# Algorithm Footprinting for Search Behaviour Characterisation

Irene Moser
*Department of Computer Science and Software Engineering*
*Swinburne University of Technology*
Melbourne, Australia
imoser@swin.edu.au

***Abstract*—Novel algorithms based on natural phenomena from insects to fish, birds, particles, genetics and physics are proposed regularly, and the boom shows no sign of abating. Typically, the introduction of a new method takes the form of a success story that reports outperformance of similar methods on a number of pertinent benchmark problems. As long as the 'inner workings' of stochastic heuristics remain a mystery, it is not possible to ascertain that a new natural paradigm also leads to a truly new search strategy. This paper makes a first attempt at characterising the algorithm 'footprint' by analysing the search steps that take an algorithm from a known to a previously unknown solution. The details of the discoveries of all new solutions are recorded as a description of the algorithm's search behaviour. Each record is considered a 'footprint' of the walk the heuristic undertakes through the search space. To identify patterns in the walk, the footprints are analysed using self-organising maps. The approach is evaluated on a scenario of differential evolution and particle swarm optimisation, as well as a hybrid of these searching two well-known benchmark functions. Unsupervised self-organising maps plausibly identify clusters of typical moves for both main algorithms. It also discovers that hybridisation leads to new types of moves not present in the algorithms involved in the hybridisation. In low dimensions, supervised self-organising map models correctly ascribe the search moves to the algorithm they belong to at least 50% of the time.**

***Index Terms*—nature-inspired, footprinting, algorithm behaviour, particle swarm, differential evolution**

## I. Introduction

Most nature-inspired stochastic methods are iterative: They create solutions by modifying previously discovered solutions, often those of higher fitness, and a degree of randomness to ensure significantly different information is discovered in the newly created solution. The natural paradigm guides the researcher in defining which solutions are used in the creation of a new solution, and how randomness is added. Most often a solution has a 'parent' that is similar to the 'child' (the new solution), and the strategy of the search can likely be read from the way a solution evolves from parent to child. We can regard these moves from base solution to new solution as the 'footprint' that defines the 'walk' of the algorithm through the decision and fitness spaces. Regardless of natural paradigm, the search strategy depends on the following decisions:

1) How to incorporate prior information in the creation of new (tentative) solutions, generally known as operators such as mutations (information from one existing solution) or crossovers (information from several solutions discovered earlier);
2) how to balance the components of this information, if multiple sources are used, and how to balance the aspects of information with randomness when creating new solutions (generally determined by tunable parameters);
3) what criteria should determine whether new information (most often in the form of solutions with a high level of fitness) is kept;
4) what dynamic aspect could optionally be introduced to vary the principles arising from decisions 1 to 3 over time.

The definition of most, if not all, nature-inspired algorithms can be reduced to these principles without losing information that would prevent an accurate implementation. For example, principle 1, is defined in particle swarm optimisation (PSO) with a star topology as using information from the globally best 'particle' (solution) and also information from the solution to be perturbed and the best solution discovered on the path to finding it. PSO employs inertia and other weights to define the tradeoffs between these aspects of information in principle 2, and principle 3 is defined as unconditional acceptance. Principle 4 is defined as static (no time line), except in a number of variations of PSO which tend to vary the inertia and balancing weights.

The postulation that natural paradigms do little to help clarify the operating principles of heuristic search algorithms is not new. Dower's thesis [1] investigates the hypothesis, "Evolutionary Algorithms are a single class of algorithms that no longer need the separation resulting from their distinct origins, and a unified model and approach will aid the understanding, development, implementation and presentation of these algorithms.", and introduces a description language to cover the operational principles. The current study asserts that the hypothesis can be extended to a larger number of nature-inspired algorithms, such as swarms, flocks and phenomena from physics such as simulated annealing, as all of them can be reduced to the four principles of heuristic design.

Knowing the exact behaviour of a successful algorithm, as compared to a less successful one, can help design better

heuristics in a more focussed way, without the detour of natural paradigms, and help improve the optimisation approaches we already have.

## II. Background

Significant progress has been made in the area of search landscapes of optimisation problems [2]. The ultimate goal is to determine groups of algorithms and groups of problems to be able to match them. Characterising fitness landscapes is the more straightforward of the tasks. Although the search landscape of a problem arises from the search algorithm, it is possible to define a 'base landscape' of a problem or problem instance through random walks [3]. Algorithm characterisation cannot easily be carried out without an application to a problem.

Traditionally, machine learning research has approached the problem of finding the best-performing algorithm for a problem from the point of view of algorithm selection [4]. In a 2008 survey, Smith-Miles coined the term of meta-knowledge and meta-learning [5], but mostly focuses on determining the properties of problems to move towards a more knowledge-based selection [6], whereas the bulk of portfolio-based algorithm selection relies on a meta-algorithm applying a heuristic [7].

More recently, researchers have begun to look into the algorithm selection and composition problem with deeper insights into search landscapes and algorithm behaviour characterisation [8]. A number of studies on bin packing have recently been published, for example Pérez et al. [9], who extracted characteristics of bin packing to determine their relationships with the performance of a number of heuristics. Quiroz et al. [10] studied the behaviour of a genetic algorithm at bin packing, extracting the five factors number of inflexion points, number of valleys, the average size of the valleys and dispersion using principal components analysis. Based on these insights, the authors were able to tune the algorithm to provide better results. Landero et al. [11] characterises Tabu Search on bin packing, built on this work, added autocorrelation, the number and variance of feasible solutions discovered to the descriptors of the 'operational behaviour' of the algorithm. It is clear that these measures are derived from the search landscape, which highlights the intertwined nature of search and problem characterisation.

All published algorithm characterisation approaches rely on abstract aggregate measures to describe algorithm behaviour. Eschewing summary measures, in the current study we examine the entire collection of footprints arising from a search, which gives us more detailed information about how the computational effort is spent and how the details of each algorithm differ.

## III. Methodology

To determine what an algorithm 'actually does' in order to find good solutions, the current study analyses the 'steps' ('moves') the algorithm takes from one solution to the next. These steps are assumed to be partly the same for all algorithms: Especially in the beginning of a search, few high-quality solutions are available to guide the search. Most searches will behave the same at this stage, picking new solutions almost entirely randomly. As more information becomes available, the search steps are expected to reflect the principles of the search. It is expected that the patterns of the moves vary between algorithms, because the performances clearly differ.

The search behaviours of three example algorithms are analysed and compared using two well-known benchmark functions, Schwefel's function and Giunta. As several variables are needed to describe a search move, a self-organising map (SOM) [12] is used to define groups of moves and their prevalence in each algorithm. A supervised SOM is applied to predict the algorithm based on the moves.

### A. Benchmark functions

For the exploratory footprint analysis, the generalised Giunta and Schwefel 1 functions were used, based on the findings by Lang and Engelbrecht [13]. The functions make part of a set that was found to represent diverse properties of benchmark functions, albeit as part of a larger suite of 24 functions. Giunta is two-dimensional, differentiable, separable, scalable and multimodal.

$$f(x) = 0.6 + \sum_{i=1}^{2} [\sin(\frac{16}{15}x_i - 1) + \sin^2(\frac{16}{15}x_i - 1) + \frac{1}{50}\sin(4(\frac{16}{15}x_i - 1))] \quad (1)$$

In the Giunta function (1), $-1 \leq x_i \leq 1$ and the global minimum is $x^* = (0.45834282, 0.45834282), f(x^*) = 0.060447$ [14]

Schwefel's first function is differentiable, partially separable, scalable and unimodal; it was set to 30 dimensions.

$$f(x) = \left(\sum_{i=1}^{D} x_i^2\right)^{\alpha} \quad (2)$$

In Schwefel's first function (2) $-100 \leq x_i \leq 100$ and the global minimum is $x^* = (0, ...0), f(x^*) = 0$ [14]. Alpha was set to the square root of $\pi$.

### B. Algorithms

Popular, generally accepted function optimisation algorithms with clearly different search approaches were used for this initial investigation. PSO and differential evolution (DE) both fit this description. As the purpose is to footprint the algorithms, a hybrid was included to investigate whether its moves could be identified by the method as belonging to either PSO or DE.

The comparison does not hinge on a performance competition and no attempts were made to optimise parameters; these were obtained from the literature. Actual search results are

only stated (in Section III-D) because they are needed for the discussion of the search behaviour.

Each algorithm was given a population of 30 and an iteration budget of 5000, or 150,000 function evaluations, leading to 150,000 footprints.

*1) Differential Evolution:* A standard implementation using a uniform random pick of target vector $x_3$ and differential vectors $x_1$ and $x_2$ was used. One of $D$ dimensions $d^* = r \times D$ was chosen uniformly randomly (using an uniform random number $r$ and rounding to the nearest integer) to ensure a change is applied to the current vector $x_i$

$$x_{i,d}(t+1) = \begin{cases} x_{3,d} + F(x_{1,d} - x_{2,d}), & \text{if } d^* = d \text{ or } r < p_{cr} \\ x_{i,d}(t), & \text{otherwise} \end{cases} \quad (3)$$

In (3), the differential weight $F = 0.8$ and crossover probability $p_{cr} = 0.5$ were set as reported by Omran, Engelbrecht and Salman [15]. In 30 dimensions, the crossover probability was set to zero on observing that higher perturbations led to very mediocre results. In spite of a lack of performance ambition in this study, there is little incentive in analysing the behaviour of substandard search algorithms.

The footprint collected assumed the position $x_i(t)$ as the base solution and $x_i(t+1)$ as the newly proposed solution. The value of 'kept' depended on whether the solution was accepted, which is equivalent to the fitness of the new solution being higher.

*2) Particle Swarm:* A star topology was used in a standard implementation using personal and global bests as well as an inertia weight.

$$v_{i,d}(t+1) = \omega v_{i,d}(t) + \phi_p r_p (x_{p,d} - x_{i,d}) + \phi_g r_g (g_d - x_{i,d}) \quad (4)$$

$$x_i(t+1) = x_i(t) + lr v_i \quad (5)$$

The inertia weight $\omega = 0.729$, the acceleration coefficients $\phi_p = 2.0412$ and $\phi_g = 0.9477$ in (4) and the learning rate $lr$ in (5) were set based on findings of Harrison, Ombuki-Berman and Engelbrecht [16].

The footprint collected assumed the particle position $x_i(t)$ as the base solution and $x_i(t+1)$ as the newly proposed solution. As PSO accepts solutions unconditionally, the value of 'kept' was always 1.

*3) SDEA-Hybrid:* Hendtlass [17] developed the swarm differential evolution algorithm (SDEA) as a simple combination of the two algorithms, adding a parameter that defines the proportion of DE moves. He found that any proportion between 25% and 50% DE moves improved the results. The parameter values used in [17] were not retained in the current study, instead the values were set as described in Sections III-B1 and III-B2 were used, so as not to render the PSO and DE moves unrecognisable in the hybrid. This eliminates the possibility of discovering new behaviour that is not due to hybridisation but parameter settings.

The footprint was collected as described for PSO and DE above depending on which algorithm was used.

### C. Search moves as data points

The challenge is to define a) the solution that counts as the 'base' or 'parent' of the new solution and b) what variables best describe the shift from one solution to the other. The choice of 'base' solution was discussed in the context of the algorithm used as examples, but the general principle to apply is to use the most 'similar' individual, in algorithms that draw on information from several individuals when probing for a new solution. Base solutions are easy to determine in the case of PSO or simulated annealing, which both move from one solution to another. In the case of evolutionary algorithms with crossover moves, a solution can have several parents as base solutions. It is possible to capture the moves from all individuals, or choose one. If several moves are used to capture the discovery of a new solution from several base solutions, single-solution moves such as mutations, if present in the same algorithms, might need to be captured multiple times to avoid inadvertent bias.

How to describe a move from one base solution to another? The position vector of both solutions could be used, but function optimisation is often carried out in a great number of dimensions, leading to a rather large number of descriptors to represent a single piece of information. Instead, the physical distance in decision space was used, as was the fitness distance. The iteration at which the sample was taken was also recorded. This information is significant, as search behaviour often changes over time [18]. Whether the new solution was kept is encoded in a binary variable. The raw fitnesses of base and new solutions were included as information about the general level of optimisation achieved. This leads to the footprint variables in Table I, which will be used in the results illustrations.

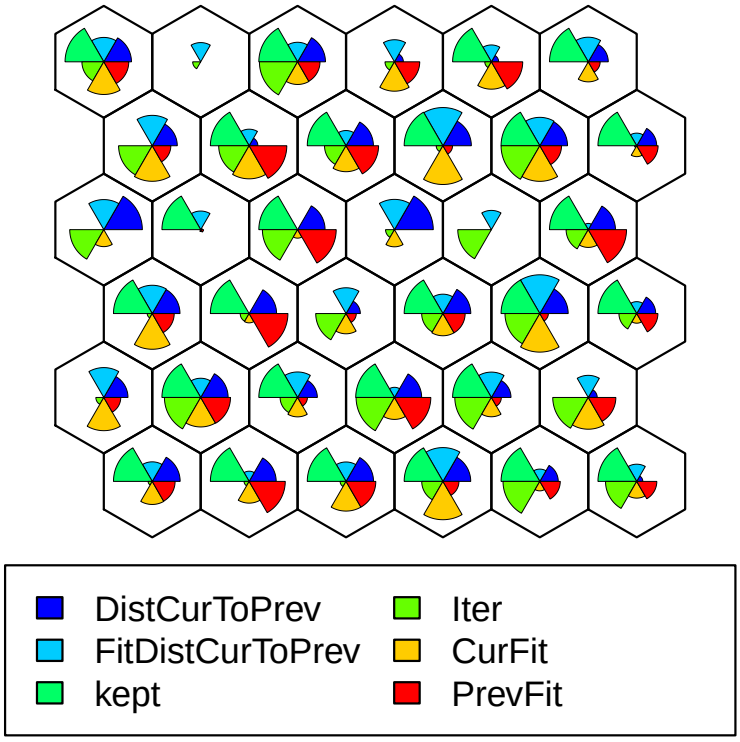
TABLE I: Content of a Footprint

| Variable | Definition |
|---|---|
| DistCurToPrev | Euclidean distance between base and new solution in vector space |
| FitDistCurToPrev | Fitness distance between base and new solution |
| kept | Binary variable, reflects whether the new solution was added to the population (1) or discarded (0) |
| Iter | Number of iterations at which the footprint move was recorded |
| CurFit | Fitness of the new solution |
| PrevFit | Fitness of the base solution |

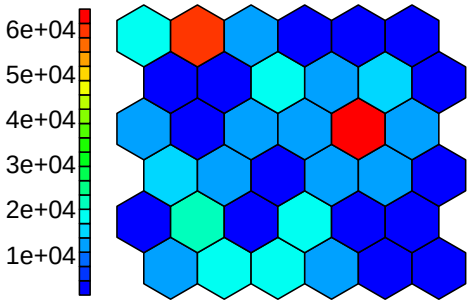
### D. Data Collection

The footprints were collected by running the algorithms and recording values for the variables in Table I from each newly proposed solution and its base solution.

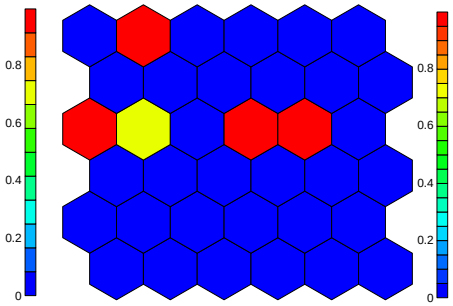
Table II lists the mean fitness of the best solution after 30 repeats of trials with 5000 iterations each, and the standard deviation in brackets. The differences are small indeed but for the sake of the discussion about how the algorithm progresses,



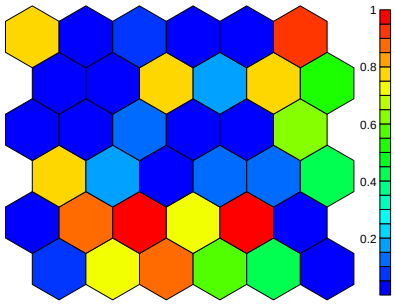

(a) Codes plot detailing combinations of footprint values



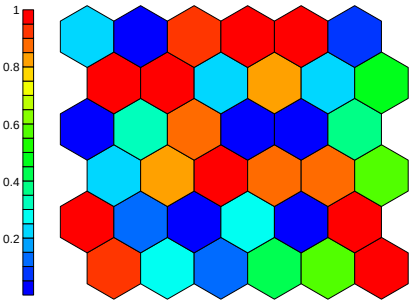

(b) Number of footprints that fall into each cluster




(c) Proportion of DE footprints




(d) Proportion of PSO footprints




(e) Proportion of SDEA footprints

Fig. 1: SOM for the footprint of the three algorithms searching the Giunta function in 2 dimensions.

TABLE II: Search Results by Algorithm

| | Giunta (2 dim) | Schwefel (30 dim) |
|---|---|---|
| DE | 0.6 (0) | 2.11E-89 (9.58E-89) |
| PSO | 0.601 (0.0015) | 1.14E-129 (6.23E-129) |
| SDEA | 0.60015 (0.00015) | 2.59E-80 (1.33E-79) |

they are included. Essentially, DE appears to lose in higher dimensions because its 'step size' seems too large for the local optimisation.

### *E. Self-organising maps*

Kohonen's SOM [12] are used to visualise highly dimensional data. It provides the means for unsupervised and supervised clustering using a neural network. In the supervised case, trained models can be used to classify new cases. In the current study, the Kohonen R package was used to analyse the DE, PSO and SDEA footprint data in an unsupervised manner on the Giunta and Schwefel functions, then the same data was used to train a model which was applied to a second footprint dataset on the Giunta function. All models were trained using a budget of 5000 iterations, with the learning rates between 0.05 and 0.01. All variables were normalised to avoid any scaling bias. For the supervised model, the algorithm names were added as a dependent variable in a factor columns.

The grid size (number of clusters) were initially determined based on the number of variables $n$. This allows for $n \times n$ combinations, assuming that a high and a low number of each variable can theoretically be accommodated. If the SOM algorithm finds fewer clusters, parts of the grid are shown with no occurrences. The number of clusters was then reduced until no clusters without occurrences was included.

## IV. Results and Discussion

Three sets of results are presented; an unsupervised model of the three algorithms' footprints solving Giunta, an equivalent model for Schwefel, and a supervised model of a Giunta optimisation with some results from the supervised model of Schwefel.

The SOM clusters are shown in hexagonal heat maps and matching heat maps for relevant details. To refer to each cluster in the discussion, the x and y coordinates are used with (1,1) defining the bottom left cluster. Although the algorithm name was not included in the clustering in the unsupervised models, the number of footprints in each cluster can be detailed per algorithm in SOM, because the owning algorithm is known for each of them.

### A. DE, PSO and SDEA solving Giunta

The 36 cluster SOM codes plot in Fig. 1a shows the values of the footprint variables of the three algorithms searching Giunta. The matching overall counts plot in Fig. 1b shows that clusters (4,5) and (6,2) take up the overwhelming majority of the steps all algorithms make. Both clusters pertain almost exclusively to DE, and represent cases where new solutions of worse quality fitness (higher CurFit) are discarded. Cluster (6,2) describes this scenario at an early stage with low iterations, while (4,5) shows the same behaviour at a later stage. In both clusters, the fitnesses have already been developed to a high quality, and the high FitDistCurToPrev indicates a considerably worse fitness in the new solution. Both clusters describe cases where the solutions are close in vector distance.

Unlike the other algorithms, DE also has a high proportion of (4,1) and (4,4) moves, and a higher than medium number of (4,2) moves. The latter are the most significant, because they are the moves that the algorithm keeps. The fitness is already high in this cluster, but the PrevFit value is higher than the CurFit value. The new solution vector is close to the base vector, as the DistCurToPrev value is imperceptible. Clusters (4,1) and (4,4) in contrast describe moves that cover a long distance and incur a significant decrease in fitness (CurFit is higher than PrevFit and FitDistCurToPrev is high). The only difference between (4,1) and (4,4) is one occurs early in the optimisation (low iterations) and the other late (high iterations). All other clusters have very few contributions from DE.

PSO has more different types of steps, with high levels of moves in clusters (2,3), (2,5) and (6,6), but also clusters (1,3) and (2,2), and another 11 describing moves that are represented with above-average counts for PSO. The most prevalent moves happen early (low iterations) in the trial that are medium distance in vector space but of much worse fitness, represented by (6,6). The same moves happen at later (cluster (2,3)) and very late (cluster (2,5)) iterations. The most common improving moves PSO makes are covered by clusters (5,3), (1,2) and (2,4). Cluster (5,3) contains significant improvements, medium distance in vector space, around one third through the iterations, while (1,2) shows small improvements early in the search, at a significant distance in vector space. Cluster (2,4) describes a very similar situation later in the search.

Although SDEA is a combination of DE and PSO moves, its most common moves overlap with neither algorithm. Early in the search, although not quite at the start, it tends to make hugely deteriorating moves, jumping significant distances. These moves are represented by (2,1) and are discarded, therefore they must be DE moves, although they hardly ever happen in DE. This is not surprising, given DE is greedy, and PSO is not. Using - necessarily mediocre - PSO solutions to determine the differential and target vectors must lead to some very unfit probes, which can only be discarded. Cluster (2,1) illustrates that alternating between PSO and DE on a random basis uses function evaluations on doomed probes.

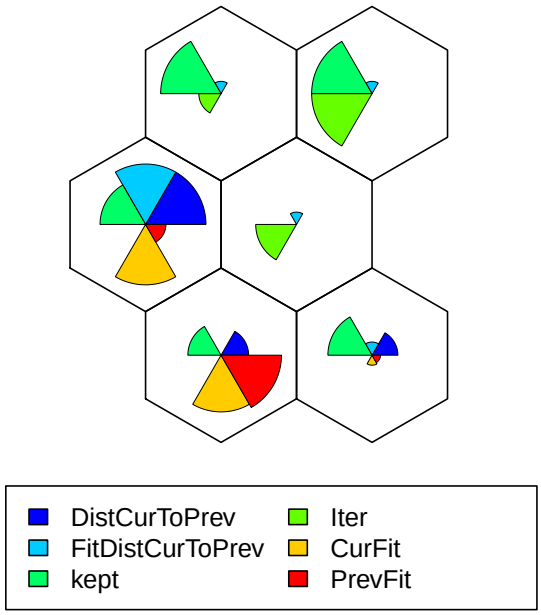
Cluster (5,2) also contains mostly SDEA data points. These are moves where the algorithm steps from a solution with bad fitness to one with a slightly better fitness over a short vector distance at an advanced stage, an indication of maintaining exploration throughout the search. Cluster (5,1) in contrast are also late-stage exploratory moves, but they are carried out using the DE side of SDEA, which is evident in the rejection of the solutions. The data points describe a reasonably fit base solution and a new probe with at least twice the fitness value. Cluster (2,6) describes a similar scenario and also contains mostly SDEA footprints. Cluster (1,6), however, shows moves at medium iterations where small steps in vector space lead to improvements. Whether the 'blundering' - exploratory moves with little hope of improvement - prevalent in SDEA make it a better or worse algorithm can only be answered by a reference to the algorithms' performances in Table II. SDEA provides better results than PSO on Giunta, but DE wins hands down.

### B. DE, PSO and SDEA solving Schwefel

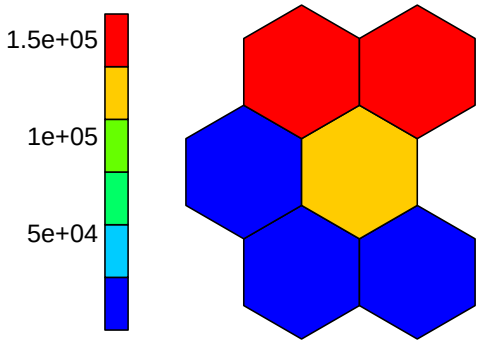
In the second unsupervised scenario, the Schwefel function was searched in 30 dimensions. In Section IV-A, the crossover probability of DE had been $p_{cr} = 0.5$. In 30 dimensions, this setting proved far too high and was reset to zero, which led to the best performance, listed in Table II. This leaves DE changing one dimension at each step. Making changes in more dimensions leads to a swift improvement of the global best fitness at the start, but fails to fine-tune the result at the end. The setting of zero was retained for SDEA for a fair comparison.

The SOM results for the Schwefel trials have few distinct clusters. Training $3 \times 3$ clusters still included one without cases. Fig. 2 shows the SOM with $3 \times 2$ clusters. High dimensionality seems to lose much detail, as almost all steps have to be comparatively small to ensure the algorithm finds good solutions. The six clusters are interpreted in Table III.

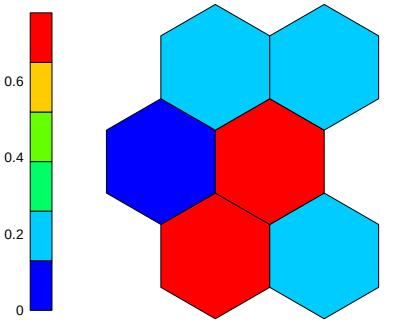
From Table II we can see that all algorithms come very close to the optimal solutions, with PSO coming the closest, and surprisingly, SDEA coming last, although it includes PSO moves. Visualised in Fig. 3, the algorithms improve the fitness quickly during the first 400 moves, then spend the remaining 4000 iterations on very small improvements.



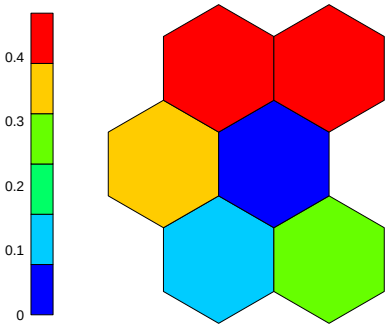

(a) Codes plot detailing combinations of footprint values



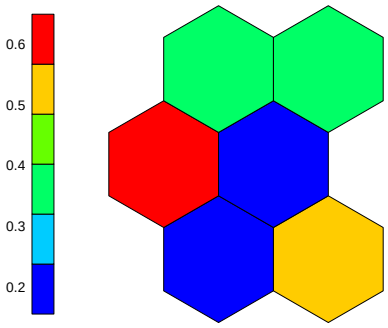

(b) Number of footprints that falls into each cluster




(c) Proportion of DE footprints




(d) Proportion of PSO footprints




(e) Proportion of SDEA footprints

Fig. 2: SOM for the footprint of the three algorithms searching the Schwefel function in 30 dimensions.

TABLE III: Clusters from unsupervised SOM of Schwefel optimisation

| Cluster | Step Interpretation | Predominant in |
|---|---|---|
| (1,1) | At the beginning of the optimisation, fitnesses are still high and small improvements are made with steps that cover large distances in vector space. Only some of them are kept, which means not all of them improve the fitness. | DE |
| (1,2) | At the early stages of the optimisation fitnesses have improved, probes are taken at large distances and these do not improve the fitness. | SDEA |
| (2,1) | Very large steps in both vector and fitness spaces made at the beginning of the optimisation, incurring a drastic deterioration in fitness. Only some of them are kept, suggesting both DE and PSO are the origin of these. | SDEA |
| (2,2) | Towards the middle of the trial, fitnesses are already very good. Small moves lead to a tangible deterioration of fitness and are discarded. | DE |
| (3,1) | After the initial stages, fitnesses are good and small moves are made, which lead to perceptible fitness changes but are presumably mostly deteriorating moves. They are kept nonetheless. | PSO |
| (3,2) | At a late stage in the trial, fitnesses are good and small moves are made, which lead to perceptible fitness changes, presumably mostly deteriorating, but mostly kept. | PSO |

Fitness improvements of magnitudes $\frac{1}{80} - \frac{1}{130}$ may be considered irrelevant from an optimisation perspective, but they highlight PSO's well-known convergence behaviour that shifts all solutions close to the optima, reducing distances in 4. DE on the other hand is tied to the distances between local optima where its solutions are likely located. This explains why PSO has twice as many moves in the (6,1) and (6,2) clusters than DE, whereas SDEA spends a similar percentage on these types of moves (the colour scale is different for each).

The plot of the fitness development in the range 250 - 400

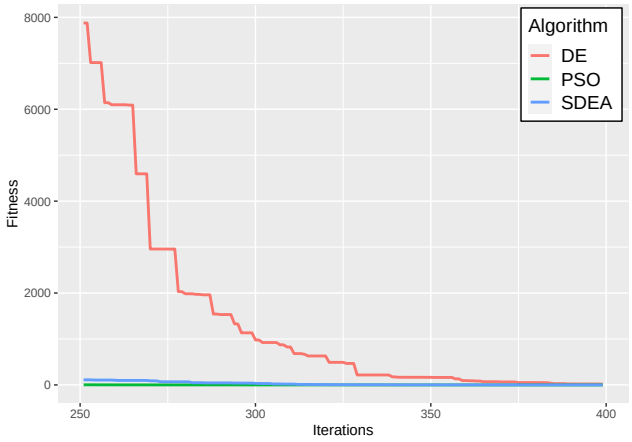



Fig. 3: Excerpt of fitness development on Schwefel

iterations in Fig. 3 shows how DE approaches these areas below unity far more slowly. This explains the high proportion of DE moves in clusters (1,1), describing the descent from high fitness values at the start, and (2,2), describing the discarded non-improving moves across the process. Over time, however, DE occasionally contributes a small improving move at elevated fitness levels (clusters (3,1) and (3,2).

It is clear from the small number of clusters that the behaviours of the algorithms are far less distinct in high dimensions, contributing more to the same clusters than when working on two-dimensional Giunta.

### C. Supervised SOM of DE, PSO and SDEA

For this investigation, two data sets from two trials of each algorithm were used. The training data set used the same footprints as Sections IV-A and IV-B. For the testing set, the trials for each algorithm were run again with a different random seed.

Given the insights gained from the unsupervised SOMs in Sections IV-A and IV-B, the notion arises that the search stagnates before the 1000th iteration, and therefore might no longer contain as many search moves that can clearly be attributed to one algorithm over the other. To verify this hypothesis, the prediction accuracy from supervised SOM were used. The accuracies measured when training on the first 1000 iterations proved indeed consistently higher, sometimes significantly, than when using footprints from all 5000 iterations. The data suggests that even at 4000 iterations with hardly any search progress, the moves continue to be typical of either PSO or DE, but the iteration variable loses its power as a distinguishing factor at lower iterations (0 - 1000) when all 5000 iterations are used in the footprint data. Therefore, the supervised SOM investigation was carried out with the footprints from the first 1000 iterations only.

Another question is how many clusters to train the SOM on. The prediction accuracies measured from the analysis of the test data sets indicate that this is a crucial parameter. Using a suboptimal number of SOM clusters may change the accuracy such that the prediction on the test set attributes the majority of

TABLE IV: Confusion matrix for Giunta

| | | Actual | | |
|---|---|---|---|---|
| | Algorithm | DE | PSO | SDEA |
| Prediction | DE | 48% | 2% | 8% |
| | PSO | 23% | 79% | 37% |
| | SDEA | 24% | 19% | 47% |

PSO moves to DE. Successive SOMs were created for both the Giunta and Schwefel data sets starting from a 6 x 6 hexagonal grid, then reducing until only one cluster had no footprints attributed to it. The model of each configuration was tested for prediction accuracy. This sensitivity analysis revealed that the best accuracy for the Giunta function was achieved with 6 x 6 clusters (Table IV), and for Schwefel with 4 x 5 (Table V).

TABLE V: Confusion matrix for Schwefel

| | | Actual | | |
|---|---|---|---|---|
| | Algorithm | DE | PSO | SDEA |
| Prediction | DE | 13% | 5% | 14% |
| | PSO | 13% | 5% | 16% |
| | SDEA | 14% | 31% | 17% |

The predictions are rather accurate in the case of Giunta, especially considering that SDEA contains some of the same moves, even though SDEA has also been shown to have its own types of footprint. Nonetheless, the footprints inaccurately assigned in Table IV are not surprising, as algorithms occasionally take similar steps, especially at the beginning, when information is scarce and the search mainly random. The predictions on the Schwefel test set are more disappointing. Only about 40% of each algorithm's footprints have been assigned to any cluster at all.

The Giunta results are shown in detail in Fig. 4. The independent footprint variables are the same as in the unsupervised clustering in Section IV-A. The difference is that this model has been trained to separate the footprints by algorithm as shown in the plot named Dependent. All clusters are distinct, belonging to only one algorithm, except (4,6) which has no footprints in it. Empty clusters are still shown with variables in the Independent plot, but these are largely due to the initialisation of the weights and carry no meaning.

The Dependent plot shows nine clusters for DE, eleven for PSO and fifteen for SDEA. The diversity of distinct types of move are expectedly larger in a hybrid. A greater diversity in PSO than DE is also easily explained with PSO taking varying influences from local and global best positions.

The DE clusters stand out by their rejection of moves (the 'kept' variable), as all DE clusters except (1,4) and (2,5) reject the probes. Fig. 5 illustrates how close to 5000 moves are spent on each of (1,6) and (2,6), while 4500 each are spent on (1,5) and (3,5), none of them successful. Only about 1500 of the 30,000 moves (population of 30 and 1000 iterations) contribute directly to the pool of better positions, as they fall into (1,4) and (2,5).

Although PSO accepts all its candidates, according to the Independent plot of Fig. 4 only clusters (4,1), (4,2) and (6,1) contain improving moves. According to Fig. 5, about 3500

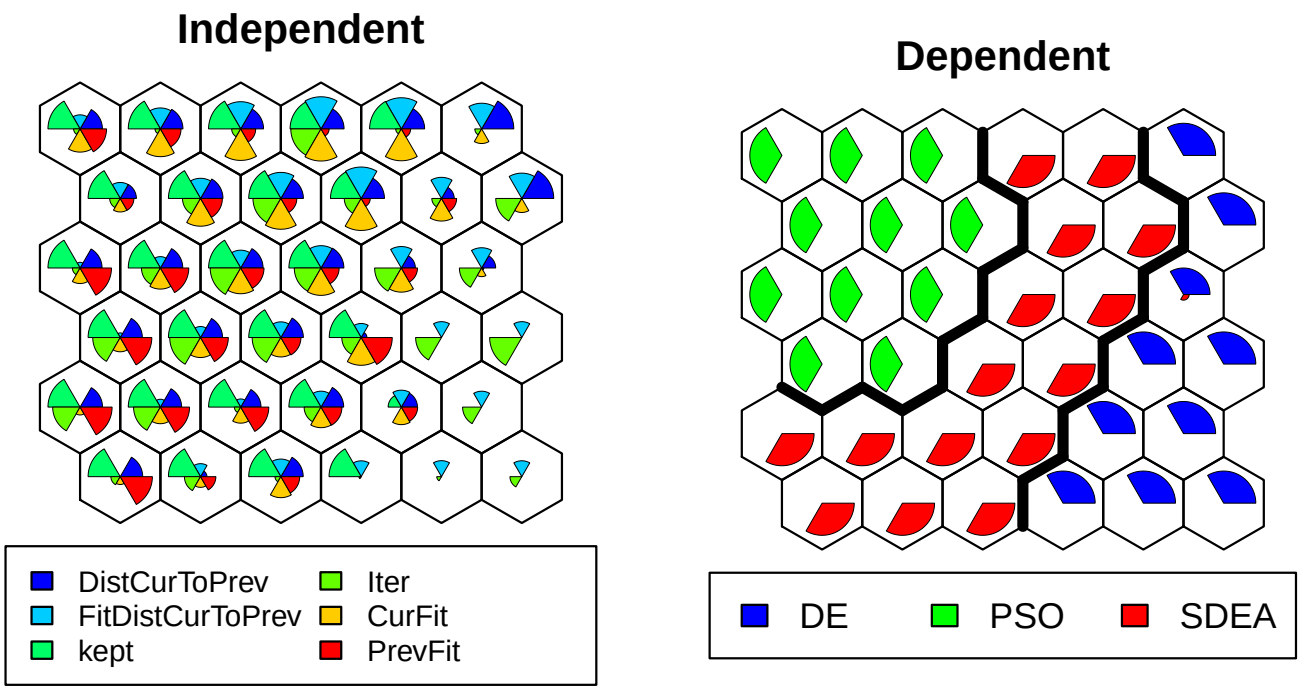



Fig. 4: Supervised SOM with footprint details in the clusters on the left and algorithm definitions on the right

moves fall into the (6,1) cluster, similar numbers into (4,2) and about 3000 into cluster (4,1). The remaining, less successful clusters contain similar numbers, which means that only about one third of the candidate solutions directly contribute to the search.

Having more move types overall, SDEA also has more types of fitness-improving moves. Clusters (2,3), (1,2) and (1,1) clearly describe the initial phases of the search, at iterations 200, 230 and 400 respectively. They take approx. 1500, 1000 and 2500 moves, while the later improvements (2,2), (2,3) and (3,4) also take 5000 between them. It appears no matter how many different types of improving moves exist, the number of improving moves stays at about one third.

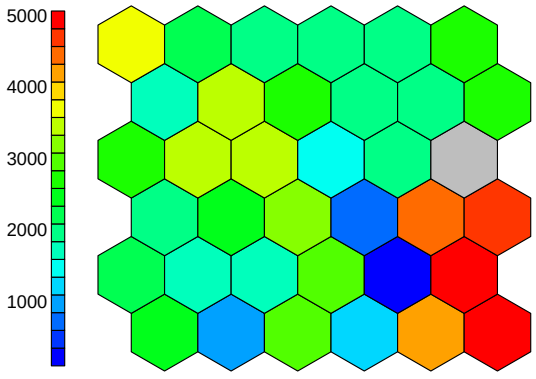



Fig. 5: Number of footprints in each cluster (none in the grey cluster (4,6))

The plots of the supervised Schwefel model are not shown to save space. They show reduced numbers of move types per algorithm. DE still has eight, but PSO has only three and SDEA four distinct type, while four types are shared between PSO and SDEA.

## V. Conclusions and Further Work

The study has shown that even with simple footprint variables, search algorithm behaviour can be described and compared in a way that both adds a detailed understanding of the search progress and a notion of comparability of algorithms. These two aspects promise a) that the method can help determine how close the search behaviour of two algorithms actually is and b) what moves contribute directly to the success of the algorithm. It also enables a researcher to manipulate the four principles of the search directly and observe the consequence of change in the footprints.

The differences between the SOMs on Giunta and Schwefel in an unsupervised context highlight that the moves of the algorithms, or footprints, are far more diverse in low dimensions. A possible hypothesis would be that in high dimensional space, all algorithms decrease their steps or miss out on optima. This reduces the number of different types of moves and blurs the lines between algorithms. This should be tested with trials of diverse algorithms and functions, the possible introduction of additional footprint variables and a performance comparison.

Building algorithm-specific clusters in the supervised setting shows that each algorithm spends around a third of its function evaluation budget on improving moves, although the number of different types of improvements varies between algorithms and is expectedly highest with the hybrid. The proportion of non-improving moves highlights the need for research on how to improve the efficiency of search heuristics.

This study has barely scratched the surface of search algorithm characterisation, and only in the context of continuous problems. Applications to solvers of discrete combinatorial problems are within reach, given that the distance between solutions in these spaces have been defined before. To demonstrate the general applicability of the method, more and more diverse algorithms have to be examined, as well as the effect of parametrisation on the search steps. The approach presented also has the drawback of being problem-specific; algorithms cannot be compared across different problems, because the footprints and their variables depend on the underlying fitness landscape.

## REFERENCES


[1] S. Dower, "Disambiguating evolutionary algorithms," Ph.D. dissertation, Swinburne University of Technology, 2012.

[2] K. M. Malan and A. P. Engelbrecht, "A survey of techniques for characterising fitness landscapes and some possible ways forward," *Information Sciences*, vol. 241, pp. 148–163, 2013.

[3] R. Lang and A. Engelbrecht, "On the robustness of random walks for fitness landscape analysis," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 1898–1906.

[4] J. R. Rice, "The algorithm selection problem," in *Advances in computers*. Elsevier, 1976, vol. 15, pp. 65–118.

[5] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, pp. 1–25, 2009.

[6] K. Smith-Miles and J. van Hemert, "Discovering the suitability of optimisation algorithms by learning from evolved instances," *Annals of Mathematics and Artificial Intelligence*, vol. 61, no. 2, pp. 87–104, 2011.

[7] L. Kotthoff, "Algorithm selection for combinatorial search problems: A survey," in *Data Mining and Constraint Programming*. Springer, 2016, pp. 149–190.

[8] S. Verweij and S. Fan, "Understanding search behavior via search landscape analysis in design optimization of optical structures," *JOSA B*, vol. 33, no. 12, pp. 2457–2471, 2016.

[9] J. Pérez, R. A. Pazos, J. Frausto, G. Rodríguez, D. Romero, and L. Cruz, "A statistical approach for algorithm selection," in *International Workshop on Experimental and Efficient Algorithms*. Springer, 2004, pp. 417–431.

[10] M. Quiroz, L. Cruz-Reyes, J. Torres-Jiménez, P. Melin *et al.*, "Improving the performance of heuristic algorithms based on exploratory data analysis," in *Recent Advances on Hybrid Intelligent Systems*. Springer, 2013, pp. 361–375.

[11] V. Landero, D. Ríos, J. Pérez, L. Cruz, and C. Collazos-Morales, "Characterizing and analyzing the relation between bin-packing problem and tabu search algorithm," in *International Conference on Computational Science and Its Applications*. Springer, 2020, pp. 149–164.

[12] T. Kohonen, "The self-organising map, a possible model of brain maps," *Perception*, vol. 26, no. 1_suppl, pp. 204–204, 1997.

[13] R. D. Lang and A. P. Engelbrecht, "An exploratory landscape analysis-based benchmark suite," *Algorithms*, vol. 14, no. 3, p. 78, 2021.

[14] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[15] M. G. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research*, vol. 196, no. 1, pp. 128–139, 2009.

[16] K. R. Harrison, B. M. Ombuki-Berman, and A. P. Engelbrecht, "A parameter-free particle swarm optimization algorithm using performance classifiers," *Information Sciences*, vol. 503, pp. 381–400, 2019.

[17] T. Hendtlass, "A combined swarm differential evolution algorithm for optimization problems," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 2001, pp. 11–18.

[18] A. Aleti and I. Moser, "Predictive parameter control," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 561–568.