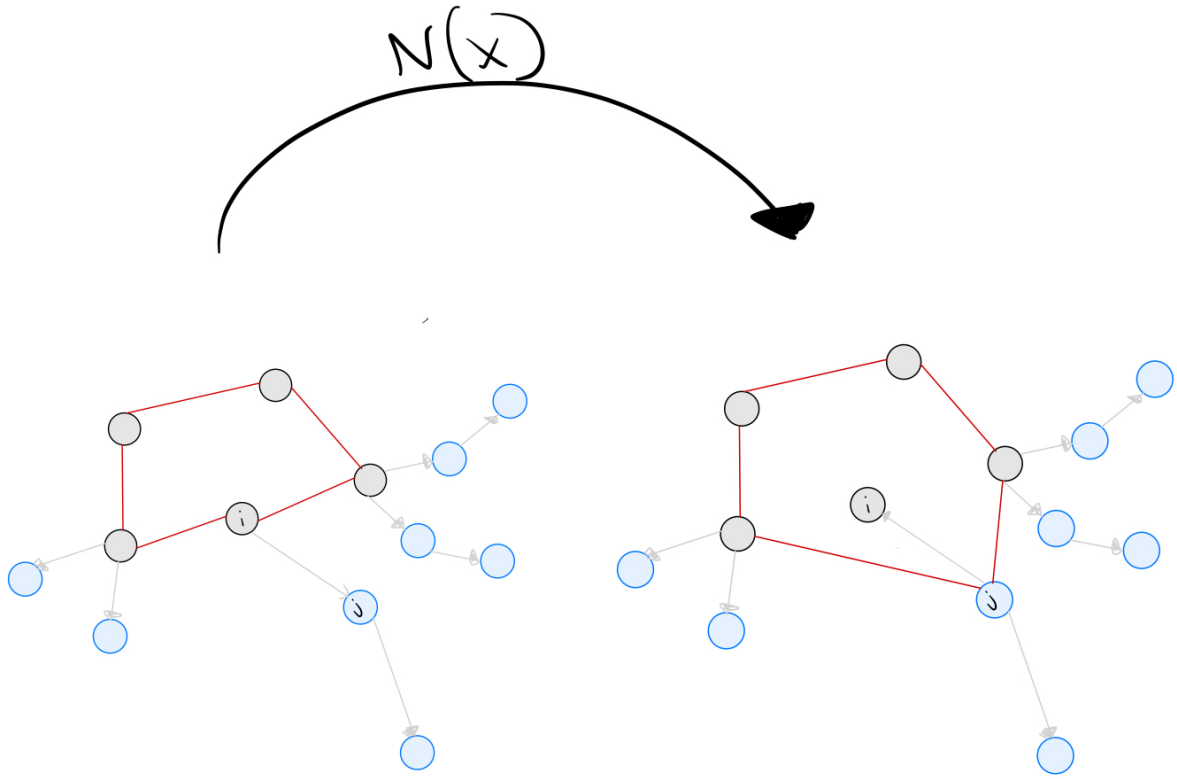

Figure 3: Illustration showing a possible neighbour using operator 1

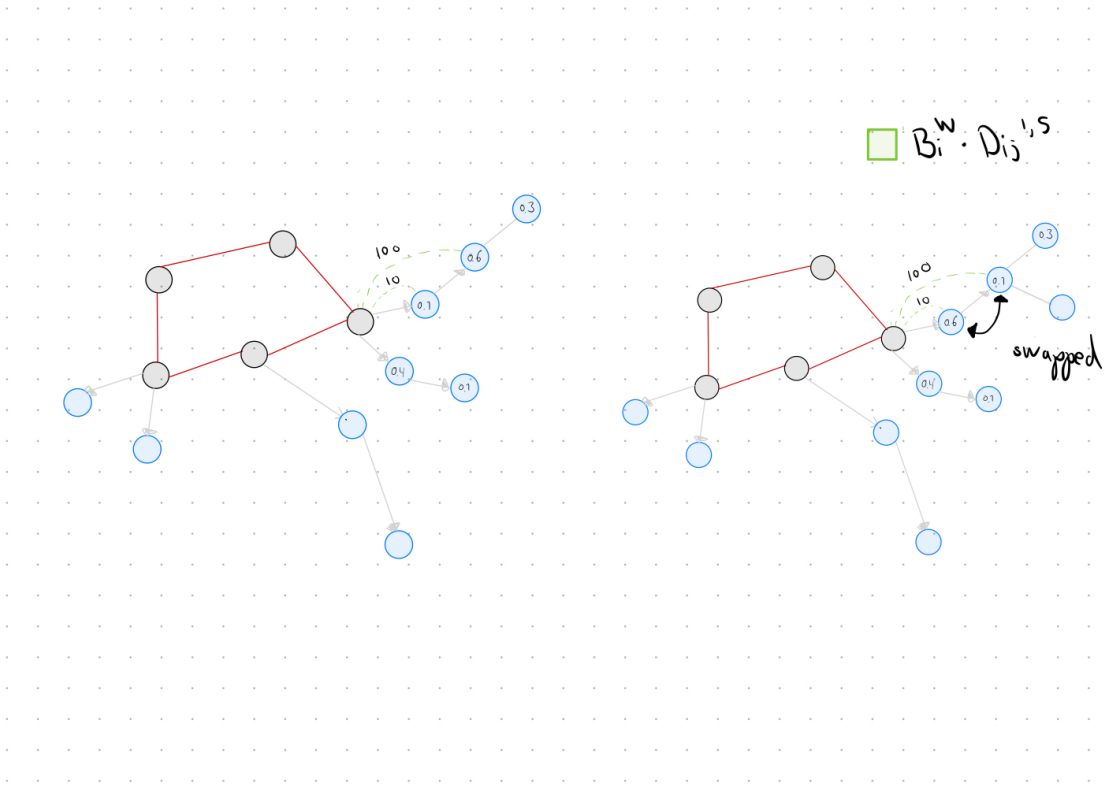

Figure 4: Illustration showing a possible neighbour using operator 2

Pseudo code for operator 2

```
Neighbourhood = {}
for each core node i :
    for each sub tree T connected to core node i where |T|>1:
        neighbour = OriginalSolution
        DemandDistanceScore = empty list
        for each sub node j in subtree T:
            score = Demand(j) * Distance(i,j)^1.5
        add (j,score) to DemandDistanceScore list
        Traverse T from root:
            if DemandDistanceScore(parent)<DemandDistanceScore(child):
                swap(children,parent)
        add neighbour to Neighbourhood
```

7.2 Cyclic core with core size from 3 to 8

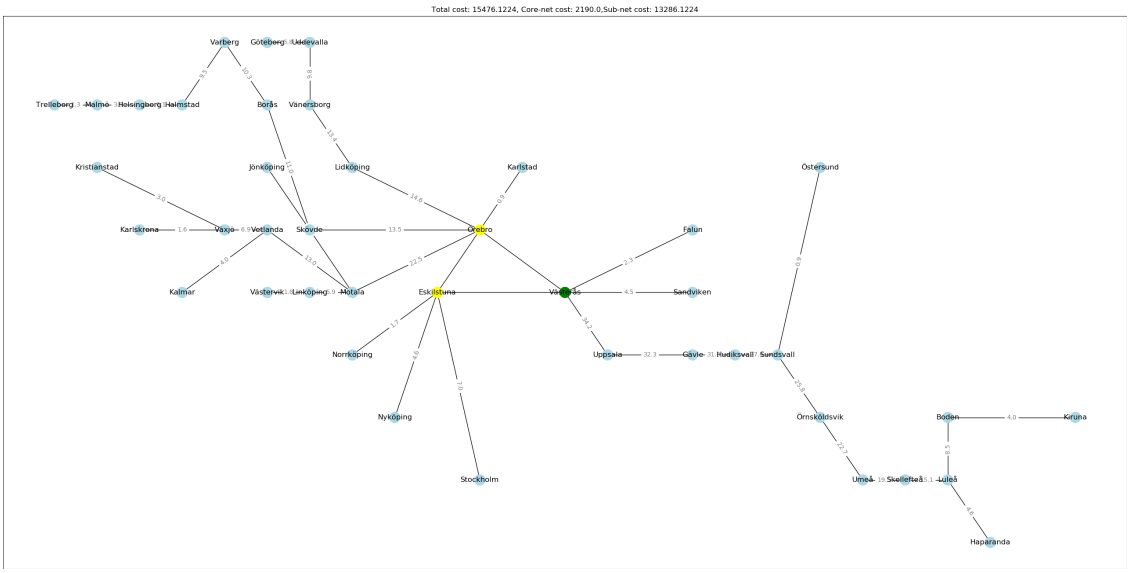

Figure 5: Cyclic core with size 3. Core net cost: 2190,00. Sub net cost: 13286.12. Total cost: 15476.12

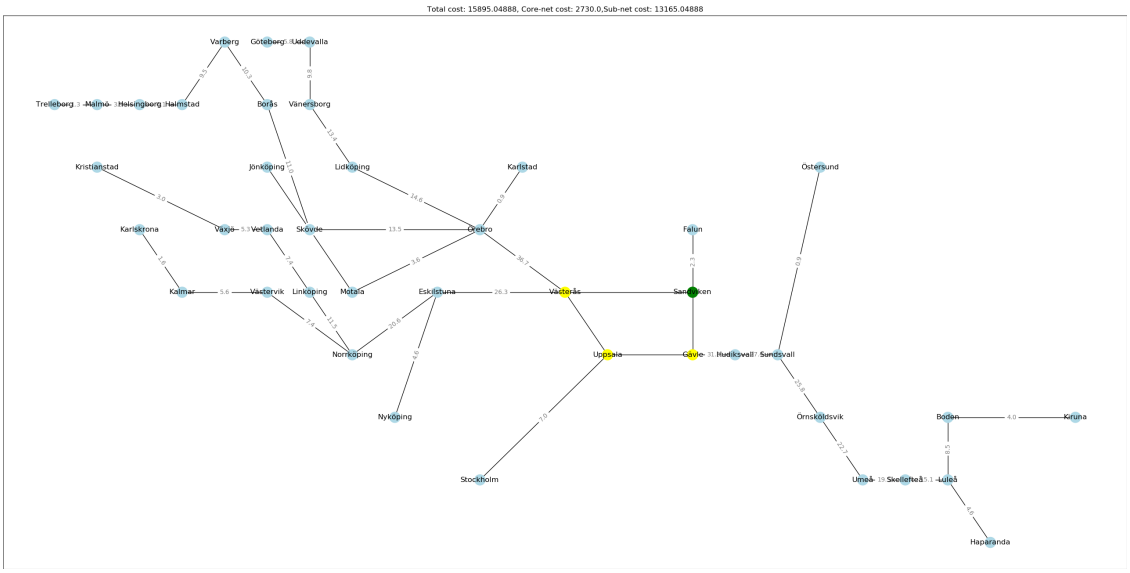

Figure 6: Cyclic core with size 4. Core net cost: 2730,00. Sub net cost: 13165,05. Total cost: 15895,05

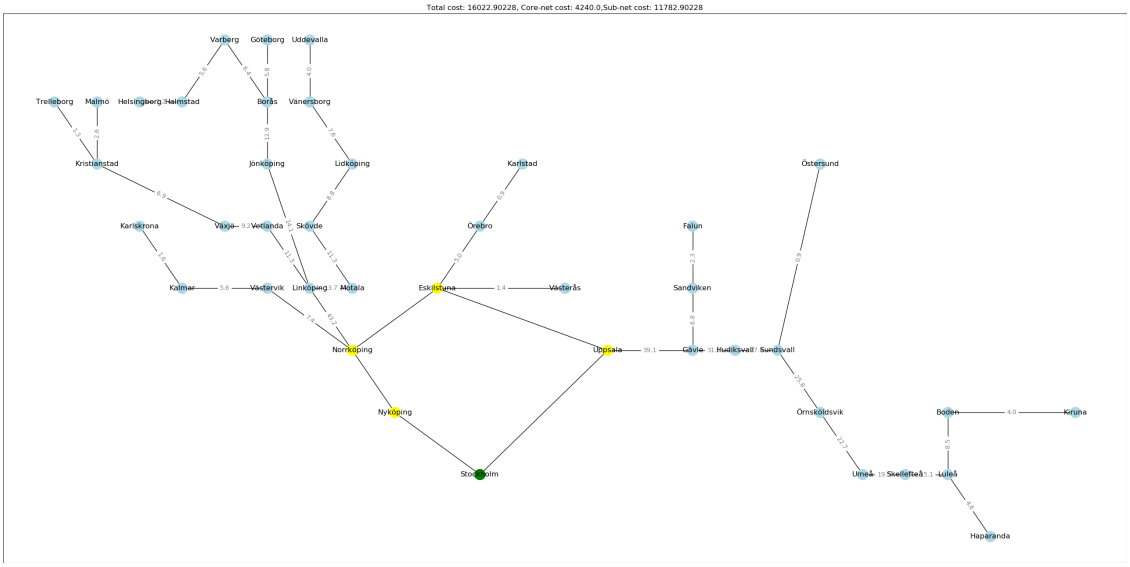

Figure 7: Cyclic core with size 5. Core net cost: 4240,00. Sub net cost: 11782,90. Total cost:16022,90

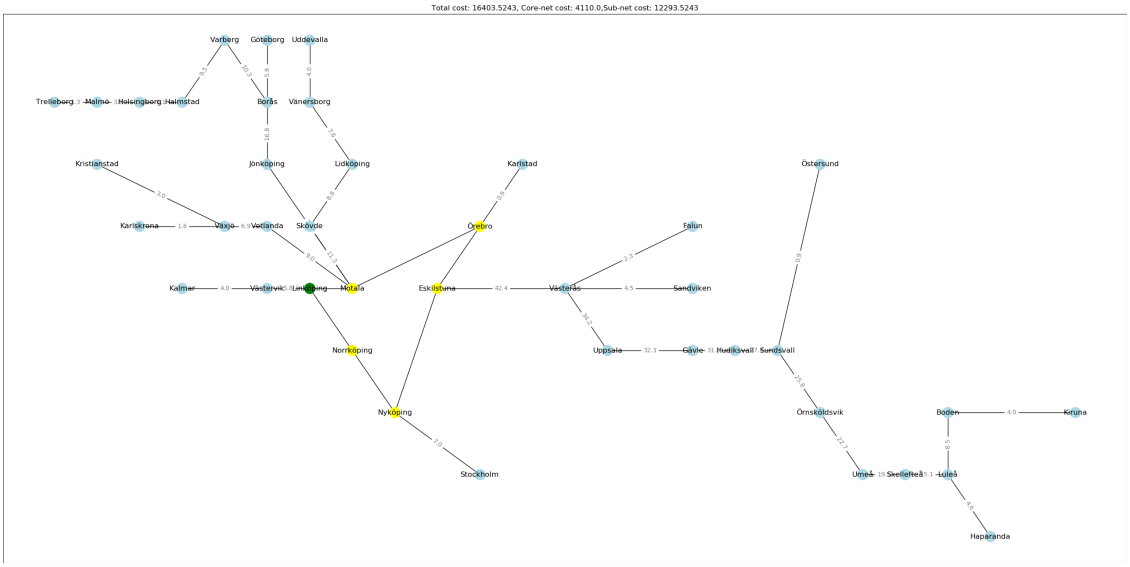

Figure 8: Cyclic core with size 6. Sub-optimal solution. Core net cost: 4110,00. Sub net cost: 12293,52. Total cost: 16403.52

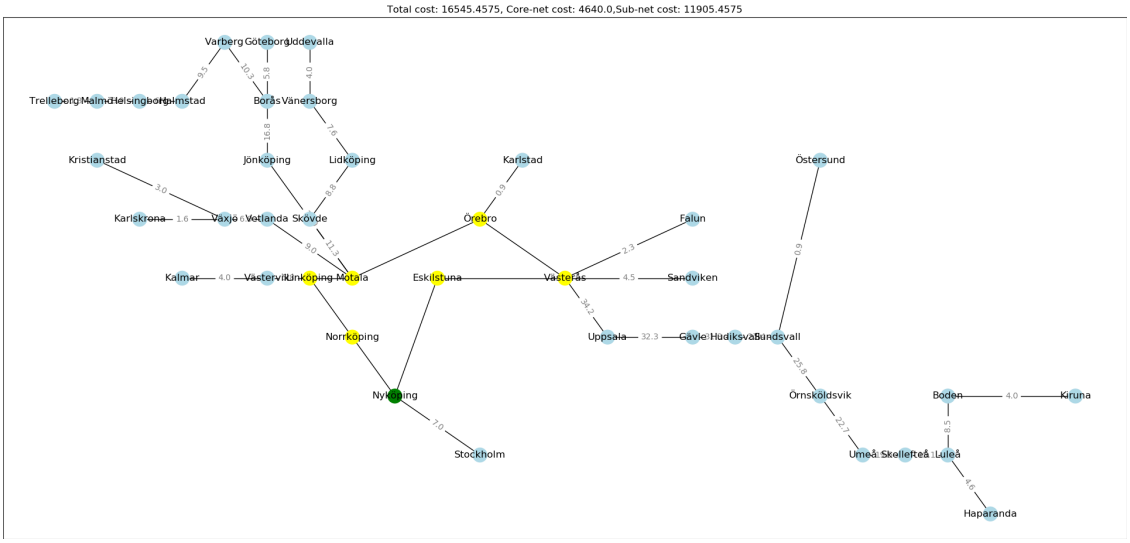

Figure 9: Cyclic core with size 7. Core net cost: 4640,00. Sub net cost: 11905,46. Total cost: 16545,46

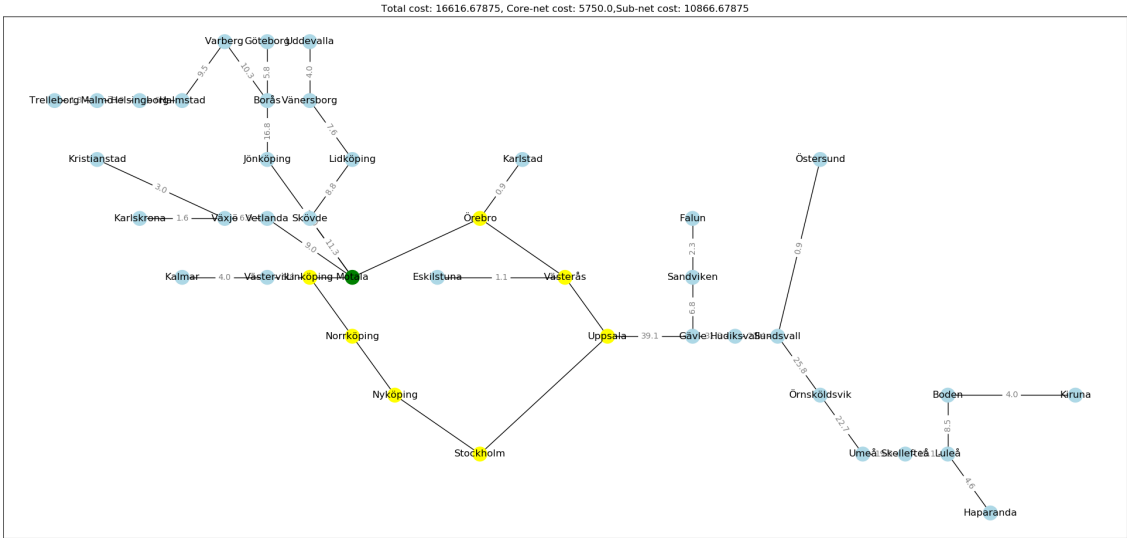

Figure 10: Cyclic core with size 8. Core net cost: 5750,00. Sub net cost: 10866,68. Total cost: 16616,68

7.3 Results for scenarios 2 and 3

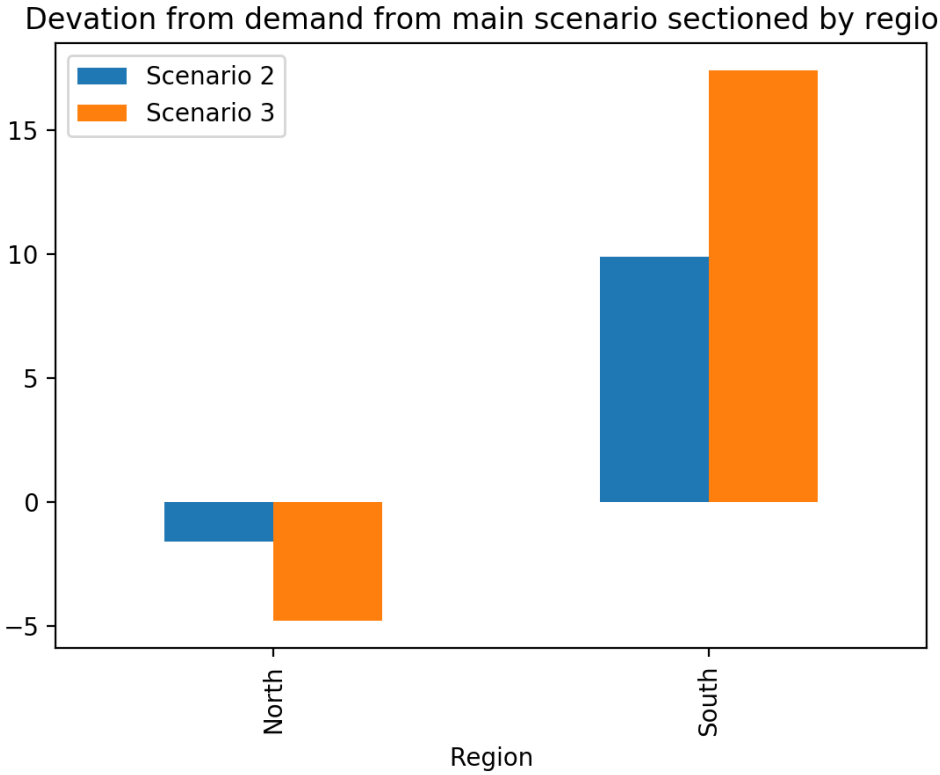

Figure 11: Demand difference from main scenario sectioned by most northern and southern cities.

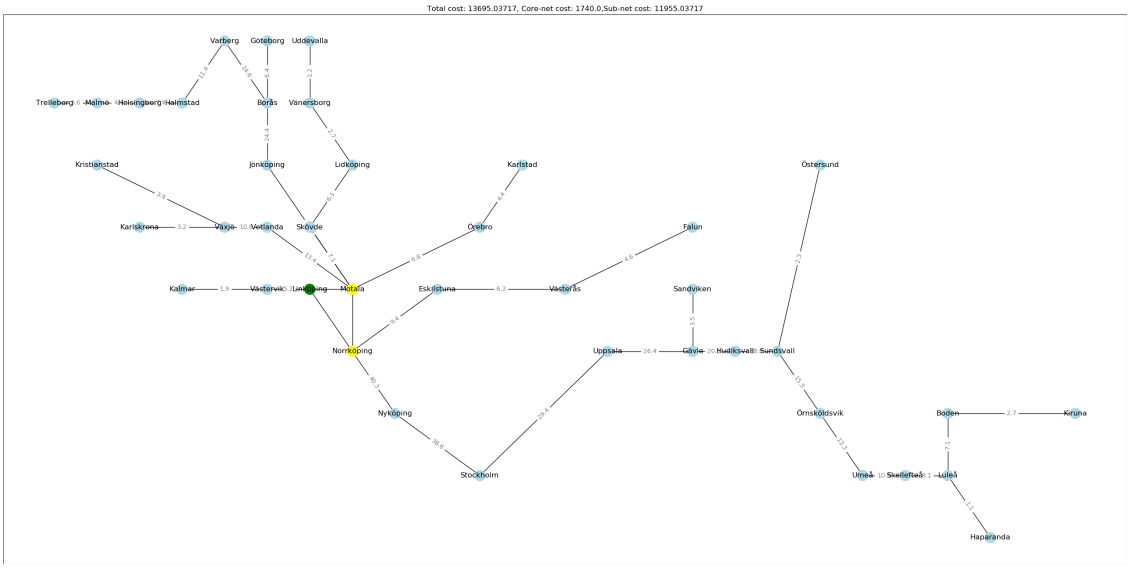

Figure 12: Scenario 2: Cyclic core with size 3. Core net cost:1740,00. Sub net cost: 11955,04. Total cost: 13695,04

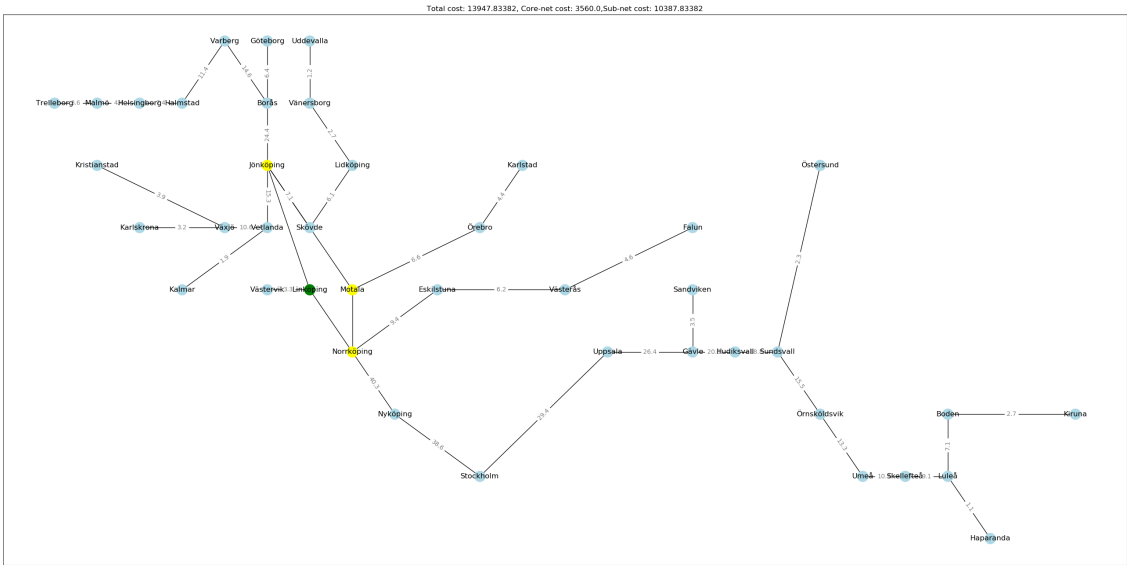

Figure 13: Scenario 2: Cyclic core with size 4. Core net cost: 3560,00. Sub net cost: 10387,83. Total cost: 13947,833.

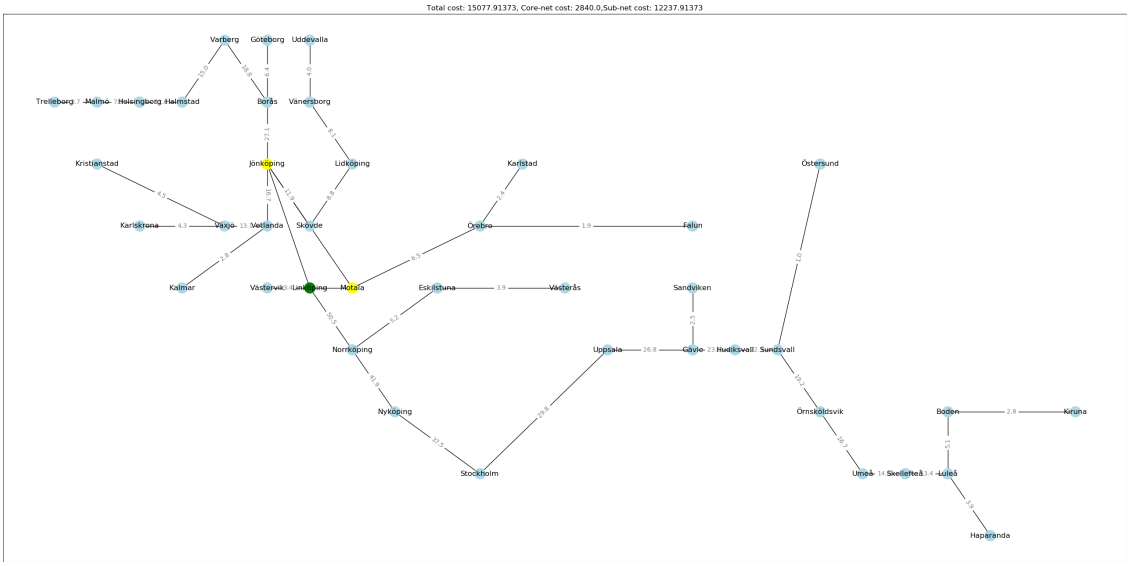

Figure 14: Scenario 3: Cyclic core with size 3. Core net cost: 2840,00. Sub net cost: 12237,91. Total cost: 15077,91.

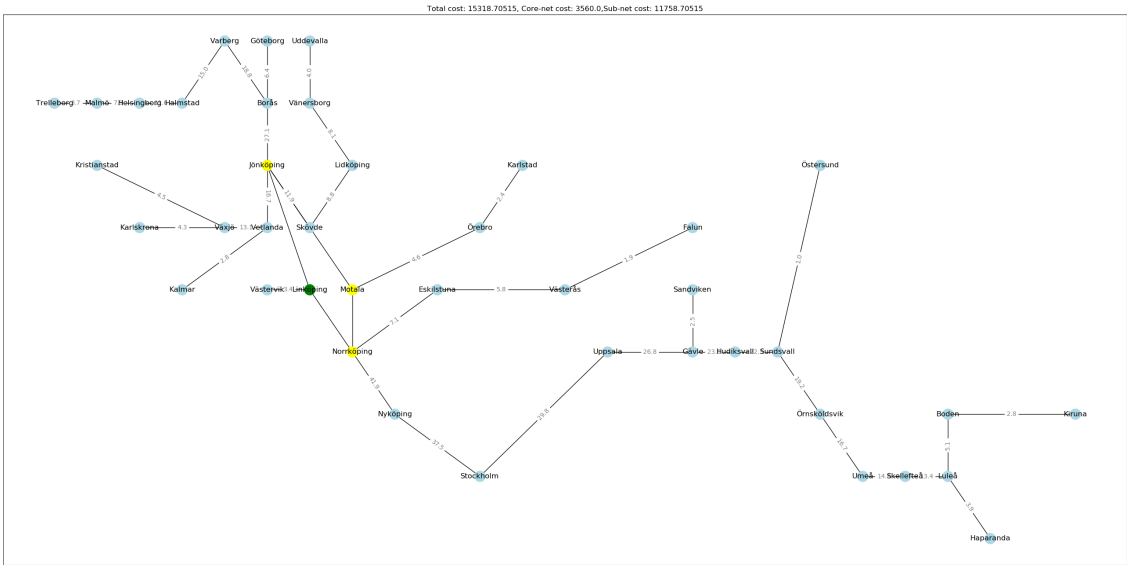

Figure 15: Scenario 3: Cyclic core with size 4. Core net cost: 3560,00. Sub net cost: 11758,71. Total cost: 15318,71

7.4 Path core with core size 3 and 4

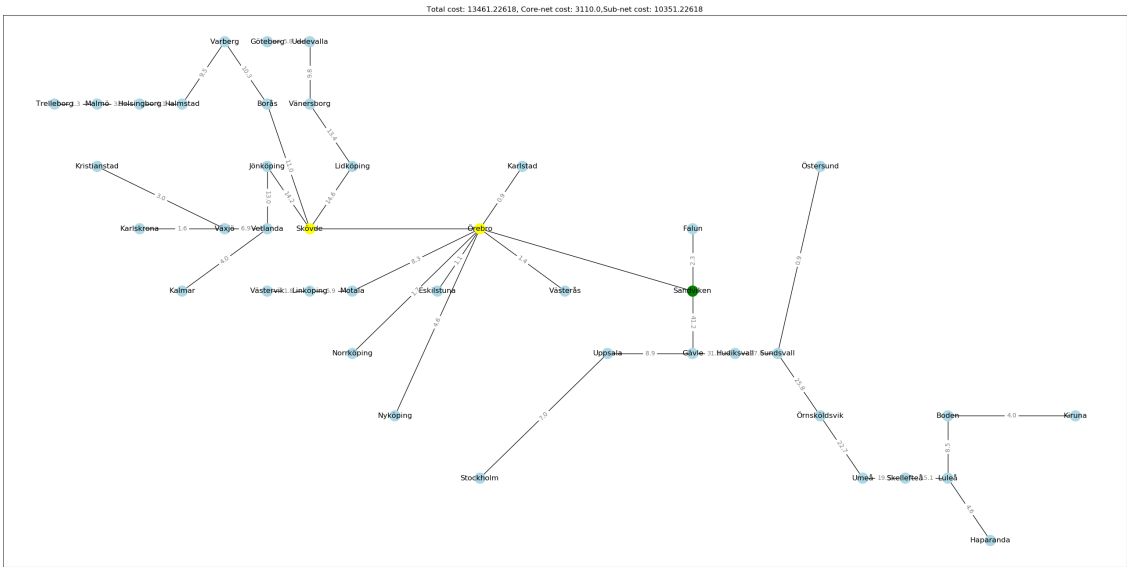

Figure 16: Path core with size 3. Core net cost: 3110,00. Sub net cost: 10351,23. Total cost: 13451.23

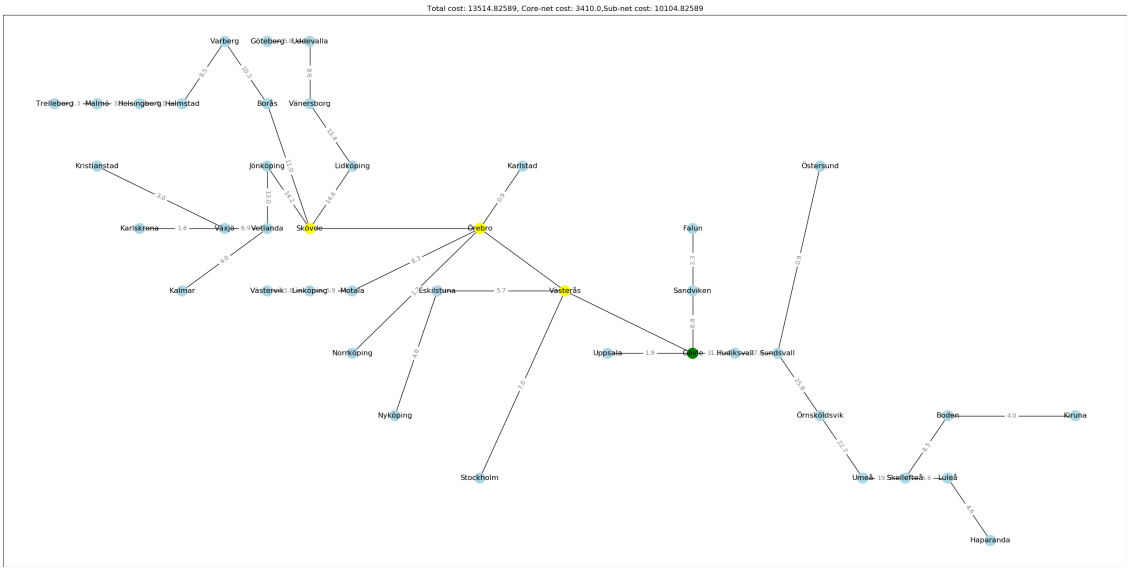

Figure 17: Path core with size 4. Sub-optimal. Core net cost: 3410. Sub net cost: 10104,83. Total cost: 13514,83

7.5 Cycle core with Östersund assigned as control center

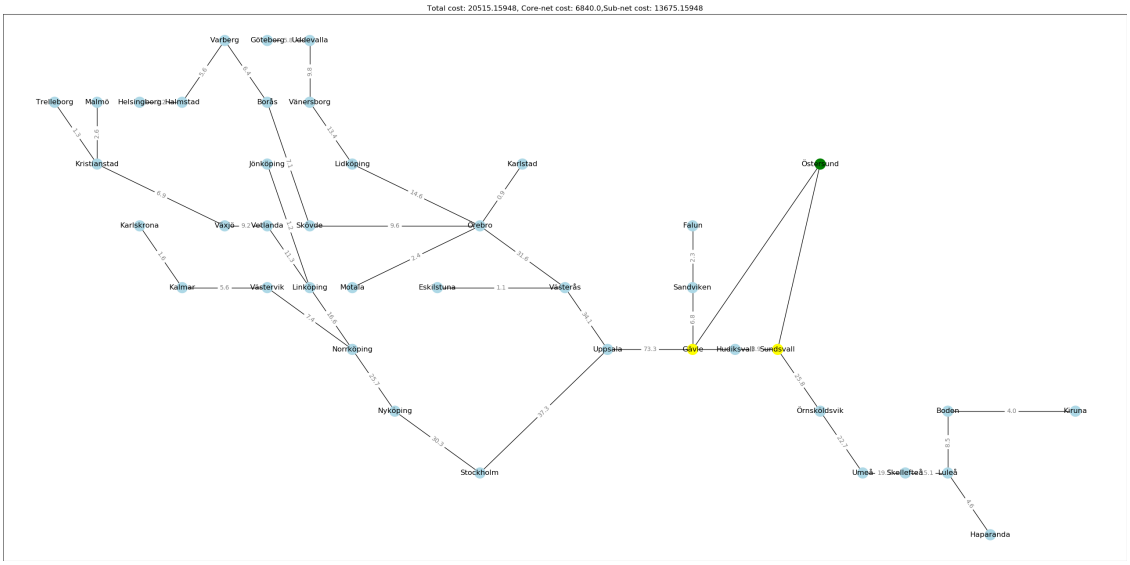

Figure 18: Cyclic core with size 3 with Östersund as control center. Core net cost: 3560,00. Sub net cost: 11758,71. Total cost: 15318,71