Comparing the Optimization Behaviour of Heuristics with Topology Based Visualization

Simon Bin[1], Sebastian Volke[2], Gerik Scheuermann[2], and Martin Middendorf[1]

[1] Parallel Computing and Complex Systems Group, Institute of Computer Science, University of Leipzig, Germany
`{sbin,middendorf}@informatik.uni-leipzig.de`
[2] Image and Signal Processing Group, Institute of Computer Science, University of Leipzig, Germany
`{volke,scheuermann}@informatik.uni-leipzig.de`

Abstract. In this paper we propose some changes and extensions of the visualization approach that is used in the visualization tool dPSO-Vis. It is shown also how corresponding visualizations can help for the analysis of fitness landscapes of combinatorial optimization problems and for understanding and comparing the optimization behaviour of heuristics. As an example we use a small instance of the Travelling Salesperson Problem (TSP) and the three heuristics Greedy, Random greedy, and Simulated Annealing.

Keywords: visualization, fitness landscape, combinatorial optimization problem, barrier landscape, heuristic, optimization behaviour.

1 Introduction

In this paper we show how visualization can be helpful for comparing and understanding the optimization behaviour of (meta)heuristics for combinatorial optimization problems. In particular, we propose some changes and extensions of the visualization approach that is used in the recent visualization tool dPSO-Vis ([11]). A one-dimensional landscape is used for the visualization of the topology of the search space and of the search process of the meta-heuristic. We concentrate here on the visualization of small problem instances for which it is possible to generate all solutions and to determine their fitness values. We also shortly discuss how the concepts that are introduced in this paper could be applied to larger problem instances. The Travelling Salesperson Problem (TSP) is used as the example optimization problem and the heuristics Greedy, Random greedy, and Simulated Annealing are the example metaheuristics.

It is assumed here that an optimization problem is given as a finite set of solutions X together with an objective function $f : X \to \mathbb{R}$ that assigns each solution its fitness. The optimization problem is then to find a solution with minimum objective value, i.e. with highest fitness. Instead of a minimization problem, a maximization problem could also be considered. In addition it is assumed that a neighbourhood relation $N \subset X \times X$ is given on the set of solutions. Here we assume that N is symmetric, i.e. $(x, y) \in N$ iff $(y, x) \in N$. Let $N(x) = \{y \in X \mid (x, y) \in N\}$ be the neighbours of x. Then (X, N) is the neighbourhood graph and (X, N, f) is called fitness landscape. In this paper we assume that (X, N) is connected. The neighbourhood relation should reflect the process used by the heuristic to create new solutions from a given solution as described in the following. Given a solution $x \in X$ the heuristic can create a new solution y from x only when $(x, y) \in N$. Alternatively, when the heuristic is allowed to perform several steps for creating a new solution, it can create y from x only when there exists solutions $x = x_1, x_2, \ldots, x_k = y$ such that $(x_i, x_{i+1}) \in N$ for $i \in \{1, 2, \ldots, k-1\}$. In some cases the neighbourhood graph might reflect not all possible transitions but only the transitions that occur with a certain likelihood. The details of which solutions are actually created and from which starting solutions depend on the particular heuristic.

The basic concept for the one-dimensional landscape that is used here for the visualization stems from Volke et al. [11]. Their approach has been implemented in the visualization tool dPSO-Vis. More details are described in Section 2. It was also shown by Volke et al. in [10] how dPSO-Vis can be used to compare the optimization behaviour of two discrete Particle Swarm Optimization algorithms (PSO) for solving the RNA folding problem. We do not give an overview on other visualization methods here but point the reader to [9].

The changes and extensions that are introduced here to the visualization concept of dPSO-Vis are explained in Section 2. The used heuristics and the test instances are described in Section 3. Results are shown in Section 4. The paper ends with conclusions in Section 5.

2 Visualization Method

The visualization of dPSO-Vis [11] is based on the barrier tree data structure which has been proposed by Flamm et al. [1] to represent the topology of a fitness landscape (X, N, f). The barrier tree can be used to partition the nodes of (X, N) into so called basins and to associate every barrier tree arc with one basin ([11]). In the following this is explained in more detail.

The barrier tree is a directed tree B where each node v of B has a height $h(v) \in \mathbb{R}$. For a node v in B let B_v be the subtree of B with node v. Let $(X, N)_\eta$ be the subgraph of (X, N) that is generated by all nodes $x \in X$ with $f(x) \leq \eta$, Similarly, let $(X, N)_{<\eta}$ be the subgraph of (X, N) that is generated by all nodes $x \in X$ with $f(x) < \eta$. In the barrier tree B each leaf represents a local minimum of (X, N, f), i.e. a node $x \in X$ such $f(x) \leq f(y)$ for all $y \in N(x)$. If there exists a set of local minima that are connected in the subgraph of (X, N) that is generated by the local minima, all nodes in the set are represented by the same single leaf in B. Each subtree of B with root v and height $h(v)$ corresponds to a connected component C of $(X, N)_{h(v)}$ such that this component is not connected in $(X, N)_{<h(v)}$, i.e. $C_{<\eta}$ is not connected. C is the component of $(X, N)_{h(v)}$ that contains the local minima which are represented by the leaves of B_v. The height $h(v)$ is called the barrier between the components of $C_{<\eta}$. Thus, an inner node

v of the barrier tree represents all nodes $x \in C$ with $h(x) = h(v)$. Some of these nodes represent a barrier between the different parts of the fitness landscape that correspond to the child nodes of v. An edge (u, v) of the barrier tree B corresponds to all nodes x of (X, N) for which it holds: i) $x \notin (X, N)_{<h(u)}$ and ii) $x \in (X, N)_{<h(v)}$ and x is connected to a leaf of B_v. This set of nodes is called basin. As a consequence, the subtree B_v represents all nodes in $(X, N)_{h(v)}$ which are connected to a local minimum that corresponds to a leaf in B_v.

Volke et al. [11] proposed to add a node v, that corresponds to the global maximum of (X, N, f), to the nodes of the barrier tree B. Also, an edge is inserted from the (original) root of the barrier tree to v. Then, every edge of the barrier tree represents a subset of the nodes of (X, N) with an objective value that lies in a certain interval of $\mathbb{R}$. Also, all nodes of (X, N) are represented by B and the edges of B partition the nodes of (X, N) into basins. It should be noted that the barrier tree can be computed with a flooding algorithm [1]. This algorithm can be easily extended to also compute the partitioning of the solution space ([11]).

In dPSO-Vis a 2D landscape profile called barrier landscape is computed that is topologically equivalent to (X, N, f) and therefore has the same barrier tree (details see [11]). The form of the 2D landscape is a height graph over a 1D line which contains a valley for every leaf node of the barrier tree and nested valleys for every subtree of it. If two subtrees are joined by an inner node, there is a corresponding mountain pass within the landscape profile that joins the corresponding valleys. Each edge (u, v) of the barrier tree corresponds to the right slope and the left slope of the corresponding valley. The middle part of the valley corresponds to the subtree B_u. This construction leads to a roughly symmetric impression of the barrier landscape with high slopes at both sides and the lowest part in the middle.

In this paper we use an asymmetric visualization of the barrier landscape where all nodes of (X, N) that correspond to a single edge of the barrier tree are drawn as a single slope which increases from left to right (examples are shown in figures 2 and 4). Moreover, for each inner node v of B and each edge (u, v) we add a line that connects the highest node of (X, N) which is represented by (u, v) to the leftmost node that is represented by v. We call these lines barrier lines. Recall that several nodes of (X, N) with the same height might be represented by v. Some additional changes with respect to the barrier landscape in dPSO-Vis are: i) All local minimum nodes are marked in the barrier landscape (if several connected minimum nodes exist, only the leftmost is marked), ii) all leftmost nodes that are represented by inner nodes of the barrier tree are marked, and iii) for each node the height of its lowest neighbour in (X, N) is shown below it.

In order to visualize the behaviour of a heuristic we mark the solutions that are the result of a run of the heuristic. Since for several runs of a heuristic a solution might have been found several times, the size of the marking is proportional to how often the solution has been found. It should be noted that the type of the used markings and their colour can by easily changed for the visualization.

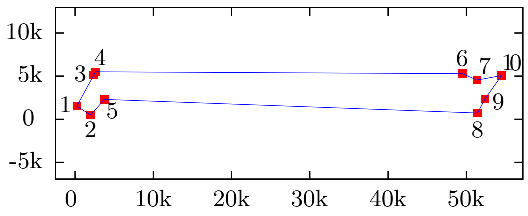

Fig. 1. The TSP test instance

3 Heuristic Algorithms and Test Instance

As a test optimization problem we use the well known Travelling Salesperson Problem (TSP). Given is a set of n cities and an $n \times n$ distance matrix $D = |d(i,j)|_{i,j \in \{1,\ldots,n\}}$ where $d(i,j)$ is the distance from city i to city j. Here we consider symmetric TSP instances only, i.e. $d(i,j) = d(j,i)$ for $i,j \in \{1,2,\ldots,n\}$. The problem then is to find a shortest round tour that contains each city exactly once, i.e. to find a permutation π of $1,2,\ldots,n$ such that $d(\pi(n),\pi(1)) + \sum_{i=1}^{n} d(\pi(i),\pi(i+1))$ is minimal. Note, that every cyclic shift of a permutation π and the corresponding inverse permutations lead to the same round tour and therefore to the same solution. Thus, the set X of solutions can be described by choosing one representative from each equivalence class of solutions, e.g. as the set of all permutations π of $\{1,2,\ldots,n\}$ with $\pi(1) = 1$ and $\pi(2) \leq \lfloor n/2 \rfloor + 1$.

The neighbourhood relations that we consider here as examples are the swap neighbourhood and the width restricted interchange neighbourhood. In the swap neighbourhood two permutations π, π' are neighboured if one of the following conditions hold: i) $i \in \{1,2,\ldots,n-1\}$ such that $\pi(i) = \pi'(i+1)$, $\pi(i+1) = \pi'(i)$ and $\pi(j) = \pi'(j)$ for all $j \in \{1,2,\ldots,n\} - \{i,i+1\}$, or ii) $\pi(n) = \pi'(1)$, $\pi(1) = \pi'(n)$ and $\pi(j) = \pi'(j)$ for all $j \in \{2,3,\ldots,n-1\}$. In the interchange neighbourhood two permutations π, π' a neighboured if one of the following conditions hold: if there exists $i,j \in \{1,2,\ldots,n\}$ such that $\pi(i) = \pi'(j)$, $\pi(j) = \pi'(i)$ and $\pi(h) = \pi'(h)$ for all $h \in \{1,2,\ldots,n\} - \{i,j\}$. The k-width restricted interchange neighbourhood has the additional restriction that $|i-j| \leq k$ must hold. Note, that there exists many other (and for heuristics often better) neighbourhoods that are used for solving the TSP problem.

As a test instance we have chosen a 10 city TSP instance that consists of 2 clusters of 5 cities each. The instance is shown in Figure 1. The distances between the cities are the Euclidean distances.

In order to demonstrate how the visualization can be used to compare the optimization behaviour of heuristics we have chosen the following three example heuristics: Random start greedy, Randomized greedy, and Simulated Annealing.

Greedy starts with a uniform randomly chosen permutation $\pi \in X$. Then it chooses the best permutation $\pi' \in N(\pi)$, i.e. the permutation π' with minimum

Algorithm 1. Simulated Annealing algorithm

```
1: $T \leftarrow 1$ probability to accept a worse solution
2: $s \leftarrow x$ where $x \in X$ is chosen randomly
3: $accept \leftarrow true$
4: while accept=true do
5:     $accept \leftarrow false$
6:     while $accept = false$ and $\exists$ untested neighbour in $N(s)$ do
7:         randomly choose untested neighbour $x \in N(s)$
8:         if $f(x) < f(s)$ or $r < T$ where $r$ is uniform randomly
             chosen in [1,0] then
9:             $s \leftarrow x$
10:            $accept \leftarrow true$
11:        end if
12:    end while
13:    $T \leftarrow (1 - \rho)T$
14: end while
15: return the best found solution
```

value $f(\pi')$. If there exists more than one such permutation one of them is chosen randomly. If $f(\pi') < f(\pi)$ then the last step is executed again (with π' instead of π). Otherwise, the algorithm stops.

Randomized greedy works similar as Greedy but with the difference that it chooses a random solution (instead of the best solution) from all solutions $\pi' \in N(\pi)$ for which $f(\pi') < f(\pi)$ holds if such a solution exists. Otherwise, the algorithm stops.

The third heuristic is a simple standard Simulated Annealing algorithm. A pseudo-code can be found in Algorithm 1. For the tests the value $\rho = 0.05$ has been used for the parameter of changing the temperature.

4 Results

In this section we show what the proposed visualization method can tell us about the fitness landscape of a problem instance, the neighbourhood relation, the optimization behaviour of the heuristics and the connections between these three aspects. If not mentioned otherwise, all results use the swap neighbourhood.

4.1 Fitness Landscape

A visualization of the whole fitness landscape of the test instance is shown in Figure 2. The figure shows that the landscape consists of 5 plateaus with a steep descent between two neighboured plateaus. The fitness differences between the solutions within each plateau are smaller than the fitness differences between two plateaus. An exception is the third (numbered from left to right) plateau in the middle where several very narrow valleys exist that have solutions with a fitness that is in the range of fitness of the second plateau. A detailed view

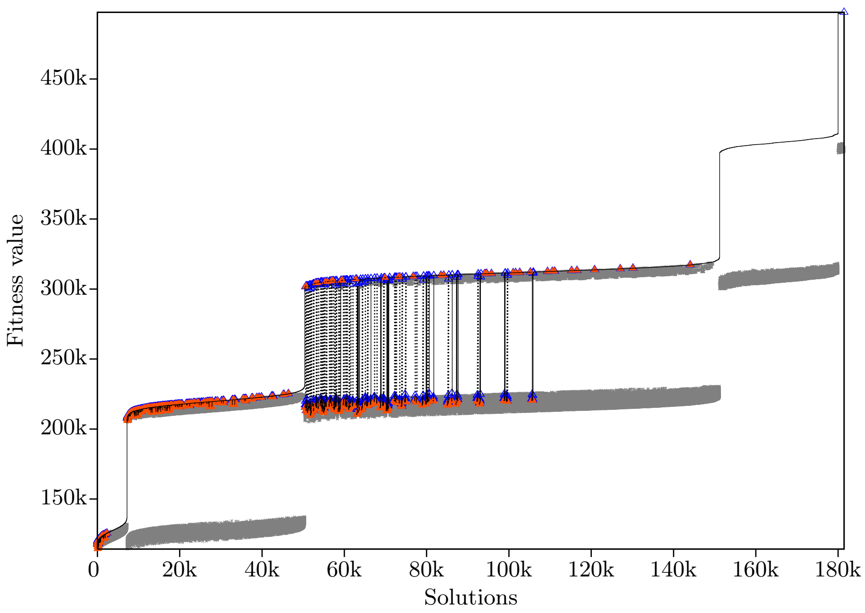

Fig. 2. Visualization of the fitness landscape of the TSP test instance: fitness of all solutions, local minima (red filled triangles), representatives for barriers (explained in the text) (blue triangles), quality of best neighbours (grey crosses).

of these valleys can be seen in Figure 3. The figure shows that only very few solutions are part of each valley. Figure 2 shows also that the plateau with the best (worst) solutions is very small and contains only about 2% of the solutions. The middle plateau, in contrast, contains more than half of the solutions.

Another interesting fact that can be seen from the visualization is, that the two worst plateaus do not contain any local minimum. For heuristic algorithms which consider the whole neighbourhood of a solution this means that there is no danger to get stuck at the very bad solutions of plateaus 4 and 5. The other three plateaus contain many local minima and most of them lay in the better (left) part of each plateau. The fitness of all these local minima is not much better than their neighbours in the plateau (with the exception of the already mentioned steep valleys that occur in the left part of middle plateau).

It can be seen that many solutions that lay within one of the plateaus 2-5 have a best neighbour which has a much better quality than the node itself. These neighbours lay in the plateau that is the next to the left. This shows that heuristics which evaluate the whole neighbourhood of a solution have good chances for a fast improvement, i.e. to jump directly from one plateau to the next better plateau.

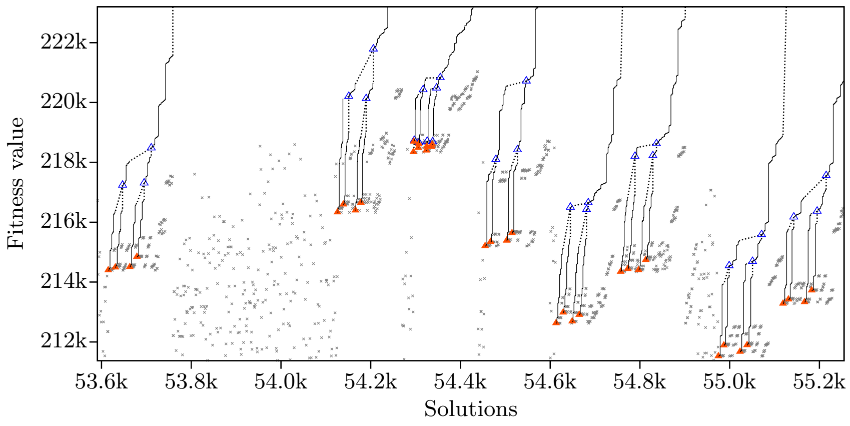

Fig. 3. Detailed view of some deep valleys of the middle plateau; local minima (red filled triangles), representatives for barriers (explained in the text) (blue triangles), quality of best neighbours (grey crosses), barrier lines (dotted lines).

A detailed view of the left part of the first plateau, i.e. the best part of the fitness landscape, is shown in Figure 4 (top). The many local minima in that plateau show that it might easily happen for a heuristic to get stuck in one of these local minima.

For the example test instance it was shown that the visualization of its fitness landscape reveals several interesting properties. Also, the extent of the properties is immediate from the visualization. However, it could not be estimated from simply looking at the problem instance alone. The visualization also leads directly to conclusions about the presumed optimization behaviour of heuristics in the respective part of the landscape. A closer look at the problem instance then gives us explanations for some of the properties. For example, the reason why there are five plateaus is the following. The problem instance consists of two clusters of five cities each. Each solution has 2, 4, 6, 8 or 10 edges that contain one city from each cluster. Obviously, the smaller the number of such edges is the better is a solution. Thus, the 5 plateaus correspond to these five classes of solutions. The middle plateau contains most of the solutions, because — as a short combinatorial calculation shows — the number of solutions with 6 edges between the clusters is much larger than the number of solutions with fewer or more such edges.

4.2 Neighbourhood Relation

The lower part of Figure 4 shows the same part of the fitness landscape for different neighbourhoods. Clearly, the quality of the solutions is independent of the neighbourhood. What changes is the number and the location of local

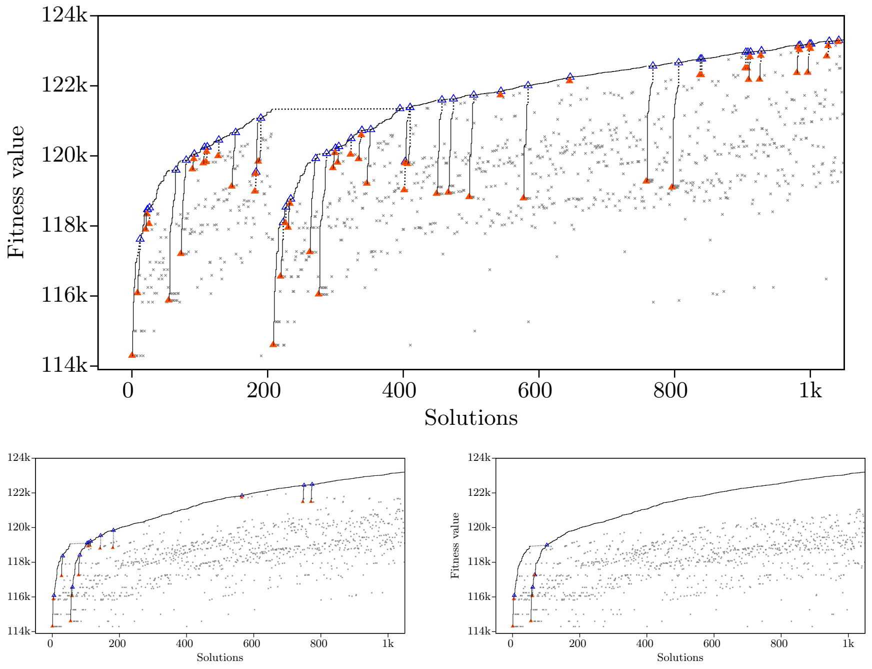

Fig. 4. Detailed view of the left (best) part of the fitness landscape; local minima (red filled triangles), representatives for barriers (explained in the text) (blue triangles), quality of best neighbours (grey crosses), barrier lines (dotted lines). Swap neighbourhood (top), 2-restricted interchange neighbourhood (bottom left) and 3-restricted interchange neighbourhood (bottom right).

minima. It can be seen clearly that the landscape with the (width restricted) interchange neighbourhood is simpler and has less local minima than the landscape with the swap neighbourhood. Moreover, the landscape with the 3-width restricted interchange neighbourhood is simpler than the landscape with the 2-width restricted interchange neighbourhood. The reason is that a solution has more neighbours in the 2-width restricted interchange neighbourhood than in the swap neighbourhood and it has more neighbours in the 3-width restricted interchange neighbourhood than in the 2-width restricted interchange neighbourhood. It may also be noted, that the 1-width restricted interchange neighbourhood is equal to the swap neighbourhood.

For a heuristic this means that the problem becomes simpler and the danger to get stuck in a local minimum is smaller with the 2- or 3-width restricted interchange neighbourhood than with the swap neighbourhood. This is at the expense of a growing neighbourhood. It should be mentioned that we do not argue here, that such observations are new. The point is, that the proposed visualization shows such effects of different neighbourhoods very clearly and also the strength of this effect is easily visible for the user.

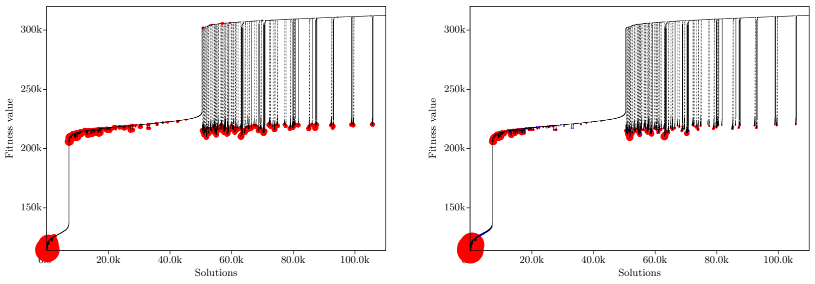

Fig. 5. Solutions of the heuristics Greedy (left) and Simulated Annealing (right), 10000 runs of each; shown is the left half of the fitness landscape; red bright dots = found local minima, blue dark dots = found non local minima.

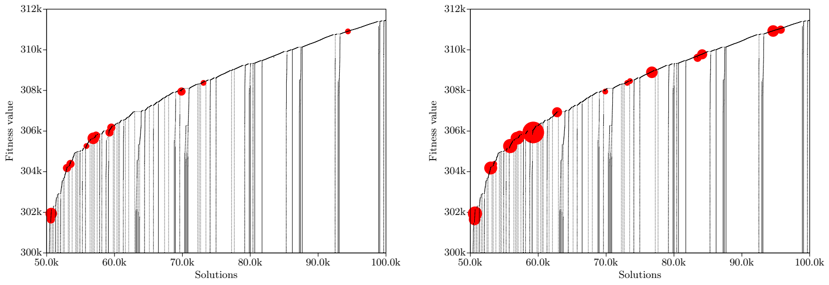

Fig. 6. Solutions found by heuristics Greedy (left) and Random greedy (right), 10000 runs of each; shown is the left upper part of plateau 3; red bright dots = found local minima.

4.3 Optimization Behaviour

The optimization behaviour of the heuristics Greedy and Simulated Annealing can be seen in Figure 5. In each figure the results of 10000 runs of the corresponding heuristic are shown. It can be seen that both heuristics often find solutions that lay within the best plateau. However, many solutions lay also in the left part of plateau 2 and also in the deep valleys of plateau 3. It can be seen that Simulated Annealing works on average slightly better than Greedy because more solutions of the former lay within plateau 1. Also the solutions of Simulated Annealing within plateau 2 lay more in the better (left) part. The visualization also shows that some solutions of Greedy lay also outside of the deep valleys on plateau 3. Since this cannot be seen in the print version of Figure 5 a detailed version of that part is shown in Figure 6. This figure shows also that Random greedy has more solutions than Greedy in this not so good part of the fitness landscape. In principle, this could have been expected. However, the visualization shows this effect quantitatively.

A detailed view of the best (left) part of plateau 1 is shown in Figure 7. All local minima have been found by at least one of the 10000 runs of each heuristic.

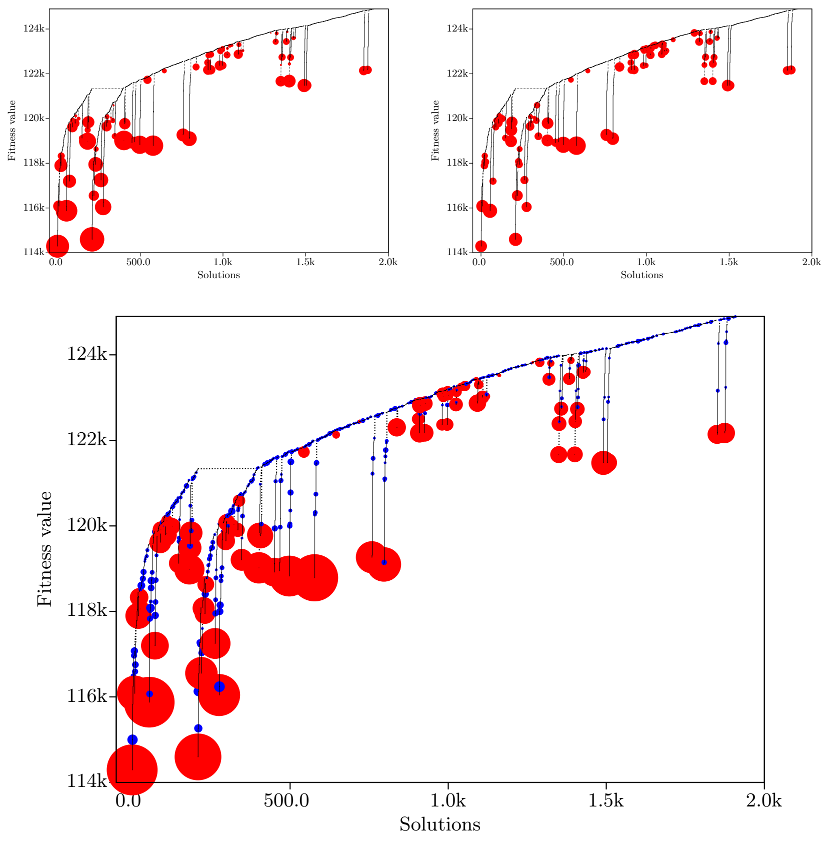

Fig. 7. Solutions found by Greedy (upper left), Random greedy (upper right), Simulated Annealing (bottom), 10000 runs of each; shown is the left (best) part of the fitness landscape; red bright dots = found local minima, blue dark dots = found non local minima.

It can be seen that Simulated Annealing found the best solutions more often than the other heuristics. Greedy found the best solutions slightly more often than Random greedy. By definition of the heuristics, all solutions of Greedy and Random greedy are local minima. This is different for Simulated Annealing: most but not all of the solutions that have been found in the best part of the fitness landscape are local minima. The reason is that Simulated Annealing might end in the basin of attraction of a local minimum which is worse than the best solution that was found during the run. The visualization shows that nearly every of the solutions in the best part of the front was the best solution in at least one of the 10000 runs of Simulated Annealing. As a consequence it might be reasonable to improve Simulated Annealing by performing one run of Greedy on the best solution found by Simulated Annealing. Of course this makes the algorithm slightly slower. The user can see from the visualization that this would improve a significant part of the good solutions that have been found.

As a final note, we want to mention that the proposed visualization method can not be directly used for large problem instances, for the simple reason that there exists too many solutions. However similar ideas can be applied to a sample of the set of all solutions. It will be interesting, to develop proper sampling techniques on which such a visualization can be based. For the case of RNA folding landscapes, which were the original motivation to introduce the barrier tree data, this has been done very recently in [5].

5 Conclusions

Some extensions of the visualization approach that is used by the visualization tool dPSO-Vis ([11]) have been proposed. Examples are: i) barrier lines are introduced that connect a valley of the landscape with a barrier, ii) local minimum nodes can be marked in the barrier landscape, and iii) the fitness of best neighbour nodes are shown. Also, for a visualization of the optimization behaviour of heuristics the solutions that it has found in one or more runs can be marked. The size of such a mark is proportional to how often the corresponding solution has been found. For the example of a small instance of the Travelling Salesperson Problem (TSP) and the three heuristics Greedy, Random greedy, and Simulated Annealing, it was shown how the introduced visualization method can help for the analysis of the fitness landscapes and for understanding the optimization behaviour of the heuristics. Clearly, some of the conclusions that were drawn from the visualization could also, at least in principle, be found by other methods. However, it was argued that the proposed visualization can easily give an approximate quantitative and detailed insight which is often difficult to get by other methods.

The proposed visualization method can be applied in principle to any combinatorial optimization problem. The only requirement is that a neighbourhood relation between solutions can be defined that is used by the heuristics to create new solutions. Future work is to extend the proposed visualization methods to sampled solution sets in order to be able to apply it to larger problem instances.

Acknowledgement. This work has been supported by the European Social Fund (ESF) and the Free State of Saxony (ESF App.No. 100098248 and App.No. 100098251).

References

1. Flamm, C., Hofacker, I.L., Stadler, P.F., Wolfinger, M.T.: Barrier Trees of Degenerate Landscapes. Z. Phys. Chem. 216, 1–19 (2002)
2. Halim, S., Yap, R.H.C.: Designing and Tuning SLS Through Animation and Graphics: An Extended Walk-Through. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2007. LNCS, vol. 4638, pp. 16–30. Springer, Heidelberg (2007)

3. Halim, S., Yap, R.H.C., Lau, H.C.: Viz: A Visual Analysis Suite for Explaining Local Search Behavior. In: Proc. 19th Annual ACM Symposium on User Interface Software and Technology (UIST 2006), pp. 57–66 (2006)
4. Halim, S., Yap, R.H.C., Lau, H.C.: Search Trajectory Visualization for Analysing Trajectory-Based Meta-Heuristic Search Algorithm. In: Proc. European Conference on Artificial Intelligence, pp. 703–704 (2006)
5. Kucharík, M., Hofacker, I., Stadler, P.F., Qin, J.: Basin Hopping Graph: a computational framework to characterize RNA folding landscapes. Bioinformatics 30(14), 2009–2017 (2014)
6. Oesterling, P., Heine, C., Jänicke, H., Scheuermann, G., Heyer, G.: Visualization of high-dimensional point clouds using their density distribution's topology. IEEE Transactions on Visualization and Computer Graphics 17(11), 1547–1559 (2011)
7. Pérez, J., Mexicano-Santoyo, A., Santaolaya, R., Alvarado, I.L., Hidalgo, M.A., De la Rosa, R.: A visual tool for analyzing the behavior of metaheuristic algorithms. International Journal of Combinatorial Optimization Problems and Informatics 3(2), 31–43 (2012)
8. Pérez, J., Mexicano-Santoyo, A., Santaolaya, R., Alvarado, I.L., Hidalgo, M.A., De la Rosa, R.: A Graphical Visualization Tool for Analyzing the Behavior of Metaheuristic Algorithms. International Journal of Emerging Technology and Advanced Engineering 3(4), 32–36 (2013)
9. Richter, H., Engelbrecht, A. (eds.): Recent Advances in the Theory and Application of Fitness Landscapes. Springer Series Emergence, Complexity and Computation, vol. 6, pp. 487–507 (2014)
10. Volke, S., Bin, S., Zeckzer, D., Middendorf, M., Scheuermann, G.: Visual Analysis of Discrete Particle Swarm Optimization using Fitness Landscapes. In: Richter, H., Engelbrecht, A. (eds.) Recent Advances in the Theory and Application of Fitness Landscapes. Series Emergence, Complexity and Computation, vol. 6, pp. 487–507. Springer (2014)
11. Volke, S., Middendorf, M., Hlawitschka, M., Kasten, J., Zeckzer, D., Scheuermann, G.: dPSO-Vis: Topological Visualization of Discrete Particle Swarm Optimization. Computer Graphics Forum 32(3), 351–360 (2013)