Mutation Operator Characterization: Exhaustiveness, Locality, and Bias

Ralph Moritz*, Tamara Ulrich†, Lothar Thiele† and Susanne Buerklen*
*Robert Bosch GmbH, Schwieberdingen, Germany
Email: firstname.lastname@de.bosch.com
†Computer Engineering and Networks Laboratory, ETH Zurich
8092 Zurich, Switzerland
Email: firstname.lastname@tik.ee.ethz.ch

***Abstract*—When designing an evolutionary algorithm, one question which arises is what a good mutation operator should look like. In order to be able to anticipate which operators may perform well and which may perform poorly, knowledge about the behavior of the mutation operator is necessary. To this end, we formally define three operator properties: Exhaustiveness, locality and unbiasedness. Furthermore, we provide statistical measures that allow to compare operators that work on the same optimization problem. The novelty of our approach is that the properties are formally defined in a unified manner, and that the measures can be calculated on arbitrary decision spaces, only assuming that a distance measure between solutions in decision space is given.**

Tests on a binary decision space using several mutation operators with known properties show that the statistical measures presented in this paper are able to reflect the properties well. Also, the measures are calculated for mutation operators of a more complex problem, namely the cluster partitioning problem. To test the validity of our measures, we introduce an exploration benchmark that measures how well the solutions can move across the decision space when applying a mutation operator to them. Tests on both the binary and the partitioning problem show that our measures reflect the operator behavior well.

***Index Terms*—Mutation Operators, Exhaustiveness, Locality, Bias, Diversity**

I. Introduction

One of the major difficulties when applying randomized search methods such as evolutionary algorithms to complex optimization problems is the choice of variation operators. Their purpose is to generate new solutions based on a selected set of known solutions, possibly consisting of a single solution only. The present paper concentrates on mutation operators which typically generate a new solution from a single one by slightly changing it in a randomized manner. The underlying assumption is usually that good new solutions can be found in the vicinity of good old solutions.

Based on our experience in using evolutionary algorithms for large and complex optimization problems in the automotive domain, defining and selecting mutation operators is a complex task. First of all, the decision space, i.e. the space containing all possible solutions to the problem, typically is very large. Secondly, depending on the class of optimization problems, solutions are represented very differently, e.g. as graph structures, continuous values, partitionings, and any hierarchical combination of discrete and continuous structures. Currently, there is not much support available for designing and evaluating mutation operators for complex decision spaces.

In general, there are two ways to design a mutation operator for a given class of optimization problems: At first, one may start from standard representations such as bitstrings or integer strings which encode each solution by means of a usually complex transformation (genotype-phenotype mapping). For these standard representations, standard mutation operators exist and their properties have been extensively studied. Secondly, one may define a mutation operator directly on the decision space without the need of mapping solutions to standard representations, i.e. the mutation operator is designed such that it changes the solution itself, a process which is independent of how the solution is represented on the computer.

In known studies on standard mutation operators, i.e. mutation operators applied to standard representations such as bitstrings, some properties were found to have a strong influence on the search. First of all, all solutions should be reachable through an iterative application of mutation (exhaustiveness). Secondly, one would like to know how similar a new mutated solution is to its old solution (locality). And third, one would like to know whether the mutation operator has an inherent preference towards a certain subclass of solutions (bias). When using standard representations, emerging difficulties are (a) to find a suitable mapping from such a standard representation to individual solutions and (b) to evaluate the properties of the mutation operator on the space of problem solutions, i.e. if the mapping from standard representations to solutions is complex, the operator behavior on solutions cannot be directly infered from its behavior on the representation. If on the other hand a mutation operator is used that is directly applied to a solution, it is not clear in the first place how to define such an operator and how its properties can be characterized.

In this paper, we formally define important properties of mutation operators, which can characterize operators that are defined via an intermediate mapping of standard representations to solutions as well as operators that directly work on the solutions. In other words, mutation is understood as a translation of one solution into another, no matter whether this happens directly or via the mapping from a standard representation. With this view on mutation, we define decision space independent measures for important properties of muta-

tion operators which enable us to compare different operators that work on the same decision space. In summary, to support the design and evaluation of mutation operators, we make the following contributions:

- We select three important properties that characterize a mutation operator, and formally define these properties in a unified manner.
- For each property, we present a method of calculating it on arbitrary decision spaces.
- We test our properties on some well known binary mutation operators, as well as on a more complex decision space of cluster partitioning problems.
- We introduce a decision space exploration benchmark to validate the importance of our properties.

In the next section, the formal setting will be introduced and a set of properties will be selected. Section III gives a short problem statement, whereas Section IV discusses related work. In Section V, the selected properties are formally defined in a unified manner and for each property, a statistical method to calculate the property in arbitrary decision spaces is given. The experimental setup including the exploration benchmark is given in Section VI, whereas Section VII presents the experiments on binary decision spaces and on the partitioning problem.

II. Selection of Properties

Consider an optimization problem where one or possibly more objective functions $f : X \to \mathbb{R}^m$ shall be minimized. An element $x \in X$ is called a solution. Here, X denotes the set of all feasible solutions, i.e the decision space. We place no assumptions on X, except that a distance measure $d : X \times X \to [0, d_{max}] \subset \mathbb{R}$ is given, where $d(x_1, x_2) \leq d_{max} < \infty$ denotes the distance between solution x_1 and x_2.

The behavior of a mutation operator can be described using ordered pairs of solutions in decision space, where the operator transforms the first solution into the second with a certain probability. This is described with $\mu : X \times X \to [0, 1]$ where $\mu(p, o)$ expresses the probability that a single application of the mutation operator will turn parent $p \in X$ into offspring $o \in X$. The probability only depends on parent and offspring, which excludes dynamic behavior of the operator, i.e. adaptive strategies that change their parameter settings during the search. Furthermore $\mu(p, \cdot) : X \to [0, 1]$ denotes the probability distribution of the offspring of a certain parent p, which must sum up to one, i.e. $\forall p \in X : \sum_{o \in X} \mu(p, o) = 1$.

Also, we will use the notion of a *walk*, as a sequence of elements of the decision space:

Definition 1: A *walk* r_k of length k is a sequence $\{a_n\}$, $0 \leq n \leq k$ of $k + 1$ elements $a_n \in X$.

Note that r_k can contain duplicate solutions. The probability $P(r_k, \mu)$ that a given mutation operator μ produces a certain walk r_k can be calculated as $P(r_k = \{a_n\}, \mu) = \prod_{0 \leq i \leq k-1} \mu(a_i, a_{i+1})$. Finally note that for a given mutation operator and when starting from a fixed solution x, all possible walks of fixed length k that have x as a starting point a_0 form a probability distribution which sums up to 1, i.e. $\forall x \in X, k \in \mathbb{N} : \sum_{r_k = \{a_0 = x, a_1, \ldots, a_k\}} \prod_{0 \leq i \leq k-1} \mu(a_i, a_{i+1}) = 1$.

There has been much discussion about what properties characterize a mutation operator. For example, Eiben and Smith [1] define mutation as a *random, unbiased change*, which means that applying a mutation operator sufficiently many times without any selection pressure should lead to a random point in decision space, see e.g. [2]. This property is called unbiasedness. Note that there are optimization problems where there is some domain-specific knowledge which can be used to bias the mutation operator towards good solutions. Nevertheless, it is useful to first characterize an operator in terms of its bias such that afterwards a problem specific bias can be added in a controlled manner.

Another well established operator property is that in contrast to random search which in one step reaches each solution with equal probability, offspring created by a mutation operator usually have a small distance to their respective parent. Such a local mutation operator allows an algorithm to traverse the decision space in small steps. It has been shown that a local motion through a smooth fitness landscape is advantageous [3], [4]. Note that usually for real-world optimization problems strong causality (as defined by Rechenberg, see [5]) holds between decision and objective space, i.e. similar solutions (according to the chosen distance measure) also have similar objective function values. This property assures that good solutions can be found in the neighborhood of other good solutions. Again, even if locality is not a desired mutation operator property in a certain optimization context, reliable information about this property is helpful for designing a suitable mutation operator.

Finally, another important property of a mutation operator is to assess whether each solution is reachable from any other solution in an iterative application (exhaustiveness). If this condition is satisfied, it can be guaranteed that all solutions in the decision space are reachable from randomly initialized populations by applying the mutation operator.

In conclusion, we here select the three properties *exhaustiveness*, *locality* and *unbiasedness* to characterize mutation operators.

III. Problem Statement

In the last section we described three properties that characterize a mutation operator. Exhaustiveness is binary, i.e. an operator either is or is not exhaustive. If it is not exhaustive, the amount of the decision space that is reachable can be quantified to distinguish different operators. Locality and unbiasedness can be expressed as continuous measures. One goal of this paper is to find definitions of these properties that satisfy the following requirements:

- They are computable on arbitrary decision spaces, only assuming that a distance measure between solutions is given.

- They place no further assumptions on the representation or on the mutation operator themselves.
- They allow the comparison of arbitrary mutation operators based on the above three properties.

Furthermore, the paper will provide efficient methods to determine the above properties and compare operators defined on the same decision space.

IV. Related Work

Little attention has been paid to measure the exhaustiveness of an operator. Nevertheless, exhaustiveness which is also called ergodicity of an operator is considered as an important property [6], [7]. Often, an informal definition is given to characterize exhaustiveness.

Locality is generally considered as an important aspect of mutation operators. Often, locality is defined as the property of a representation, and it states to what degree the distance between two solutions in the genotypic space (i.e. the space spanned by the chosen representation) corresponds to the distance of the two solutions in the phenotypic space (i.e. our decision space X), see e.g. [8], [9], [3]. It is assumed that a mutation operator only introduces small changes in the genotype of a solution, and using a local representation, this results in a small phenotypic change. Note that this definition of locality does not contradict the setting described in Section II. In this paper, we only care about changes in the phenotypic space, as we do not restrict our measures to operators that use representations or those who do not. Some authors measure locality as the influence of small genotypic changes on the objective values of the solutions, see e.g. [10]. This is not contrary to our problem setting, as the distance measure between solutions in decision space can be defined as their objective value difference. Gottlieb and Raidl [9] calculate locality by generating random solutions, mutating them once and then calculating the distance between original and mutated solution. They herewith assume that it is possible to sample the decision space evenly, or at least that the operator shows similar behavior all over the decision space. Both might not always be the case in real-world problems. Nevertheless, this approach is the closest one to the approach presented in our paper.

There are also some attempts to measure bias. Gottlieb and Raidl [9] sample random solutions and analyze them in some problem-specific manner in order to find out what type of solution is more frequently generated. Bullock [11] specializes in a bounded, one-dimensional, real valued decision space, where he finds that several standard mutation operators have a bias towards the extreme values. Spears [12] considered only string representations where each allele (i.e. each element of the string) can take a value from a given set of values. He then identifies bias as a non-uniform distribution of allele changes. Finally, Rothlauf [8] defined bias as a property of the representation rather than of the operator, stating that an unbiased representation must represent each phenotype with an equal number of genotypes.

To sum up, most existing approaches to characterize exhaustiveness and unbiasedness either place some assumptions on the given decision space or the used representation, or they define formal measures that have to be analytically proven for each problem and each chosen representation/mutation operator pair (e.g. Droste and Wiesmann [2]). In this paper we place no assumptions on the used representations and mutation operators, and we would like to define measures which can be calculated in arbitrary decision spaces based on a given distance measure only. Furthermore, we present the definitions of all three measures in a unified manner, which allows for an interpretation of the relations between the properties.

V. Proposed Definition of Properties

The characteristics of a mutation operator show themselves in walks that are done with this mutation operator, i.e. starting with a given solution, the mutation operator is iteratively applied to the solution, thereby going from one solution to the next. The distribution of the solutions visited during walks contains all the information needed to characterize a mutation operator. For example an exhaustive operator should be able to visit all solutions, no matter from which solution the walk was started. When using an operator with high locality, on the other hand, solutions that are close to each other in the walk sequence should be similar according to the chosen distance measure. Finally, when using an unbiased operator, a walk of sufficient length should cover the decision space well.

A walk can be characterized by the coverage of the solutions it visits. But how can the coverage of the decision space be measured? One standard way is to calculate the diversity of the set. The diversity $D(P) \in \mathbb{R}$ here is defined as a set measure which assigns each multiset P containing elements of X to a real value which indicates the diversity of P. There are many diversity measures available, although not all of them are suitable to measure coverage. Apart from the fact that the diversity measure should not need more information than the distance between all pairs of solutions in P, a suitable diversity measure should have the following properties (according to Solow and Polasky [13], see [14] for a more formal definition of these properties):

- **Twinning:** Adding duplicates should not change D, as duplicates do not add to the coverage. This important property also ensures that the diversity of a multiset is equal to the diversity of the corresponding set.
- **Monotonicity in Varieties:** Adding new solutions (i.e. solutions not yet contained in P) should increase the diversity, as adding a new solution always increases the coverage.
- **Monotonicity in Distance:** If comparing the diversity of two sets A and B of equal size, and the pairwise distance between any two solutions in A is larger than or equal to the pairwise distance between any two solutions in B, then the diversity of A should not be lower than the diversity of B. This property reflects the notion that further apart solutions are always more diverse than close solutions.

Until now we found that the characteristics of a mutation operator show themselves in walks, and that a walk in turn can be characterized by the diversity of the solutions visited during the walk. We therefore propose to use the expected value of the diversity over all walks of length k with equally probable starting solutions as a measure to characterize an operator:

$$E^{Div}(\mu, k) = \frac{1}{|X|} \sum_{\{a_0,...,a_k\}} D(\{a_0, ..., a_k\}) \cdot \prod_{i=0}^{k-1} \mu(a_i, a_{i+1}) \quad (1)$$

where $a_i \in X$. Note that the term $\frac{1}{|X|}$ comes from the fact that each solution has equal probability to be chosen as the starting point of a walk.

A. Exhaustiveness

As stated before, an exhaustive operator must be able to reach every solution by mutating any solution several times, which in turn means that for every pair of solutions $x, y \in X$, there exists a walk $r_k = \{a_0, ..., a_k\}$ of finite length k, whose probability $P(r_k, \mu)$ is strictly larger than zero, such that $a_0 = x$ and $a_k = y$, i.e. which leads from x to y. If any solution y can be reached from any other solution x, then in expectation, an infinite walk starting from any solution, contains the whole decision space. Therefore, we can state that the expected diversity of all infinite walks of an exhaustive operator is $E^{Div}(\mu_{Exhaustive}, \infty) = D(X)$, which is the largest possible value, due to the diversity measure requirement *Monotonicity in Varieties* and *Twinning*, see above. When using a non-exhaustive operator, the diversity of the walks decreases, and hence also the expected diversity. We therefore propose to measure the exhaustiveness of a given mutation operator by calculating the expected diversity of infinite walks:

$$M_E^{exact}(\mu) = E^{Div}(\mu, \infty)$$

In practice, we face a number of problems: First, we cannot do walks of infinite length, so we have to use long walks of length $k \gg 1$ instead. Here, it is unclear how large this k should be such that we can assume that all solutions that can be reached actually have been reached. Second, depending on the complexity of the diversity calculation, we cannot efficiently compute the diversity of very large sets. Third, for large decision spaces the number of initial solutions as starting points of walks explodes.

We therefore propose to use the following approximations: We start by sampling the starting solutions randomly. Here, we denote by S the set containing $|S|$ sampled starting solutions. For each starting solution $s \in S$, we would like to determine the solutions that can be reached from s, and how good their coverage is. To calculate the coverage of solutions reachable from s, we propose to copy each starting solution s r times to yield a starting multiset $C = \{s, ..., s\}$, and then optimize this population according to diversity, i.e. try to find a set $P^* = \text{argmax}_{P \subset X^{(\mu,s)}, |P|=r} D(P)$ of maximum diversity, where $X^{(\mu,s)}$ contains all solutions $s' \in X$ to which at least one walk starting from s exists that has a probability larger than zero. As testing all possible subsets P is infeasible due to combinatorial explosion, we propose a simple greedy strategy to find an approximation of P^*, see Algorithm 1. The algorithm takes a population P, optimizes this population according to diversity, and stops if the diversity could not be improved in a total of c iterations. Subfunction VARIATEPOP mutates each solution in P once. Subfunction SELECTDIV is responsible for finding the subset of size n that maximizes diversity, i.e. $P'' = \text{argmax}_{Q \subset \{P \cup P'\}, |Q|=n} D(Q)$. As testing all possible subsets Q is not feasible, we suggest to use the standard greedy procedure of discarding one solution after another, until the population only contains n solutions. In each step, the solution with the lowest contribution to the diversity of the set is discarded, where the contribution of a solution p_i to the diversity of a set P can be calculated as $D(P) - D(P \backslash \{p_i\})$, i.e. the diversity of the set with the solution minus the diversity of the set without the solution. Finally, the diversity-optimized set P is returned, which can be used to measure how well the decision space can be covered when starting from s.

To summarize, we sample the decision space, and for each sampled solution s, we try to find the maximum diversity of any set P of fixed size that only contains solutions reachable from s

$$M_E^{approx}(\mu, S, r) = \frac{1}{|S|} \sum_{s \in S} D(\text{DIVOPT}(P_{(s,r)}, \mu))$$

where S is a randomly sampled set of solutions and $P_{(s,r)} = \{s, ..., s\}$, $|P_{(s,r)}| = r$ is the set containing r duplicates of s.

Algorithm 1 Diversity optimization DIVOPT. Input: Population P that will be optimized; a mutation operator μ. Output: diversity-optimized population.

```
function DIVOPT(P, μ)
    c = 1000
    n = |P|
    i = 0
    while i < c do
        P' = VARIATEPOP(P, μ)
        P'' = SELECTDIV({P ∪ P'}, n)
        if D(P'') > D(P) then
            P = P''
        else
            i = i + 1
        end if
    end while
    return P
end function
```

B. Locality

A local mutation operator should produce offspring with a low distance to its parent. We propose to measure locality as the expected distance of a single mutation, averaged over all parents, i.e. $\sum_{x_i \in X} \frac{1}{|X|} \sum_{x_j \in X} d(x_i, x_j) \mu(x_i, x_j)$. Due to our diversity measure requirement *Monotonicity in Distance*, the diversity of a set of two solutions is monotone in the distance between these two solutions. Hence, the expected

distance of a single mutation increases monotonically with the expected diversity of two solutions, and therefore the locality of a mutation operator can be expressed using (1) as

$$M_L^{exact}(\mu) = E^{Div}(\mu, 1)$$

which is the expected value of the diversity of all walks of length 1. We can approximate this expected diversity by sampling the distances

$$M_L^{approx}(\mu, n) = \frac{1}{n^*} \sum_{1 \leq i \leq n} d(x_i, x_i^o)$$

where n is the number of samples and for each sample $x_i \in X$, one offspring x_i^o is generated by mutating x_i once. Here, n^* denotes the number of offspring whose distance to its parent is strictly larger than zero. Operators with a bad locality are therefore not able to compensate for that bad locality by producing many neutral mutations (i.e. where the offspring is equal to the parent). This is in concordance with Gottlieb and Raidl [9], who also provide a more in-depth discussion of this issue.

C. Unbiasedness

In contrast to locality where the diversity of short walks (in our case walks of length 1) has to be as small as possible, an unbiased operator should have a high diversity in a reasonably long walk which can be expressed by the measure

$$M_B^{exact}(\mu, k) = E^{Div}(\mu, k)$$

Note that the choice of k depends on the locality of an operator and on the size of the decision space. If k is chosen too small, the locality of the operator is measured. If on the other hand k is chosen too large, we measure the exhaustiveness of the operator.

To approximate this measure, we first determine a set of walks by sampling a set of starting solutions S, and then we generate one walk of length k for each starting solution. We again face the problem that if k is large, it is infeasible to calculate the diversity of the whole walk r_k. We therefore propose to additionally sample the walk in order to approximate its diversity. The approximate measure hence becomes

$$M_B^{approx}(\mu, k, n) = \frac{1}{|S|} \sum_{s \in S} D(R \subseteq r_k, |R| = n)$$

where S is a random sample of the decision space, r_k is a random walk of length k and starting from s, and R is a random sample of r_k of length n. Another advantage of sampling the walk is that regions which are visited more often have a higher probability of being represented in the sample. This is in contrast to exhaustiveness, which measures what regions *can* be reached, whereas the unbiasedness measure M_B^{approx} reflects what regions *typically are* reached when iteratively applying a mutation operator.

VI. Experimental Setup

Before performing experiments, we first need to select a diversity measure. Second, we need to define a sampling method for the decision space and third, we need to define a benchmark function in order to be able to quantify the expressiveness of our chosen properties.

A. Chosen Diversity Measure

Section V lists the requirements a diversity measure must satisfy in order to be applicable for our property calculation. Previous results [14] discuss several diversity measures and conclude that the measure proposed by Solow and Polasky [13] fulfills these requirements best.

The Solow-Polasky diversity measure $D(P)$ of a population $P = \{p_1, ..., p_n\}, p_i \in X$ can be calculated as follows: First, a transformed pairwise distance matrix M is defined, whose elements $m_{i,j}$ are defined as $m_{i,j} = \exp(-\theta \cdot d(p_i, p_j))$, where θ is a user defined parameter and $d(p_i, p_j)$ is the distance in decision space between solutions p_i and p_j. Then, the Solow-Polasky measure is defined as $D(P) = e \cdot M^{-1} \cdot e^T$, where e is the unit vector $(1, ..., 1)$ and e^T is its transpose, i.e. $D(P)$ is the sum of all elements of M^{-1}, the inverse of M.

The Solow-Polasky measure typically returns a real value in the interval $[1, |P|]$, and denotes the number of distinct species found in population P, where close solutions belong to the same species. The parameter θ tunes the relation between distance and number of species. We found that the choice of θ is not sensitive, as long as the values of M are within reasonable bounds, i.e. $10^{-5} \leq m_{i,j} \ll 1$.

B. Decision Space Sampling

All measures require that a representative, unbiased random sample of the decision space can be generated. However, initialization of solutions can sometimes introduce a bias, in which case the sample does not cover the whole decision space evenly. In order to compensate for any sampling bias, we suggest to diversity optimize this random sample using Algorithm 1. Such a diversity-optimized sample then represents a set of solutions which covers the decision space in the best possible manner.

C. Exploration Benchmark

To validate our measures, we introduce a further empirical measure called the *Exploration Benchmark* which directly measures the mobility of the solutions in the decision space. The corresponding hypothesis is that there is a relation between the values of the unbiasedness, locality and exhaustiveness measures of a mutation operator and the ability of an iteratively mutated solution to move freely in the decision space. For example, it is expected that a non-exhaustive mutation operator will not be able to reach certain regions in decision space, a biased operator will often get stuck in its favored regions and a non-local operator will have difficulties to converge towards some specific solution as it tends to make large jumps in decision space.

The exploration benchmark measures the mobility of a solution by giving it a number of random targets $\{t_1, ...t_n\}, t_i \in X$ which the solution must reach in a fixed random order. The faster the solution can complete this tour, the higher its mobility. A detailed description of the exploration benchmark is shown in Algorithm 2. First, a random solution is generated. Then, the solution is iteratively mutated. After each mutation, the distance of the old and the new solution to the current target solution is calculated and the solution closer to the target is selected. If the distance of the solution to the target gets smaller than some predefined value ϵ, the target is considered reached and the solution can move on to the next target. Finally, the total number of iterations needed to reach the last target is stored. Note that only exhaustive operators may be tested with this exploration benchmark, as non-exhaustive operators will not be able to complete any tour that contains targets in regions it cannot reach.

Algorithm 2 Exploration benchmark. Inputs: a population $T = \{t_1, ...t_n\}$ of n target solutions in random order; distance ϵ below which a target is considered reached. Output: the number of iterations c needed to traverse all targets in the given order.

```
function ExplorationBenchmark(T, ε)
    x = random solution
    c = 0
    i = 1
    while True do
        x' = mutate(x)
        if d(x', t_i) ≤ d(x, t_i) then
            x = x'
        end if
        c = c + 1
        if d(x, t_i) < ε then
            if i == n then
                break
            else
                i = i + 1
            end if
        end if
    end while
    return c
end function
```

VII. Experiments

A. Binary Decision Space

In order to test the expressiveness of our statistical measures, we first apply them to well known operators. We consider the 10-dimensional binary decision space, i.e. $X = \{0,1\}^{10}$. A solution $x \in X$ can be represented as a bitstring of length 10, where each bit can have the value 0 or 1. As a distance measure we use the normalized Hamming distance, i.e. the number of different bits in two strings divided by the total number of bits. Mutation works on the bits by flipping them, i.e. by transforming a 0 into a 1 and vice versa. Random initialization of a solution is done by randomly setting each bit to 0 or 1 with probability 0.5. For our tests, we compare the following mutation operators for their exhaustiveness, locality and unbiasedness:

- *rand*: Random Search, i.e. flip each bit with probability 0.5.
- *single*: Single bit-flip, i.e. choose one of the 10 bits at random and flip it.
- *unif1*: Uniform bit-flip 1, i.e. flip each bit with probability $1/10 = 0.1$.
- *unif2*: Uniform bit-flip 2, i.e. flip each bit with probability $2/10 = 0.2$.
- *pref1*: Prefer flip to 1, i.e. flip each 1 with probability 0.1 towards 0 and each 0 with probability 0.3 towards 1.
- *3fix*: The first 3 bits are never changed, the remaining 7 bits are uniformly flipped with probability $1/7$.
- *3lowprob*: The first 3 bits are flipped uniformly with probability 0.01 and the remaining 7 with probability 0.1

TABLE I
Properties of selected binary mutation operators.

Operator	Exhaustive	Local	Unbiased
rand	Yes	No	Yes
single	Yes	Yes	Yes
unif1	Yes	Yes	Yes
unif2	Yes	Yes	Yes
pref1	Yes	No	No
3fix	No	Yes	No
3lowprob	Yes	Yes	No

As these mutation operators are very simple, their properties can be anticipated intuitively, see Table I. In computing our statistical measures we can check whether these anticipated properties are reflected well.

For the experiments, we use the following setup: For each operator μ, we calculate the exhaustiveness measure $M_E^{approx}(\mu, P, r = 10)$, the locality measure $M_L^{approx}(\mu, n = 10)$ and the unbiasedness measure $M_B^{approx}(\mu, k = 5000, n = 200)$. As these are empirical measures, we calculate each one 30 times. In each measure a random initial set P of size 10 is used. For the binary space a random initialization is possible, and therefore no diversity optimization is needed to create this set. Starting with each element of the initial set, M_E^{approx} optimizes the diversity of a set containing 10 copies of the respective element of the initial set until no improvement could be achieved for a total of 1000 iterations. M_L^{approx} mutates each element once and calculates the diversity between mutated and original solution. M_B^{approx} performs a walk of length 5000 from each element of the initial set and randomly samples 200 solutions from this walk. Diversity is calculated using the Solow-Polasky measure with $\theta = 10$. Additionally we assess the performance of the binary operators with the proposed exploration benchmark. Here, ϵ was set to 0.15 and 10 target solutions were created by a diversity optimization. If an operator is not able to reach all targets within 50,000 iterations, the run terminates.

Figure 1 shows the results of the statistical measures. It can be seen that all anticipated properties from Table I are reflected well by the measures, i.e. all operators that have a *No* in Table I have significantly lower values than those with a *Yes*. When inspecting the results more closely, it can be seen that the unbiased operators *rand*, *single*, *unif1* and *unif2* show some unexpected behavior. One would expect that the unbiased

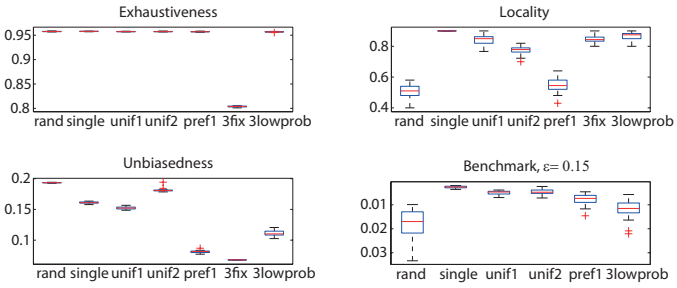

Fig. 1. Boxplots of statistical measures of selected mutation operators for the binary decision spaces. Values are normalized to $[0, 1]$, where larger values indicate that the operator exhibits the property more strongly. Additionally, the results for the exploration benchmark are shown in the lower right (without the non-exhaustive operator *3fix*), where values higher in the diagram are better, i.e. show a smaller number of mutations to reach all target solutions (here a 0/1 in the diagram corresponds to reaching all the targets in the benchmark in 0/50,000 mutation steps).

operators would have similarly high unbiasedness values. The apparent differences in unbiasedness can be explained by the locality of the operators, as locality can be seen as a short-term bias towards the parent solution. Therefore it is not surprising that local operator can never achieve the same unbiasedness value as random search.

The exploration benchmark results show the importance of our properties as the only three operators that are known to be both local and unbiased, i.e. *single*, *unif1*, and *unif2*, also perform significantly better than the remaining operators. What is unexpected here is that *pref1* which has a lower unbiasedness as well as a lower locality than *3lowprob* performs better on the benchmark. The reason for this behavior can be seen in the unbiasedness measure. While the bias of *pref1* always favors the same region of the decision space, *3lowprob* has a short-term, region-specific bias to stay in the larger vicinity around the current solution, given by the first 3 entries in the bitstring. This bias seems to impede the operators performance on the specific exploration benchmark. Note that in this experiment, the walks for calculating unbiasedness were chosen to be of length $k = 5000$, which for a decision space of size $|X| = 2^{10} = 1,024$ is quite large. While a large k is useful to identify long-term bias, smaller values of k should be used to identify such a short-term bias.

B. Partitioning Problem

Partitioning problems arise in a multitude of areas, e.g. in bin packing, task assignment or graph coloring, see [15] for an overview. In a partitioning, we assume that a number of n elements $S = \{s_1, ..., s_n\}$ is given. A partitioning $\mathcal{P}$ then divides these n elements into k non-overlapping clusters, i.e. $\mathcal{P} = \{c_1, ..., c_k\}$, where $c_i \subseteq S$. The clusters must contain all elements, i.e. $\cup_{1 \leq i \leq k} c_i = S$ and no element can be present in more than one cluster, i.e. $s_i \in c_j \wedge s_i \in c_l \Rightarrow j = l$. Note, that k is not fixed, and therefore the decision space contains partitionings which use from one up to n cluster.

A common distance measure for partitionings which we will also use here is the minimal number of elements that have to be removed from S such that the two partitioning of the remaining elements are equal. A good discussion about this distance measure as well as a method of complexity $\mathcal{O}(n^3)$ to calculate the exact minimal number of elements to remove is given in Konovalov et al. [16]. Note that the maximum distance is the number of elements to be clustered minus one, and therefore, we can easily normalize our distance measure to the interval $[0, 1]$.

Now we need to define a mutation operator on these partitioning problems. Falkenauer [15] did extensive research in this area, and compared several common representations and operators on partitioning problems, including the integer representation (where each element is assigned an integer value and elements with the same value belong to the same cluster) as well as representing element permutations (and using a problem dependent decoding scheme to get the actual partitioning). He finally finds that partitionings are best represented directly as what they are, i.e. groups of elements. He also states that mutation operators must be able to create and to eliminate clusters, and may additionally have the ability to shuffle a few elements between their respective clusters. Following Falkenauers work, we here propose to use the following elementary mutation operators:

- Split: Randomly select a cluster and split it into two new clusters by randomly reassigning the elements it contains.
- Merge: Randomly select two clusters and join their elements.
- Move: Randomly move one element to another existing cluster.

On purpose none of these elementary operators is exhaustive when used alone. Split can not reduce the number of clusters while merge and move can not create new clusters. The move operator has the best achievable locality, as it only moves one element per mutation. Split and merge have a worse locality, especially merge, whose locality is governed by the size of the smaller cluster to be merged. Also, split has a bias towards many small clusters, whereas merge has a bias towards a few large clusters. Move also has a bias towards a few large clusters, as it is able to destroy clusters, but not to create new clusters. Of course, different implementations of Split, Move and Merge exist, that differ in these properties.

In the following we consider combinations of these elementary operators. In each mutation step, exactly one elementary operator is executed. To this end, each elementary operator is assigned a certain probability (p_{Split}, p_{Merge} and p_{Move}). Table II lists the combinations we are testing in this paper.

To calculate our statistical measures we consider the case that 20 elements have to be clustered. For all four measures, we use a population size of $|P| = n = 20$. The remaining parameters including the exploration benchmark setup are the same as in Section VII-A. A partitioning is randomly initialized by first uniformly selecting the number of clusters in the interval $[1, n]$, and then randomly assigning the elements to these clusters (but assuring that each cluster contains at least one element). Note that we here do not know whether this random initialization truly samples the decision

TABLE II
CHOICE OF ELEMENTARY OPERATOR COMBINATIONS.

Name	p_{Move}	p_{Split}	p_{Merge}
μ^1	1	0	0
μ^2	0	1	0
μ^3	0	0	1
μ^4	0.5	0	0.5
μ^5	0.5	0.5	0
μ^6	0	0.5	0.5
μ^7	1/3	1/3	1/3
μ^8	0.5	0.25	0.25

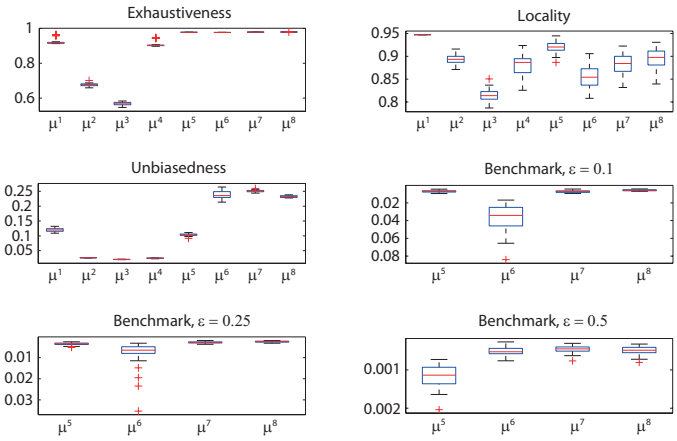

Fig. 2. Statistical measures of selected mutation operators for the partitioning problem. Values are normalized to $[0, 1]$, where larger values indicate that the operator exhibits the property more strongly. Additionally, the results for the exploration benchmark with three different ϵ values are shown in the lower right (only for exhaustive operators). Same normalization as in Figure 1.

space uniformly. Therefore, we diversity-optimize each initial population P before our statistical measures are calculated.

The results for exhaustiveness, locality and unbiasedness can be seen in the upper diagrams of Figure 2. It is easy to discriminate the non-exhaustive operators μ^1, μ^2, μ^3 and μ^4 from the exhaustive operators μ^5, μ^6, μ^7, and μ^8. Among the exhaustive operators, μ^5 achieves the highest locality, but at the same time the lowest unbiasedness value. The reason is that μ^5 uses the same ratio of split and move operations, but a single move can rarely undo a split operation. Therefore, there is a bias towards many small clusters.

The remaining plots in Figure 2 show the results from the exploration benchmark with $\epsilon = 0.1$, $\epsilon = 0.25$ and $\epsilon = 0.5$ for the exhaustive operators μ^5, μ^6, μ^7, and μ^8. It is expected that when using a small ϵ, locality plays an important role (remember, ϵ denotes the distance below which a target is considered reached), whereas for a large ϵ, the importance of locality diminishes and unbiasedness becomes the leading property. This expected behavior can be observed in Figure 2, where the local but biased operator μ^5 performs better on low ϵ benchmarks than on high ϵ benchmarks. The non-local and unbiased operator μ^6 has the opposite behavior, i.e. works well on high ϵ benchmarks but not on low ϵ benchmarks. μ^7 and μ^8, which have both an acceptable locality and an acceptable unbiasedness perform well on all three benchmarks.

VIII. CONCLUSIONS

In this paper we have chosen three properties that characterize mutation operators: Exhaustiveness, locality and unbiasedness. We gave a formal definition of these properties in a unified manner, using the expected diversity of walks done with a given mutation operator. We found that exhaustiveness can be measured by the expected diversity of infinitely long walks, whereas locality can be measured by the expected diversity of walks of length 1. Unbiasedness is positioned in between, i.e. can be measured by the expected diversity of walks of sufficient length. Finally, we presented a way how each of these properties can be calculated on arbitrary mutation operators working on arbitrary decision spaces.

Tests on operators with known properties showed that our measures are able to reflect these properties well. Furthermore, we introduced an exploration benchmark that measures how well an exhaustive mutation operator can move a solution across the decision space. We found that locality is especially useful if the solution should be able to hit a small region, whereas unbiasedness helps if the solution should reach a larger region in decision space in a small number of steps.

In the future, it would be interesting to investigate different types of bias. Also, absolute rather than relative measures would be desirable. Finally, new methods, possibly based on our measures, could be developed to assess the characteristics of adaptive operators.

REFERENCES

[1] A.E. Eiben and J.E. Smith, *Introduction to Evolutionary Computing*. Springer, 2003.

[2] S. Droste and D. Wiesmann, "On representation and genetic operators in evolutionary algorithms," *Evol. Comput.*, 1998.

[3] B. Sendhoff, M. Kreutz, and W.V. Seelen, "A condition for the genotype-phenotype mapping: Causality," in *Conf. on Genetic Algorithms*, 1997.

[4] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[5] I. Rechenberg, *Evolutionsstrategie '94*. frommann-holzboog, 1994.

[6] C. Palmer, "An approach to a problem in network design using genetic algorithms," Ph.D. dissertation, Brooklyn, NY, USA, 1994.

[7] P.D. Surry and N.J. Radcliffe, "Formal algorithms + formal representations = search strategies," in *PPSN*, 1996.

[8] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*. Physica, 2002.

[9] J. Gottlieb and G.R. Raidl, "Characterizing locality in decoder-based EAs for the multidimensional knapsack problem," in *Proc. of Artificial Evolution*. Springer, 1999.

[10] J. Tavares, F.B. Pereira, and E. Costa, "Multidimensional knapsack problem: A fitness landscape analysis," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, no. 3, pp. 604–616, 2008.

[11] S. Bullock, "Smooth operator? understanding and visualising mutation bias," in *Conf. on Artificial Life*. Springer, 2001.

[12] W. Spears, *Evolutionary Algorithms: The Role of Mutation and Recombination*. Springer, 2000.

[13] A.R. Solow and S. Polasky, "Measuring bilological diversity," *Environmental and Ecological Statistics*, vol. 1, pp. 95–103, 1994.

[14] T. Ulrich, J. Bader, and L. Thiele, "Defining and optimizing indicator-based diversity measures in multiobjective search," in *PPSN*. Springer, 2010.

[15] E. Falkenauer, *Genetic Algorithms and Grouping Problems*. Wiley, 1998.

[16] D.A. Konovalov, B.E. Litow, and N. Bajema, "Partition-distance via the assignment problem," *Bioinformatics*, vol. 21, no. 10, pp. 2463–2468, 2005.