Designing and Characterising Fitness Landscapes with Various Operators

Marius Gheorghita
Swinburne University of Technology
Melbourne, Australia
mgheorghita@swin.edu.au

Irene Moser
Swinburne University of Technology
Melbourne, Australia
imoser@swin.edu.au

Aldeida Aleti
Monash University
Melbourne, Australia
aldeida.aleti@monash.edu

***Abstract*—Stochastic optimisers such as Evolutionary Algorithms, Estimation of Distribution Algorithm are suitable methods when problems are highly complex and deterministic algorithms cannot be expected to produce acceptable results. Generally, when the search process produces the optimised solutions, there is no indication how successful the search has been. In previous work, we introduced Predictive Diagnostic Optimisation (PDO), a local-search-based solver which can predict with certain accuracy the quality of local optima and that can help decide which of the initial solutions is appropriate to optimise. The neighbourhood created by the swap operator was used in exploration of the search space and the number of predictors created is a metric for the homogeneity of the landscape. The advantage of PDO is that it provides information regarding the difficulty of the search landscape alongside the optimisation results. In this work we extend PDO by employing three more neighbourhood operators to allow a comparison between the performances of different types of local search. Each neighbourhood operator has its own group of predictors and the difficulty in predicting the local optima is quantified by a new metric, the prediction error. To provide an assessment of the characterisation ability for the algorithm, a set of landscapes with various degrees of difficulty has been designed by manipulating the matrices of the test problems instances. We show that the metric is able to identify the degree of difficulty that we expect the landscapes to pose for the employed local search operators.**

I. Introduction

Stochastic optimisers are approximate algorithms that generally perform better than random search, finding good approximate solutions to multimodal problems by exploiting gradients in the fitness landscape. A fitness landscape is a structure defined by three elements: the solution space of the problem, the fitness function that measures the quality of a solution and a neighbourhood operator that defines how we move between solutions [18]. A neighbourhood structure is defined by the possible changes that may be applied to a single solution. Local search is a successful class of approximation algorithms which has been shown to achieve near-optimal solutions to many difficult problems when coupled with a suitable global search [19]. It is an iterative improvement method in which, at each step, the current solution is improved by a predefined small modification. If, at each step, all possible modifications of the current solution are explored and the change with the biggest fitness improvement is made permanent, the local search type is steepest descent, assuming minimisation. Local search ends when no further improving moves are possible and the solution obtained is a local optimum. In multimodal landscapes, which stochastic algorithms are usually applied to, the solution obtained by a local search along with the path from the initial solution to the local optimum follows a gradient into the basin of attraction.

Starting with an initial solution, the modification that we apply in each step to obtain a fitness improvement is determined by the nature of the local search operator. Each operator will create a different search neighbourhood and the quality of the end solution where the local search stops will influence what move we decide to apply. In previous work, we used a swap operator to explore the neighourhood and for each complete local search we used the first step to project the final quality to be expected when no further improvement is possible. At the core of PDO, the ratio of the improvement achieved by the first step and the fitness improvement after the search stops forms a predictor. A predictor can then be matched to new initial solutions and their first moves. As predictors are created dynamically whenever the existing predictors are unable to predict the quality of the optimum to a predefined margin of accuracy, the number of predictors created during the run can be used to indicate how homogeneous is the created landscape for the swap operator.

This work improves the algorithm to extend the analysis of the various operators and to collect more information about the structure of the fitness landscapes than the initial PDO [17]. In particular, it aims at estimating the diversity of the local optima and of the basins by looking at the prediction error as a characterisation metric. In combination with the predictor number, this will provide a more robust characterisation of the search landscape, helping evaluate the approximate success of the local search in the case of problems where the global optimum is unknown.

Search space characterisation is usually conducted separately from the optimisation process with the aim of learning about properties of the search space to optimise the choice of the solver. The latter is dependent of the former, therefore to learn about search space while optimising is a step towards a better understanding of the optimisation process. Sampling a fitness landscape simply to characterise its structure to benefit future optimisation may be an impractical approach. The function evaluations may be costly, or there may be few instances to optimise and the knowledge gathered during the

diagnostic walk may be useful for few actual optimisation processes. Therefore, a feature detection algorithm should ideally undertake not only a landscape analysis but also a search for the (approximate) optimum.

This paper contributes to the Diagnostic Optimisation area extending PDO with a new metric based on prediction error that can show the suitability and limitations of the local search operators applied to the test problems and therefore an approximate indication of the solutions provided by the search. The reliability of the information extracted during the analysis was evaluated with the help of a set of modified problem instances to reflect various degrees of difficulty. We obtained this set of modified instances from two combinatorial problems, Quadratic Assignment Problem (QAP) and Flow Shop Scheduling Problem (FSSP), both represented by matrices. By introducing zeros into the test matrices, the sparsity of the matrices was used to control the difficulty of the problem and to verify the ability of the metric to differentiate between instances. The test instances origin from two sources: new instances from problem generators and well-known studied instances from the library.

II. Fitness Landscape Characterisation

A question that has intrigued researchers over time, is how we can determine when a problem type (instances) is easy for a particular search technique. Intuitively, describing the characteristics of a fitness landscape and finding features that makes the search hard will help understand the suitability and limitations of the heuristic. Since the aim of this work is to devise new means of describing features of the search space, one of the main tasks is to show whether the findings of the new method coincide with existing characterisation techniques. Few metrics capable of describing fitness landscapes have been discovered to date. Autocorrelation and its derivate correlation length [21] were arguably the first descriptive metrics conceived, followed by Fitness Distance Correlation (FDC) [9]. The correlation of neighbouring fitness values can be described by autocorrelation and is calculated as given in eq. 1. Given a number s of independent walks through the solution space with a predefined number of k neighbourhood steps each, we calculate the autocorrelation coefficient:

$$\rho(k) = \frac{\frac{1}{s}\sum_{i=0}^{s}(\overline{f}(x^0) - f(x_i^0))(\overline{f}(x^k) - f(x_i^k))}{\sigma_{f(x^0)}\sigma_{f(x^k)}} \quad (1)$$

In this equation, x^0 is the initial random solution and $f(x^0)$ its fitness value. The solution x^k is the final solution after k steps. $\overline{f}(x^0)$ and $\overline{f}(x^k)$ are the averages of the initial fitnesses and after k steps, respectively $\sigma_f(x^0)$ and $\sigma_f(x^k)$ represent the standard deviation of them. To measure the autocorrelation for all the test instances, a number of $s = 1000$ random walks were used, each with a number of $k = 10$ neighbourhood steps. The swap operator was used to create the neighbourhood.

For the problems represented by matrices, the sparsity of the matrix (represented by the percentages of zeros) seemed to correlate with problem difficulty as observed by Mitchell and Borchers [16]. Based on the assumption that sparser matrices lead to more difficult problems, we used the sparsity level to create a set of instances with a controllable degree of difficulty. To have a representative scale, we chose to insert zeros in the matrices to obtain sparsity levels of 40% and 80%. We have computed the autocorrelation with random walk for this set of instances and the results show that for a given problem the autocorrelation coefficient is largely the same between different levels of sparsity as observed in Tables I-II.

TABLE I: Common fitness landscape metric (autocorrelation with random walk) applied to FSSP instances.

Problem		Random walk k=10
JC	20x20	0.2962
	40%	0.2477
	80%	0.2611
	50x20	0.4870
	40%	0.5156
	80%	0.5198
	100x20	0.7398
	40%	0.6883
	80%	0.6793
MC	20x20	0.0092
	40%	0.2351
	80%	0.1839
	50x20	0.4001
	40%	0.4291
	80%	0.4917
	100x20	0.3973
	40%	0.6907
	80%	0.6773
MXC	20x20	0.0457
	40%	0.2258
	80%	0.1866
	50x20	0.4018
	40%	0.4219
	80%	0.4978
	100x20	0.5474
	40%	0.6823
	80%	0.6772

To measure the FDC, the Hamming distance between the known global optimum and the 100 local optima sampled has been calculated using the difference in assignments to locations. The 100 local optima were found using steepest descent (SD).

$$C_{FD} = \frac{\frac{1}{s}\sum_{i=0}^{s}(\overline{f} - f_i)(\overline{d} - d_i)}{\sigma_f \sigma_d} \quad (2)$$

Eq. 2 describes the calculation of the FDC, which is designed to establish the correlation of local and global optima with respect to their fitnesses and their distances to the global optima. $\overline{f}$ and $\overline{d}$ are the averages of the fitnesses and distances, while $\sigma_{f()}$ and $\sigma_{d()}$ are the standard deviations calculated for fitnesses respective distances. Values of a positive correlation show that fitnesses of the local optima increase with increasing proximity to the global optimum. FDC values around 0.0 are considered to describe difficult problems and a negative FDC indicates misleading problems where the quality of the local optima increases with increasing distance to the global optimum.

FDC is an a posteriori measure of difficulty and we could apply it only to the well-known instances from the QAP

TABLE II: Common fitness landscape metric (autocorrelation with random walk) applied to QAP instances.

Problem		Random walk k=10
Generator	uni20	0.1239
	40%	0.1242
	80%	0.1633
	uni40	0.4013
	40%	0.3444
	80%	0.3482
	uni60	0.5355
	40%	0.5296
	80%	0.4514
	uni80	0.6044
	40%	0.6170
	80%	0.6178
	uni100	0.6843
	40%	0.6476
	80%	0.67
QAPlib	nug20	0.1526
	40%	0.1491
	80%	0.1372
	tai20a	0.0756
	40%	0.1272
	80%	0.1638
	nug30	0.3473
	40%	0.3006
	80%	0.2831
	tai30b	0.2663
	40%	0.2856
	80%	0.2570
	kra30a	0.2539
	40%	0.2867
	80%	0.3378
	kra32	0.3806
	40%	0.3713
	80%	0.3141
	ste36a	0.3846
	40%	0.3662
	80%	0.3066
	ste36b	0.4369
	40%	0.3917
	80%	0.2649

library, since the newly generated problems do not have a known global optimum.

TABLE III: Common fitness landscape metric applied to QAP library instances.

Problem		FDC
QAPlib	nug20	0.0248
	tai20a	-0.2798
	nug30	-0.1675
	tai30b	0.1681
	kra30a	0.1281
	kra32	-0.1594
	ste36a	0.2209
	ste36b	0.1932

Kra* instances are variations of a hospital design problem and can therefore be expected to be homogeneous and similar. While the FDC provides a weak positive correlation for Kra30a, there is no correlation for Kra32. Tai**a are considered rugged [15] and as the FDC shows Tai20a is among the least correlated of all problem instances.

III. Predictive Diagnostic Optimisation

The success of stochastic optimisers depends on an appropriate balance between exploration and exploitation and most heuristics apply global and local search in different phases [1]. Optimisers such as EAs implicitly alternate between global and local search phases. Others, such as Ant Colony Optimisation [14] explicitly choose one or more results of a global search phase to apply a hill-climbing local search such as steepest ascent to.

A major research question in this context is which solutions should be chosen for local optimisation. One contribution of PDO is the prediction of the ultimate solution quality after a local search has been applied, solving the problem of choice of initial solution for the local search. The prediction is based on the fitness improvement ensuing from the first step of the local search applied to an initial solution. In contrast to the previous PDO implementation [17], the method applied in the current work creates predictors in an initial learning phase.

A. Predictor Definition

A local search is performed starting from an initial random solution x_0. Assuming a minimisation problem, after the *first step* of the local search, the ratio of improvement [17] between the initial solution x_0 and the improved solution x_1 is calculated as:

$$p_1 = \frac{n(f(x_0) - f(x_1))}{f(x_0)} \tag{3}$$

where n is the cardinality of the problem. The fitness at the end of the optimisation is used to calculate the ratio between the initial improvement of the first step and the fitness of the local optimum $f(x_k)$ as follows:

$$p_2 = \frac{f(x_0) - f(x_k)}{f(x_0) - f(x_1)} \tag{4}$$

Both ratios are part of a predictor. When considering a new starting solution for local optimisation, this solution is evaluated according to the objective function, which yields the value $f(x_0)$. Then, the first local search step is performed and $f(x_1)$ is obtained. The first ratio is calculated and compared to the first ratio of each available predictor. The closest matching predictor is then used to project the expected final fitness $f(x_k)$ of the solution after k optimisation steps, where k will vary depending on the number of steps needed to reach the bottom of the basin.

B. Predictor Discovery

PDO samples the solution space uniformly randomly to find the initial solutions from which the group of predictors are created. Unlike in previous work [17], no bias towards promising areas of the search space is maintained. The goal is to have an accurate represention of the whole landscape with the group of predictors created.

When a new solution x_0^b is created and improved to x_1^b by applying one local search move, the ratios p_1^b and p_2^b are calculated on the basis of the fitness difference between x_0^b and its improved solution x_1^b according to Eq. 3 and 4. PDO then determines if the new predictor is significantly different from the predictors created in the previous iterations. The similarity between two predictors is established using the Canberra distance defined as follows:

$$\mathcal{C}[(p_1^a, p_2^a), (p_1^b, p_2^b)] = \frac{|p_1^a - p_1^b|}{|p_1^a| + |p_1^b|} + \frac{|p_2^a - p_2^b|}{|p_2^a| + |p_2^b|} \quad (5)$$

where (p_1^b, p_2^b) and (p_1^a, p_2^a) represent the ratios of two predictors derived from solutions x^a and x^b. The distance is defined in the interval [0,2) with 0 representing identical predictors. The greater the distance, the more different are the predictors. The diversity of the gradients in the search space can be accurately represented if we accept predictors that are significantly different one from another, therefore the threshold for accepting a new predictor was set to $\epsilon = 0.1$. Each time a new predictor is created, it is compared with all predictors in the predictor pool using the Canberra distance (Eq. 5), a mathematical function used to sort objects by their similarity. In our case, the object is represented by a predictor, described by its ratios. If the distance of the new predictor to other predictors in the pool is bigger than this threshold, a new predictor is introduced into the group. If, between loops of $N = 100$ local searches, no new predictors have been created, the algorithm stops. The steps of the algorithm for predictor discovery loop using the Canberra distance are shown in the algorithmic listing 1.

Algorithm 1 Predictive Diagnostic Optimisation

procedure PREDICTORDISCOVERY(N, ϵ)
  $groupSize \leftarrow 1$
  $previousSize \leftarrow 0$
  while $groupSize \neq previousSize$ **do**
    for $i \leftarrow 1, N$ **do**
      $x_0^i \leftarrow randomSolution$
      x_1^i = FIRSTLOCALSEARCHSTEP(x_0^i)
      $p_1^i = \frac{n(f(x_0^i) - f(x_1^i))}{f(x_0^i)}$
      x_k^i = LOCALSEARCH(x_1^i)
      $p_2^i = \frac{f(x_0^i) - f(x_k^i)}{f(x_0^i) - f(x_1^i)}$
      $different \leftarrow true$
      for $j \leftarrow 1, predSize$ **do**
        $\mathcal{C}[(p_1^j, p_2^j), (p_1^i, p_2^i)] = \frac{|p_1^j - p_1^i|}{|p_1^j| + |p_1^i|} + \frac{|p_2^j - p_2^i|}{|p_2^j| + |p_2^i|}$
        if $\mathcal{C}[(p_1^j, p_2^j), (p_1^i, p_2^i)] < \epsilon$ **then**
          $different \leftarrow false$
        end if
      end for
      if $different$ **then**
        SAVEPREDICTOR(p_1^j,p_2^j)
      end if
    end for
    $previousSize \leftarrow groupSize$
    $groupSize \leftarrow predNumber$
  end while
end procedure

For every combinatorial problem, a number of possible neighbourhood moves exist. It is generally believed that some moves render a local search more prone to premature stagnation than others [18]. It is intuitive to implement more than one neighbourhood move when search space diagnostics is a desired byproduct of the search.

For each neighbourhood move, PDO maintains its own group of predictors. For each neighbourhood, the predictor discovery procedure is applied as described in listing 1. When the stopping criterion is met, the assumption is that the existing predictors for each group are able to represent the entire search space if searched with the group-specific operator. After the group creation phase, it follows the predictor application phase where we test the capability of the predictors to guide the choice of solutions to optimise. In this phase, random candidate solutions x_0^i are created, and improved locally once to obtain x_1^i. Based on this first step, a predictor is matched and the ultimate fitness of the local optimum is predicted. When the local search is complete on the same solution, the final fitness of x_k^i is compared with the prediction and a prediction error is recorded. The average prediction error of each predictor group that represents a particular neighbourhood is used to interpret the type of the landscape and the predictability of the gradients. Given that the number of predictors depends on the similarity of the gradients that are encountered in the search space, the number of predictors created can be used as an indicator of the homogeneity of the basins of attraction (assuming minimisation). If the slopes to these basins, or troughs with a local minimum at the bottom, all have slopes with the same shape, a single predictor will suffice to predict the outcome of a full local search. In addition to the number of predictors created, the distribution of usage of these predictors elucidates the relative frequency of a certain shape of the path to a local optimum.

C. Sampling the neighbourhood

The initial PDO implementation [17] uses steepest descent (SD) as its deterministic optimisation method, an expensive approach which exhaustively explores the complete neighbourhood of a solution before making the move that leads to the best fitness improvement. For most types of search moves and problems of relevant sizes, the resulting neighbourhood is prohibitively large. To reduce the cost of the local search during the exploration of the neighbourhood, we have introduced an improvement to the algorithm that is comprised of a sampling procedure [7]. The primary goal is to reduce the cost of the local search and at the same time to preserve approximately the same prediction accuracy and fitness improvement as the exhaustive SD approach. Compared to SD, sampling reduces the neighbourhood size to the sample size, therefore the exploration is limited to the sample. Instead of searching the neighbourhood exhaustively, the sampling procedure selects the best solution found in a representative sample. A representative sample is one that finds an approximate best fitness improvement. For determining the approximate best improvement from the neighbourhood by sampling, a statistical significance test was proposed, where a dynamic stopping criterion based on accuracy monitoring determines when the sample reaches the confidence level for the neighbourhood.

D. Diagnostic metric

In the predictor application phase, the prediction error is calculated as a percentage error between the predicted and the

actual fitness for each of the solutions that were optimised during this phase. As in previous work, a predictor is selected for making predictions based on the minimum difference between its first ratio and the candidate solution first step's ratio of improvement. Then the solution is optimised to the local optimum and the predicted fitness is compared to the predicted fitness to obtain the difference, which is the prediction error. If the error is not within a predefined margin of tolerance, the remaining predictors are examined for a match within the tolerance, which considers both the ratio between the first step (eq. 3) and the prediction (eq. 4). The best-matching predictor is used to determine the prediction error reported alongside the results. This procedure is repeated for all groups of predictors. The resulting value is reported as the prediction error for the neighbourhood moves. The prediction error calculated as a percentage is considered an indicator of the accuracy of the predictions, the suitability of the local search and the problem difficulty.

IV. Experiments

The purpose of the experiments is to to demonstrate that the measure we propose coincides with what we already know about the problems. In order to test the ability of the metric to detect the changes in the difficulty of a given landscape, we create problem instances with different levels of difficulty. Sparsity of the matrices is used to create these instances, since the more zeroes we will have in a matrix the more probable is to produce 'plateaus' in the fitness landscape that offers no direction for the search and isolates the local optima. This feature is assumed to introduce different degrees of isolation for the optima and by that making the search difficult for an algorithm that exploit the gradients.

A. Creation of instances with various degree of difficulty

We produced several instances from the generators as reference instances, and generated different problem instances from these with two different levels of sparsity: 40% and 80%. The simplest way to generate these modified instances with variable sparsity is to replace randomly a number of nonzero elements, until the desired percentage of zeroes was obtained. The original instances are in general uniform, and exhibit a low prediction error. What we expect from the modified instances is to have a correlation between the level of sparsity and their degree of difficulty measured direct by the prediction error metric. For the experiments two combinatorial problems were chosen, Quadratic Assignment Problem (QAP) with a instance generator having details provided in [10] and Flow-Shop-Scheduling Problem (FSSP) with a instance generator described by Watson [20].

B. Benchmark problems

QAP [11] is the problem of allocating a set of facilities to a set of locations, with a cost function associated to the distance and flow between the facilities.

We are given two n x n input matrices with real elements $H = [h_{ij}]$ and $D = [d_{kl}]$, where h_{ij} is the flow between facility i and facility j and d_{kl} is the distance between location k and location l. Given n facilities, the solution of the QAP is codified as a permutation $\sigma = (\sigma_1, ..., \sigma_n)$ where each σ_i (i = 1, . . . , n) represents the facility that is allocated to the i-th location. The fitness of the permutation is given by the following objective function:

$$F(\sigma_1, \sigma_2 ... \sigma_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} h_{ij} * d_{\sigma_i \sigma_j} \tag{6}$$

The objective is to assign each facility to a location such that the total cost is minimised. The quality of the solution is determined by the absolute position of each index (facility) in the permutation as regards the absolute position of the remaining indexes as observed by Ceberio et al. [4]. The first subset of QAP instances is provided by a generator who produces uniformly random instances and the second subset is a well-known set of instances in the QAP library [3]. This type of instances assure that for larger problem size they are more interesting to study than the 'real-like' instances, as was observed by Knowles and Corne [10]. The reason is that are more local optima close to the global optimum, but it is difficult to direct the search to the one that is better.

FSSP [8] describes the task of scheduling n jobs $(i = 1, ..., n)$ on m machines $(j = 1, ..., m)$. A job consists of m operations and the j-th operation of each job must be processed on machine j for a specific time. A job can start on the j-th machine when its $(j-1)$-th operation has finished on machine $(j-1)$, and if machine j is free. The goal of the optimisation is to minimise the processing time of all the jobs, or in other words, to minimise the processing time of the last job.

The solution is codified as a permutation of length n that represents the ordering in which the jobs are going to be processed. This means that for each machine the order of the jobs is the same and it is given as a permutation.

Let $p_{i,j}$ denote the processing time for job i on machine j, and $c_{i,j}$ denote the completion time of job i on machine j. Then $c_{\sigma_{i,j}}$ is the completion time of the job scheduled in the i-th position in the sequence on machine j. $c_{\sigma_{i,j}}$ is computed as $c_{\sigma_{i,j}} = p_{\sigma_{i,j}} + \max\left\{c_{\sigma i,j-1}, c_{\sigma_{i-1,j}}\right\}$. The objective function F is given by Eq.7.

$$F(\sigma_1, \sigma_2, ..., \sigma_n) = c_{\sigma_{n,m}} \tag{7}$$

The generator for FSSP produces instances based on features found in some real-world scheduling problems. Three types of instances are used 'job-correlated' (JC), 'machine-correlated' (MC) and 'mixed-correlated' (MXC). In JC instances, job processing times are independent of the machine. In MC instances, job processing times are dependent upon the machine. MXC instances are a form of MC types in which the relative ranks of job processing times are generally consistent across the machines. What the authors observed is that the majority of the instances produced were solved easily by both simple and complex algorithms. They investigated the search space using the next-descent search algorithm [6] under two

shifting operators and found a distribution of local optima in a topography like 'big-valley'. In this distribution of 'big-valley' was observed that local optima are closer together than randomly chosen solutions and better local optima tend to be closer to global optima. This is assumed to be due to 'smoother' fitness regions at the bottom of the valley, especially since local optima near one another have similar evaluations.

C. Neighbourhood Definition

In order to test the consistency of the diagnostics, several neighbourhoods with different characteristics were used. The expectation is that the algorithm works well with different neighbourhoods, however some neighbourhoods may be more suitable than others. For example, on a more rugged landscape, it is expected that a fitness-improvement-driven neighbourhood such as k-opt performs better than a simple neighbourhood such as 2-opt.

2-opt operator [5] is a simple local search algorithm often used for solving the Travelling Salesman Problem (TSP). In permutation-based problems the algorithm is defined as a pair-exchange neighbourhood which consists of all permutations obtained by applying a transposition of two elements in the original permutation. It keeps its characteristics as a simple move and is one of the most frequently used.

3-opt operator [2], [12] is another simple local search algorithm proposed for solving TSP and related network optimisation problems. The algorithm is an extension of 2-opt, in that it performs two pair-exchanges instead of one. The second pair-exchange is applied to the resulting permutation from the first pair exchange.

k-opt operator [13]; for a fixed k, where k-opt is the natural generalization of 2-opt and 3-opt performs up to k pair exchanges to the original permutation with the condition that the intermediary permutations display a fitness improvement. k is established by the immediate fitness deterioration compared to the k-1 pair-exchange. It is a conditional move based on improving exchanges, and the exit criterion is the fitness deterioration.

k-opt-d (k-opt with deterioration) is a new operator introduced in this paper to address the drawbacks of k-opt. k-opt-d uses a deterioration acceptance threshold, i.e. if there is no fitness improvement in the current solution after applying a pair-exchange to its parent, the algorithm compares the fitness difference of the current solution and the fitness of the parent of its parent. If the fitness of the current solution has deteriorated from the fitness of its "grand-parent", then the algorithm stops. Similar to k-opt, k-opt-d is intended for more rugged neighbourhoods, where the fitness improvement is not continuous.

V. Results

Each of the QAP and FSSP problem instances was optimised 30 times using the PDO algorithm and the resulting prediction errors were averaged for each of the four investigated neighbourhood operators.

TABLE IV: Prediction error for the 40% and 80% sparsity level - FSSP instances.

Problem		2-opt	3-opt	k-opt	k-opt-d
JC	20x20	0.238	0.216	0.231	0.234
	40%	7.097	6.686	5.872	5.339
	80%	16.452	16.659	14.837	12.633
	50x20	2.326	2.330	2.263	2.130
	40%	6.404	5.967	5.593	5.204
	80%	13.449	12.762	12.123	11.229
	100x20	2.181	2.174	2.232	2.120
	40%	5.282	5.029	4.806	4.633
	80%	12.290	12.025	11.427	10.929
MC	20x20	0.616	0.578	0.558	0.572
	40%	6.290	5.748	5.337	4.932
	80%	14.307	14.133	13.957	12.217
	50x20	2.259	2.138	2.038	1.977
	40%	6.204	5.729	5.454	5.273
	80%	13.520	12.740	12.594	11.707
	100x20	0.481	0.504	0.489	0.509
	40%	4.157	4.058	3.963	4.056
	80%	11.289	10.946	10.575	10.270
MXC	20x20	0.913	0.843	0.856	0.873
	40%	6.003	5.375	5.349	5.012
	80%	14.655	13.169	12.759	11.837
	50x20	2.157	2.108	2.112	2.005
	40%	6.085	5.669	5.754	5.191
	80%	13.455	12.761	13.080	11.710
	100x20	0.696	0.634	0.742	0.735
	40%	4.560	4.215	4.186	4.126
	80%	11.008	10.298	10.849	9.881

An observed trend in Table IV is that the prediction error increases gradually with the sparsity level for all the FSSP instances and for all neighbourhoods. For the QAP instances the difference in prediction error between the original instance and the ones with sparsity controlled is more accentuated as observed in Table V. Between the modified instances, the 80% sparsity instances have the highest prediction error showing a very difficult landscape to search. It is clear that the ruggedness induced by the added sparsity poses a problem to all neighbourhood moves. Not even an operator that accepts deterioration can overcome this challenge.

Where the modified instances are affected by high sparsity, the landscapes have increasingly isolated local optima, the gradients become less predictable and that is quantified by a high prediction error metric. The information obtained during the search shows the limitations of the local search on the instances affected by the high sparsity. In the FSSP case, for the instances that are the hardest to predict, the operator with deterioration indicates the lowest prediction error. For the QAP case on the original instances and 40% sparsity the results indicate specialised moves (k-opt, k-opt-d) have the lowest error. On 80% sparsity becomes more difficult to associate a specialised move with a better predictability, the lowest error being shown by 3-opt operator.

When the landscape is not uniform and very difficult to search, making specialised moves based on improvement or deterioration like k-opt, k-opt-d it shows no better results than only random-based moves in neighbourhood as 3-opt. Overall, the moves confirm that the prediction metric is reliable independent of the chosen neighbourhood, and if that is combined with another metric to be extracted, like the average best fitness discovered during the search process, could be

beneficial in recomending a certain move as fit for a certain type of instance/problem.

TABLE V: Prediction error for the 40% and 80% sparsity level - QAP instances.

Problem		2-opt	3-opt	k-opt	k-opt-d
Generator	uni20	3.661	3.668	2.986	2.956
	40%	10.763	10.189	9.902	8.960
	80%	756.634	199.604	694.584	915.058
	uni40	2.145	2.070	2.043	1.962
	40%	5.519	5.198	5.046	4.825
	80%	40.513	31.497	39.496	39.936
	uni60	1.699	1.618	1.554	1.576
	40%	3.927	3.539	3.442	3.409
	80%	24.634	21.299	23.566	23.432
	uni80	1.393	1.295	1.324	1.216
	40%	3.091	2.780	2.850	2.716
	80%	18.713	16.418	18.285	16.716
	uni100	1.076	1.058	1.057	1.013
	40%	2.483	2.465	2.450	2.398
	80%	14.783	13.311	14.589	13.648
QAPlib	nug20	4.500	4.527	3.930	3.657
	40%	9.800	9.528	8.761	7.702
	80%	204.073	109.781	204.653	191.193
	tai20a	3.611	3.376	3.234	3.020
	40%	10.456	10.298	9.316	8.723
	80%	915.568	218.709	989.525	911.447
	nug30	3.652	3.691	3.498	3.366
	40%	7.146	6.924	6.095	6.241
	80%	61.135	44.980	61.693	61.383
	tai30b	13.700	14.289	12.855	11.632
	40%	25.822	23.964	24.383	23.821
	80%	18984.661	4080.113	16380.792	16622.982
	kra30a	5.505	5.513	5.132	4.876
	40%	6.819	6.598	6.300	5.781
	80%	74.928	52.327	75.834	77.059
	kra32	5.722	6.091	5.361	5.107
	40%	6.648	6.732	5.875	5.850
	80%	74.636	53.404	73.502	75.620
	ste36a	10.847	11.204	10.289	9.913
	40%	11.753	11.692	11.276	10.987
	80%	151.152	82.061	147.342	145.800
	ste36b	22.180	21.572	21.833	20.691
	40%	23.494	21.151	21.731	20.998
	80%	315.602	138.527	304.766	284.203

VI. Conclusion

In this paper, we have extended PDO, an optimisation approach which is based on predictive local search. A metric for measuring the problem difficulty based on the prediction error was proposed and we have created and 'manipulated' the landscapes for the tested problems to obtain a various range of instances that would challenge the algorithm's capacity of diagnostic.

Four different neighbourhood operators have led to the creation of four distinct groups of predictors used to analyse the suitability of the local search. The neighbourhoods validate the consistency of the results for the diagnostic. When the problem is predictable, the low prediction error confirms it and when we have higher prediction error we deal with more rugged landscapes of difficult problems. The information returned by the algorithm describes the landscapes in accordance with the existing knowledge about the problem instances and the values obtained for the metrics vary accordingly with the difficulty of the designed landscapes. An interesting direction for further research is to extend the search space analysis to a larger number of of instances and examine more closely the relationship of sparsity to instance hardness.

References

[1] A. Aleti. *An Adaptive Approach to Controlling Parameters of Evolutionary Algorithms*. PhD thesis, Swinburne University of Technology, 2012.

[2] F. Bock. An algorithm for solving traveling-salesman and related network optimization problems. unpublished manuscript associated with talk presented at the 14th orsa national meeting, 1958.

[3] R. E. Burkard, S. Karisch, and F. Rendl. Qaplib-a quadratic assignment problem library. *European Journal of Operational Research*, 55(1):115–119, 1991.

[4] J. Ceberio, E. Irurozki, A. Mendiburu, and J. A. Lozano. A review on estimation of distribution algorithms in permutation-based combinatorial problems. Technical report, University of the Basque Country, 2011.

[5] G. Croes. A method for solving traveling salesman problems. *Operations Research*, 6:791–812, 1958.

[6] M. Dietzfelbinger, J. E. Rowe, I. Wegener, and P. Woelfel. Precision, local search and unimodal functions. *Algorithmica*, 59(3):301–322, 2011.

[7] M. Gheorghita, I. Moser, and A. Aleti. Characterising fitness landscapes using predictive local search. GECCO '13 Companion. ACM, 2013.

[8] J. N. Gupta and E. F. S. Jr. Flow shop scheduling research after five decades. *European Journal of Operational Research*, 169:699–711, 2006.

[9] T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.

[10] J. Knowles and D. Corne. Instance Generators and Test Suites for the Multiobjective Quadratic Assignment Problem. In *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 295–310. Springer. Lecture Notes in Computer Science. Volume 2632, 2003.

[11] T. C. Koopmans and M. Beckmann. *Assignment Problems and the Location of Economic Activities*. Cowles Foundation for Research in Economics, Yale University, 1955.

[12] S. Lin. Computer solutions of the travelling salesman problem. *Bell Syst. Tech J.*, 44:2245–2269, 1965.

[13] S. Lin and B. Kernighan. An effective heuristic algorithm for the travelling salesman problem. *Operations Res.*, 21:498–516, 1973.

[14] N. Liouane, I. Saad, S. Hammadi, and P. Borne. Ant systems and local search optimization for flexible job shop scheduling production. *International Journal of Computers, Communications and Control*, II(2):174–184, 2007.

[15] P. Merz and B. Freisleben. Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. *Evolutionary Computation*, 8:61–91, 2000.

[16] J. E. Mitchell and B. Borchers. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. *High Performance Optimisation*, pages 349 – 366, 2000.

[17] I. Moser and M. Gheorghita. Combining search space diagnostics and optimisation. In *Congress on Computational Intelligence, Proceedings*, pages 897–904. IEEE, 2012.

[18] C. M. Reidys and P. F. Stadler. Combinatorial landscapes. *SIAM Review*, 44(1):3–54, 2002.

[19] T. Stutzle. *Local search algorithms for combinatorial problems - analysis, improvements, and new applications*, volume 220 of *DISKI*. Infix, 1999.

[20] J.-P. Watson, L. Barbulescu, L. D. Whitley, and A. E. Howe. Contrasting structured and random permutation flow-shop scheduling problems: Search-space topology and algorithm performance. *INFORMS Journal on Computing*, 14:98–123, 2002.

[21] E. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63:325–336, 1990.