Comparison of heuristic methods for achieving minimum-cost capacitated networks with a new metaheuristic based on node valency

Christopher Yeates[1], Cornelia Schmidt-Hattenberger[1], Wolfgang Weinzierl[1], David Bruhn[1][2]

1: Geoenergy Department, GFZ Potsdam, Germany

2: Faculty of Civil Engineering and Geosciences, Delft University of Technology, Netherlands

Abstract

Designing low-cost networks is an essential step in planning linked infrastructure. For the case of capacitated trees, such as oil or gas pipeline networks, the cost is usually a function of both pipeline thickness (i.e. capacity) and pipeline length. Minimizing cost becomes particularly difficult as network topology itself dictates local flow material balances, rendering the optimization space non-linear. The combinatorial nature of potential trees requires the use of graph optimization heuristics to achieve good solutions in reasonable time. In this work we perform a comparison of known literature network optimization heuristics and metaheuristics, and propose novel algorithms, including a metaheuristic based on transferring edges of high valency nodes. Our metaheuristic achieves performance above similar algorithms studied, especially for larger graphs, usually producing a significantly higher proportion of optimal solutions, while remaining in line with time-complexity of algorithms found in the literature. Data points for graph node positions and capacities are first randomly generated, and secondly obtained from the German emissions trading CO_2 source registry. Driven by the increasing necessity to find applications and storage for industry CO_2 emissions, finding minimum-cost networks increases the business case for large-scale CO_2 transportation pipeline infrastructure.

Keywords: Network Design, Heuristics, Cost Minimization, Pipeline networks, CO_2 transport

Declarations

Funding: The Helmholtz Climate Initiative (HI-CAM) is funded by the Helmholtz Association's Initiative and Networking Fund. The authors are responsible for the content of this publication.

Conflicts of interest/Competing interests: There are no conflicts to declare.

Availability of data and material: The data used to test the codes here, as well as the results, are provided at:
https://gitext.gfz-potsdam.de/yeates/network-design/-/tree/master/paper_materials_valency_shuffle/

Code availability: The code for all the algorithms used here, as well all data analysis and plotting codes are available at:
https://gitext.gfz-potsdam.de/yeates/network-design/

1. Introduction

Network design problems arise in a multitude of scenarios. For fluid transport problems, such as oil or gas transport, the aim is to design networks linking all fixed location points of the network via pipelines in the way that meets system requirements while minimizing a defined cost-function. System requirements may take different forms such as maximal pressure limitations for safe use, minimal pressure requirements for ease of transport (e.g. supercritical CO_2), prohibited transport through certain areas (e.g. natural parks), or preferential transport through other areas (e.g. existing pipeline corridors). While some networks are created iteratively as new nodes are added (i.e. historical domestic gas distribution networks), potentially resulting in suboptimal network designs, networks can be *designed from scratch* as new incentives or legislative restrictions arise. Creating novel networks that re-purpose existing liquid commodity sources, such as CO_2 storage or utilisation from industry CO_2 emissions (van den Broek et al. 2009; Alhajaj and Shah 2020), slurry collection from dairy plants (Bietresato et al. 2013), or otherwise networks that distribute a new commodity to a series of known recipients such as Hydrogen (André et al. 2013) or biogas networks (Heijnen et al. 2020), all undergo a step of network design. This involves linking a series of fixed sources/sinks points via pipelines with associated capacity and cost.

Finding optimal networks connecting sources to sinks is an example of finding Minimum-cost Capacitated Spanning Trees (MCST), i.e. the minimum cost tree linking all elements of the network and respecting flow requirements. In this work, the cost is given by a combined function of the *pipeline length and a positive capacity value.* Furthermore, as the relative pipeline cost is expected to flatten with increasing capacity, i.e. considering the economy of scale, a *concave cost function* of the capacity is used.

While finding Minimum cost spanning trees without capacity is known to be solvable in polynomial time with algorithms such as Kruskal's (1956) algorithm, the addition of the capacity constraint adds computational complexity. Such problems are difficult to optimize as the attribution of capacity (and therefore cost) to each pipeline will depend on the exact network topology chosen, creating a highly non-linear optimization space, where switching a single edge may alter all the attributed capacities of the network, significantly altering the cost. As a result, local minima are easy to find and discovering the optimal MSCT requires iterating over all trees, attributing capacity, then calculating associated cost. The combinatorial nature of trees renders this problem NP-hard.

The further addition of intermediate relay nodes, Steiner points, to the network is expected to reduce overall costs, creating Minimum cost Gilbert trees (MCGT)(Gilbert 1967). As such, each MCST serves as starting point for the introduction of Steiner points. Steiner points do not require any capacity addition to the pipelines and simply provide a way of reducing the overall pipeline lengths. It has been shown (Heijnen et al. 2020) that the cost of the Minimum cost Gilbert tree (MCGT) is highly dependent on the MCST from which it originates and is expected to reach lower final costs for lower cost input MCSTs .

In this regard, finding MCSTs which are as close as possible to the optimal MCST also becomes a valuable undertaking the search for the more general MCGT. Due to the large number of possible trees (Cayley's (1857) formula shows that n^{n-2} spanning trees exist for n network nodes), iteration over all possible spanning trees in hope of finding the optimal MCST rapidly becomes impractical. Instead, graph optimization schemes are used to achieve lower cost trees rapidly, to the detriment of not insuring true optimal soluitions. Multiple approaches exist, including multi-Mixed Integer Linear Programming, that make use of traditional solver tools from mathematical programming and optimization (van den Broek et al. 2009; Brimberg et al. 2003; Sun & Chen 2016), agent-based methods such as Ant Swarm Optimization (Maier et al. 2003), Genetic Algorithms (Weihs et al. 2011),

or graph-based heuristic algorithms (Kazmierczak et al. 2009; Bietresato et al. 2013; André et al. 2013; Heijnen et al. 2020). A formal mathematical description of the problem can be found in Heijnen et al. (2020) or Xue et al. (1999). Heuristic algorithms offer the advantages of providing quick, good results while allowing decision-makers and other stakeholders to understand them easily, due to their conceptual simplicity. Finally, they are easily modifiable to integrate extra system requirements due to transparency and code shortness.

This work will therefore focus on establishing a baseline of MCST heuristic algorithms in combination with our proposed algorithms with associated performance statistics. Comparisons to optimal MCST solutions are given where possible (i.e. when not overly computationally demanding).

2. Materials and methods

2.1. General procedure

In a first step we give a non-exhaustive review of some commonly used applicable heuristics and metaheuristics and introduce some of our own. Heijnen et al. (2020) do a more comprehensive comparison of applicable algorithms, but here we only pick the best performing, with some additions from our own literature review. To do so, we make use of discrete optimization terms and concepts to describe some of the literature algorithms. At the same time, we describe the concepts and algorithms in an accessible way that can be understood by scientists and stakeholders in network design alike.

In a second step, we apply the algorithms directly to a series of examples with differing numbers of graph nodes, comparing the minimum-cost network found for each algorithm with an optimal solution, when possible. After providing some performance statistics based on a random generation of data points, some one-to-one algorithm comparison results are given using existing clusters of CO_2 industry emission sources in Germany provided by the European Emissions trading scheme data (German Environment Federal Office 2018).

The general procedure we use to obtain a minimum-cost capacitated spanning tree is given in **Fig. 1**. This procedure was also used by Yeates et al. (2020) and Heijnen et al. (2020). A series of sources are initially connected by the unique minimum-spanning tree to a chosen sink, capacity is attributed to the edges, and cost is calculated as a starting point before entering the network optimization algorithms that we explore in this paper. The flow requirement of the single sink is set by the opposite of the sum of the flow requirements of the sources. The location of the sink is chosen either randomly, along with the positions and flow requirements of the sources (Section 3.1), or as a set of multiple positions in a regular grid surrounding the sources (Section 3.2). The capacity attribution is unique for a given network topology and matching total sink-source capacity. This is calculated using the capacity allocation procedure proposed by Heijnen et al. (2020).

The cost function for a pipeline used throughout this work is a simple continuous concave function of flow capacity C as determined by the capacity attribution algorithm, and is proportional to the pipeline length L given by: $Cost(p_i) = LC^{0.6}$. The network cost is given by the sum of all pipeline costs. The exponent used here 0.6, is typical of CO_2 networks (Kazmierczak et al. 2009) and integrates notions of economy of scale as it is lower than one. The methods in this paper are applicable to any exponent between 0 and 1. The cheapest networks will represent the best *trade-off between overall pipeline length reduction and joining of pipelines into higher capacity*, comparatively cheaper pipes.

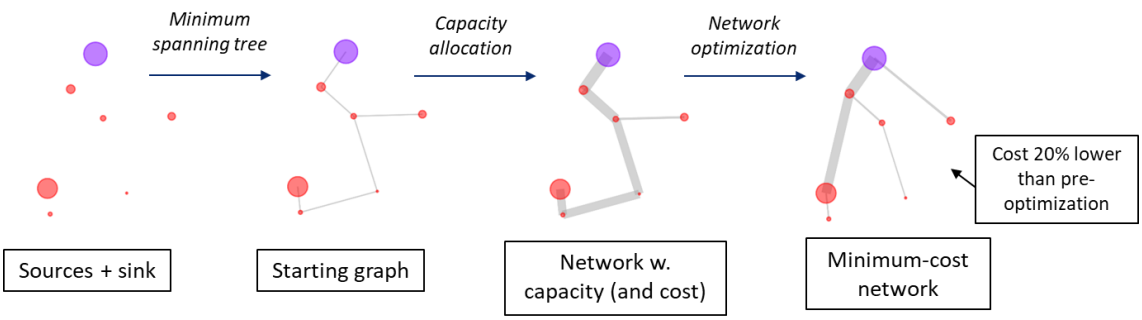

Fig. 1 *Procedure for finding a minimum-cost network. Sources are shown in red, while the sink is shown in purple. Inversing requirements and thus sources and sinks yield the equivalent problem*

2.2. Algorithms

In this section we provide a series of historical and recent examples of heuristics and metaheuristics to compare against our algorithm proposals. As seen in **Fig. 1**, the network optimization algorithms (second last to last tiles in **Fig. 1**) take a tree with associated cost as input and attempt to find lower-cost capacitated trees through modifications of the tree topology, adaptation of the pipeline capacity requirements where needed, and recalculation of cost. Topology modifications typically involve breaking the tree at an existing edge and recombining the two disconnected parts of the initial tree via a previously unconsidered edge. Such a transformation (i.e. breaking and recombining once) is known as *1-neighbourhood transformation*. These transformations can be described by a formal metric, a distance between solutions, most simply given by the *symmetric difference between the sets of edges*. As such, a transformation of distance 1 occurs when an edge is removed, and the distinct parts of the tree are combined in a new manner. A transformation of distance 2 occurs when two such transformations are done simultaneously, providing a solution within the 2-neighbourhood of the initial solution.

Local search heuristics involve only 1-transformations in a *single algorithm step.* As soon as a lower-cost solution is found, a first-descent local heuristic would then use the lower-cost solution as the new incumbent solution and repeat the process until no further single step improvement is made, i.e. a local minimum is found. A highest-descent local heuristic scans the entirety of its available 1-neighbourhood, then choosing the cheapest lower-cost solution and restarting the process until no further single step improvement is made.

While a heuristic provides a way of making a transformation between two solutions, a metaheuristic considers more general strategies for reaching optimal solutions within the broader optimization space, by attempting carefully chosen jumps into non-immediate neighbourhoods by either invoking memory, allowing exploratory moves towards higher-cost solutions or doing searches at different starting points in the optimization space and converging towards a single solution.

We now provide three examples of local heuristics that serve as basis for exploring metaheuristics later, also serving as baselines for comparison to our contributions. Python source code is given for each algorithm in the provided Github repository.

2.2.1. Heuristics

a. Delta Change

The Delta Change (DC) algorithm was first introduced by Rothfarb et al. (1970) focusing on the optimal design of offshore gas pipelines. The procedure consists of taking an initial spanning tree solution, for example the MST, creating new candidate solutions by choosing two disconnected nodes, adding an edge between them to create a cycle in the structure, then breaking the cycle

elsewhere by removing one of the pre-existing edges, before recalculating required pipeline capacities and cost. As soon as a better solution is found, including within the iterations over a given cycle, the algorithm is restarted, categorizing the procedure as a first-descent heuristic. A similar algorithm is described by Andre et al. (2003) although the authors include a randomization of the list of nodes to initiate a cycle from, and limit the number of explored node pairs by only selecting the closest ones to the initial node choice.

b. Local Search

The Delta Change cycle-based heuristic is also used in an algorithm described by Brimberg et al. (2003) referred to as "Local Search" (LS). However, in Local Search, the steepest-descent form is used, in which all the possible cycles of the 1-neighbourhood are explored before the minimum-cost solution is chosen, if an improvement is found.

c. Edge Turn

Recently, another 1-neighbourhood search heuristic was proposed by Heijnen et al. (2020) that we label the Edge Turn (ET) algorithm. The authors describe *edge turns*, which involve choosing an edge to remove, and reconnecting the disconnected two parts of the tree via a new edge connecting to one of the initial two nodes. As the breaking and recombining is done in the same step, the elementary edge-turn process can be compared to the one used in Delta Change, as each edge turn can be retrieved through an associated corresponding cycle. However, here the number of allowed transformations is reduced as the recombination of the two parts of the tree must necessarily be done at one of the nodes of the removed edge. Heijnen et al. scan over all possible edge turns before choosing the lowest-cost solution, in a steepest-descent fashion.

2.2.2. Extensions into 2-neighbourhood - Nested heuristics.

Extensions of the above heuristics can be conceived to permit jumps into the 2-neighbourhood of a graph solution in a single algorithmic step. This can allow discovery of lower cost solutions, inaccessible via a 1-neighbourhood local search due to existence of local minima. Put simply, such algorithms enable discovery of better solutions that involve breaking and recombining the trees twice simultaneously, rather than twice in succession, in which case the first iteration would not lead to a lower-cost solution and the second iteration would hence not be explored. These algorithms can be described as nested versions of the 1-neighbourhood heuristics. The computational complexity of such algorithms increases vastly as another local heuristic (i.e. into the 2-neighbourhood) is performed for each tentative initial local heuristic (i.e. into the 1-neighbourhood). We shall include 2-neighbourhood extensions of the Delta Change and Edge Turn algorithms named Nested Delta Change (NDC) and Nested Edge Turn (NET). A nested version of the Local Search algorithm was attempted but rapidly disregarded as the long computational times were considered impractical. In the case of the Nested Delta Change algorithm, the *first-descent* characteristic is maintained, such that as soon as a better solution is found, either in the initial 1-neighbourhood move or any of the associated 2-neighbourhood moves, the algorithm is stopped and restarted at this incumbent solution. For the Nested Edge Turn algorithm, the *steepest-descent* characteristic is maintained over the full 2-neighbourhood, such that all 2-neighbourhood solutions are tried, and only the best new solution (if any) is chosen as the incumbent.

2.2.3. Metaheuristics

Metaheuristics are more general search strategies that orient the lower-level heuristics in the context of the larger optimization space, mainly by providing mechanisms to overcome local minima. We give 2 literature examples of metaheuristics, Variable Neighbourhood Search and Tabu Search, and finally provide our contribution algorithm, the High Valency Shuffle.

a. Tabu Search

Tabu search is a general method proposed by Glover et al. (1989) and adapted to an oil pipeline design problem by Brimberg et al. (2003). It is based on exploring solutions around a local minimum that may not immediately be lower-cost, but add the least amount of cost to the local minimum. The lowest cost solution is still stored and ultimately chosen if no better solution is found, but *exploration to other zones is permitted*. In the case of a local minimum, the 1-transformation that increases the cost the least is chosen as the solution from which the next local search shall be initiated. This transformation (or exchange of graph edges) is then added to the Tabu list, and *the reverse transformation is prohibited* by the algorithm. The process is then repeated until a user-defined stopping criterion is reached, and the overall minimum cost solution is returned. The stopping criterion used here is chosen as 10 times the duration of the previously defined Local Search, and the Tabu list has a maximal length of 7, identical choices to Brimberg et al. (2003).

b. VNS

Variable neighbourhood search was initially proposed by Mladenović & Hansen (1997) and was used by Brimberg et al. (2003). Starting from an initial local minimum solution, it makes tentative jumps into solutions in further away neighbourhoods in a single step, reaching exploratory solutions that can have increased cost versus the current best solution, before attempting a local search in those new locations to find lower-cost solution. In the simple form proposed by Brimberg el al. implemented in this work*, random jumps into new neighbourhoods* are performed, successively reaching points at larger and larger radii (or distances) from the local minimum until a better solution is found. Note that radius is defined by the specific metric used, here the metric described in the introduction. If no better solutions are found up to a chosen neighbourhood radius (here up to 5 consecutive 1-neighbourhood transformations, as per Brimberg's implementation), the algorithm restarts at lower distance jumps again. The algorithm stops when a user-defined stopping criterion is reached. The stopping criterion used here is chosen as 10 times the duration of a Local Search, as used by the authors.

c. High Valency Shuffle

The High Valency Shuffle (VS) proposed here is based on the observation that many local minima solutions share the characteristic of having at least one *high valency node* (>2 edges) that is distinct to a high valency node found in the optimal solution. For a given high valency node in the optimal solution, a node situated in proximity of it may play a similar role in distributing flow but for a higher cost. The latter solution represents a local minimum, and metaheuristics and heuristics alike can often be trapped in it. Indeed, none of the local transformations can transfer all the edges in a single step to another node, and incremental transfers over multiple steps do not necessarily provide lower-cost solutions. In some cases, only *a full single-step switch of high valency nodes* can reach a lower cost solution. While this transformation may be accessible via Variable Neighbourhood Search or Tabu search as they explore the wider optimization space outside of the local minima, they are sometimes too general to create such a node switch in a reasonable number of steps. The High Valency Shuffle metaheuristic is used in combination with a lower-level local heuristic such a Local Search, Delta Change or Edge Turn. The High Valency Shuffle is described in Table 1.

Table 1 *High Valency Shuffle Metaheuristic*

Step 0:	Start from an initial good solution obtained with a local heuristic such as Local Search, Delta Change or Edge Turn. This solution is the starting incumbent solution
Step 1:	Initiate empty solution list
Step 2:	Identify all the nodes n_{HV} with high valency (3 and above edges)
Step 3a:	***For*** each node n_{HVi} of these n_{HV}: - Identify closest (in Euclidian distance) nodes n_C (e.g. 4 closest ones), connected or not to n_{HVi}
Step 3b:	***For*** each node n_{Ci} of these n_C: - Transfer all the edges from n_{HVi} to n_{Ci} - Connect n_{HVi} to n_{Ci} if not already done
Step 3c:	***If*** a cycle is detected in the graph: ***For*** each of the edges in the cycle: - Tentatively remove edge from cycle - Run a lower-level local heuristic on resulting tree - Add the obtained solution to the solution list ***Else***: - Run a lower-level local heuristic on the tree - Add the obtained solution to the solution list
Step 4:	Find minimal-cost solution from solution list
Step 5:	***If*** this solution is better than current incumbent solution: - Set this new solution as incumbent solution - Restart algorithm from **Step 1** ***Else:*** - End algorithm and return incumbent solution

In this study, we use three lower-level local heuristics (seen in Step 3c) in combination with the High Valency Shuffle: Edge Turn (VSET), Delta Change (VSDC), and Local Search (VSLS), all described previously. An example of a transformation obtained in a single step with a High Valency Shuffle is shown in **Fig. 6a**.

3. Results

3.1. Random graphs

To evaluate the proposed heuristics and metaheuristics and compare with literature options, we initially perform tests on generated data, with randomly placed source and sink locations as well as source capacities to gauge the algorithm's effectiveness on a variety of input data. We use the aforementioned algorithms on an increasing number of sources and compare calculation times to the fraction of optimal networks found. For the smaller network comparisons, as displayed in **Fig. 2**, optimal networks were discovered through *a brute force method*, obtained by comparing all the possible trees, obtained from Prüfer sequences (1918). For larger network comparisons, shown in **Fig. 3**, due to impractical computational times for the exhaustive method, *the best-found tree for all the algorithms was used for the optimal comparison*. In each random graph the sources and sink were placed randomly on a square and source capacities C were attributed randomly according to the following law: $C = X^3$ where X is a random uniformly distributed variable between 0-100. The sink capacity was then set as the opposite of the sum of the source capacities to ensure material balance. The minimum spanning tree was used as a starting point for each algorithm, as shown in **Fig. 1**.

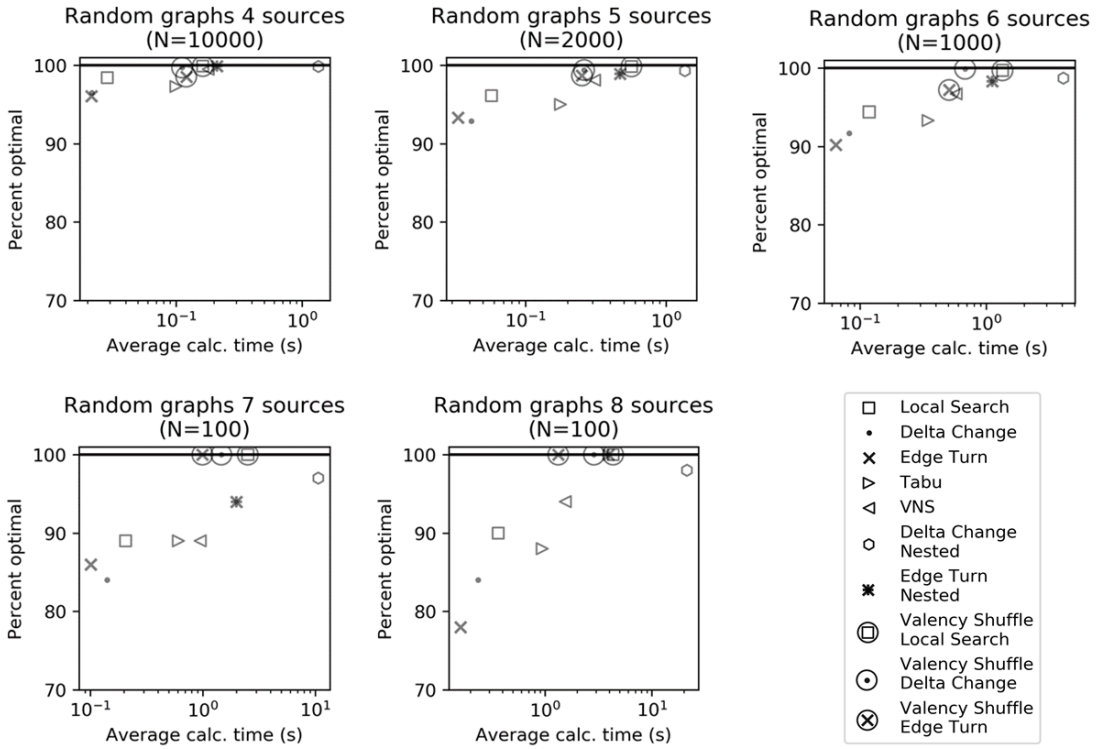

Fig. 2 *Algorithm performance comparison for small networks with a direct comparison to guaranteed optimal network*

We make a few observations regarding the performance of the various algorithms on the small networks shown in **Fig. 2**. There exists a general trend in which increased calculation time leads to increased rates of optimal solutions. All the new algorithms proposed here (Nested Edge Turn and Nested Delta Change, and all High Valency Shuffle Algorithms) lead to better optimality at the expense of longer calculation times. The High Valency Shuffle Algorithms obtain a 100% optimality rate in most of the cases while being faster than the Nested Algorithms. As the number of sources increases, calculation time for all algorithms increases, and the optimality of all algorithms decreases except for the High Valency Shuffle Algorithms and the Nested Algorithms.

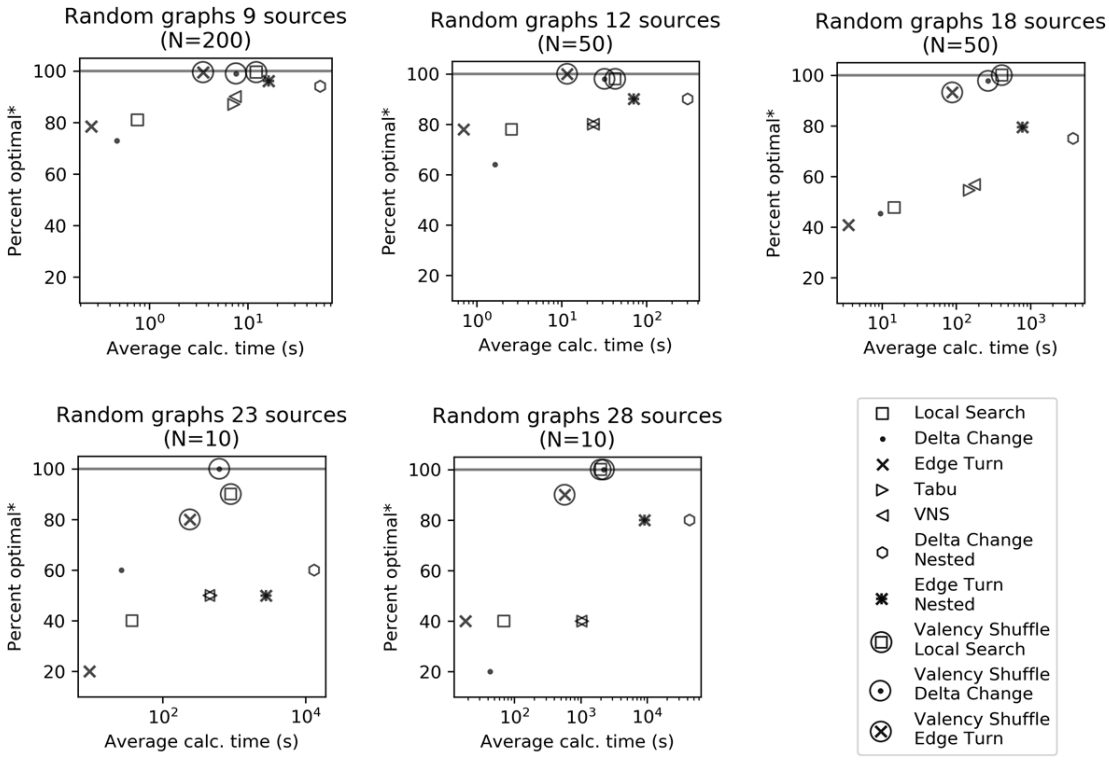

Fig. 3 *Algorithm performance comparison for larger networks. The asterisk refers to the fact that the optimal network comparisons are not guaranteed to be globally minimal*

As the number of sources increase, we continue to observe a loss of optimality for all algorithms, except the High Valency Shuffle Algorithms, which have optimality rates upwards of 90% in most cases. The Nested Algorithms, despite their large calculation times, fail to capture the best network solutions in the larger networks. These will therefore be excluded in some of the case study examples as they are impractical.

3.2. Case study: grid search procedure over CO_2 sources

As a case study of the algorithms, we use clusters of German industry CO_2 source locations and emission quantities as initial input data. As well as using these sources, we create a grid of potential sink points surrounding the clusters and iterate the network optimization algorithms over each potential sink location. As such, we both test the speed and efficiency of the algorithms in finding the optimal graph solution and get an indication as to where the optimal sink location is placed.

We present the CO_2 source clusters used as input data for the sink grid search. The CO_2 source clusters considered are shown by different colours in **Fig. 3**. They are numbered from 1 to 5 in order of increasing number of points. Source clustering was done using a DBSCAN (Ester et al. 1996) algorithm. We limit ourselves to only a few emission clusters to demonstrate the algorithm effectiveness. The sources in grey without a cluster number are therefore not considered here initially. Emission quantities were taken as a 3-year average of years 2015-2017 for emitters listed in the EU Emissions Trading Scheme database and georeferenced to provide coordinates. Some further alterations to the dataset were made. Emissions whose primary activity was energy production were removed, as these sources (mainly coal plants) have an uncertain future in the German emission landscape with the planned phasing out of large coal-fired plants before 2038 at the latest (German Federal Ministry for Economic Affairs and Energy 2020). Through this omission, 73% of the total emission quantity is not considered. Finally, individual sources emitting annually less than 50000 tons of CO_2 per year were excluded to reduce the number of points in the potential networks. These

represent a further reduction of 3% in total emission quantity. In some cases, different emission sources occur in the same location, and are listed as different contributors. For the network optimization procedure, sources with the same coordinates are combined as a single node with emissions summed together.

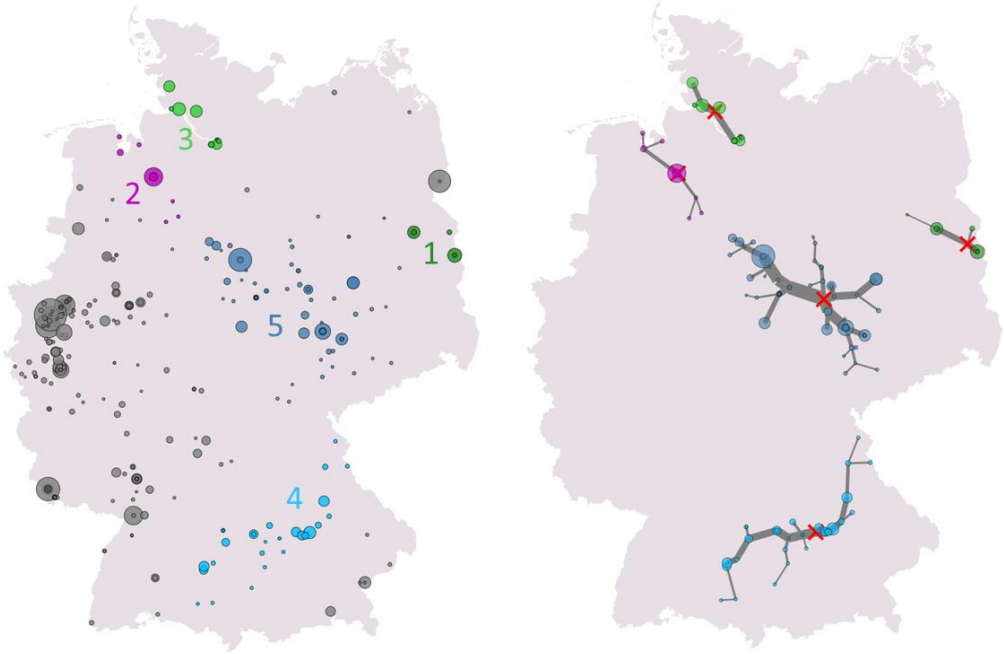

Fig. 4 *Left: CO_2 source clusters considered in this work. Right: lowest-cost networks obtained for each cluster, with optimal sink location shown as a red cross*

Cluster details can be found in Table 2.

Table 2 *Details of CO_2 source clusters*

Cluster number	Geographical region	Number of sources	Number of nodes (with sink)
1	Berlin-Brandenburg	7	5
2	Bremen	8	8
3	Hamburg	12	9
4	Southern Germany	28	26
5	Central Germany	50	38

The potential sink locations were chosen within a grid surrounding the overall shape of the cluster up to 50 km around the convex hull of the shape described by the sources in the cluster. The number of sink locations was adapted each time to the network complexity to account for large calculation times. The algorithms and number of sink points tried for each cluster are given in Table 3. The algorithms considered are Edge Turn (ET), Delta Change (DC), Local Search (LS), Nested Edge Turn (NET), Nested Delta Change (NDC), Tabu Search, Variable Neighbourhood Search (VNS), High Valency Shuffle with Edge Turn (VSET), High Valency Shuffle with Delta Change (VSDC), High Valency Shuffle with Local Search (VSLS). For the three largest clusters, some of the algorithms required extremely large calculation times. For the clusters 3, 4 and 5 the nested algorithms were therefore omitted, also owing to their lower performance demonstrated in the previous section.

Finally, a comparison to a known optimal solution was only possible for the smallest cluster. Iterating over all possible trees rapidly becomes computationally too impractical for other clusters. For the

other clusters, the optimal tree considered was simply taken as the minimum cost tree for all the algorithms but is not guaranteed to be globally minimal.

Table 3 *Numerical details for the grid search around the source clusters*

Cluster number	Algorithms omitted	Number of sink locations considered	Comparison to a guaranteed global minimum?
1	Ø	1666	Yes
2	Ø	1193	No
3	Ø	1076	No
4	NET,NDC	125	No
5	NET,NDC	65	No

Grid search results are summarized in **Fig. 5** with colours consistent with clusters shown in **Fig. 4**.

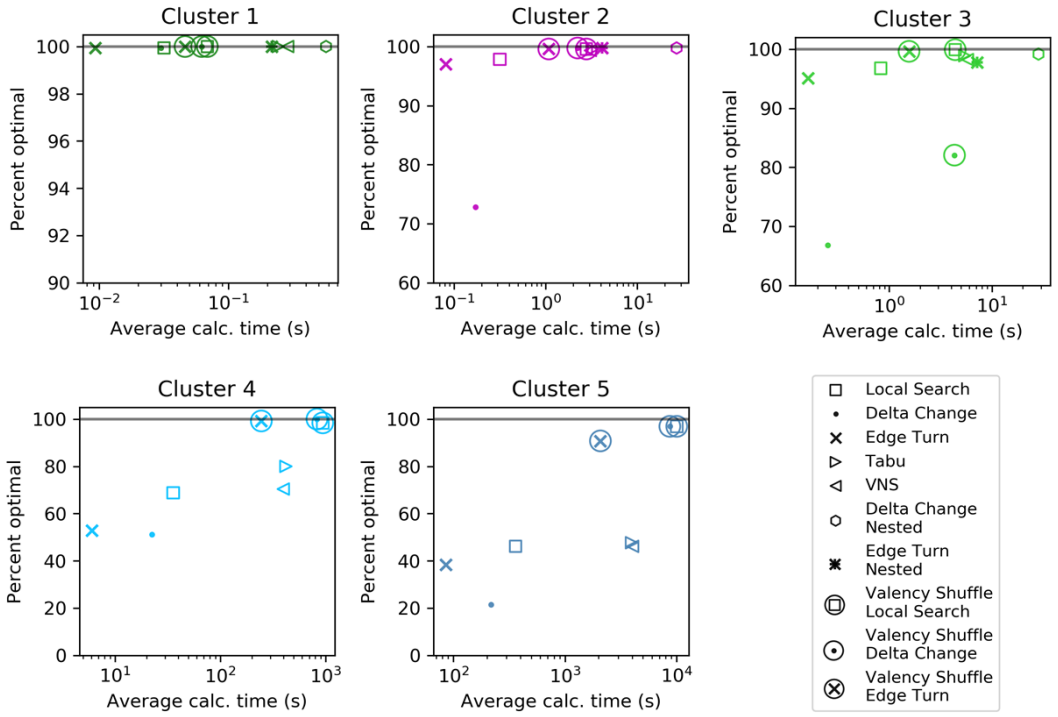

Fig. 5 *Grid search results comparison. Only for Cluster 1 is the comparison to a guaranteed global minimum possible*

Through these grid searches we see that the conclusions obtained with the random graphs are maintained, notably the graphs solutions obtained from the Valency Shuffle algorithms are near-universally optimal. Their performance is particularly superior for the largest clusters, such as cluster 5, and well-above all other literature algorithms considered with similar calculation times such as Tabu and VNS.

One valuable comment concerns the choice of lower-level heuristic to use in combination with the high valency shuffle. A fast but mid-fidelity lower-level heuristic (such as the Edge Turn algorithm) achieves almost equivalent optimality rates when combined with the High Valency Shuffle metaheuristic (VSET) as better performing, but slower, lower-level heuristics such as Local Search (combined as VSLS). In other words, differences in performance of the lower-level heuristic do not seem significantly affect the result obtained when used in combination with the High Valency Shuffle metaheuristic. In this regard, it becomes useful to employ a fast lower-level heuristic (such as the Edge Turn algorithm) in combination with the High Valency Shuffle for the best combination of

optimality and speed. This leads us to speculate that an *even faster lower-level heuristic exploring a reduced fraction of the 1-neighborhood might be sufficient* to provide similarly high optimality rates in combination with the high valency shuffle and further decrease the overall calculation time.

In the Appendices we display the detailed results of the grid searches for all sink locations, presented on top of the original sources. The results are compared through maps showing the location of non-optimal solutions and deviation from optimal cost.

4. Discussion

4.1. High valency metric – another route for improvement.

As described in the Materials and Methods section, the High Valency Shuffle heuristic switches the edges of high valency nodes to their closest neighbours and eliminates (if any) created cycles before applying a local heuristic and then choosing the lowest cost solution. This metaheuristic is an interplay between two distinct metrics when used with the lower-level heuristics shown here. The first is a classical metric based on the *symmetric difference between sets of edges.* This metric is used by the lower level heuristic that breaks single edges and reassembles trees once at each step. The higher-level metaheuristic transfers multiple edges in one step. It can be nonetheless described as making 1-transformations within the metric space described by the *symmetric difference between the sets of nodes with more than 2 edges*.

A typical transformation allowed by the High Valency Shuffle metaheuristic with Local Search is shown in **Fig. 6a**. In this example, taken from the grid search of Cluster 2, the potential sink is represented by a black circle and the sources are coloured as displayed in **Fig. 4**. The sink location remains the same within both panels. The graph to the left of panel a) is the result obtained from the Local Search. The graph to the right of panel a) is the result obtained from the High Valency Shuffle with Local Search, which is found to have a cost 5% lower. The edges of a high valency node are transferred to a different node, as expected. To obtain this graph from the first, the number of 1-transformations in the symmetric difference of edges metric is three. Within the proposed high valency metric, this is equivalent to a local 1-transformation. Indeed, the set of nodes possessing more than two edges has changed by one element.

However, in panel b), the graph to the left is a suboptimal result obtained from the High Valency Shuffle with Local Search. The graph on the right is a lower cost solution found with the Nested Delta Change algorithm. We can therefore see that the High Valency Shuffle was *unable to add* an extra element in the set of high valency nodes (i.e. two high valency nodes rather than one) as it instead *relies on switching* each high valency node to another node. The transformation seen in panel b) is nonetheless a *1-transformation in the high valency metric space* but is not captured by the High Valency Shuffle metaheuristic. A number of suboptimal solutions provided by the High Valency Shuffle metaheuristic can be included in this category of transformations.

The inclusion of a heuristic that could test selected candidate nodes to become high valency nodes without transfer from a prior high valency node would be beneficial for achieving higher optimality rates.

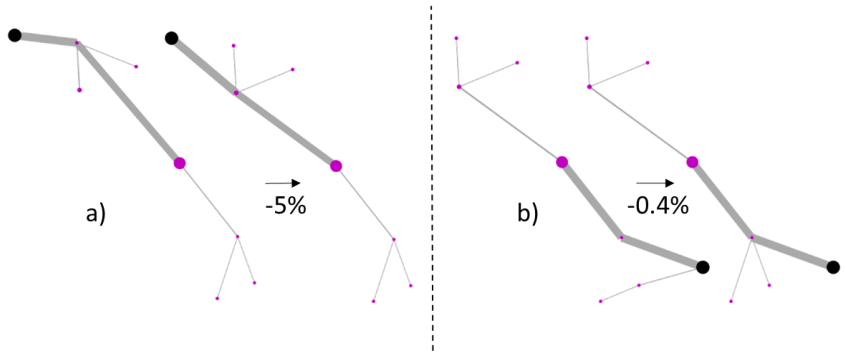

Fig. 6 *Final graphs obtained using different algorithms. In both cases the transformation to the lower cost graph involves a local 1-transformation in the proposed high valency metric space*

4.2. Time complexity analysis and application to large graphs

The calculation times from randomly generated graphs in the previous section are displayed as a function of the number of sources in the network. The straight lines seen in the log-log plot are indicative of polynomial time algorithms. The curves are therefore fitted to a power function of the form: $t(N) = AN^B$, where $t(N)$ is the calculation time in seconds, given in terms of the number of sources N. A and B are the variables to fit. To achieve this, the curves are fitted by minimum least-squares to a linear function in log-log space, with a weight function given by $w = \sqrt{t}$. The weighting is included to avoid the intrinsic bias of fitting a transformed function in log-space in which a disproportional weight is given to the points with smaller x-axis values. The results from the curve-fitting are shown in Table 4.

Table 4 *Power law fit parameters for the time complexity analysis: The algorithms' performance is fitted with the function $t(N) = AN^B$ with t in seconds*

	LS	DC	ET	Tabu	VNS	DCN	ETN	VSLS	VSDC	VSET
A	2.28×10^{-4}	2.09×10^{-4}	0.64×10^{-4}	4.80×10^{-4}	8.82×10^{-4}	2.05×10^{-4}	0.51×10^{-4}	25.82×10^{-4}	0.76×10^{-4}	2.34×10^{-4}
B	3.80	3.69	3.78	4.39	4.19	5.75	5.70	4.07	5.15	4.42

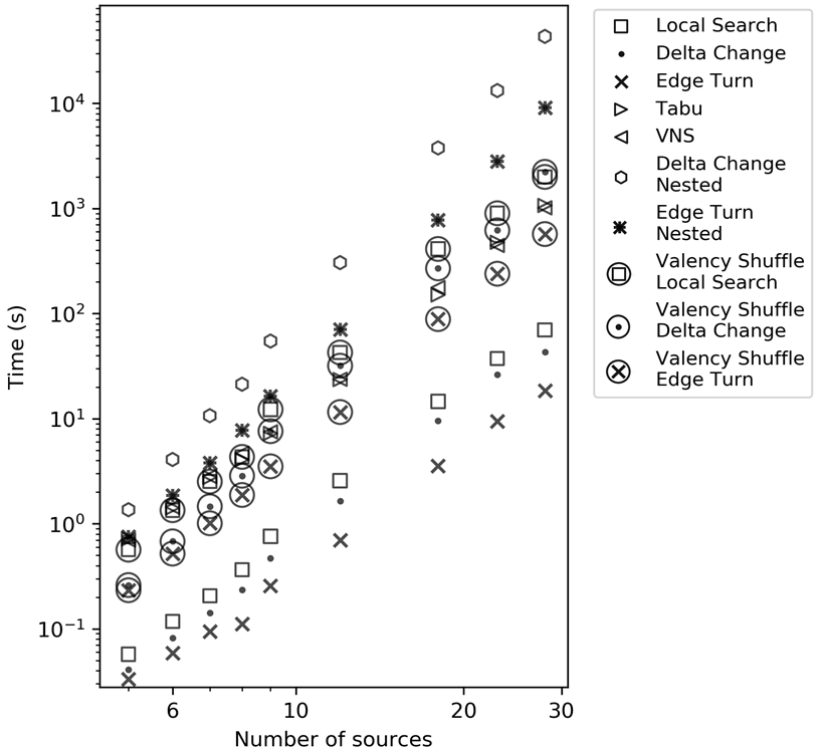

Fig. 7 *Time complexity comparison of the algorithms described in this study*

Finally, a test is done on a large graph composed of CO_2 sources previously not included in the clusters from the previous section. Within this CO_2 source cluster, there are 143 emitters at 94 distinct locations. This cluster is seen at the top left of **Fig. 7**. Combining the sources from the same location gives a network of 95 nodes, including the sink. Due to the large number of nodes in the cluster, only a *single sink* location was considered and placed at the *centre of gravity* of all the sources, weighted by emission quantity. All algorithms were tested but stopped if not completed before a reasonable period of time (48h). The lowest-cost graph was *only obtained using the High Valency metaheuristic* in combination with the Edge Turn heuristic. Predicted calculation times, measured calculation times obtained relative graph costs, when applicable, are shown in Table 5. The cost difference of the network created from the Minimum Spanning Tree (MST), which simply minimizes network length, is also given for comparison.

Table 5 *Details of the network design modelling of the large final CO_2 source cluster*

Algorithm	MST	LS	ET	DC	Tabu	VNS	VSET
Predicted time (h)	0	2.03	0.51	1.14	61.22	45.36	33.51
Measured time (h)	0	2.09	0.31	0.93	25.89	24.89	25.24
Cost difference	+33.6%	+0.19%	+2.6%	+0.94%	+0.03%	+0.19%	0

To better visualise the results from this final example, the spatial networks are shown in **Fig. 8**. The compared algorithms are Local Search (LS), Variable Neighbourhood Search (VNS), Delta Change (DC), Edge Turn (ET), Tabu Search (Tabu) and High Valency Shuffle with Edge Turn (VSET). A comparison with the solution obtained from the Minimum Spanning Tree (MST) is also given. The lowest-cost network obtained with the VSET algorithm is shown with pipelines in grey. Network differences with the lowest-cost network for other solutions are shown in red, whereas sections of the networks shared with the lowest-cost solution are also left in grey. For clarity, the sources are

displayed with a small, constant size, despite contributing differently to the network flow. However, pipeline thickness remains representative of flow quantity. The sink node is shown as a black square.

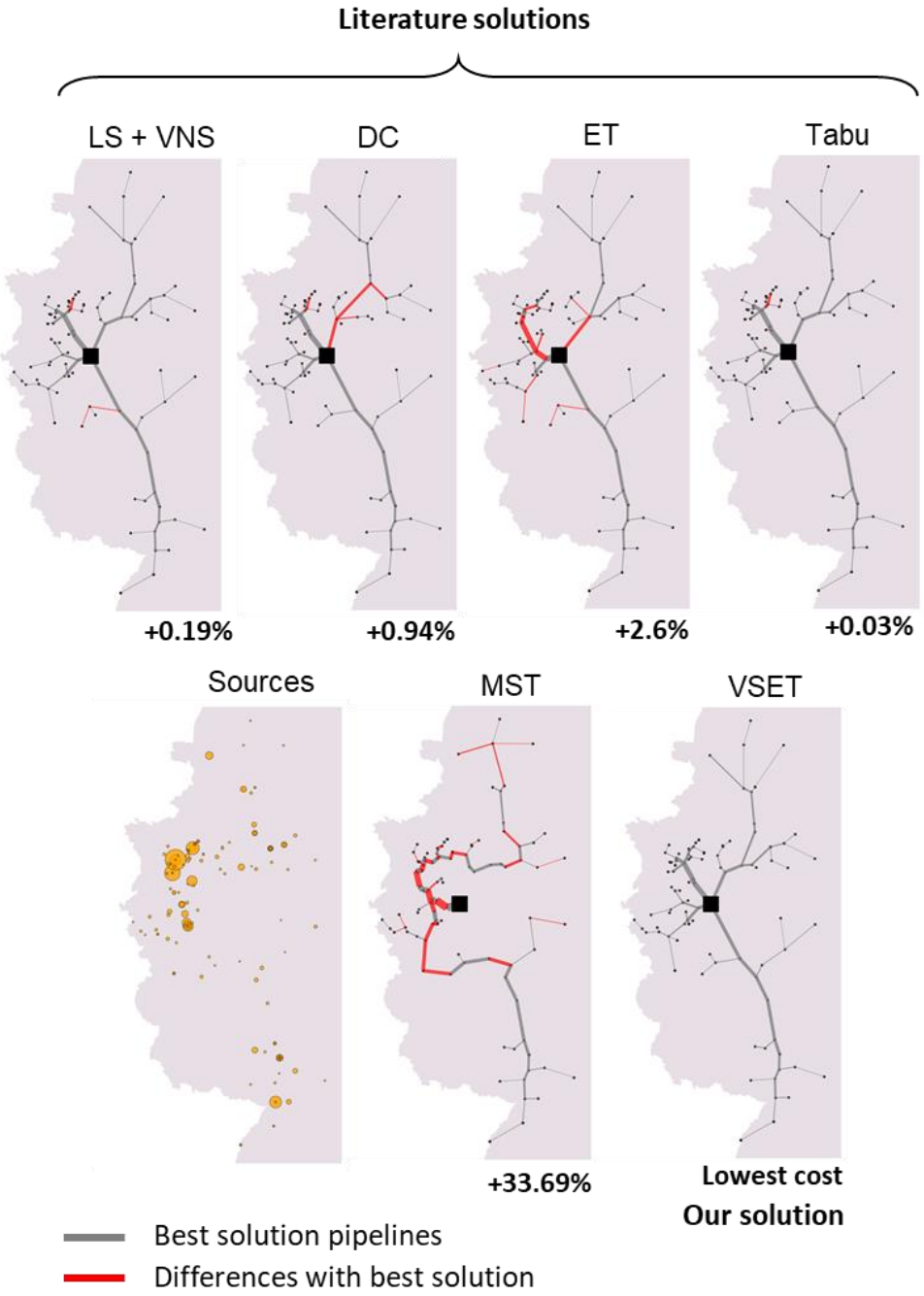

Fig. 8 *Final comparison on algorithms with reasonable (<48h) calculation times on a graph with 95 nodes*

5. Conclusion

In this paper we have conducted performance comparisons on graph optimization heuristics and metaheuristics aimed at achieving minimum-cost capacitated trees within the context of pipeline network design. In the search for lower-cost solutions, we attempt nested solutions of literature algorithms that enable unoriented explorations into wider optimization spaces in a single algorithmic step. These solutions provided higher rates of optimal networks than literature options, at the expense of larger calculation times. A new metaheuristic and associated distance metric were developed that captured useful network transformations easily. Such transformations had required a large amount of algorithm steps previously and were often not accessed at all. The High Valency

Shuffle, switching the edges of nodes with more than two edges to neighbours, proved to achieve almost 100% optimal networks in most examples used. The calculation times for the new metaheuristic, while large, remain comparable to current literature options such as Tabu or Variable Neighbourhood search while achieving greater proportions of optimal solutions. The new metaheuristic is used in combination of a lower-level heuristic, the choice of which seems to influence only the resulting solution cost but has a large effect on calculation time. It seems therefore beneficial to combine the High Valency Shuffle with a fast low-level metaheuristic which may by itself only achieve low rates of optimal solutions.

References

Alhajaj A, Shah N (2020) Multiscale design and analysis of CO2 networks, International Journal of Greenhouse Gas Control, 94, 102925. https://doi.org/10.1016/j.ijggc.2019.102925

André J, Auray S, Brac J, De Wolf D, Maisonnier G, Ould-Sidi MM, Simonnet A (2013) Design and dimensioning of hydrogen transmission pipeline networks, European Journal of Operational Research, 229(1):239-251. https://doi.org/10.1016/j.ejor.2013.02.036

Bietresato M, Friso D, Sartori L (2013) An operative approach for designing and optimising a pipeline network for slurry collection from dairy farms across a wide geographical area, Biosystems Engineering, 115(3):354-368. https://doi.org/10.1016/j.biosystemseng.2013.01.008

Brimberg J, Hansen P, Lin K, Mladenovi N, Breton M, Brimberg, J (2003) An oil pipeline design problem. Operations Research, 51(2):228–239. https://doi.org/10.1287/opre.51.2.228.12786

van den Broek M, Brederode E, Ramírez A, Kramers L, van der Kuip M, Wildenborg T, Faaij A, Turkenburg W (2009) An integrated GIS-MARKAL toolbox for designing a CO2 infrastructure network in the Netherlands, Energy Procedia, 1(1):4071-4078. https://doi.org/10.1016/j.egypro.2009.02.214

Cayley, A (1857) On the Theory of the Analytical Forms Called Trees, Philosophical Magazine, 4(13):172–176. https://doi.org/10.1080/14786445708642275

Ester,M et al. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96). AAAI Press, 226–231. https://dl.acm.org/doi/10.5555/3001460.3001507

German Environment Federal Office (2019) Installations covered by ETS Germany in 2018, Deutsche Emissionshandelsstelle. Available at: https://www.dehst.de/SharedDocs/downloads/EN/installation_lists/2018.pdf (Accessed: 2 September 2020)

German Federal Ministry for Economic Affairs and Energy (2020). Final decision to launch the coal-phase out – a project for a generation [Press release]. 3 July. Available at: https://www.bmwi.de/Redaktion/EN/Pressemitteilungen/2020/20200703-final-decision-to-launch-the-coal-phase-out.html (Accessed: 26 November 2020)

Gilbert EN (1967) Minimum cost communication networks. AT&T Tech J. 46(9):2209–2227. https://doi.org/10.1002/j.1538-7305.1967.tb04250.x

Glover F (1989) Tabu Search – Part 1. ORSA Journal on Computing. 1(2):190–206. https://doi.org/10.1287/ijoc.1.3.190

Heijnen P, Chappin E, Herder P (2020) A method for designing minimum-cost multisource multisink network layouts, Systems Engineering, 23:14-35. https://doi.org/10.1002/sys.21492

Kazmierczak T, Brandsma R, Neele F, Hendriks C (2009) Algorithm to create a CCS low-cost pipeline network. Energy Procedia. 1(1):1617–1623. https://doi.org/10.1016/j.egypro.2009.01.212

Kruskal, J. B. (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society. 7:48-50. https://doi.org/10.1090/S0002-9939-1956-0078686-7

Maier HR, Simpson AR, Zecchin AC (2003) Ant colony optimization for design of water distribution systems. Journal of Water Resource Planning and Management, 129(3):200–209. https://doi.org/10.1061/(ASCE)0733-9496(2003)129:3(200)

Mladenovic N, Hansen P (1997) Variable Neighborhood Search, Computers and Operations Research, 24:1097-1100. https://doi.org/10.1016/S0305-0548(97)00031-2

Prüfer, H (1918). Neuer Beweis eines Satzes über Permutationen. Arch. Math. Phys. 27:742–744.

Sun L, Chen W (2016) Development and application of a multi-stage CCUS source – sink matching model. Applied Energy. 185:1–9. https://doi.org/10.1016/j.apenergy.2016.01.009

Rothfarb B, Frank H, Rosenbaum DM, Steiglitz K, Kleitman DJ (1970) Optimal Design of Offshore Natural-Gas Pipeline Systems, Operations Research. 18(6):992-1020. http://doi.org/10.1287/opre.18.6.992

Weihs GAF, Wiley DE, Ho M (2011) Steady-state optimization of CCS pipeline networks for cases with multiple emission sources and injection sites: South-East Queensland case study. Energy Procedia. 4:2748–2755. https://doi.org/10.1016/j.egypro.2011.02.177

Xue G, Lillys TP, Dougherty DE (1999) Computing the minimum cost pipe network interconnecting one sink and many sources. SIAM J Optim. 10(1):22–42. https://doi.org/10.1137/S1052623496313684

Yeates C, Schmidt-Hattenberger C, Bruhn D (2020) Potential CO2 Networks for Carbon Storage in a German Net-Zero Emission Landscape, EGU General Assembly 2020, Online, 4–8 May 2020, EGU2020-20085. https://doi.org/10.5194/egusphere-egu2020-20085

Appendix

Grid search results

The grid search results are displayed on top of the original sources, showing for each potential sink location, a small square if the solution found is not optimal. Furthermore, the difference in cost from the optimal solution is displayed as a colour gradient within the square. The cost of the minimum cost network found (either guaranteed optimal for cluster 1 or minimum overall solution for the other clusters) is shown on a separate map and named “optimal cost”. The lowest-cost sink of all potential sink locations is indicated by a red cross. The map of isomorphic graph zones for the overall lowest cost solutions is shown for some clusters. Isomorphic graph zones display areas in which the graphs share the same list of edges, i.e. are structurally the same but may have varying edge lengths or costs. As well as being illustrative of the variety of optimal solutions occurring over a large space of potential sink locations, the isomorphic graph zones can also represent a weak proof of global optimality of solutions. Indeed, continuous zone boundaries and a lack of “island” solutions (i.e. a single location or small number of solutions fully surrounded by a different, single isomorphic graph zone) are to be expected with globally optimal isomorphic graph zones. For the clusters with a lower number of sink locations (i.e. clusters 4 and 5), we do not display the optimal isomorphic zones as the

low spatial density of sink locations leads to each sink location usually belonging to its own individual zone. For each sink location, we do not display the obtained graphs as the optimal topology is of little importance with regards to the algorithm's effectiveness, and this would be impractical due to the large number of sink locations. For completeness, we give the minimum cost tree for the lowest cost sink location in each of the 5 CO_2 source clusters considered in **Fig. 3**. Some algorithm results are not displayed in this map form as they achieve optimal solutions at every sink location. Finally, in cluster 1, the results obtained for certain algorithms were the same and are shown in a single map.

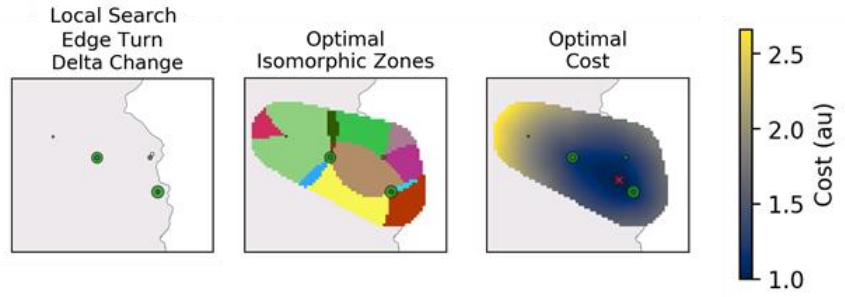

Fig. 9 *Cluster 1 grid search results. Local Search, Edge Turn and Delta Change here only show one non-optimal solution adjacent to the top rightmost source*

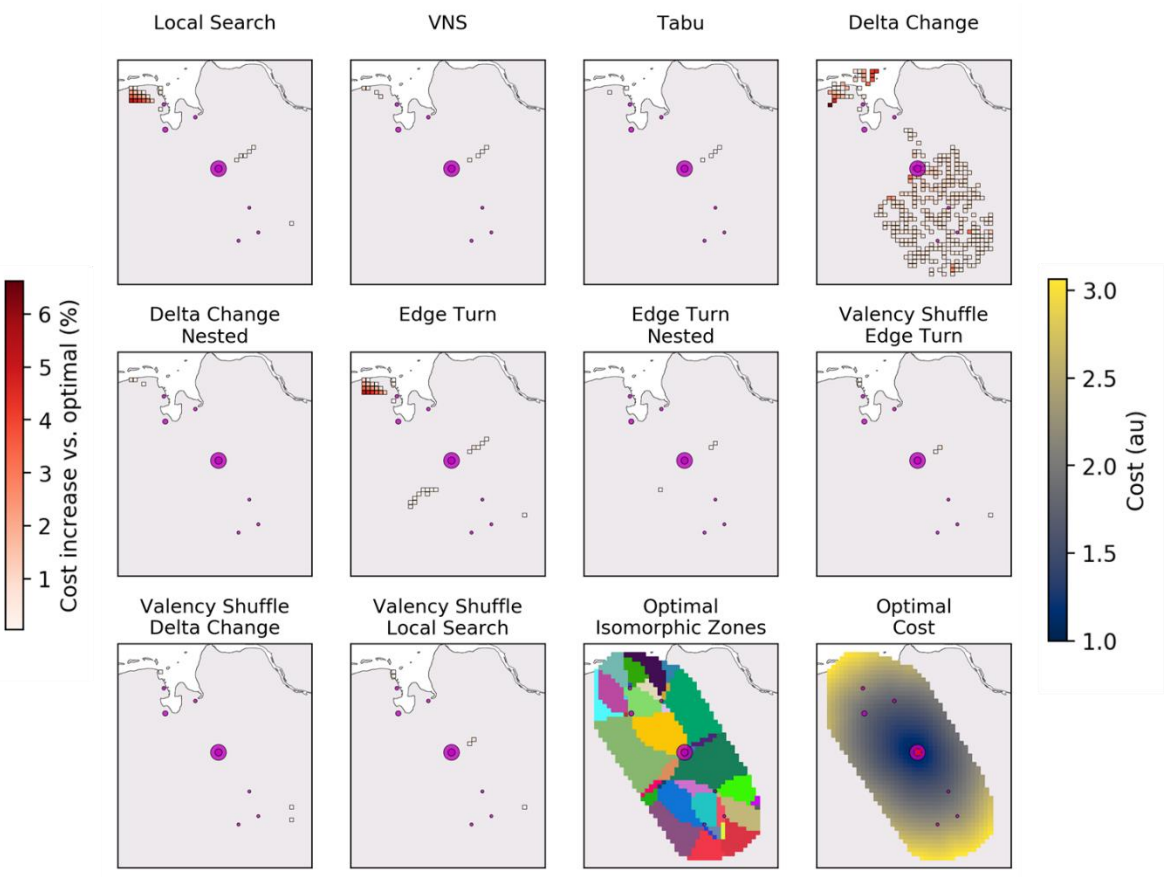

Fig. 10 *Cluster 2 grid search results*

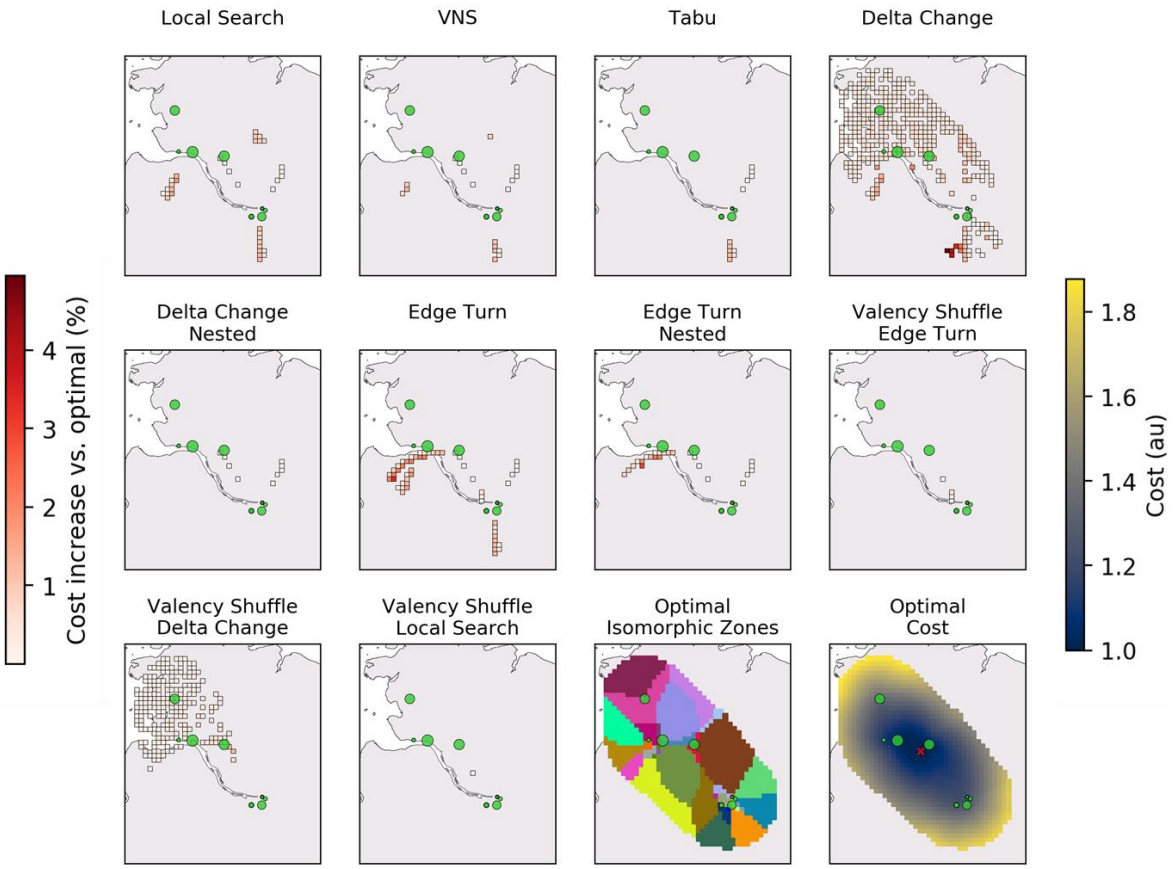

Fig. 11 *Cluster 3 grid search results*

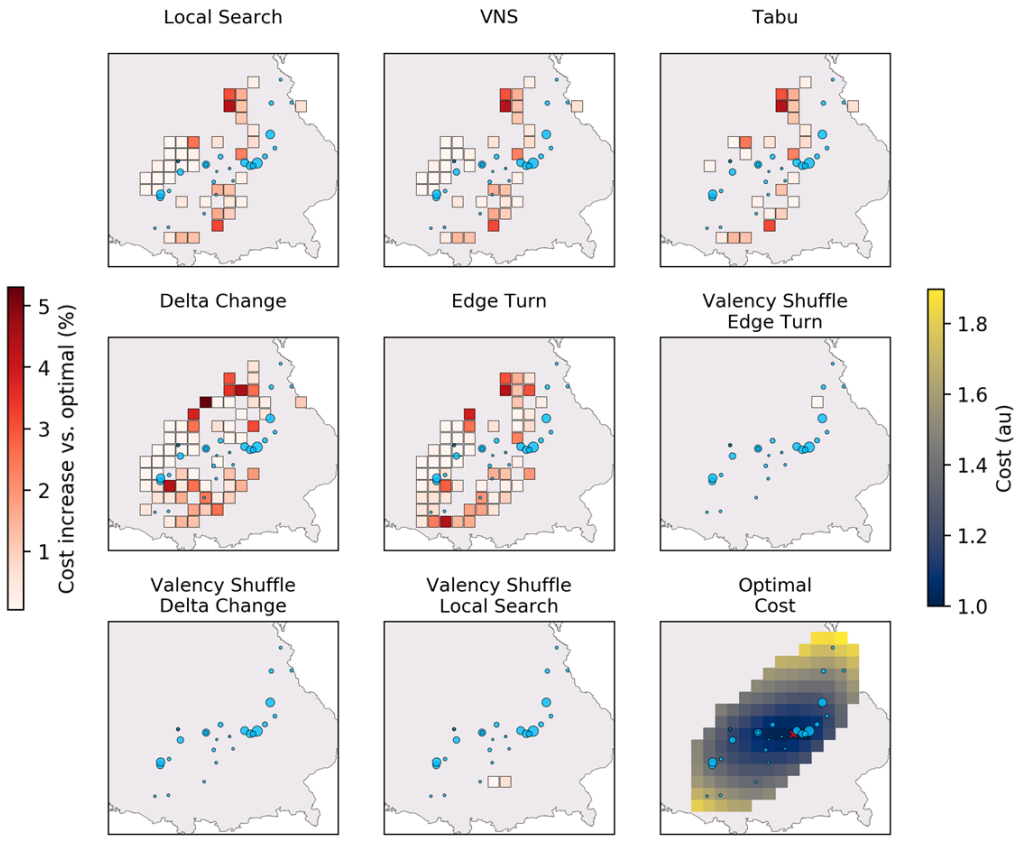

Fig. 12 *Cluster 4 grid search results*

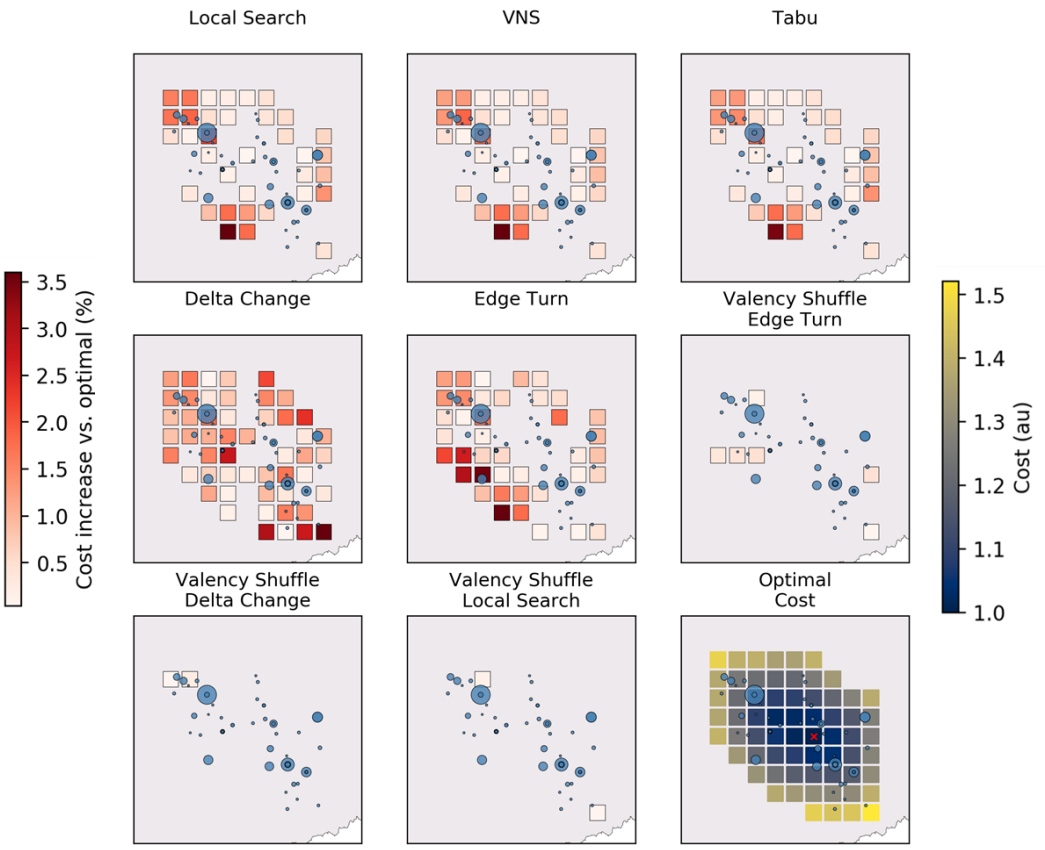

Fig. 13 *Cluster 5 grid search results*