Generating Diverse and Discriminatory Knapsack Instances by Searching for Novelty in Variable Dimensions of Feature-Space

Alejandro Marrero
Universidad de La Laguna
San Cristóbal de La Laguna, Spain

Eduardo Segredo
Universidad de La Laguna
San Cristóbal de La Laguna, Spain

Emma Hart
Edinburgh Napier University
Edinburgh, United Kingdom

Jakob Bossek
RWTH Aachen University
Aachen, Germany

Aneta Neumann
The University of Adelaide
Adelaide, Australia

ABSTRACT

Generating new instances via evolutionary methods is commonly used to create new benchmarking data-sets, with a focus on attempting to cover an instance-space as completely as possible. Recent approaches have exploited Quality-Diversity methods to evolve sets of instances that are both diverse and discriminatory with respect to a portfolio of solvers, but these methods can be challenging when attempting to find diversity in a high-dimensional feature-space. We address this issue by training a model based on Principal Component Analysis on existing instances to create a low-dimension projection of the high-dimension feature-vectors, and then apply Novelty Search directly in the new low-dimension space. We conduct experiments to evolve diverse and discriminatory instances of Knapsack Problems, comparing the use of Novelty Search in the original feature-space to using Novelty Search in a low-dimensional projection, and repeat over a given set of dimensions. We find that the methods are *complementary*: if treated as an ensemble, they collectively provide increased coverage of the space. Specifically, searching for novelty in a low-dimension space contributes 56% of the filled regions of the space, while searching directly in the feature-space covers the remaining 44%.

KEYWORDS

Instance generation, instance-space analysis, knapsack problem, novelty search, evolutionary computation

ACM Reference Format:
Alejandro Marrero, Eduardo Segredo, Emma Hart, Jakob Bossek, and Aneta Neumann. 2023. Generating Diverse and Discriminatory Knapsack Instances by Searching for Novelty in Variable Dimensions of Feature-Space. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583131.3590504

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590504

1 INTRODUCTION

Methods for generating new sets of synthetic instances for a given problem domain have attracted significant attention in the Evolutionary Algorithm (EA) literature: they can be used to generate new instances to cover areas of an instance-space currently lacking in instances [4] and to generate instances that are hard or easy for a particular solver or discriminatory with respect to a portfolio of solvers [6, 15]. This helps to provide new insights into the relationships between solver performance and instance characteristics. It also informs the development of both better algorithm-selection and algorithm-configuration methods.

The majority of existing methods tend to focus on one or the other of the above tasks: either they generate instances that are discriminatory without explicitly requiring diversity [2] or they generate instances that are diverse but not necessarily discriminatory [26]. However, very recent work has demonstrated that the field of instance-generation could benefit from the class of EAs known as Quality-Diversity (QD) algorithms [9]. Since their introduction around a decade ago, these algorithms are now prolific in the field of robotics due to their ability to increase exploration (by rewarding diversity in a feature-space of interest defined by a user) while simultaneously promoting quality, with the benefit of returning multiple high-quality solutions in a single run [11, 32]. They have recently begun to gain traction within combinatorial optimisation, e.g. [30] in increasing user-choice and most recently, to generate instances that are both diverse *and* discriminatory with respect to an algorithm portfolio [5, 20]. The term QD refers to two main sub-classes of algorithms — MAP-Elites [22] and Novelty Search (NS) [18]. In the former, a dynamically expanding population of solutions is mapped to an $n-$dimensional archive discretised into cells. Each dimension refers to a user-defined feature derived from a solution. Each cell retains the solution with the highest objective value found (i.e. its ability to discriminate between solvers). On the other hand, in NS, a fixed size population of solutions is augmented with a dynamically growing archive. A feature-vector is calculated from each solution, and used to calculate the novelty of the solution with respect to the current population and the archive. Selection to create future generations is driven by a combination of objective fitness and novelty.

MAP-Elites has shown promise in generating instances for the Travelling Salesperson Problem (TSP) [5] but does not scale well as the number of axes used to define the archive increases: for example, in the former paper, just combinations of two features are explored from a set of over 150 potential candidates to create the

archive. On the other hand, [20] demonstrated that NS could be used to generate knapsack instances using an 8d feature-vector to calculate novelty. While this yielded promising results, the method relies on calculating novelty as the distance between two feature-vectors: this metric is problematic for high dimensional vectors when distances become diminishingly small (the well known curse of dimensionality [1]).

In this paper, we address these weaknesses. The innovation of the work is to modify the approach proposed by Marrero et al. [20] to search for novelty in a new low-dimensional space obtained by projecting from the original feature-space associated with an instance. This builds on the intuition that it is easier to explore in a low-dimension space and therefore to locate new instances. Furthermore, it opens up the possibility of scaling the method to situations where the feature-vector contains a large number of features[1] by enabling all features to be captured in a low number of dimensions in which it is meaningful to measure distance between points. The contribution of this work is to evaluate an approach to transforming the novelty space into a lower dimension, and to show that by using an ensemble of models that create different transformations, we are able to significantly improve instance space coverage.

We test our hypothesis in the zero-one knapsack domain. A thorough set of experiments examines whether the length of the original feature-vector transformed to 2D influences results. Our findings show that in fact the methods are *complementary*: NS conducted in various 2D spaces is able to find instances in regions of the instance-space not covered by NS conducted directly in the original space and vice-versa, while there are some areas of the space that both methods cover. We conclude that using an *ensemble* of methods results in the widest coverage of the space.

We provide an overview of related work, followed by some brief background regarding NS. This is followed by a detailed description of the methods employed and analysis of the experimental results.

2 RELATED WORK

As noted in the previous section, existing approaches to instance-generation tend to focus either on generating discriminatory instances or space-filling. With respect to the former, Smith-Miles *et al.* [27] evolved instances for TSP that are either easy or hard for two heuristic solvers. In the knapsack domain, a similar approach is used to evolve instances that are discriminatory with respect to a portfolio of solvers [24], while in bin-packing, a large set of instances are evolved to discriminate between four heuristic solvers [2]. However, while successful in delivering discriminatory instances, the approaches just mentioned make no effort to also deliver diversity, hence new instances can be concentrated in small regions of the potential space. Some attempts have been made to *implicitly* address this: in [14], a new selection method is proposed that favours offspring that maintains diversity with respect to a chosen feature (as long as the offspring have a performance gap over a given threshold), while Bossek *et al.* [4] proposed novel mutation operators for the TSP domain which encourage exploration of the feature-space while still optimising for discrimination without explicit diversity preservation methods.

[1]For example, the R package `salesperson` returns more than 150 features for the TSP.

Algorithm 1: Novelty Search

Input: N, k, $MaxEvals$, $portfolio$, pca
initialise($population$, N);
evaluate($population$, $portfolio$);
archive = $\emptyset$;
for $i = 0$ *to* $MaxEvals$ **do**
  parents = select($population$);
  offspring = reproduce($parents$);
  offspring = evaluate($offspring$, $portfolio$, $archive$, k, pca) (see Algorithm 2);
  population = update($population$, $offspring$);
  archive = update_archive($population$, $archive$);
  solution_set = update_ss($population$, $solution\_set$);
end
return $solution\_set$

On the other hand, another line of work from Smith-Miles *et al.* [26] focuses directly on evolving *diverse* instances that fill regions of an instance-space where there are currently gaps, without considering portfolios of solvers. Specifically, an instance-space is created by first describing instances using a high-dimensional feature-vector, and then projecting those instances into a 2D space, e.g. using Principal Component Analysis (PCA) [26]. A target location in the 2D space is identified and then an EA is used to search for instances that match the target. However, there is no guarantee the evolved instances will be discriminatory with respect to a portfolio of interest.

There are two recent examples of attempts to evolve instances that are both diverse and discriminatory, addressing weaknesses in the approaches described above, both of which use methods from the growing field of QD and discussed in Section 1. A MAP-Elites algorithm was used to generate new discriminatory TSP instances that were diverse with respect to multiple combinations of two features chosen by the authors in [5]. NS was used by Marrero *et al.* [20] to generate new knapsack instances that are diverse with respect to an 8-dimensional feature-vector while discriminatory with respect to a portfolio of four EA solvers. Clearly there is scope for improvement. Existing QD literature indicates that defining suitable descriptors for both QD methods (i.e. to define the axes of the archive in MAP-Elites or the novelty descriptor in NS) is both crucial and challenging, and can have significant influence on the dynamics of the algorithm [25]. More recent work in the QD field in the context of robotics and generative design has indicated that searching in a transformed low-dimensional space derived from the solution-space can also improve results [10]. Here we propose to use PCA to transform a high-dimensional feature-vector in 2D and measure novelty in this space. While alternative dimensionality reduction approaches are possible (e.g. autoencoders [29] or UMAP [21]), we selected PCA given that it is both quick to train and computationally inexpensive to calculate the transformations of new instances produced by the EA.

3 NOVELTY SEARCH: BACKGROUND

The paper builds directly on the work of Marrero *et al.* [20], which uses NS to discover instances that are novel w.r.t instance features, and show discriminatory behaviour regarding a portfolio of solvers.

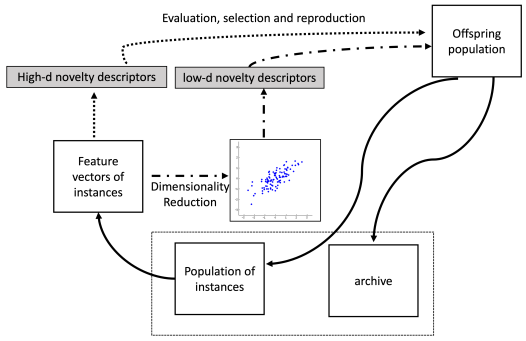

Figure 1: Flow of NS algorithm: dotted lines show the execution of the NS_f method (high-dimensional novelty vector), while dashed lines show the run of the NS_{PCA} method (low-dimensional projected novelty vector). Solid lines show common paths. The low-dimensional projection model is pre-trained before the execution of each NS method.

NS was first introduced by Lehman *et al.* [18] as an attempt to mitigate the problem of finding optimal solution in deceptive landscapes, with a focus on the control problems. The core idea replaces the objective function in a standard evolutionary search process with a function that rewards novelty rather than a performance-based fitness value to force exploration of the search-space.

An overview of the algorithm proposed by Marrero [20] is given in Algorithm 1. In brief, a population defining new instances is evolved that are discriminatory with respect to a target algorithm selected from a portfolio. Uniform crossover and Uniform One mutation are applied to create new child instances. The evaluation function (Algorithm 2) assigns a fitness value based on a linearly weighted combination of *novelty* and *objective fitness*. The former measures the novelty of an instance with respect to a user-defined *descriptor* that denotes a set of characteristics of a solution, e.g. a set of features describing the instance. For each solution, novelty is calculated as the average distance between its descriptor and that of its k nearest neighbours. Neighbours are calculated w.r.t the current population *and* an archive of previously discovered solutions, which is updated at each generation of the algorithm and provides a historical record of places visited in the search-space. On the other hand, objective fitness is calculated as the difference in performance between the target solver and the next best solver in the portfolio, with the goal of maximising this value. After each iteration, the archive of past solutions is updated in two manners. Firstly, we randomly select a sample of individuals from the current population and insert it in the archive with a probability of 1% following common practice in the literature [28]. Then, whichever individual from the current generation with a novelty score greater than a pre-defined threshold t_a is also included to the archive. The algorithm returns a *set* of solutions.

The process is depicted diagrammatically in Figure 1. As noted in Section 1, the key modification to the original approach proposed in this paper is to alter the descriptor used to calculate novelty. We pre-train a dimensionality reduction method to reduce the feature-vector described in [20] to two dimensions. Hence, each time a new instance is produced, its feature vector is calculated and then it is projected to 2D using the pre-trained model. This projection is then used in the novelty calculation in step 11 of Algorithm 2.

Specific details of the dimensionality-reduction approach used, as well as details of the algorithm portfolio and instance definition are given in the next section.

Algorithm 2: Evaluation

Input: $offspring$, $portfolio$, $archive$, number of nearest neighbours k, low-dimension model pca
for *instance in* $offspring$ **do**
  for *algorithm in* $portfolio$ **do**
    apply *algorithm* to solve *instance* R times;
    calculate mean profit of *algorithm*
  end
  if NS_{PCA} **then**
    descriptors ← reduceDimensionality($offspring$, pca)
  else
    descriptors ← featureVector ($offspring$)
  end
  calculate the novelty score(*descriptors*, *archive*, k);
  calculate the performance score($offspring$);
  calculate fitness($offspring$);
end
return $offspring$

4 METHODS

This section provides specific detail on each of the steps of the approach described above.

4.1 Instance Representation

We apply the approach to generating instances for the zero-one Knapsack Problem (KP) domain, a commonly studied [8, 23] combinatorial optimisation problem with many practical applications. The KP requires the selection of a subset of items from a larger set of N items, each with profit p and weight w in such a way that the total profit is maximised while satisfying a constraint that the weight remains under the knapsack capacity C. Each instance is described by an array of real numbers of size $2 \times N$, where N is the dimension (number of items) of the instance of the KP we want to create, with the weights and profits of the items stored at the even and odd positions of the array, respectively. The capacity C of the knapsack for each new individual generated is defined as 80% of the sum of the weights of all items since using a fixed capacity would tend to create insolvable instances. This value can be tweaked to create easier or harder instances to be solved for each solver. We evolve randomly generated and fixed size instances containing $N = 50$ items, hence each individual describing an instance contains *100* values describing pairs of (profit, weight). In addition, upper and lower bounds were set to delimit the maximum and minimum values of both profits and weights.[2]

[2]The description of an instance follows the general method of [24], except that they converted the real-valued profits/weights to a binary representation.

Table 1: Pearson correlation coefficients P_{f,t_p} of each feature f to the target performance t_p.

Feature f	P_{f,t_p}
C	0.849306
std	0.186009
mean	0.084504
min_p	0.054122
max_w	0.036205
min_w	0.007127
avg_eff	-0.005272
max_p	-0.054382

4.2 Algorithm Portfolio

While the portfolio can contain any number or type of solvers; i.e Marrero *et al.* used a portfolio of EA configurations [19], we restrict experiments to a portfolio containing a set of four deterministic heuristics described in [24]. *Default* (Def) selects the first item available to be inserted into the knapsack; *Max Profit* (MaP) sorts the items by profit and selects those items with largest profit first; *Max Profit per Weight* (MPW) sorts the items according to efficiency (ratio between the profit and weight of each item) and selects those items with largest ratio first; and finally, *Min Weight* (MiW), which selects items with the lowest weight first. The reason behind this portfolio definition is to lighten the computational work required for the experimental evaluation. The selected KP heuristics are considerable fast algorithms in comparison to previous portfolios of EA solvers [20]. Moreover, the deterministic nature of the solvers allows an instance to be evaluated only once, instead of performing several repetitions per instance and solver.

4.3 Novelty Descriptors

As noted, two different types of novelty descriptors are defined.

- **High-dimension**: An n-dimensional descriptor defined by Marrero *et al.* [20] where $n \in \{4, 6, 8\}$.
- **Low-dimension**: A 2D descriptor which is calculated by projecting an n-dimensional feature-vector to two dimensions using a pre-trained dimensionality reduction model. To investigate the influence of the reduction in dimensionality, we evaluate models that project from 8D, 6D and 4D to 2D.

For the high-dimensional methods, the 8D descriptor is defined using a set of 8 commonly used features from the literature: *capacity of the knapsack (C); minimum weight/profit (min_w, min_p); maximum weight/profit (max_w, max_p); average item efficiency (also known as correlation); mean distribution of values between profits and weights (mean); standard deviation of values between profits and weights (std).* The 8D descriptor uses the 8 features just described. For 6D and 4D descriptors, n features are selected in each case by ranking the original 8 features in terms of their correlation to the target performance (t_p) across the training set (see Table 1) and choosing the n most correlated in each case. Therefore, 6D descriptors are constructed with *C, std, mean, min_p, max_w, min_w* features, while 4D descriptors consider *C, std, mean, min_p.*

The target performance t_p is calculated as the mean profit achieved in R repetitions by the target algorithm. As is usual in the KP domain, *profit* is defined as the sum of the profits of those items included into the knapsack.

Table 2: Parameter settings for NS which evolves the diverse population of instances.

Parameter	Value
Knapsack items (N)	50
Weight and profit upper bound	1000
Weight and profit lower bound	1
Population size	10
Crossover rate	0.8
Mutation rate	$1 / (N \times 2)$
Generations	1000
Repetitions (R)	1
Distance metric	Euclidean Distance
Neighbourhood size (k)	3
Threshold (t_a)	3.0, 0.0001

To obtain the low-dimension descriptors, we use PCA trained on a large dataset. This dataset was obtained by running the NS_f algorithm from Marrero *et al.* [20] to generate instances that are discriminatory w.r.t the four deterministic solvers described above using a 8D feature vector. 600 instances were generated per solver giving a training set of 2,400 instances each with an 8D feature-vector associated. As PCA assumes that there is correlation among features, Barlett's Test of Sphericity was applied to the 2,400 instance dataset to check this assumption holds. The p-value obtained ($p = 1.367e - 311$) confirms that the assumption of correlation is valid. All features were standardised by removing the mean and scaling to unit variance. PCA was then applied to the scaled data to obtain a 2D projection of an n-dimensional feature vector on a set of training instances. We trained three models to project from {8D, 6D, 4D} feature-vectors to 2D. All models were trained using the same data-set by considering the n features selected as described above. Each time a new instance is discovered by the search process, its n features are calculated and then these are projected to 2D by the corresponding pre-trained model to calculate novelty (see Algorithm 2, step 7). It is important to note that all PCA procedures map the descriptors to a 2D space for two main reasons: for faster calculations in NS_{PCA}, and to obtain clearer visualisations of the instance distribution.

5 RESULTS

Each of the two NS algorithms (NS_f for the high-dimensional space and NS_{PCA} for the 2D projected space) was run for 1,000 generations and 10 times for each of the four target heuristics. The instances for each run per target were combined. The parameter ϕ describing the performance/novelty balance was set to 0.85 in all cases. Parameters for both approaches were taken directly from [20] and are given in Table 2. The only exception arises regarding the threshold values given in the last row of the said table. This parameter determines the minimum novelty score for new instances to be inserted into the archive and to be included in the final solution set. For high-dimensional spaces (such as 8D, 6D and 4D) we followed the steps from Marrero *et al.* [20] and set $t_a = 3.0$. In

contrast, considering NS_{PCA}, we set $t_a = 0.0001$, as the range of values for the novelty scores measured in the 2D space occur on a smaller scale. The values shown were obtained from an extensive parameter tuning evaluation by means of generating 10 evenly spaced numbers in the range $[0.0, 3.0]$ and evaluating the number of instances generated and the space coverage (more details about its calculation are given below). For a more in-depth analysis on the impact of the parameters, please refer to [20]. Finally, we note that all algorithms were written in C++.[3]

The experimental assessment was conducted to address two main questions using data obtained from six different experiments: three experiments using a high-dimensional feature vector (8D, 6D, 4D) to calculate novelty using NS_f, and three experiments using the 2D projection obtained from the corresponding high-dimensional model to calculate novelty using NS_{PCA}. Particularly, we address the following research questions:

(1) To what extent does each of the six different methods locate instances in different parts of the instance-space?
(2) To what extent is each of the six methods able to discover instances that are discriminatory for each solver in the portfolio?

By examining the results collectively, we then evaluate whether an *ensemble* composed of a mixture of the above approaches outperforms subsets of methods. In the following diagrams and tables, we use the terms NS_f and NS_{PCA} for the high-dimensional and reduced feature-space NS algorithms, respectively, and subscripts {8D, 6D, 4D} to refer to each feature-space. Moreover, subscript c refers to a combination of all runs for each method.

5.1 Coverage

We first provide a qualitative analysis of the results. Instances generated from all methods were combined into a single dataset: this was used to obtain a PCA projection of the entire dataset for easy visualisation. The said projection was created by calculating the 8D feature-vector of every instance (regardless of how it was generated), and using the 8D vector in combination with the instance information as input to PCA. Again, the input was scaled before applying the PCA.

Results are shown in Figure 2. The left-hand side shows the distribution of instances generated by each run of NS_{PCA} from using the three different projection models (8D, 6D and 4D to 2D). As expected, there is some overlap between some methods, particularly in the centre of the space. However, we notice that $NS_{PCA_{6D}}$ and $NS_{PCA_{4D}}$ approaches were able to locate instances in different regions of the space, and therefore *complement* $NS_{PCA_{8D}}$. Figure 2 (middle) shows the same information but for NS_f runs instead. In contrast to the NS_{PCA} runs, the instances for NS_f are considerably more clustered in the middle of the space. $NS_{f_{4D}}$ however does locate some instances in the bottom left corner of the space. Finally, the right-hand side of Figure 2 shows the combination of the previous representations on one diagram. Notice that the instances generated by NS_{PCA} not only are able to cover regions of the space which NS_f cannot, but also some instances from $NS_{PCA_{8D}}$ and $NS_{PCA_{4D}}$ are filling the gaps between the small cluster of $NS_{f_{4D}}$ on the bottom left and the centre of the space.

[3]The source code, instances generated and results obtained are available through a GitHub repository: https://github.com/PAL-ULL/ns_pca_gecco23.

Table 3: Summary of the U metric values and number of instances generated per each method. Contribution shows the percentage of how much each method contributed to the final set of instances.

Method	U	Inst. Generated	Contribution
NS_{PCA_c}	**0.5719**	5320	
NS_{f_c}	0.5348	4363	
$NS_{f_{8D}}$	0.5133	1623	16.76
$NS_{PCA_{8D}}$	0.4701	1801	18.59
$NS_{PCA_{6D}}$	0.4629	1919	**19.81**
$NS_{f_{4D}}$	0.4313	1420	14.66
$NS_{f_{6D}}$	0.4305	1320	13.63
$NS_{PCA_{4D}}$	0.4193	1600	16.52

In order to quantitatively evaluate the extent to which the evolved instances cover the instance space, we calculate the exploration uniformity (U) metric, previously proposed in [16, 17]. This compares the distribution of solutions in the space with a hypothetical Uniform Distribution (UD) and is calculated as follows. First, the environment obtained from all the information available is divided into a grid of 25×25 cells, after which the number of solutions in each cell is counted. Next, the Jensen-Shannon divergence (JSD) [12] is used to compare the distance between the distribution of solutions and the ideal UD. The U metric is then calculated as

$$U(\delta) = 1 - JSD(P_\delta, UD). \quad (1)$$

In Equation 1, δ denotes a *descriptor* associated with a solution. Obtaining a score of 1 proves a perfect uniformly distributed set of solutions. This descriptor is defined as the first two principal components of each solution according to the PCA projection used for visualisation described at the beginning of this section.

Table 3 summarises the number of instances generated and the coverage metric U per each method. In addition, it shows the contribution of each method as a percentage of all filled cells in the 25×25 grid. Figure 3 provides a graphical representation of the distribution of cells covered by each method, following the same layout as Figure 2. The purple colour on the left and middle plots of Figure 3 represents cells of the space that are covered by more than one approach. There are only 30 cells that are filled by at least two of the NS_{PCA} runs, and it can be observed that $NS_{PCA_{6D}}$ covers the larger area in comparison to other approaches. However, for the NS_f approaches, this number rises to 62, with $NS_{f_{8D}}$ representing the approach which covers most space. This provides a clear insight that the instances generated by the NS_f approaches suffer from more overlap when compared to the instances generated by NS_{PCA}.

Table 4 shows the number of instances generated for each target solver in the portfolio by each method after 10 independent runs. Although NS_f seems to generate more instances than NS_{PCA}, the total amount of instances generated by each approach is relatively similar. In addition, Table 5 summarises the number of unique instances encountered in the final combined set. We define unique instances in terms of non-duplicated 8D feature-vectors of

Figure 2: Instances generated for different NS algorithms represented in the same space. Yellow crosses represent the instances from NS_{PCA} projecting 8D to 2D; green squares for NS_{PCA} projecting 6D to 2D; blue circles for NS_{PCA} projecting 4D to 2D; red pluses for NS_f searching on 8D feature space; light-blue triangles for NS_f searching on 6D; and dark-red diamonds for NS_f searching on 4D.

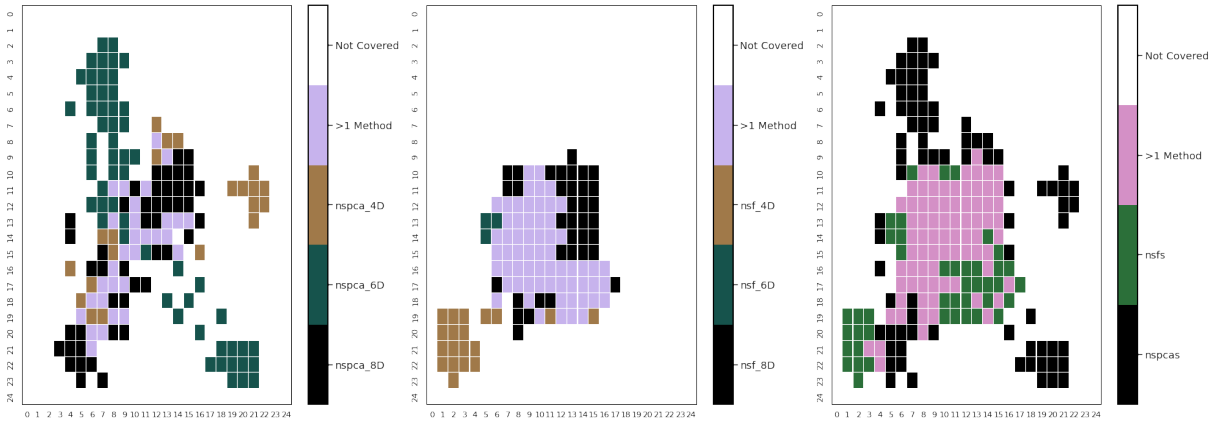

Figure 3: Cells filled by each method in the 25×25 grid obtained from all the information available.

instances. Results show that 31.67% of instances generated for Def solver across all methods are unique (228 out of 720), 53.57% for MPW, 33,84% for MaP and 32,61% for MiW. Results evidence that the generation of discriminatory instances for MPW is considerably difficult across all approaches, while some methods struggle to generate instances for Default solver as well; i.e $NS_{f_{4D}}$ and $NS_{PCA_{6D}}$ provide fewer than 100 instances for Default.

5.2 Instance Overlap

We now provide additional insight into the overlap of instances generated by the different methods. We compare the instances generated for every pair of methods based on their 8D feature vectors regardless of the selected target solver. Results are given in Table 6. Each cell shows the fraction of overlap (op_{ij}) between the set of instances generated by each pair (i, j) of methods, and therefore has a value between 0 and 1. For every (i, j) method pair, the overlap is calculated as $op_{ij} = (total_{i+j} - unique_{i+j})/total_{i+j}$. Here $total_{i+j}$ is the union of 8D feature vectors extracted from those instances generated by methods i, j, while $unique_{i+j}$ are the non-duplicated 8D feature vectors from the previous union set. Notice that although there is some overlap between all pairs, each method is able to generate some instances that no other method produces. There is high overlap between pairs of NS_{PCA} methods, while interestingly, less overlap between pairs of NS_f methods. Now the key takeaway from this table is that the ensemble is beneficial; removal of any single method would reduce overall coverage.

5.3 Discriminatory Ability

Finally, we provide further insight into the *performance diversity* of the evolved instances (see Figures 4 and 5). We follow the method of Marrero *et al.* [20] to calculate the performance score ps of each instance (see Equation 2). Thus, ps is defined by the difference between the mean target performance and the maximum of the mean performance achieved in R repetitions by the remaining approaches of the portfolio defined as o_p.

$$ps = t_p - \max(o_p) \tag{2}$$

Table 4: Summary of the number of instances generated for each target after 10 repetitions of every NS method. The minimum, mean, standard deviation and maximum number of instances generated by each method regardless of the solver are also given.

Method	Def	MPW	MaP	MiW	Min	Mean	Std	Max
$NS_{f_{8D}}$	131	31	774	687	1	40.57	34.73	105
$NS_{f_{6D}}$	100	10	650	560	1	33.0	28	65
$NS_{f_{4D}}$	40	20	570	790	2	35.5	33.85	79
NS_{f_c}	271	61	1994	2037				
$NS_{PCA_{8D}}$	140	21	900	740	1	45.02	38.12	90
$NS_{PCA_{6D}}$	39	20	960	900	2	62.67	43.70	96
$NS_{PCA_{4D}}$	270	10	640	680	1	40	27.96	68
NS_{PCA_c}	449	51	2500	2320				

Table 5: Summary of unique instances generated for each target in the portfolio across all methods. Unique instances are calculated based on non-duplicated 8D feature vectors.

Target	Unique Instances	Generated
Def	228	720
MPW	60	112
MaP	1521	4494
MiW	1421	4357

Table 6: Summary of overlap in instances generated (regardless of the target solver) for every pair of methods. Instances are compared based on their 8D feature vectors. Each cell shows the fraction of overlapping instances.

	$NS_{f_{8D}}$	$NS_{f_{6D}}$	$NS_{f_{4D}}$	$NS_{PCA_{8D}}$	$NS_{PCA_{6D}}$
$NS_{f_{6D}}$	0.35				
$NS_{f_{4D}}$	0.23	0.63			
$NS_{PCA_{8D}}$	0.47	0.84	0.72		
$NS_{PCA_{6D}}$	0.48	0.84	0.72	0.89	
$NS_{PCA_{4D}}$	0.45	0.84	0.70	0.90	0.89

Figures 4 and 5 show the spread in the magnitude of the performance difference as detailed in Equation 2 for instances generated for the MaP heuristic using NS_f and NS_{PCA} for 6D and 4D spaces, respectively.[4] We note that both approaches are able to generate diverse instances in terms of this metric, while a minor number of instances have a relatively small gap. Applying a Shapiro test to check for normality shows that no significant departure from normality was found ($p < 0.05$) for each of the resulting distributions.

Hence, we demonstrate that instances exhibit not only diversity in terms of space coverage but also performance diversity with respect to the solvers.

[4]Plots for the remaining heuristics are available at the aforementioned GitHub repository.

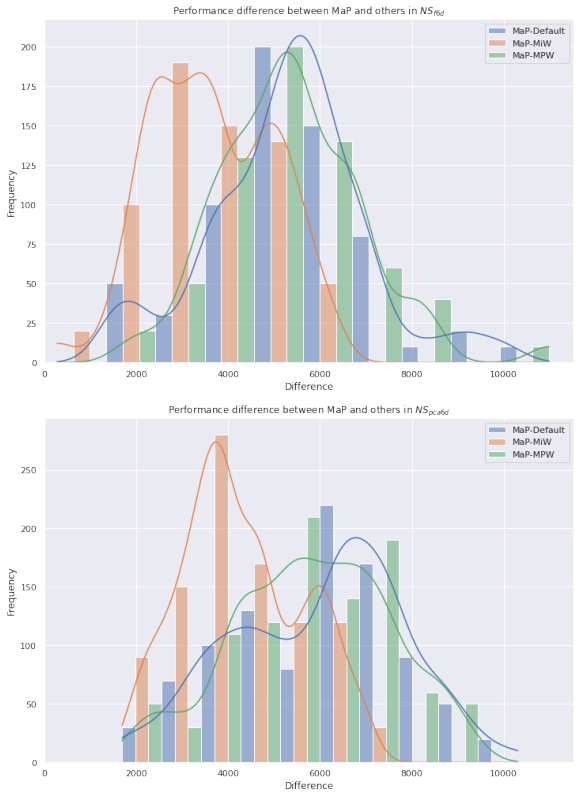

Figure 4: Distribution of performance gap between $NS_{f_{6D}}$ (top) and $NS_{PCA_{6D}}$ (bottom) approaches by considering the instances generated for MaP solver.

6 DISCUSSION

In this article, we proposed a modification to an NS algorithm designed to generate diverse instances with the goal of enhancing its ability to find instances in regions not covered by the original method. The proposed approach searches for novelty in a transformed search-space, obtained by creating a 2D projection of the novelty vector originally proposed in [20]. The results in the previous section clearly demonstrate that measuring novelty in different spaces leads to the identification of instances in different regions of the space. While there is obviously some overlap, the ensemble of methods collectively leads to better coverage.

It is well known in the QD literature that identification of suitable descriptors to define the space in which diversity is measured is crucial [7, 13], with the choice having a strong influence on the results obtained. Furthermore, the QD literature also points to the fact that if the space defined by the descriptors is high-dimensional, then a suitable dimensionality reduction technique is required to maintain a limited number of niches for the algorithm to search. Although never specifically investigated in the context of instance-generation, the generative design literature that often exploits QD techniques has demonstrated that in fact *searching* for an appropriate transformation of the space is beneficial in increasing diversity of solutions found. For example, [13] use a data-driven approach that uses an autoencoder [29] to periodically learn a mapping into

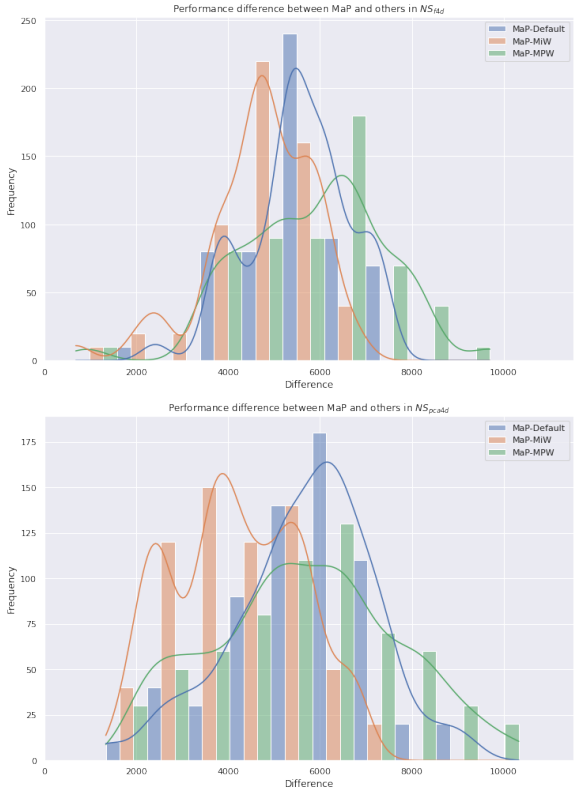

Figure 5: Distribution of performance gap between $NS_{f_{4D}}$ (top) and $NS_{PCA_{4D}}$ (bottom) approaches by considering the instances generated for MaP solver.

a new space from data points discovered during the search while Bossens *et al.* [7] use a *meta-QD* approach that searches for new transformations that then lead to better results if the QD algorithm is applied in the suggested space. Although this is demonstrated to be successful, it has the disadvantage of being extremely computationally expensive. The method we propose takes a more pragmatic approach of directly specifying a set of transformations, evaluating each individually and then combining the results as an ensemble. However, there is considerable scope for investigating more complex methods along the lines just discussed.

We focus on one specific category of transformation, i.e. transforming a high-dimensional space to a lower one. The choice of method used to create the transformation from is clearly important. We elected to use PCA to obtain the 2D projections required, which is a linear projection model. However, considering alternative dimensionality-reduction techniques is important to extend this work. Several candidates exist: the main criteria is that any method must learn a *model* that can be used to find a projection of unseen instances due to the mechanics of the NS algorithm. This rules out the popular visualisation technique t-SNE [31] for example. Potential candidates include UMAP [21], a non-linear method that has tunable parameters, as well as methods from machine-learning such as auto-encoders, although the latter tend to require large volumes of data and have long training times.

7 CONCLUSIONS

This paper proposed an ensemble approach to generating large sets of instances that are diverse with respect to a feature-space and discriminatory with respect to a portfolio of solvers. Our approach builds on previous approaches that showed that QD methods, such as MAP-Elites and NS, are useful in this respect. In contrast to previous work in which novelty was defined with respect to one descriptor defining the space in which novelty is measured, our methods consider a range of descriptors that facilitate search in different spaces. Specifically, we propose the use of new descriptors that are obtained by mapping from a high-dimensional feature-vector to a lower one. This specific choice has multiple advantages. Firstly, it is able to exploit existing dimensionality-reduction methods to create new transformations of existing data. Secondly, it opens up the possibility of applying the method to generating instances that are described by a very large set of features. Current approaches tend to simply select a small number of features (e.g. [5]) which fails to capture the full description of an instance. Furthermore, using a dimensionality-reduction technique to create the transform addresses the potential issue concerning measuring distance in very high-dimensional spaces (see Section 1). Finally, as noted by [7], searching in a space with lower dimensionality reduces the number of niches the algorithm needs to search.

The results clearly demonstrate that searching for novelty in six different spaces (three defined by NS_f and three by NS_{PCA}) covers more of the space than any individual method, and that each of the six methods contributes towards the final collection of instances (see Table 3). Therefore, there is value in using an ensemble approach. Furthermore, we also demonstrate that as well as being diverse, the instances are also discriminatory (Table 4). Note that it is much more difficult to generate discriminatory instances for the two heuristics MPW and Def than MaP and MiW. It is not clear whether this is due to the fact that these heuristics are intrinsically weak compared to the other two and hence there are very few cases in which they outperform the other methods, or whether the algorithm fails to locate them. We leave this for further work.

Future work will be directed towards evaluating the method in additional domains, for example in TSP, where there has already been some interest in generating diverse instances [4, 5]; and in bin-packing, where there has also been some preliminary work in trying to produce diverse instances [3]. As noted in Section 6, an obvious avenue for future investigation is also to consider different methods of learning new descriptors. Finally, we intend to investigate how the method scales to both very large feature-spaces and with the size of the portfolio of solvers.

ACKNOWLEDGEMENTS

The work of **Alejandro Marrero** is funded by the Canary Islands Government "Agencia Canaria de Investigación Innovación y Sociedad de la Información - ACIISI" [contract number TESIS2020010005]. **Emma Hart** is partially funded by EPSRC EP/V026534/1

The authors would like to thank Coromoto León and Frank Neumann for their advice.

REFERENCES

[1] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. 2001. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *Database Theory — ICDT 2001*, Jan den Bussche and Victor Vianu (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 420–434. https://doi.org/10.1007/3-540-44503-X_27

[2] Mohamad Alissa, Kevin Sim, and Emma Hart. 2019. Algorithm Selection Using Deep Learning without Feature Extraction. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 198–206. https://doi.org/10.1145/3321707.3321845

[3] Mohamad Alissa, Kevin Sim, and Emma Hart. 2023. Automated Algorithm Selection: from Feature-Based to Feature-Free Approaches. *Journal of Heuristics* 29, 1 (2023), 1–38. https://doi.org/10.1007/s10732-022-09505-4

[4] Jakob Bossek, Pascal Kerschke, Aneta Neumann, Markus Wagner, Frank Neumann, and Heike Trautmann. 2019. Evolving Diverse TSP Instances by Means of Novel and Creative Mutation Operators. In *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms* (Potsdam, Germany) *(FOGA '19)*. Association for Computing Machinery, New York, NY, USA, 58–71. https://doi.org/10.1145/3299904.3340307

[5] Jakob Bossek and Frank Neumann. 2022. Exploring the Feature Space of TSP Instances Using Quality Diversity. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 186–194. https://doi.org/10.1145/3512290.3528851

[6] Jakob Bossek and Heike Trautmann. 2016. Understanding Characteristics of Evolved Instances for State-of-the-Art Inexact TSP Solvers with Maximum Performance Difference. In *Proceedings of the XV International Conference of the Italian Association for Artificial Intelligence on Advances in Artificial Intelligence - Volume 10037 (AI*IA 2016)*. Springer-Verlag, Berlin, Heidelberg, 3–12. https://doi.org/10.1007/978-3-319-49130-1_1

[7] David M. Bossens and Danesh Tarapore. 2022. Quality-Diversity Meta-Evolution: Customizing Behavior Spaces to a Meta-Objective. *IEEE Transactions on Evolutionary Computation* 26, 5 (2022), 1171–1181. https://doi.org/10.1109/TEVC.2022.3152384

[8] Valentina Cacchiani, Manuel Iori, Alberto Locatelli, and Silvano Martello. 2022. Knapsack Problems — An Overview of Recent Advances. Part I: Single Knapsack Problems. *Comput. Oper. Res.* 143, C (jul 2022), 13 pages. https://doi.org/10.1016/j.cor.2021.105692

[9] Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. 2021. Quality-Diversity Optimization: A Novel Branch of Stochastic Optimization. In *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, Panos M Pardalos, Varvara Rasskazova, and Michael N Vrahatis (Eds.). Springer International Publishing, Cham, 109–135. https://doi.org/10.1007/978-3-030-66515-9_4

[10] Antoine Cully. 2019. Autonomous Skill Discovery with Quality-Diversity and Unsupervised Descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 81–89. https://doi.org/10.1145/3321707.3321804

[11] Antoine Cully and Jean-Baptiste Mouret. 2013. Behavioral Repertoire Learning in Robotics. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation* (Amsterdam, The Netherlands) *(GECCO '13)*. Association for Computing Machinery, New York, NY, USA, 175–182. https://doi.org/10.1145/2463372.2463399

[12] B. Fuglede and F. Topsoe. 2004. Jensen-Shannon divergence and Hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.* 31–. https://doi.org/10.1109/ISIT.2004.1365067

[13] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. 2020. Discovering Representations for Black-Box Optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 103–111. https://doi.org/10.1145/3377930.3390221

[14] Wanru Gao, Samadhi Nallaperuma, and Frank Neumann. 2016. Feature-Based Diversity Optimization for Problem Instance Classification. In *Parallel Problem Solving from Nature – PPSN XIV*, Julia Handl, Emma Hart, Peter R Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter (Eds.). Springer International Publishing, Cham, 869–879. https://doi.org/10.1007/978-3-319-45823-6_81

[15] Wanru Gao, Samadhi Nallaperuma, and Frank Neumann. 2021. Feature-based diversity optimization for problem instance classification. *Evolutionary Computation* 29, 1 (2021), 107–128. https://doi.org/10.1162/evco_a_00274

[16] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. 2015. Devising Effective Novelty Search Algorithms: A Comprehensive Empirical Study. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Madrid, Spain) *(GECCO '15)*. Association for Computing Machinery, New York, NY, USA, 943–950. https://doi.org/10.1145/2739480.2754736

[17] Léni K Le Goff, Emma Hart, Alexandre Coninx, and Stéphane Doncieux. 2020. On Pros and Cons of Evolving Topologies with Novelty Search. , 423–431 pages. https://doi.org/10.1162/isal_a_00291

[18] Joel Lehman and Kenneth O. Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation* 19, 2 (2011), 189–222. https://doi.org/10.1162/EVCO_a_00025

[19] Alejandro Marrero, Eduardo Segredo, and Coromoto Leon. 2021. A Parallel Genetic Algorithm to Speed up the Resolution of the Algorithm Selection Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Lille, France) *(GECCO '21)*. Association for Computing Machinery, New York, NY, USA, 1978–1981. https://doi.org/10.1145/3449726.3463160

[20] Alejandro Marrero, Eduardo Segredo, Coromoto León, and Emma Hart. 2022. A Novelty-Search Approach To Filling An Instance-Space With Diverse And Discriminatory Instances For The Knapsack Problem. In *Parallel Problem Solving from Nature – PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I* (Dortmund, Germany). Springer-Verlag, Berlin, Heidelberg, 223–236. https://doi.org/10.1007/978-3-031-14714-2_16

[21] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software* 3, 29 (2018), 861. https://doi.org/10.21105/joss.00861

[22] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* abs/1504.04909 (2015). https://doi.org/10.48550/arXiv.1504.04909

[23] David Pisinger. 2005. Where are the hard knapsack problems? *Computers and Operations Research* 32, 9 (2005), 2271–2284. https://doi.org/10.1016/j.cor.2004.03.002

[24] Luis Fernando Plata-González, Ivan Amaya, José Carlos Ortiz-Bayliss, Santiago Enrique Conant-Pablos, Hugo Terashima-Marín, and Carlos A. Coello Coello. 2019. Evolutionary-based tailoring of synthetic instances for the Knapsack problem. *Soft Computing* 23, 23 (2019), 12711–12728. https://doi.org/10.1007/s00500-019-03822-w

[25] Justin K. Pugh, L. B. Soros, Paul A. Szerlip, and Kenneth O. Stanley. 2015. Confronting the Challenge of Quality Diversity. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Madrid, Spain) *(GECCO '15)*. Association for Computing Machinery, New York, NY, USA, 967–974. https://doi.org/10.1145/2739480.2754664

[26] Kate Smith-Miles and Simon Bowly. 2015. Generating new test instances by evolving in instance space. *Computers and Operations Research* 63 (2015), 102–113. https://doi.org/10.1016/j.cor.2015.04.022

[27] Kate Smith-Miles, Jano van Hemert, and Xin Yu Lim. 2010. Understanding TSP Difficulty by Learning from Evolved Instances. In *Learning and Intelligent Optimization*, Christian Blum and Roberto Battiti (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 266–280. https://doi.org/10.1007/978-3-642-13800-3_29

[28] Paul A. Szerlip, Gregory Morse, Justin K. Pugh, and Kenneth O. Stanley. 2015. Unsupervised Feature Learning through Divergent Discriminative Feature Accumulation. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15)*. AAAI Press, Austin, Texas, 2979–2985. https://doi.org/10.1609/aaai.v29i1.9601

[29] Michael Tschannen, Olivier Bachem, and Mario Lucic. 2018. Recent advances in autoencoder-based representation learning. In *Third workshop on Bayesian Deep Learning (NeurIPS 2018)*. arXiv. https://doi.org/10.48550/arXiv.1812.05069

[30] Neil Urquhart and Emma Hart. 2018. Optimisation and Illumination of a Real-World Workforce Scheduling and Routing Application (WSRP) via Map-Elites. In *Parallel Problem Solving from Nature – PPSN XV*, Anne Auger, Carlos M Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and Darrell Whitley (Eds.). Springer International Publishing, Cham, 488–499. https://doi.org/10.1007/978-3-319-99253-2_39

[31] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605. http://www.jmlr.org/papers/v9/vandermaaten08a.html

[32] Enrico Zardini, Davide Zappetti, Davide Zambrano, Giovanni Iacca, and Dario Floreano. 2021. Seeking Quality Diversity in Evolutionary Co-Design of Morphology and Control of Soft Tensegrity Modular Robots. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '21)*. Association for Computing Machinery, New York, NY, USA, 189–197. https://doi.org/10.1145/3449639.3459311