



Search Trajectory Networks of Population-Based Algorithms in Continuous Spaces

Gabriela Ochoa¹, Katherine M. Malan², and Christian Blum³

¹ University of Stirling, Scotland, UK
gabriela.ochoa@stir.ac.uk

² Department of Decision Sciences, University of South Africa, Pretoria, South Africa
malankm@unisa.ac.za

³ Artificial Intelligence Research Institute (IIIA-CSIC), Campus of the UAB,
Bellaterra, Spain
christian.blum@iiia.csic.es

Abstract. We introduce *search trajectory networks* (STNs) as a tool to analyse and visualise the behaviour of population-based algorithms in continuous spaces. Inspired by local optima networks (LONs) that model the global structure of search spaces, STNs model the search trajectories of algorithms. Unlike LONs, the nodes of the network are not restricted to local optima but instead represent a given state of the search process. Edges represent search progression between consecutive states. This extends the power and applicability of network-based models to understand heuristic search algorithms. We extract and analyse STNs for two well-known population-based algorithms: particle swarm optimisation and differential evolution when applied to benchmark continuous optimisation problems. We also offer a comparative visual analysis of the search dynamics in terms of merged search trajectory networks.

Keywords: Continuous optimisation · Local optima networks · Metaheuristics behaviour · Search trajectory networks

1 Introduction

There is a lack of tools for understanding the dynamics of heuristic search algorithms and the global structure of fitness landscapes. It is also difficult to visualise high-dimensional search spaces. Local optima networks (LONs) [15, 24] help to fill this gap by providing a compressed model of landscapes, where nodes are local optima and edges possible transitions among them. LONs model the distribution and connectivity pattern of local optima, and thus help to characterise the underlying landscape global structure. Once a network model has been constructed, it can be visualised and analysed with the plethora of powerful analytical and visualisation tools provided by the science of complex networks [14]. However, there are limitations to LONs, as they have been applied mainly to

fully enumerated networks [10,15,24], combinatorial optimisation in the context of single-point metaheuristics [16,22] or population-based approaches where there is a local search component [4,23]. The major limitation of LONs is that the nodes have been restricted to local optima of standard neighbourhood operators, and only recently have been extended to large-neighbourhoods within hybrid metaheuristics [1]. The contributions of this article are as follows:

- To propose search trajectory networks (STNs) as a tool to analyse and visualise the behaviour of population-based algorithms.
- To show how the concept of STNs could be implemented in the context of continuous search spaces.
- To conduct STN analyses of two well-known evolutionary algorithms in continuous optimisation.
- To conduct a visual comparative analysis of the studied algorithms using merged STNs.

2 Population-Based Algorithm Behaviour

In the last few decades there has been a huge increase in the number of metaheuristics inspired by different natural and social phenomena. One of the problems with all of these new algorithms is that it is not clear whether they are in fact “new”. When the metaphor is stripped away, is the search process any different from the search process of existing established algorithms [20]?

Combined with this increase in the choice of algorithms is the lack of expert knowledge required to use the algorithms effectively. It has taken many decades of empirical and theoretical research for well-established metaheuristics to be understood even to a very limited extent. Every new approach comes with a blank record of established knowledge around behaviour with respect to algorithm setup, parameter choices and suitable or unsuitable problem classes. A need clearly exists for approaches to analysing search algorithm behaviour.

It is often stated that the success of any metaheuristic boils down to finding the right balance between exploration and exploitation (or the broader concept of intensification/diversification [2]). However, there is no generally accepted understanding of the concept in the evolutionary computing research community [8]. Part of the problem is that controlling this aspect of algorithmic behaviour is not trivial. In the case of evolutionary algorithms, there are three levels at which exploration/exploitation can be controlled [8]: at individual level (when solutions share information with each other), at sub-individual level (when solutions are combined) and at gene level (when components of individual solutions are modified).

In population-based algorithms, exploration/exploitation is related to the notion of diversity. The more diverse or spread-out the solutions in the population are, the more the algorithm is exploring. Conversely, if the solutions are clustered closely together in the search space, then the algorithm is exploiting a specific part of the search space. The way in which diversity changes over

time is therefore one approach to characterising the behaviour of a population-based algorithm. Bosman and Engelbrecht [3] proposed a single numerical measure called diversity rate of change (DRoC) for characterising the exploration-exploitation trade-off in particle swarms. Their premise was that the profile of the reduction in diversity (measured using the average Euclidean distance around the centre of the swarm [17]) could be captured by the slopes of a two-piecewise linear approximation of the diversity over time. Although diversity provides one important view of algorithm behaviour, it ignores where in the search space the population is moving and hence whether convergence is premature or not.

The STN model proposed in this paper provides a complementary view of the behaviour of population-based algorithms. Instead of studying the diversity of the population over time, we study the trajectory of a representative solution (the current best solution) over time. The visualisations and metrics of STNs provide an additional tool for analysing algorithm behaviour that may provide insights that are not captured by commonly used convergence plots or the DRoC measure.

3 Search Trajectory Networks (STNs)

Our proposal to provide new insights into the search dynamics of different algorithms is to model their search trajectories by means of a network model. Specifically, we modify the local optima networks (LON) model [15] to analyse the trajectories of population-based algorithms. We also use the recently proposed idea [1] of merging the network models induced by two different algorithms in order to compare their trajectories with a graphical support.

In order to define a network model, we need to specify the nodes and edges. The relevant definitions are given below, as well as a description of the sampling process to construct the network models.

Trajectory. A sequence of solutions corresponding to the best solution in the population over time. The frequency of recording the best solution is controlled by a parameter.

Location. A partition of the search space containing a subset of solutions within a predefined neighbourhood.

Nodes. The nodes correspond to the locations of the corresponding best solutions in a trajectory. The set of nodes is denoted by N .

Edges. Edges are directed and connect two consecutive locations of best solutions in the search trajectory. Edges are weighted with the number of times a transition between two given nodes occurred during the process of sampling and constructing the STN. The set of edges is denoted by E .

Search Trajectory Network (STN). Is the directed graph $G^{\text{STN}} = (N, E)$, with node set N , and edge set E as defined above.

Sampling and STN Model Construction. The STNs were generated for a selection of well-studied benchmark instances. For each of these instances an STN was constructed by aggregating all the unique nodes and edges encountered across 10 independent runs (search trajectories) of each algorithm. The details of the sampling and STN setup are given in Sect. 4.4.

4 Experimental Setting

This section describes the algorithms and problems used in the experiments as well as the setup required for the STN model.

4.1 Candidate Algorithms

Two candidate population-based algorithms were chosen for testing the proposed STN model – one evolutionary, namely differential evolution (DE), and one swarm-based, namely particle swarm optimisation (PSO). The particular version of DE used in the study was DE/rand/1 [21], with uniform crossover, a population size of 50, a scale factor of 0.5, and a crossover rate of 0.5. The version of PSO used in the study was traditional global best PSO [6, 12] with an inertia weight term [19], 50 particles, 1.496 for both the cognitive and social acceleration constants, and 0.7298 for the inertia weight (although the optimal choice of parameters is problem dependent, this is a common choice that works reasonably well for many problems [7]).

Note that no parameter tuning was performed on the two algorithms. The purpose of this paper is to introduce a mechanism for understanding and contrasting population-based algorithm behaviour in continuous spaces. No judgements are made on the performance of DE in relation to PSO. We are rather showing that if one search process is more successful than another on a particular problem, the STN model can shed some light on why this is the case.

The behaviour of the two candidate algorithms can be understood on a high level as follows:

- DE/rand/1: At each iteration, new perturbed solutions are formed for each individual of the population by adding a scaled weighted difference between two other random solutions in the population to another random solution. A trial solution is then formed through crossover of the current solution with the perturbed solution. If the trial solution is better, then it will replace the current solution. On a high level DE/rand/1 can be understood as each solution being attracted towards a combination of three other random solutions of the population, but only moving if the new combination is better.
- Global best PSO: At each iteration, the position of each solution is influenced by three terms: the solution’s previous velocity, the position of the best solution in the individual trajectory and the position of the best solution of the population. All solutions therefore have two attractors: the current best solution in the population and the previous best solution in the individual’s trajectory. The solution will move regardless of whether it is better or not.

A significant difference between the two algorithms is in the size of the neighbourhood for sharing information. In the case of DE/rand/1, the neighbourhood is three other individuals, whereas for global best PSO, the neighbourhood is the entire population, since all solutions are attracted towards one global best solution.

4.2 Benchmark Functions

A sample of five minimisation benchmark functions (defined in Table 1) with different characteristics were chosen for testing the proposed STN model. Function instances were chosen that demonstrated performance differences between the two candidate algorithms in our preliminary experiments. The purpose was to investigate whether the STN visualisations and metrics could be used to explain relative algorithm success and failure for problems in different dimensions. Two-dimensional plots of the functions are provided in Fig. 1 to provide insight into the global structure of the problems.

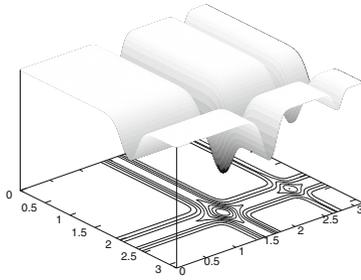
Quadric (also known as Schwefel 1.2) [25] is the only unimodal function. Michalewicz [13] is multimodal, but also has large plateaus at high fitness values. Schwefel 2.26 [25] is multimodal and also multi-funnelled. Both Salomon [18] and Rana [18] are extremely rugged, but Salomon has a single-funnel global structure (evident in Fig. 1d), whereas Rana has a multi-funnel structure. Figure 1e shows the Salomon function zoomed in to the domain around the origin (the global optimum), showing that the function “resembles a pond with ripples” [18] on a micro scale.

Table 1. Benchmark functions

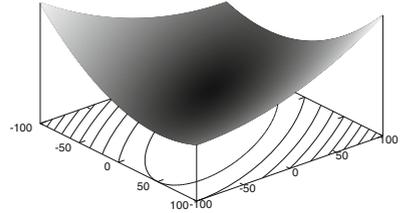
Function	Definition	Domain
Michalewicz	$f(\mathbf{x}) = -\sum_{i=1}^D \sin(x_i) (\sin(ix_i^2/\pi))^{2p}$	$x_i \in [0, \pi]$
Quadric	$f(\mathbf{x}) = \sum_{i=1}^D \left(\sum_{j=1}^i x_j \right)^2$	$x_i \in [-100, 100]$
Rana	$f(\mathbf{x}) = \sum_{i=1}^D x_i \sin(\alpha) \cos(\beta) \\ + (x_{(i+1) \bmod D} + 1) \cos(\alpha) \sin(\beta), \quad D \geq 2, \\ \alpha = \sqrt{ x_{i+1} + 1 - x_i }, \quad \beta = \sqrt{ x_i + x_{i+1} + 1 }$	$x_i \in [-512, 512]$
Salomon	$f(\mathbf{x}) = -\cos\left(2\pi \sum_{i=1}^D x_i^2\right) + 0.1\sqrt{\sum_{i=1}^D x_i^2} + 1$	$x_i \in [-100, 100]$
Schwefel 2.26	$f(\mathbf{x}) = -\sum_{i=1}^D \left(x_i \sin(\sqrt{ x_i }) \right)$	$x_i \in [-500, 500]$

4.3 Experimental Runs

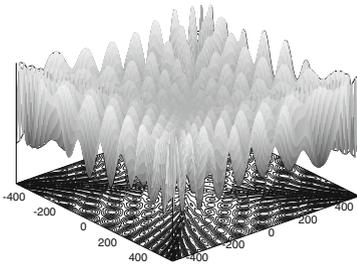
Ten runs of each of the two candidate algorithms (using the parameters specified in Sect. 4.1) were executed on the five benchmark problem instances: Michalewicz



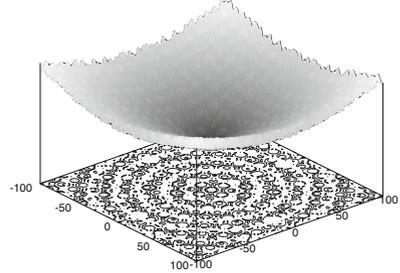
(a) Michalewicz



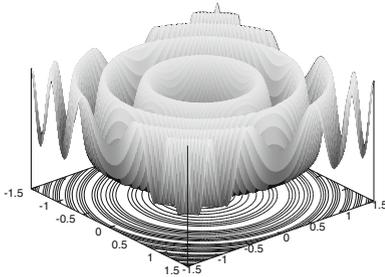
(b) Quadric



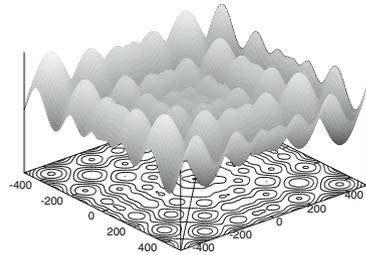
(c) Rana



(d) Salomon



(e) Salomon (smaller domain)



(f) Schwefel 2.26

Fig. 1. Plots of two-dimensional versions of the benchmark functions used in the experiments

in 5D, Quadric in 10D, Rana in 3D, Salomon in 3D and Schwefel 2.26 in 5D. Each run had a budget of $5000 \times D$ function evaluations ($100 \times D$ iterations for a population size of 50). The positions and fitness values of the best solutions in the current population were saved to represent the trajectories.

To ensure that the visualisations were not too cluttered, the best solutions of every D^{th} iteration were stored. Problems of different dimensions therefore

had representative trajectories of equal length (100 in this case). The fitness of solutions was stored rounded off to a precision of 10^{-8} .

4.4 STN Setup

In a discrete domain, nodes of the STN could be modelled as unique solutions. In the continuous domain, however, considering unique solutions is not feasible, so each node instead represents a number of solutions that are within a specified non-overlapping sub-space of the solution space, which we call a *location*. A location can therefore be thought of as a small portion of the search space through which trajectories might pass. Each solution in the trajectory is represented by one node in the STN, but the same node may represent multiple solutions (meaning that the trajectories passed through the location multiple times).

In this study, a solution precision parameter (SP) was used to portion the continuous search space into equal-sized discrete portions equal to a hypercube with length 10^{-SP} . For example, if $SP = 2$, then the solution space is divided into hypercubes of size 10^{-2} and each node in the STN is equivalent to one of these hypercubes or locations. Solutions are mapped to locations by rounding off all components of the position to the nearest 10^{-SP} to determine the identity of the enclosing hypercube.

To extract meaningful insights from the STN model, the value of the parameter SP should decrease as the search space increases. In this study, (assuming that the domain of values is the same for each dimension of the problem) the value for SP was expressed as a function of the range of the domain ($x_{max} - x_{min}$) and dimension (D) of the problem as follows: SP is set to $2 - n$, where n is the largest integer for which the following is true:

$$(x_{max} - x_{min}) \times D \geq 10^n. \quad (1)$$

For example, given a problem in 3 dimensions with domain $[-1, 1]$ in all dimensions, SP would be set to 2, since $2 \times 3 \geq 10^0$, so $n = 2 - 0 = 2$. For this problem, a location/node in the STN would be equivalent to a unique $10^{-2} \times 10^{-2} \times 10^{-2}$ cube in the search space.

Since each location comprises multiple solutions, there are many different fitness values for each location. For visualisation purposes, each location in the trajectory was assigned a representative fitness value equal to the minimum fitness value of all visited solutions within that location.

5 Results

5.1 Visualisation

Visualisation is a powerful tool for network data analysis, allowing us to appreciate structural features difficult to infer from the raw data and statistical analysis. The network visualisations in Figs. 2 and 3 model the search trajectories

traversed by 10 independent runs of both candidate algorithms on the benchmark instances. Plots were produced with the R statistics package, using graph layout methods implemented in the `igraph` library [5]. Specifically, we considered *force-directed* layout algorithms, such as Fruchterman-Reignold [9] and Kamada-Kawai [11]. Force-directed layout algorithms are based on physical analogies and do not rely on any assumptions about the structure of the networks. These algorithms strive to satisfy the following generally accepted criteria [9]:

- Vertices are distributed roughly evenly on the plane (a circle in the `igraph` implementation).
- The number of crossing edges is minimised.
- The lengths of edges are approximately uniform.
- The inherent symmetries in the networks are respected, i.e., sub-networks with similar inherent structure are usually laid out in a similar manner.

The left plots in Figs. 2, 3 and 4, show the STNs 2D force-directed layouts, while the right plots give a 3D visualisation of the same layouts where the z coordinate indicates the fitness values. In the 3D plots, locations appearing lower in the plot have better fitness as we are dealing with minimisation problems. The features of the nodes and edges in all the STN visualisations (Figs. 2, 3 and 4) reflect properties of the search dynamics. The size of the nodes is proportional to their incoming degree (number of incoming edges), which indicates how often a node was visited and thus ‘attracts’ the search process. The nodes and edges visited by only one of the two algorithms are distinguished in different colours; light orange for PSO, and blue for DE. The initial locations for all runs are visualised as yellow nodes. Both algorithms started from the same 10 randomly generated initial solutions, so each yellow node has one blue (DE) and one orange (PSO) outgoing edge. Red nodes illustrate the location of the global optimum. Nodes in green indicate locations that were visited (shared) by both algorithms in their combined search trajectories, while dark grey nodes represent the end point of search trajectories, i.e. where the final location was not the location containing the global optimum. A visualisation legend summarising the colours used in Figs. 2, 3 and 4 is given in Fig. 2(a).

The STN visualisation of the Salomon function (Fig. 3(b)) appears crowded towards its centre. In order to have a clearer perspective, Fig. 4 shows a zoomed view near the global optimum. Specifically, the plots in Fig. 4 visualise the sub-graph containing the nodes which are within the first quantile in fitness value; that is, the best 25% of the set of nodes.

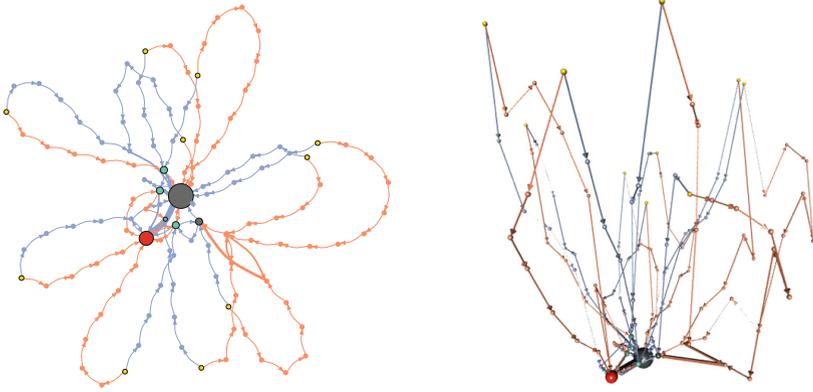
5.2 Structural and Performance Metrics

Table 2 reports the following STN metrics for each algorithm:

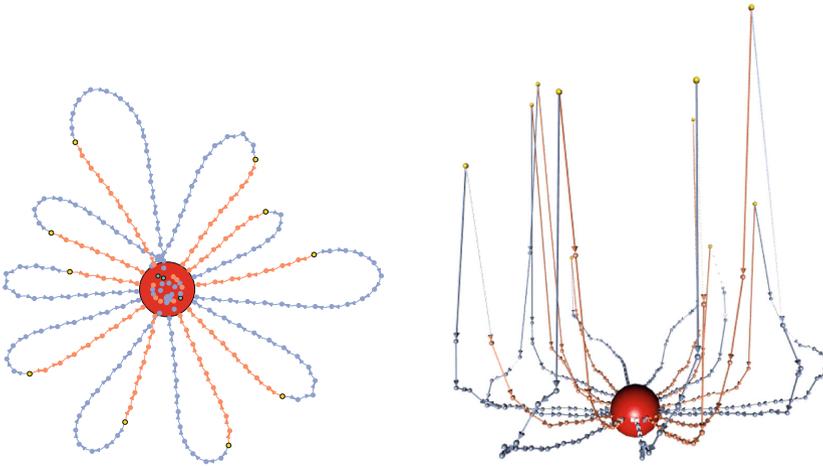
- nodes: The total number of nodes, which corresponds to the number of unique locations visited.
- edges: The total number of edges, which corresponds to the number of unique search transitions between locations.

- Locations visited by PSO
- Search transitions by PSO
- Locations visited by DE
- Search transitions by DE
- Locations visited by both algorithms
- Locations at the start of runs
- Locations at the end of runs
- Location of the global optimum

(a) Visualisation legend.



(b) Michalewicz function $D = 5$, $SP = 1$.



(c) Quadric function $D = 10$, $SP = -1$.

Fig. 2. Merged STN visualisations for functions 1 and 2. The visualisation legend is shown at the top. The dimension D , and solution precision SP are indicated in the captions. The left plots use a force-directed layout, while the right plots use the same layout and adds a 3rd dimension indicating fitness. (Color figure online)

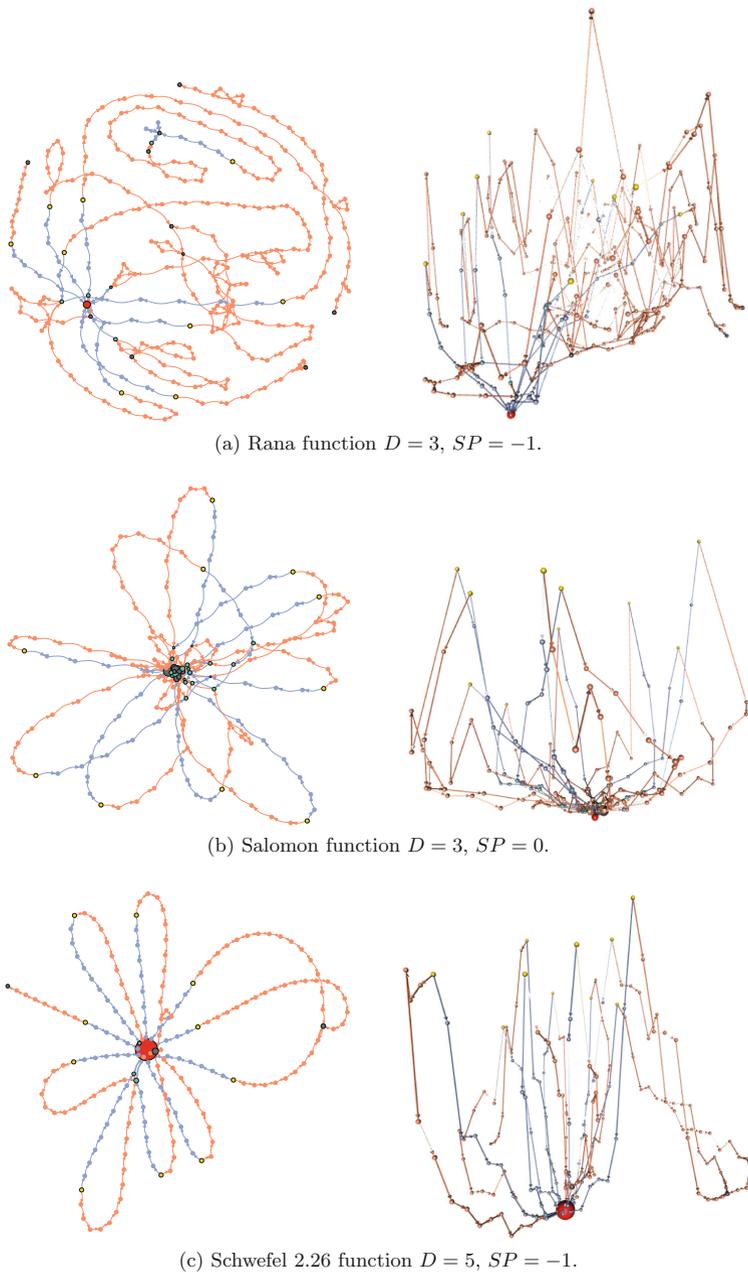


Fig. 3. Merged STN visualisations for functions 3, 4 and 5. The dimension D , and solution precision SP used for each function are indicated in the captions. For each function, the left plot uses a force-directed layout, while the right plot considers the same layout and adds a 3rd dimension indicating fitness. (Color figure online)

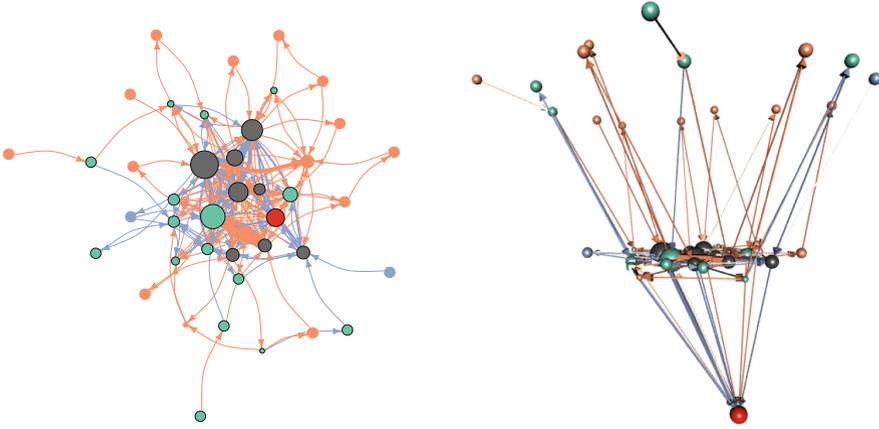


Fig. 4. Zoomed merged STN visualisation for the Salomon function. The plots show only those nodes within the first quantile of fitness values (i.e. the best 25% solutions). The left plot uses a force-directed layout, while the right plot considers the same layout and adds a 3rd dimension indicating fitness. (Color figure online)

- ends: The number of (unique) nodes corresponding to the end of runs. These include both the global optimum and other locations to which the runs converged.
- avg. path length: Average (with standard deviation) path length from start to end nodes. The length of a path is the number of edges it contains.

The table also reports the success rate of the algorithms as a percentage of the 10 runs that ended in the global optimum location. For example, in the case of DE on the Michalewicz problem instance, 8 out of the 10 runs reached the node that contains the global minimum.

The values in Table 2 indicate that the PSO trajectories are generally longer than those of DE, with the exception of the Quadric function. This means that the best solution in the PSO population traversed through more nodes than the best solution in the DE population. This could be as a result of the absence of elitism in the PSO implementation, resulting in more exploration than DE (that did use elitism), except in the unimodal function (Quadric) where the single global best attractor resulted in faster convergence to the global optimum.

Table 3 reports network metrics for the merged STNs that include the trajectories of both DE and PSO. The columns report, from left to right, the total number of nodes, the total number of edges, the number of end nodes, and the number of *shared* nodes, that is, locations visited by both algorithms. Remember that both algorithms start from the same 10 initial solutions, so the 10 start nodes are always shared. The global optimum is also a shared visited node in most functions, except Rana where PSO was not able to converge to it. Table 3 indicates that other search points are shared between the algorithms in all functions, with Salomon having the largest number of shared nodes.

Table 2. STN structural metrics and success rate of each algorithm.

	Nodes		Edges		Ends		Avg. path length		Success (%)	
	DE	PSO	DE	PSO	DE	PSO	DE	PSO	DE	PSO
Michalewicz	61	76	71	81	2	3	7.0 _{1.88}	7.5 _{2.63}	80	40
Quadric	154	93	160	92	1	1	15.1 _{3.51}	9.2 _{1.55}	100	100
Rana	49	211	54	249	2	10	4.7 _{0.95}	13.5 _{7.22}	90	0
Salomon	85	131	129	221	3	7	8.3 _{1.21}	10.4 _{2.08}	80	30
Schwefel 2.26	81	138	80	143	1	4	8.0 _{2.11}	12.6 _{3.23}	100	50

Table 3. Merged STN structural metrics.

	Nodes	Edges	Ends	Shared
Michalewicz	120	151	2	17
Quadric	233	252	1	14
Rana	245	303	10	15
Salomon	178	338	3	38
Schwefel 2.26	205	220	4	14

Metrics on merged STNs, such as those given in Table 3, can be used to provide insight into the nature of problems in relation to each other. For example, in the case of the Michalewicz function there are far fewer nodes than for the other algorithms (recall that all recorded trajectories were of equal length). This could be due to the neutrality present in the function (see Fig. 1a), resulting in less movement of the best solution, due to lack of information in the landscape.

5.3 Performance Comparison

In the following we analyse the behaviour of the two candidate algorithms on the basis of the graphics from Figs. 2, 3 and 4 and the metrics from Tables 2, 3 and 4. Remember that both the graphics and the metrics were generated on the basis of 10 runs per algorithm and per function.

Michalewicz Function. The STN concerning the Michalewicz function (see Fig. 2(a)) shows two important attractors: the global minimum (red dot) and the large grey dot. In addition, smaller attractors such as the smaller grey dot can be identified. The success rate of DE for this problem is 80%, while the one for PSO is 40%. The graphics clearly show that, although both algorithms are attracted by the portion of the search space representing the large grey dot, DE is much better than PSO at escaping from this basin of attraction (see the thick blue edge to the global optimum node). Moreover, note that a number of PSO runs are attracted by the smaller grey dot, but that none of these runs is able to escape from there. The opposite is the case for DE: although two runs are

attracted by the smaller grey dot, the algorithm is able to leave from there and to proceed with the search. Finally, note that the search trajectories of PSO can be seen to be generally longer than those of DE (also reflected in the average path length in Table 2).

Quadric Function. The STN obtained for the Quadric function shows that all runs—for both DE and PSO—are attracted by the global minimum. This is reasonable, because this function does not have any local minima. Moreover, with respect to the considered precision, all runs of PSO and DE finally reach the location containing the global minimum (success rate of 100% – see Table 2). However, consulting Table 4 we can see that the DE runs finish, on average, further from the global minimum than the PSO runs. This indicates that DE suffers from premature convergence in the case of function Quadric. Interestingly, the search trajectories of DE (in terms of the number of steps) are clearly longer than those of PSO (visually and seen in Table 2) indicating that PSO took a more direct route in the direction of the global optimum.

Rana Function. Already a first visual inspection of the STN for the Rana function (see Fig. 3(a)) indicates that this function is very different from the Michalewicz and Quadric functions. This is also evident in the merged metrics of Table 3, showing more unique nodes than for all other functions. In fact, PSO does not seem to be attracted by any particular part of the search space and can be seen to have 10 different end points (Table 2), compared to only 2 for DE. The normalised average distance from the end of the PSO trajectories to the global minimum is ≈ 0.5 (Table 4), which is about half the distance between the two most distant points in the search space. This divergent behaviour is also reflected in the low success rate – no PSO runs end in the location of the global minimum, while DE has a success rate of 90%, that is, nine out of 10 runs end in the global minimum node. Moreover, the trajectories of DE are rather short and seem to move towards the global minimum without suffering from too many detours (also reflected in the low average path length in Table 2). On the opposite, PSO comes often back to the same solutions (portions of the search space), which is indicated by directed cycles in the STN.

Salomon Function. The Salomon function is another example of a function that is characterised by a big valley structure (like Quadric, for example). However, instead of being smooth, there are many ripples (see Fig. 1(d) and Fig. 1(e)). In each of these ripples we find numerous local minima of the same quality. The STN for this function (see Fig. 3(b)) nicely shows that both algorithms move rather quickly towards the global minimum. Then, however, they get often stuck in the last or in the second-last ripple. In fact, note that DE has a success rate of 80%, while PSO has a success rate of 30% (Table 2). The fact that both algorithms have difficulties in overcoming the last ripple before reaching the global minimum is shown in a zoomed visualisation of the merged STN shown in Fig. 4. Note that some of the green and grey dots located on that ripple are actually larger in size

Table 4. Normalised average distances of the end of the search trajectories from the global minimum of each considered function. The value of the algorithm that, on average, finishes closer to the global minimum is indicated in bold font.

Algorithm	Benchmark function				
	Michalewicz	Quadric	Rana	Salomon	Schwefel 2.26
DE	2.305e-02	5.702e-06	8.080e-02	6.645e-04	4.637e-08
PSO	6.426e-02	7.947e-14	5.058e-01	2.020e-03	1.752e-01

than the red dot representing the global minimum. This means that these nodes have more incoming edges than the global minimum node.

Schwefel 2.26 Function. From the 2D graphics of the considered functions in Fig. 1 it can be seen that Schwefel 2.26 is somewhat related to Rana, in the sense that there are rather high-quality basins of attraction scattered all over the search space. This observation goes in line with the fact that the end of the PSO trajectories are, on average, again much further away from the global minimum than the ones of DE (see Table 4). Moreover, studying the STN graphics from Fig. 3(c) it can be observed that there are four of the 10 PSO runs that converge to basins of attraction rather far away from the global minimum. Interestingly, three of these four runs are attracted by the same basin of attraction. Moreover, as in the case of Rana, the PSO trajectories are—in terms of the number of steps—longer than those of DE.

As a general conclusion we might say that for those functions with rugged landscapes and high-quality solutions scattered all over the search space, an algorithm with elitism and multiple attractors (as with DE/rand/1) seems to be more successful than an algorithm without elitism and a shared global attractor (as with global best PSO).

6 Conclusion

We proposed a network-based model to characterise and visualise the search behaviour of population-based metaheuristics: search trajectory networks (STNs). We tested the model by studying the search process of two well-known algorithms (DE and PSO) when optimising a set of continuous benchmark functions with different characteristics and dimensions. Our analysis illustrates that the qualitative (visualisations) and quantitative (network metrics) analysis of STNs give insight into the convergence behaviour of algorithms and their performance differences. STNs allow us to observe and quantify which portions of the search space attract the process and thus act as traps in the way of locating the global optimum. We can also identify frequently traversed areas of the search space by a given algorithm or pair of algorithms. We argue that this information gives new insights in understanding the dynamics of metaheuristics, and thus

can be used to improve their design and to inform the selection of the most suitable algorithm for a given problem.

Future work will generalise the model to combinatorial optimisation and other metaheuristics, and will analyse real-world optimisation problems. We will also explore further case scenarios where intriguing performance differences among algorithms are observed, which can potentially be clarified with our proposed analysis.

References

1. Blum, C., Ochoa, G.: A comparative analysis of large neighborhood search and construct, merge, solve & adapt by means of merged local optima networks (submitted)
2. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization. *ACM Comput. Surv.* **35**(3), 268–308 (2003)
3. Bosman, P., Engelbrecht, A.P.: Diversity rate of change measurement for particle swarm optimisers. In: Dorigo, M., et al. (eds.) ANTS 2014. LNCS, vol. 8667, pp. 86–97. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-09952-1_8
4. Chicano, F., Whitley, D., Ochoa, G., Tinós, R.: Optimizing one million variable NK landscapes by hybridizing deterministic recombination and local search. In: Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 753–760. ACM (2017)
5. Csardi, G., Nepusz, T.: The igraph software package for complex network research. *Int. J. Complex Syst.* **1695**, 1–9 (2006)
6. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science, pp. 39–43 (1995)
7. Eberhart, R., Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 84–88 (2000)
8. Eiben, A.E., Schippers, C.A.: On evolutionary exploration and exploitation. *Fundam. Inform.* **35**(1–4), 35–50 (1998)
9. Fruchterman, T.M.J., Reingold, E.M.: Graph drawing by force-directed placement. *Softw. Pract. Exper.* **21**(11), 1129–1164 (1991)
10. Herrmann, S., Ochoa, G., Rothlauf, F.: Pagerank centrality for performance prediction: the impact of the local optima network model. *J. Heuristics* **24**(3), 243–264 (2018)
11. Kamada, T., Kawai, S.: An algorithm for drawing general undirected graphs. *Inf. Process. Lett.* **31**(1), 7–15 (1989)
12. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1942–1948 (1995)
13. Mishra, S.K.: Performance of repulsive particle swarm method in global optimization of some important test functions: a Fortran program. Technical report, Social Science Research Network (SSRN), August 2006
14. Newman, M.E.J.: *Networks: An Introduction*. Oxford University Press, Oxford (2010)
15. Ochoa, G., Tomassini, M., Verel, S., Darabos, C.: A study of NK landscapes’ basins and local optima networks. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 555–562. ACM (2008)

16. Ochoa, G., Veerapen, N.: Deconstructing the big valley search space hypothesis. In: Chicano, F., Hu, B., García-Sánchez, P. (eds.) *EvoCOP 2016*. LNCS, vol. 9595, pp. 58–73. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30698-8_5
17. Olorunda, O., Engelbrecht, A.P.: Measuring exploration/exploitation in particle swarms using swarm diversity. In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE, June 2008
18. Price, K.V., Storn, R.M., Lampinen, J.A.: Appendix A.1: Unconstrained uni-modal test functions. In: Price, K.V., Storn, R.M., Lampinen, J.A. (eds.) *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series, pp. 514–533. Springer, Berlin (2005). <https://doi.org/10.1007/3-540-31306-0>
19. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, pp. 69–73 (1998)
20. Sörensen, K.: Metaheuristics-the metaphor exposed. *Int. Trans. Oper. Res.* **22**(1), 3–18 (2013)
21. Storn, R., Price, K.: Minimizing the real functions of the ICEC'96 contest by differential evolution. In: *Proceedings of the International Conference on Evolutionary Computation*, pp. 842–844 (1996)
22. Thomson, S.L., Ochoa, G., Verel, S.: Clarifying the difference in local optima network sampling algorithms. In: Liefoghe, A., Paquete, L. (eds.) *EvoCOP 2019*. LNCS, vol. 11452, pp. 163–178. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-16711-0_11
23. Veerapen, N., Ochoa, G., Tinós, R., Whitley, D.: Tunnelling crossover networks for the asymmetric TSP. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) *PPSN 2016*. LNCS, vol. 9921, pp. 994–1003. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45823-6_93
24. Verel, S., Ochoa, G., Tomassini, M.: Local optima networks of NK landscapes with neutrality. *IEEE Trans. Evol. Comput.* **15**(6), 783–797 (2011)
25. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **3**(2), 82–102 (1999)