Using Exploration Focused Techniques to Augment Search-Based Software Testing: An Experimental Evaluation

Bogdan Marculescu*, Robert Feldt* and Richard Torkar*†
*Blekinge Institute of Technology
School of Computing
Karlskrona, Sweden
†Chalmers and the University of Gothenburg
Dept. of Computer Science and Engineering
Gothenburg, Sweden

***Abstract*—Search-based software testing (SBST) often uses objective-based approaches to solve testing problems. There are, however, situations where the validity and completeness of objectives cannot be ascertained, or where there is insufficient information to define objectives at all. Incomplete or incorrect objectives may steer the search away from interesting behavior of the software under test (SUT) and from potentially useful test cases.**

This papers investigates the degree to which exploration-based algorithms can be used to complement an objective-based tool we have previously developed and evaluated in industry. In particular, we would like to assess how exploration-based algorithms perform in situations where little information on the behavior space is available *a priori*.

We have conducted an experiment comparing the performance of an exploration-based algorithm with an objective-based one on a problem with a high-dimensional behavior space. In addition, we evaluate to what extent that performance degrades in situations where computational resources are limited.

Our experiment shows that exploration-based algorithms are useful in covering a larger area of the behavior space and result in a more diverse solution population. Typically, of the candidate solutions that exploration-based algorithms propose, more than 80% were not covered by their objective-based counterpart. This increased diversity is present in the resulting population even when computational resources are limited.

We conclude that exploration-focused algorithms are a useful means of investigating high-dimensional spaces, even in situations where limited information and limited resources are available.

I. Introduction

Search-based software testing (SBST) applies meta-heuristic search algorithms to problems in software testing [1], [2]. In practice, this means using search-based techniques to optimize quantifiable aspects of the behavior of the system under test (SUT) and/or of the test cases themselves. These aspects are defined in terms of search objectives and there is often a multitude of them. For example one objective might be full (structural) code coverage, another one to reach a high mutation score and a third one that the test case is as short as possible.

Lehman and Stanley [3], draw attention to the limitations of search-based systems that are based solely on the optimization of objectives, particularly in the case of *deceptive* problems. They define a deceptive problem as a problem where one "must seemingly move farther from the goal to ever have the hope of reaching it."

Feldt and Poulding [4] also call for a broadening of the existing notion of search-based technique and support that argument with their finding that Genetic Algorithms (GA), either in their single-objective or multi-objective forms, are the most prevalent algorithm applied in recent work in software testing.

In the general case, the problem of exploring the behavior space becomes a many-objective problem. Ishibuchi et al.[5] define a many-objective problem as a multi-objective problem with four or more objectives.

Our previous work on developing and industrially evaluating an Interactive Search Based Software Testing (ISBST) tool [6], [7] has also focused on objective-based optimization. In that case we relied on domain expertise provided by our industrial partner to provide meaningful objectives and to validate our ISBST tool. This ensured that the objectives selected for the ISBST system were relevant to the domain, and as complete as possible in describing the interesting characteristics of the tests and the SUT.

However, such detailed knowledge of the domain may be unavailable in other companies or prohibitive in terms of the cost or time involved. In such cases, the need arises for a exploration-focused approach that can investigate the behavior space of a SUT with little prior knowledge of the topology of that space. Moreover, if the relationship between the inputs of the SUT and its behavior or objective is non-trivial, we can thing of the SUT as a deceptive optimization problem. Then the search cannot target higher fitness areas of the search space, since the position, and even existence, of such areas cannot be known *a priori*.

We define the "codomain" of a SUT as the set of all possible characteristics of that SUT: all the outputs, or functions of those outputs, are included in this set. Further, we define "observed behavior space", or just "behavior space", as the subset of the codomain into which the measured outputs of the SUT, or any functions of those

outputs, is constrained to fall. It differs from the codomain in that the behavior space only refers to those outputs that are measured or the functions of those outputs that are computed. Additional behavior dimensions may exist that are not measured, not calculated from other measurements, and potentially not even known. As such, the behavior space does not completely describe all the characteristics of the SUT and may be incomplete. Moreover, the mechanisms used for the measurements may be flawed. The result is that the behavior space is fluid, as new dimensions may be added and new values observed.

This creates a problem for objective-based SBST algorithms, since only optimizing those behavior dimensions that are being measured can lead the search away from interesting behaviors. This issue is even more problematic, as what constitutes "interesting behavior" might not be known from the outset, and behaviors can be more or less interesting as the search progresses and more information becomes available. Moreover, using incomplete behavior dimensions is a difficult problem to identify when using an objective-based algorithm. Therefore, any bias induced by the existence of additional relevant behavior dimensions is also difficult to ascertain.

In an ISBST tool, what is needed is a general way to explore the behavior space and the properties of the generated test cases when it is not yet clear what are interesting or even good behaviors that we are looking for.

In recent years a number of such, exploration-focused algorithms have been proposed and in this paper we evaluate their relative merits for interactive, search-based software test generation: Novelty Search [8], Viability Evolution [9], and MAP-Elites or Illumination Search [10]. These algorithms will be compared with an objective-based alternative: a Differential Evolution (DE) algorithm [11]. Differential Evolution was used in previous validations of the ISBST tool in both an industrial and a laboratory setting, so we chose it as a baseline to evaluate exploration-focused alternatives.

In this paper, we show that objective-based and exploration-based algorithms investigate different areas of the test behavior space. Since they cover a wider area of this space, exploration-based algorithms can be used as methods to investigate and maintain population diversity, and therefore enhance existing search-based software testing techniques. Moreover, by enhancing existing techniques with exploration-based algorithms, they would be less vulnerable to incompletely or incorrectly defined search objectives, which is often the case in real-world, industrial testing problems.

In software testing terms, this means that the two approaches identify different behaviors of the SUT and a combined test suite would have greater diversity. Work by Feldt et al. suggests that higher diversity in test suites is linked to higher structural and fault coverage [12], [13].

Next, Section II presents the research questions. Section III provides more detailed information on the context, the implementation of the algorithms included in the study, considerations on how the resulting data was analyzed, and details on the practical execution of the experiment. The results are discussed and analyzed in Section IV, and their implications are discussed in Section V. Sections VI and VII conclude the paper and describe our ideas for future work.

II. Research Questions

This paper studies the following research questions:

RQ1. To what extent do exploration-focused algorithms investigate a different area of the behavior space than the objective-based ISBST tool?

Our hypothesis is that exploration-focused algorithms will tend to investigate a different area of the behavior space than the objective focused approach. Intuitively, this is a result of a lack of pressure to optimize the objectives. If this hypothesis holds, we can use exploration-based algorithms to maintain diversity in the solution population and make the overall SBST system less vulnerable to incomplete or inaccurate behavior objectives. Essentially, instead of requiring the tester to know up front which testing objectives to fulfill and set specific targets for each of them, an exploration-based system can automatically explore the space of system and test behaviors and present a more varied set to the tester from which he/she can then choose.

An interesting question to investigate is that of relevance of the behavior areas investigated. Since an exploration-focused algorithm will have a different set of pressures on the population of test case candidates, it is to be expected that it will investigate different behaviors. Comparisons of candidate solutions in multi-objective problems can be done by means of Pareto efficiency or dominance [14]. It would be interesting to see if all of the behaviors found by means of exploration are dominated by those found by objective-based search.

If exploring the behavior space only results in large numbers of dominated solutions, then exploration is only relevant as a first step in investigating a completely unknown behavior space. Conversely, if exploration based techniques still find solutions that are non-dominated, this can mean that the contribution such a technique can bring extends beyond the first look at the behavior space when specific target values are set for the objectives, or some of the objectives.

RQ2. Do exploration-focused algorithms provide the same benefit to diversity when running with restricted resources, i.e. a reduced number of available optimization steps.

Our previous studies indicate that there is a practical limit on the resources available to an algorithm in an interactive setting such as ISBST. Long waiting times can lead to the human specialist becoming bored or disengaged, with direct repercussions on the quality of their input and of their guidance of the testing system. Our hypothesis is that exploration-focused algorithms can provide a boost to diversity, even with the limited number of optimization steps available.

If this hypothesis holds, exploration-focused algorithms can be used either to complement objective based algorithms, or to replace them where the latter are inapplicable.

III. Experimental Setup

This section will provide an overview of the experiment, along with the research instruments, system under test, methods of analysis, and other practical considerations on the execution of the experiment.

The goal of the experiment is to investigate the areas of the behavior space that each of the algorithms covers and, thus, provide answers to our two research questions. In particular, we want to investigate how the algorithms behave in situations where little initial information is available on the behavior space.

A. Context

The impetus for this work came out of a previous study [7], describing the Interactive Search-Based Software Testing (ISBST) tool. The ISBST tool uses interaction to enable a human domain specialist to guide the search process by weighting a number of objectives based on their relevance at any given time. That work, however, relied on the experience and knowledge of domain specialists to define and validate the search objectives. In practice, this means that the technique is difficult to apply in situations where such detailed domain knowledge is unavailable.

For a more general application of ISBST, we looked at techniques that allow automated exploration of the behavior space, with little *a priori* knowledge of its topology or any domain specific limitations.

The information that is initially available on the behavior space is the number of behavior dimensions. Additional information will be obtained as the search proceeds and new values of each of the behavior dimensions will be observed. Thus, all the information that is obtained, is derived from the exploration itself, and can be considered reliable.

Thus, even if the search objectives are incomplete, the focus on exploration diversity means no potentially interesting SUT behaviors are ignored by the search. In the case of inaccurately defined search objectives, the added population diversity means that relevant behaviors are maintained in the population until the problem is identified and corrected.

B. System under Test

Previous experience also informed our choice of system under test. The selected SUT would have to use a large number of inputs and its behavior would have to be expressed as a many-objective problem.

The system under test selected for the experiment is a k-means clustering algorithm implemented in the Julia programming language [1]. The SUT itself is part of the Julia Clustering package [2], and is actively maintained by its developers.

For the purpose of the experiment, each of the investigated search algorithms will search for a group $n_{inputs} = 61$ inputs, representing a group of $n_{points} = 30$ two-dimensional points, and the $n_{clusters}$ number of desired clusters. The algorithm will employ the SUT, the k-means algorithm from the Clustering package, to create $n_{clusters}$ clusters from the input points.

[1] http://julialang.org/

[2] https://github.com/JuliaStats/Clustering.jl

No.	Objective	Description
1	Number of Clusters	Number of clusters to be found. (The k-means algorithm required the number of clusters to be found as an input).
2	Total Cost	A measure of the distance between the center of a cluster and each element in a cluster. Low values indicate well-defined, tightly packed clusters. High values indicate loosely-defined and indistinct clusters.
3	Number of Iterations	k-means clustering is done in several iterations, until the clusters are stable.
4	Mean Silhouette	The Silhouette is a quantitative way to measure how well each item lies in its own cluster. The Silhouette of each point has a value between 0 and 1, with higher values (closer to 1) indicating that the point lies well within its own cluster and there is no meaningful alternative cluster it could be assigned to. The mean is computed to provide an overview of how well the points belong to their respective clusters across the entire population.
5	Silhouette Range	This is the absolute distance between the lowest and the highest silhouette values found in the current candidate. A high value for this attribute means that the test case contains both well-defined and ill-defined clusters. A low value indicates that the test case contains only one of the two options.
6	Mean Weight	The weight of a cluster is the sum of the weights of all the points within it. For the purpose of this experiment, each point has the same weight: $weight = 1$. Larger clusters, with more widely dispersed points, will get high values for this quality objective. Small, tightly packed, clusters will get low values for this objective.
7	Weight Range	Test cases containing a combination of large and small clusters will obtain a high value with respect to this objective.

TABLE I
Overview of the quality objectives used in the experiment.

The behavior space consists of $n_{objectives} = 7$ dimensions that characterize the resulting clustering. The behavior dimensions are described in more detail in Table I. The behavior dimensions are based on the clustering validation information already provided by the implementation of the k-means algorithm; this does represent a use case where little is really known about the behavior space. The developer/tester might not know which test cases and output behavior that are valid and important for the testing to be of high quality. One potential solution consists of one set of inputs and a set of values for each of the behavior dimensions.

For the objective-based system, each search objective can be written to be maximized or minimized. In our case, we wrote the search objective so that smaller values are better, for all cases. This is done solely to simplify presentation and interpretation of the results.

Note that the number of clusters is both an input and a search objective. An input, because the number of clusters is a requirement of the k-means algorithm. However, it is difficult to know *a priori* how many clusters are needed. So the number of clusters is to be minimized, but test cases with a higher number of clusters may provide benefits with respect to other objectives. The number of clusters must be part of the fitness evaluation

and have an influence on the fitness score, for the system to be able to make such trade-offs.

The Clustering implementation we use as SUT is a good representation of the system that our industrial partner commonly uses: it has a large input space, a many-objective behavior space, and is too vast to be explored manually.

For this evaluation, we have also purposefully ignored domain knowledge about the values and limitations of the behavior dimensions. This would re-create the general case scenario discussed earlier, where the only information regarding the behavior space that is available is the number of dimensions. The practical effect of this decision is to allow exploration-focused algorithms the freedom to investigate the behavior space at will, and provide no negative impact on the objective based system.

C. Research Instruments

All the search algorithms being compared were implemented in the Julia programming language, to ensure that no bias be traced to language specificities. Moreover, this also provided confidence that interfacing with the SUT would not provide additional complications, either during development or while running the experiment.

All the algorithms use the same mechanism to generate an initial population of test cases, and the initial population is the same size for all algorithms: $N_{population} = 100$. The initial population is generated randomly from a uniform distribution, and covers the entire, allowed input space. For this experiment we consider an initial population with floating point values $V_{coord} \in (-1000.0, 1000.0)$ for the point coordinates, and with $n_{clusters} \in (3,11)$.

The exploration-focused algorithms are all evolutionary algorithms and need genetic operators to be applied. To only compare their ability to explore the spec of test behaviors we used the same mutation operators for all of them. The candidate selected for mutation has one of the input dimensions mutated. The mutation is based on a normal distribution, with a high likelihood of a small change, but with larger changes possible. The mutation is bounded by the extrema of the input dimension, so no invalid inputs are obtained.

Note that, while the mutation mechanism is the same for all the algorithms, selecting which candidate to mutate varies and will be explained in more detail for each algorithm.

The stop condition for all the algorithms will be reaching a maximum number of optimization steps n_{steps}. The value of n_{steps} is the same for all the algorithms, thus ensuring a fair evaluation of their capabilities.

The four algorithms that are being evaluated are presented below.

Viability Evolution.

Viability Evolution [9] is a technique that allows users to specify a set of desired values of the behavior dimensions, that the algorithm will evolve towards.

For each dimension of the behavior, a set of boundaries is defined. The initial boundaries are selected to encompass the entire population. The boundaries are then updated to exclude a fraction $f_{excluded}$ of the candidates. In our implementation, each boundary update renders at least $f_{excluded} = 0.33$ of the population unviable.

For each optimization step, one of the remaining candidates is selected and mutated. If the mutant is viable, it is added to the population. If the mutant is not viable, it is discarded. Once the population numbers have returned to previous levels, the boundaries are again updated and the process resumes. The algorithms stops when the current population boundaries match the target behavior boundaries, or the maximum number of optimization steps has been reached.

Viability Evolution employs a family system as a mechanism to ensure population diversity. Each candidate in the initial population becomes the progenitor of a family. Each mutant resulting from the candidate is part of the same family. Selecting which candidate to mutate is based on the family. The chance for a family to be selected is inversely proportional to its size. Once a family has been selected, a candidate for mutation is picked randomly, from a uniform distribution.

As stated previously, the behavior of the SUT is defined in terms of dimensions that are to be minimized. This enables us to simplify the definition of the target behavior for Viability Evolution. It also means that the algorithm will not stop until the maximum allowed number of optimization steps n_{steps} has been reached.

It should be noted that some familiarity with the behavior space is required, to allow a target for Viability Search to be defined in a meaningful way.

Novelty Search.

Novelty Search, is defined by Lehman and Stanley as an evolutionary algorithm that differs from the norm by "replacing the fitness function with a novelty metric" [3]. Solutions that are novel, are then added to an archive for future comparisons. However, defining a meaningful novelty metric is not as simple a task as it sounds. The initial paper describes a maze navigating robot, and defines the novelty metric as the Euclidean distance between the point the current solution reaches, and the points reached by previous attempts, as stored in the archive. Since the authors are looking at behavior in a two-dimensional space, Euclidean distance is a reasonable choice.

In our case, we assume little knowledge of which of the dimensions of the behavior space is relevant. As a result, we would like to evaluate SUT behavior based on all identified dimensions. This turns the relevant behavior space from two-dimensional to many-dimensional. For many-objective problems, however, work by Aggarwal et al. [15] suggests that Euclidean distance is less meaningful. Instead, they suggest fractional distance metrics

$$dist_d^p(x,y) = \left[\sum_{i=1}^{d} ||x^i - y^i||^f\right]^{(1/f)}$$

In the formula above, d is the number of dimensions and f is of the form $1/l$, where l is some integer. Their work argues that fractional distance metrics is more appropriate for high-dimensional data. In our

implementation of Novelty Search we used the fractional distance metric described above. The value, $l = 7$, was based on the number of behavior dimensions.

Novelty Search computes the sparseness for each candidate, as the distance to the k-nearest candidates. The sparseness is then used for transferring candidates to the archive and as a mechanism to select which candidates are to be mutated.

The candidates with the highest sparseness are those considered for inclusion in the archive. A candidate thus selected, is then assessed with respect to its distance from the other candidates in the archive. If the solution between the current candidate and the closest candidate already in the archive is higher than a threshold value $\rho_{threshold}$, the current candidate is included in the archive. The first member of the archive is the candidate with the highest sparseness in the initial population. The value of $\rho_{threshold}$ is based on the distance between the initial member of the archive and the most distant candidate from it, i.e. the highest observed distance in the initial population.

The likelihood that a candidate is selected for mutation is directly proportional to the sparseness value of that candidate.

As with the other algorithms, the Novelty Search algorithms will use the maximum allowed number of optimization steps as its only stopping condition, to ensure a fair evaluation.

We also note that Novelty Search requires that a meaningful measure of "novelty" be provided. This suggests that some understanding of the behavior space is required to ensure correct application of the algorithm. However, a general novelty metric such as the one we use here might be suitable in many cases given a vector of values for the behavior dimensions.

MAP-Elites (Illumination Search).

MAP-Elites is an algorithm proposed by Mouret and Clune [10] to explore a search space while seeking to avoid local optima. Since they define their algorithms as "illuminating search spaces", we have taken to calling it Illumination Search.

The algorithm splits the behavior space into a number of cells, each cell holding at most a single candidate. At each optimization step, a cell is chosen randomly, from a uniform distribution. The candidate within the selected cell is mutated, it is supplied as a test case to the SUT and the behavior recorded. The appropriate cell for the mutant is then found. If that cell is empty, the mutant is assigned to occupy it and the process resumes. If the cell is occupied, the mutant replaces the existing occupant only if its performance is better than that of the occupant. The algorithms then resumes. Note that it needs a definition of performance in addition to the definition of behavioral dimensions.

A few things need to be discussed regarding Illumination Search, and our implementation of the algorithm. First of all, defining performance is not a trivial task, given the multi-dimensional behavior space. We settled on defining performance as Pareto dominance. Thus, a mutant replaces the current occupant of a cell if it is Pareto dominant. In this case, this was possible since information on what would constitute a "better" candidate is available for each dimension. In situations where this information is not available, alternative measurements of performance would have to be defined.

The second issue is that Illumination Search divides the behavior space into cells, but the exact mechanism is not described in detail in the original paper [10]. In our work, we assume limited knowledge of the behavior space, meaning that theoretical extrema cannot be defined for each dimension. Our solution to this problem was to define cell size in terms of the random population generated at the beginning of the algorithm. This is a solution which is similar how Novelty Search decides the value of its threshold from the initial population.

Two versions resulted from this approach. The first divided the space defined by the initial population, and split that into several cells, adding a cell for higher and a cell for lower values for each dimension. This first version provides good resolution for the behavior area covered by the initial population, and allows the search to cover the initial area and identify good solutions within existing maxima and minima. The drawback is that searching outside the extrema of the initial population is problematic with this approach, effectively limiting the search. The search could thus be stymied by an initial population that does not well represent the whole set of behaviors that can be found.

The second version used the initial population to define a cell size, and uses that cell size to classify further output, even outside the min and max values for each dimension as seen in the initial population. This places no limit on the number of cells that can be investigated and does not limit the search. The goal is to preserve the underlying philosophy of MAP-Elites, which is to ensure that each potential solution competes against similar candidates.

Note that, given the way we implemented Illumination Search, the population used to calibrate the cells is an essential factor. Using a more diverse set of candidates to define the cells will likely result in a faster exploration of the behavior space. This resulted in two extra subversions, one using only the initial population of $N_{population} = 100$ candidates to calibrate the cells. The other uses the cumulative population of all the algorithms (investigated prior to Illumination Search) to allow for the purpose. In practical terms, the first is equivalent to the algorithm running on its own. The second is representative of running the algorithm as part of a large system, with other techniques available to provide the missing information.

Differential Evolution.

The traditional, objective-based search algorithm that forms the basis for our comparison comparison is a Julia implementation of the ISBST tool and its Differential Evolution search algorithm [7]. The tool presented there consisted of an interaction focused component, the *Outer Cycle*; and a search-based component, the *Inner Cycle*. Since the *Outer Cycle* is concerned with interacting with the domain specialist, it has no impact on the work presented here.

The *Inner Cycle* consists of a Differential Evolution algorithm [11]. Differential Evolution is a parallel, direct search method. Each potential solution is a vector of real numbers. The initial population is chosen randomly from a uniform distribution, and covers the entire parameter space. New parameter vectors are added by mutation: adding the weighted difference between two population vectors to a third vector. For each target vector $x_{i,G}$, where $i = 1,2,...,N_{population}$ a mutant vector is generated as follows:

$$v_{i,G+1} = x_{r_1,G} + F*(x_{r_2,G} - x_{r_3,G}) \quad (1)$$

where $r_1, r_2, r_3 \in 1,2,...,N_{population}$, are integers, and mutually different, and different from the running index i. F is a real and constant factor $\in (0,2]$ which controls the amplification of the differential variation $(x_{r_2,G} - x_{r_3,G})$.

The result $v_{i,G+1}$ is then subjected to crossover, by mixing its parameters with those of another predetermined vector, and the outcome of this operation is called trial vector. If the trial vector is an improvement over the target vector, it replaces it in the following generation [11].

The crossover rate we used is $cr = 0.5$, the scale factor is $F = 0.7$, and the population size is $population = 100$. The mutation strategy is that proposed by Storn and Price [11]: DE/rand/1/bin. The strategy uses a differential evolution algorithm (DE); the vector to be mutated is randomly chosen (rand); one difference vector is used (1); the crossover scheme is binomial (bin).

To allow the single objective DE to handle multi-objective and many-objective problems, we used the Sum of Weighted Global Averages [16].

This approach normalizes all the values in a generation to an interval between the largest and the smallest values observed for a given objective, both in the current and previous generations. Each solution is assessed and receives a score for each of the quality objectives. The weights are then used to combine the scores into a single fitness value for each candidate.

$$IFF(j) = \sum_{i=1}^{\text{nObjectives}} \text{Weight}_i * \text{Value}_{i,j} \quad (2)$$

where $IFF(j)$ is the fitness value of candidate j, Weight_i is the current weight of the objective i, and $\text{Value}_{i,j}$ is the fitness value of candidate j measured by objective i. The value of $IFF(j)$ is the sum of the weighted fitness values for all $nObjective$ objectives.

For this study, we leave aside the interactive component and assume no intervention from any human domain specialist. As a result, all the objectives have the same, default, weight:

$$Weight_i = 0.5$$

The ISBST serves as a reference, and data obtained from our previous and ongoing empirical evaluations of the ISBST system is used to calibrate the current experiment. We can thus compare the automated search algorithms in this study also to results from manual interaction with the tool by human testers.

D. Analysis of exploration results

The goal of our analysis is to determine the degree to which the exploration-focused algorithms investigate the same (or, conversely, different) areas of the behavior space as the differential evolution component of the ISBST. However, the behavior space is vast and complex, so some simplifications were made to enable a more expressive analysis.

The behavior space was divided into cells based on the maximum and minimum values observed in all the populations resulting from all runs of the algorithms. The candidates in the populations were assigned to the cells. Each of the exploration-focused algorithms was compared against the differential evolution algorithm in terms of the number of cells that overlapped, i.e. cells where both algorithms had at least one candidate present, and the number of cells that were exclusively occupied by candidates from one algorithm.

We assess whether an algorithm has explored a different area of the search space from the objective-based approach based on the number of cells that had occupants from that algorithm but not from the DE.

We had concerns regarding the lack of pressure to provide interesting candidates in the exploration-focused algorithms, as opposed to the clear drive to optimize in the objective-based approach. The vast behavior space means that there is no easy way to determine if the extra behaviors that are explored are meaningful. To address this concern we introduced a new measure: the number of candidates found by an exploration-based algorithm that are not dominated by candidates found by DE.

E. Experiment Execution

To ensure that the algorithm comparison is fair, each had the same amount of 'effort' available to work with. We define effort as the number of optimization steps available to each of the algorithms. This definition is based on the definition of Črepinšek et al. [17], that suggest measuring evolutionary algorithm performance based on the number of fitness evaluations. In our case, each optimization step results in a single fitness evaluation, for all the algorithms, so the measurements are equivalent. We chose optimization steps because they provide a convenient stop condition for the algorithm runs and because we can more easily relate optimization steps to previous uses of the ISBST and, therefore, to practical experience.

We used four values for the available budget of optimization steps, based on relevant values observed in previous, practical assessments of the ISBST system: $nSteps_1 = 250, nSteps_2 = 5000, nSteps_3 = 21500, nSteps_4 = 250000$. A more detailed explanation for the reason for choosing each of the variables can be found in Table II.

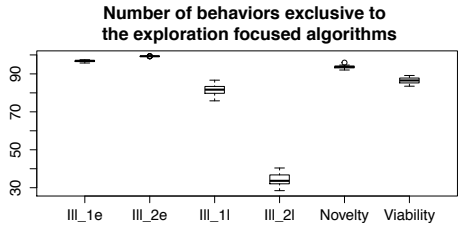
The three values for the budget are based on observed behavior of human domain specialists interacting with the ISBST system. While it is difficult to comment on other values, we will state that we have observed users interact with the system in 45-minute sessions without fatigue affecting their behavior. We will use the observed

Value	Motivation
250	The number of optimization steps between two interaction events in ISBST. The ISBST system performs 250 optimization steps after receiving input from the human specialist. As a result, we use this value to signify the smallest optimization step budget that an algorithm would have available.
5000	Practical experience during the use of the ISBST system indicates that the average number of times that a human domain specialist interacts with the ISBST system in one session is $n_{interactions} = 20$. This value results in a budget of $n_{steps} = 5000$ optimization steps.
21500	The highest number of observed interactions between a human domain specialist and the ISBST system, in one session, was $n_{interactions} = 86$. This results in a budget of $n_{steps} = 21500$.
250000	The equivalent of around 50 specialists interacting with the system for one 45-minute session. This is used as an extreme value.

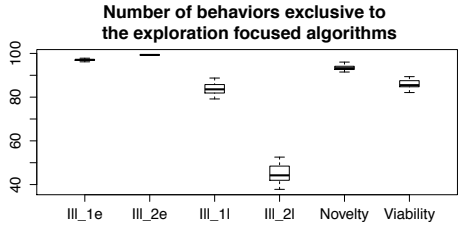
TABLE II
BRIEF EXPLANATION OF THE OPTIMIZATION STEPS BUDGET AVAILABLE TO EACH ALGORITHM.

Designation	Algorithm
Ill_1e	Illumination Search. 1 - with the cells focused on the initial population; e - early, i.e. the cells are based on the initial, random, population.
Ill_2e	Illumination Search. 2 - the initial population defines the size of the cell, but does not limit the number of cells; e - early, i.e. the cells are based on the initial, random, population.
Ill_1l	Illumination Search. 1 - with the cells focused on the initial population; l - late, i.e. the cell are defined based on the extrema of all the populations seen across all algorithms for the current run.
Ill_2l	Illumination Search. 2 - the initial population defines the size of the cell, but does not limit the number of cells; l - late, i.e. the cell are defined based on the extrema of all the populations seen across all algorithms for the current run.
Novelty	Novelty Search
Viability	Viability Evolution

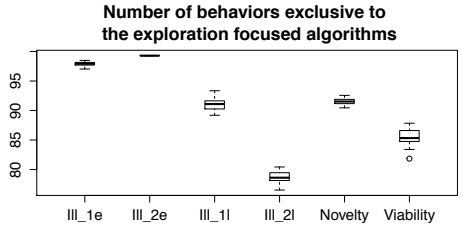
TABLE III
DESIGNATIONS OF THE EXPLORATION-BASED ALGORITHMS, AND (WHERE APPLICABLE) THEIR VARIANTS.



(a) at 21500 optimization steps.



(b) at 5000 optimization steps.



(c) at 250 optimization steps.

Fig. 1. Areas of the behavior space covered exclusively by the exploration based algorithms, measured in percentage of the number of cells in the behavior space.

number of interactions over a 45-minute session as the basis for the available budgets for the evaluation.

For each of the budget values, each algorithm was run 30 times.

IV. Results and Analysis

Throughout this section, we will refer to the different algorithm and algorithm versions by a number of designations. Those designations are clarified in Table III.

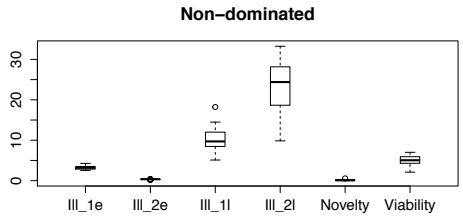
As discussed, we base the analysis on splitting the observed behavior space into a number of cells. The observed behavior space consists of the extreme values in the cumulative candidate population for all of the algorithms. We evaluate each algorithm by comparison against the reference: the objective-based DE algorithm.

A. Exploratory Power

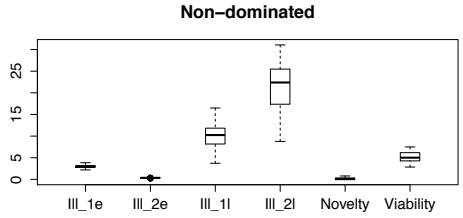
First, a quick overview allows us to see that all the algorithms explore significant areas of the behavior space that do not overlap with those investigated by the objective-based technique.

The comparison between the number of cells that are investigated by the exploration-focused algorithms exclusively, measured in terms of percentage of the total number of cells covered, can be found in Figure 1. From these results we can conclude that regardless of the optimization step budget, exploration-focused algorithms tend to investigate a much larger area of the behavior space than objective-based algorithms.

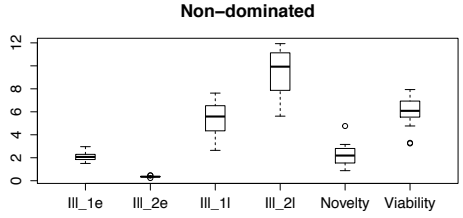
The next step in the analysis is to see how many of these cells are relevant. Since both the search space and the behavior space, are many-dimensional, it is reasonable to expect that there are many areas that an exploration-focused algorithm can explore. It may be worth considering how many of these behaviors are relevant. While it is difficult to assess, we suggest that a potentially interesting candidate is one that is not Pareto dominated by existing candidates. This is computed as follows: for each exploration focused algorithm, a mixed population is created consisting of the candidates developed by that algorithm and the objective-based algorithm. We report the number of non-dominated candidates, from among this mixed population, developed by each exploration focused algorithm.While it is difficult to assess, we suggest that a potentially interesting candidate is one that is not Pareto dominated by existing candidates. This is computed as follows: for each explo-



(a) at 21500 optimization steps.



(b) at 5000 optimization steps.



(c) at 250 optimization steps.

Fig. 2. Number of exclusive and non-dominated candidates developed by each algorithm, expressed at percentages of the total number of occupied cells.

ration focused algorithm, a mixed population is created consisting of the candidates developed by that algorithm and the objective-based algorithm. We report the number of non-dominated candidates, from among this mixed population, developed by each exploration focused algorithm. The numbers of non-dominated candidates produced by each algorithm can be seen in Figure 2, measured as percentages of the total number of occupied cells.

It is worth noting that, while all algorithms seem to investigate different areas of the behavior space from the objective-based system, a relatively small number of the resulting candidates are non-dominated.

This suggests that exploration is a powerful force in terms of investigating behaviors that might not otherwise be observed. Large areas of the behavior space can be investigated, that would not be reached by objective-based means, even in situations where little information is available about the behavior space.

The highest number of non-dominated candidates are obtained by the generous implementations of Illumination Search, that benefit from a more diverse population for calibration. Additionally, Viability Search also produces relatively high number of non-dominated solutions, but this algorithm contains a clear element of optimization that guides the exploration more towards a non-dominated solution front. The downside of these algorithms is that they require a very diverse population, for the former, and a clear set of goals, for the latter.

Of the exploration-based algorithms, the sub-variant of Illumination Search that performs worst in terms of exploring behavior, performs best in terms of generating non-dominated solutions. This seems to indicate that the drive to explore the behavior space and the drive to optimize the objectives are contradicting.

From this, we conclude that exploration-focused algorithms are extremely useful tools for maintaining and increasing population diversity, but less useful when it comes to driving the search towards optima. Given a vast behavior space to explore, it is not unreasonable that these algorithms expend their optimization step budget on expanding the covered area, and thus are less focused on finding optimal values.

The candidate populations developed by most of exploration-based algorithms consist of 80% or more candidates that cover cells of the behavior space that are not covered by the objective-based algorithm.

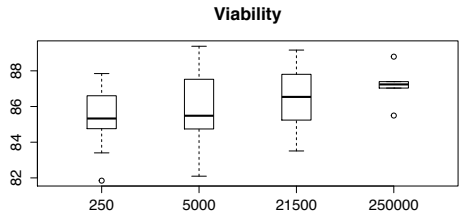
The one exception seems to be one of the sub-variants of Illumination Search. The same variant has, on the other hand, a higher number of non-dominated solutions than any of its peers. This seems to suggest that the exploration focus is not compatible with the drive to optimize.

Moreover, the lack of focus on optimizing a set of given objectives means that the exploration based algorithms are less sensitive to mistakes in the definitions of those objectives. In situations such as those mentioned above, where validating objective completeness and correctness is not possible, exploration-focused algorithms are useful alternative to objective-based algorithms.

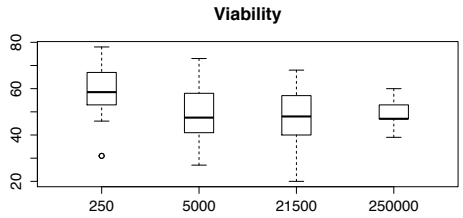
Thus, the answer to RQ1 is that the exploration-based algorithms spend most of their effort investigating areas of the search space that objective-based functions ignore. Each exploration-based algorithms investigates $n_{cells} = 650$ or more cells that objective-based algorithms do not, amounting to 70% or more of the candidates developed.

B. Optimization Step Budget

The second of the research questions posed some concerns on the effect of restricting the available optimization step budget. The hypothesis is that the exploration-focused algorithms exhibit the same benefits even when the optimization step budget is low. This is important to consider given that in our previous evaluations, we determined that the ISBST system's current Julia implementation could perform around $n_{steps} = 250$ optimization steps between two interactions with the human domain specialist. This number is high enough that the domain specialists will see improvement in the candidate population, but low enough to ensure that they will not become bored or disengaged with using the testing tool: usually amounting to fewer than 10 seconds. This number, however, is implementation specific, as

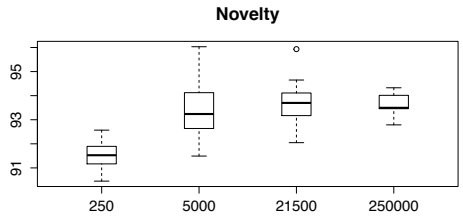


(a) Exclusive cells.

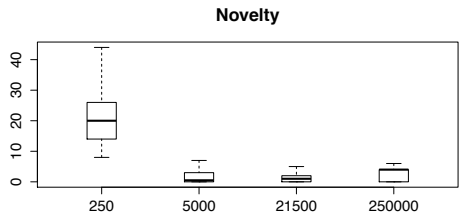


(b) Non Dominated.

Fig. 3. The Viability Evolution algorithm behavior with different number of optimization steps available.



(a) Exclusive cells.



(b) Non Dominated.

Fig. 4. The Novelty Search algorithm behavior with different number of optimization steps available.

different programming languages may be slower, and thus be able to conduct fewer steps in the same amount of time.

Figures 3 and 4 show a closer look at the behavior of two of the algorithms, compared across several values of the available optimization step budget. The two figures show clear differences between the algorithms.

Viability Evolution, shown in Figure 3, shows expected behavior. As the number of optimization steps available increases, more areas are investigated. The growth in the number of cells that are exclusively covered is not proportional to the additional resources available. The number of non-dominated solutions decreases, as the additional optimization steps go towards investigating non-optimal areas of the space. Since the final comparison is done against the entire area covered, and the larger optimization budget is available to the competition as well, a slight drop in the number of non-dominated solutions is to be expected.

Novelty Search, on the other hand, has somewhat unintuitive behavior, seen in Figure 4. The area that it covers also grows, and the same diminishing of the growth rate can be observed. However, in the case of Novelty Search, the number of non-dominated solutions drops significantly. We suggest that this is due to the absence of any pressure to optimize in Novelty Search.

We can see, however, that both algorithms perform their exploration duties well, even with the lower budget available. The answer to RQ2, therefore, is that exploration-based algorithms provide benefits to diversity even when running with a small number of optimization steps. This means that such algorithms can be used in interactive search-based systems, in spite of the reduced resources available.

V. Discussion

We have shown that exploration-based algorithms can help alleviate some of the problems inherent in attempting to search a behavior space where little information is available.

We started from the assumption that it will not always be possible to ensure the completeness and correctness of search objectives for the ISBST system. This is problematic, as incomplete search objectives mean the search can be steered away from potentially interesting candidates, in our case, potentially useful test cases and behaviors.

As an alternative, we focused on exploration-focused algorithms, to conduct an initial exploration of the behavior space.

It must be noted, however, that all of the exploration-focused algorithms discussed in this paper require some knowledge of the domain and of the behavior space; in practice they cannot work effectively without some information about the domain. Novelty Search requires a domain-relevant distance to be used. We used a fractional distance metric, as the most general type of distance for high-dimensional spaces. Undoubtedly, the algorithm will perform better with domain-specific distance metrics.

Viability Evolution reduces the space of viable solutions, trying to guide the population towards optimal values for each of the behavior dimensions. As a result, one must know for what would be optimal values for every behavior dimension. In the most general case, one should be aware of which dimensions are to be minimized and which are to be maximized, and use absolute extrema to update the boundaries. We have chosen to implement this most general case, which likely affected the performance of this algorithm.

MAP-Elites, or Illumination Search, requires that the behavior space be split into cells, based on maximum and minimum values. We supposed that such values are not

available, and defined four versions of this algorithm. All of the versions we defined are somewhat hampered by the reduced information. The ideal case, where the extrema of each behavior dimension are known, will likely yield better results.

We agree that in trying to make all these algorithms completely domain agnostic, their performance may have suffered. Nevertheless, the algorithms have provided a useful mechanism for exploring a high-dimensional behavior space, even with the information about the behavior space being limited.

The approach we used for analysis should also be discussed. Again, we assumed a situation with minimal knowledge of the behavior space. So the analysis relies on the cumulated final population of all the algorithms. This makes comparison more difficult, as the final population is unlikely to be identical in any two runs.

The final population, however, is the result of accumulating a large number of candidates from all the different algorithms. This ensures a diverse population that covers a large area of the behavior space for the SUT. Moreover, each run was conducted 30 times. Given the large number of diverse candidates in these final populations, and the large number of runs conducted without incident, we would argue that the final cumulated population is stable enough to serve as a reference and to allow for a useful analysis.

These findings open the possibility of hybrid SBST and ISBST systems. Exploration would be used to investigate the behavior space, and to define suitable objectives. Optimal solutions, as defined by those objectives, could then be found and proposed. Exploration-based search could also be used as a mechanism for maintaining population diversity, reducing the risk of objective-based search being stuck in local optima.

VI. Conclusions

In this study, we have compared exploration-focused algorithms, Novelty Search, Viability Evolution, and four versions of MAP-Elites or Illumination Search, against an objective-based algorithm, i.e. Differential Evolution.

We have observed that exploration-focused algorithms can investigate the behavior space, even in situations where there is little information available about that space. Not surprisingly, these algorithms are not as effective in driving towards an optimal solution if the tester is interested in a specific part of the behavior of the test or SUT. However, in situations where the objectives to be optimized are incorrect or incomplete or when little is known about what type of behavior the system can or should have, exploration-focused can still be applied and provide important behavior about the SUT and its tests.

In addition, we conclude that the exploration-focused algorithms provide useful results even in situations where there are few optimization steps available. This ability to explore the behavior space, even with limited resources, offers a useful mechanism for an initial exploration of an unknown behavior space. Typically, of the candidate solutions developed by exploration-based algorithms, more that 80% were not found by their objective-based counterpart. Thus we propose that exploration-focused search algorithms can be an important future component in interactive as well as non-interactive search-based software testing systems.

VII. Future Work

This study was driven by the need to find a way to explore the behavior of complex problems, with little information regarding the topology of that behavior space. For a complex problem, one with high dimensional input and output space, it may also be difficult to ensure the validity, correctness, and completeness of any defined objectives. As a result, exploration-focused algorithms provide a useful means of exploring the behavior space, without being affected by any fault in the defined optimization objectives, or even in the absence of optimization objectives.

We propose, therefore, a hybrid type of search. One where exploration is conducted in parallel with objective-based optimization. In the context of the ISBST tool, the human domain specialist can decide, based on their knowledge and their confidence, whether to explore the behavior space for a particular SUT, or to define quality objectives and to search for more clearly optimized candidates.

Alternatively, exploration can be a background process, to ensure that the diversity of the candidate population is maintained and that the objective-based optimization can avoid getting stuck in local optima.

References

[1] P. McMinn, "Search-based software testing: Past, present and future," *Fourth International Conference on Software Testing, Verification and Validation Workshops*, pp. 153–163, 2011.

[2] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Information and Software Technology*, vol. 51, no. 6, pp. 957–976, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2008.12.005

[3] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1162/EVCO_a_00025

[4] R. Feldt and S. Poulding, "Broadening the search in search-based software testing: It need not be evolutionary," in *Search-based Software Testing (SBST), 2015 IEEE Eigth Int. Workshop on*. IEEE, 2015. [Online]. Available: www.robertfeldt.net/publications/feldt_2015_broadening_the_sbst_search.pdf

[5] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, June 2008, pp. 2419–2426.

[6] B. Marculescu, R. Feldt, and R. Torkar, "A concept for an interactive search-based software testing system," in *Search Based Software Engineering*, ser. Lecture Notes in Computer Science, G. Fraser and J. Teixeira de Souza, Eds. Springer Berlin Heidelberg, 2012, vol. 7515, pp. 273–278. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33119-0_21

[7] B. Marculescu, R. Feldt, R. Torkar, and S. Poulding, "An initial industrial evaluation of interactive search-based testing for embedded software," *Applied Soft Computing*, 2014.

[8] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '11. New York, NY, USA: ACM, 2011, pp. 211–218. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001606

[9] A. Maesani, P. R. Fernando, and D. Floreano, "Artificial Evolution by Viability Rather Than Competition," *PLOS One*, vol. 9, no. 1, p. e86831, 2014.

[10] J. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *CoRR*, vol. abs/1504.04909, 2015. [Online]. Available: http://arxiv.org/abs/1504.04909

[11] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1008202821328

[12] R. Feldt, R. Torkar, T. Gorschek, and W. Afzal, "Searching for cognitively diverse tests: Towards universal test diversity metrics," in *Software Testing Verification and Validation Workshop, 2008. ICSTW '08. IEEE International Conference on*, April 2008, pp. 178–186.

[13] R. Feldt, S. M. Poulding, D. Clark, and S. Yoo, "Test set diameter: Quantifying the diversity of sets of test cases," *CoRR*, vol. abs/1506.03482, 2015. [Online]. Available: http://arxiv.org/abs/1506.03482

[14] S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in *Proceedings of the 2007 International Symposium on Software Testing and Analysis*, ser. ISSTA '07. New York, NY, USA: ACM, 2007, pp. 140–150. [Online]. Available: http://doi.acm.org/10.1145/1273463.1273483

[15] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Lecture Notes in Computer Science*. Springer, 2001, pp. 420–434.

[16] P. Bentley and J. Wakefield, "Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms," in *Soft Computing in Engineering Design and Manufacturing*, P. Chawdhry, R. Roy, and R. Pant, Eds. Springer London, 1998, pp. 231–240.

[17] M. Črepinšek, S.-H. Liu, and M. Mernik, "Replication and comparison of computational experiments in applied evolutionary computing: Common pitfalls and guidelines to avoid them," *Applied Soft Computing*, vol. 19, no. 0, pp. 161 – 170, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494614000787