Runtime Analysis of Convex Evolutionary Search

Alberto Moraglio
CERCIA, University of Birmingham
Birmingham B15 2TT, UK

Dirk Sudholt
University of Sheffield
Sheffield S1 4DP, UK

ABSTRACT

Geometric crossover formalises the notion of crossover operator across representations. In previous work, it was shown that all evolutionary algorithms with geometric crossover (but with no mutation) do a generalised form of convex search. Furthermore, it was suggested that these search algorithms could perform well on concave and approximately concave fitness landscapes. In this paper, we study the runtime of a generalised form of convex search on concave fitness landscapes. This is a first step towards linking a geometric theory of representations and runtime analysis in the attempt to (i) set the basis for a more general/unified approach for the runtime analysis of evolutionary algorithms across representations, and (ii) identify the essential matching features of evolutionary search behaviour and landscape topography that cause polynomial performance. Our convex search algorithm optimises LeadingOnes in $O(n \log n)$ fitness evaluations, which is faster than all unbiased unary black-box algorithms.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

Keywords

Geometric crossover, recombination, runtime analysis, convex search, pure adaptive search

1. INTRODUCTION

For the No Free Lunch (NFL) theorem [20], no search algorithm is better than any other on all fitness landscapes. This implies that any non-futile theory of search algorithms that aims at proving performance better than random search has to focus on a restricted class of search algorithms and on a corresponding "well-matched" restricted class of fitness landscapes. Matching classes of search algorithms and fitness landscapes is an important open problem in the field.

GECCO'12, July 7–11, 2012, Philadelphia, Pennsylvania, USA.

The class of evolutionary algorithms (EAs) with geometric crossover [11] encompasses and formalises many well-known representation-specific EAs [9]. In recent work [10], Moraglio showed that this class of algorithms (without mutation) does a generalised/abstract form of convex search. The unity in behaviour of EAs across representations calls for a single class of fitness landscapes well-matched to them all – a matching that goes beyond the specific representation. Well-matched "topographic features" of fitness landscapes are those essential features that can be exploited by the common "behavioural features" characterising this class of search algorithms, and produce good search performance. Intuition on continuous spaces suggests that convex search may be well-matched with functions which have a globally concave trend (when maximising). There is some formal evidence that this intuition is correct and that it extends naturally to general metric spaces [10].

Runtime analysis is the standard approach to analyse analytically algorithmic performance. In the last decade it has been applied, with an ever increasing success, to randomised search heuristics and it is establishing itself as a leading theory [14, 1]. Despite its success, it has been difficult to analyse EAs with population and crossover, although there are some results [4, 5, 16, 2, 6]. Furthermore, the analysis is done on a per-algorithm-and-per-problem basis, and approaches that have worked on a algorithm-problem pair do not usually transfer to others. Clearly, a more general approach to determine the performance of a *class* of algorithms on a *class* of problems would be a major progress.

In this paper, we start studying the runtime of a generalised form of convex search on concave fitness landscapes. This is a first step towards linking a geometric theory of representations and runtime analysis in the attempt to set the basis for a more general/unified approach for the runtime analysis of evolutionary algorithms across representations.

We strip to the bare essentials both the specific class of evolutionary algorithms and the matching class of problems considered. Whereas both algorithms and problems are not by themselves of practical interest/relevance, they allow us to identify the essential matching features of evolutionary search behaviour and landscape topography that cause polynomial performance, on problems on which Pure Random Search (PRS) runs in exponential time. Interestingly, their simplicity allows us to draw tight links with the work on Pure Adaptive Search (PAS) [15], an *ideal* algorithm which has been studied since the eighties and which presents an exponential speed-up on PRS on almost all problems. The main open challenge about PAS is finding how to implement

it efficiently. We will also discuss how the results presented here could be extended, in the future, to EAs in actual use and to interesting combinatorial optimisation problems, making this line of theory potentially relevant to practice.

2. ABSTRACT CONVEX EVOLUTIONARY SEARCH

We give some necessary definitions and previous results[1] from [10]. For each definition, we give the general version for a generic metric space, and the corresponding definition specialised to the Hamming distance on binary strings (obtained by substituting the Hamming distance in the general definition, and rephrasing the outcome in terms on representation-specific language, i.e., binary strings, bits and schemata).

A recombination operator is a *geometric crossover* if, for some metric, all offspring are in the metric segment between their parents, for any pair of parents. Many well-known recombination operators across representations are geometric crossovers, e. g., all mask-based crossovers on binary strings are geometric crossovers w. r. t. Hamming distance.

A set of points in a metric space is *geodetically convex* if all metric segments between any two points in the set are entirely contained in the set. In the Hamming space on binary strings, geodetically convex sets coincide with schemata.

The *metric convex hull* of a set of points in a metric space is the smallest geodetically convex set that includes the set. In the Hamming space on binary strings, the metric convex hull of a set of binary strings is the smallest schema (with the least number of '*' symbols) matching all those strings.

Regardless of the specific metric space and associated representation, and of the specific fitness landscape considered, all evolutionary algorithms with geometric crossover (and selection and replacement) but with no mutation do a form of *convex search* (see Figure 1). The term $co()$ denotes the metric convex hull operator, and P_n, P'_n and P_{n+1} are the current population, the set of selected parents, and the offspring population, respectively. Figure 1 shows that the convex hull of the population of offspring is always included in the convex hull of the population of parents. So, the overall search forms a nested chain of convex hulls of populations reducing in size with time. Specifically for the Hamming space, this result says that once a bit is fixed in a population to a certain value (i.e., 0 or 1) that bit will stay fixed in all subsequent populations, or equivalently, that the smallest schemata spanning subsequent populations are increasingly more specific (they can only lose '*' symbols).

On the Euclidean space, intuition suggests that convex search may do well on (approximately) concave functions when maximising [2] (see Figure 2). Various traditional notions of concave function can be generalised to general metric spaces, hence to combinatorial spaces associated with any representation, by replacing the notion of (Euclidean) segment with that of metric segment in the original definitions. However, caution is needed, as the resulting generalisation may not be suitable for combinatorial spaces (only degenerate landscapes on combinatorial spaces may be encompassed

[1]The framework in this section may look related to the field of Convex Optimisation, but it differs fundamentally from it. See [10] for a discussion of the differences.

[2]In this work, we assume maximisation on concave landscapes. Analogous considerations hold unchanged for minimisation on convex landscapes.

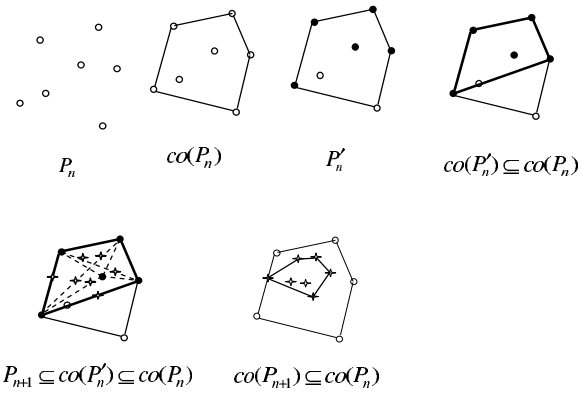

Figure 1: Convex evolutionary search.

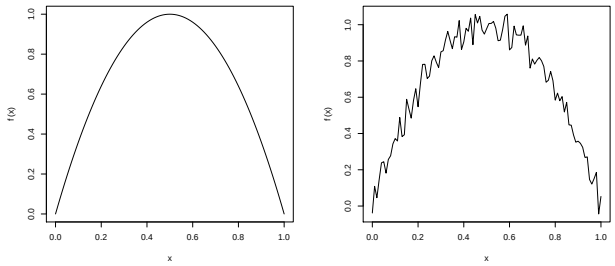

Figure 2: Examples of concave function (left) and function with a concave trend (right).

by the definition, e.g., only flat landscapes). The following well-behaved generalisations were proposed in [10]:

Average-concave landscapes for all $x, y : z \sim U([x, y])$, $E[f(z)] \geq (f(x) + f(y))/2$.

Quasi-concave landscapes for all $x, y : z \in [x, y], f(z) \geq \min(f(x), f(y))$.

They specialise to the space of binary strings with the Hamming distance by considering the Hamming segment in their definition. It can be shown that the OneMax problem is average-concave and average-convex (i.e., it is average-affine), and the LeadingOnes problem is quasi-concave.

Adding an ϵ-bounded perturbation function to the above definitions, we obtain *approximately average-concave* and *approximately quasi-concave* landscapes: $E[f(z)] \geq (f(x) + f(y))/2 - \epsilon$ and $f(z) \geq \min(f(x), f(y)) - \epsilon$. Any function is approximately average/quasi concave for ϵ large enough.

In [10] it was shown that, *regardless of the specific metric, on average/quasi concave landscapes, convex search with geometric crossover and selection produces steady improvements*: for any population, the average/worst fitness of the next population is in expectation/deterministically no worse than the average/worst fitness of the given population (even without selection). This result degrades graciously as the landscape becomes less concave (for increasing ϵ). Whereas this result in general does not imply neither convergence to the optimum nor efficient performance, it is a strong one-step performance result that holds across all representations.

In the reminder, we investigate the possibility of a general runtime analysis of evolutionary algorithms on concave landscapes *across representations* starting from the result above. Is a general result possible *regardless of the underlying space*? Does convex search on concave landscapes have exponentially better runtime than random search *regardless*

of the underlying space? If not, what are the features of the underlying space which affect critically the runtime, making it change from polynomial to exponential? In order to start answering these questions we need further assumptions on the specific algorithm, landscape and performance measure.

3. ALGORITHM, LANDSCAPE AND PERFORMANCE MEASURE

We strip to the bare essentials both the specific class of EAs and the matching class of problems considered. This will allow us to pinpoint the essential mechanism that links matching features of evolutionary search behaviour and landscape topography to polynomial performance, on problems on which random search runs in exponential time.

3.1 Algorithm

The search algorithm considered is the Convex Search Algorithm (Algorithm 1). This is an *abstract/representation-independent* algorithm as it is well-defined over any metric space, as all its algorithmic elements are well-defined on any metric space. The *metric convex hull uniform recombination* returns an offspring sampled uniformly at random from the (metric) convex hull formed by its parents. This recombination operator is a multi-parental recombination operator which like geometric crossover performs a convex search when used in a EA without mutation, regardless of the specific underlying space and representation.

Algorithm 1 Convex Search Algorithm

```
Input: k: population size
Output: individual in the last population

Initialise Population Uniformly at Random
while Population has not converged to the same individual do
    Rank individuals on fitness
    if there are at least two fitness values in the current population then
        remove all individuals with the worst fitness
    end if
    Apply k times Metric Convex Hull Uniform Recombination to the remaining individuals to create the next population
end while
return any individual in the last population
```

The specific convex hull recombination for binary strings with Hamming distance can be formally obtained by plugging in the Hamming distance in the general definition of metric convex hull recombination. It can then be turned into an operational (i.e., algorithmic) definition by re-writing the operator in terms of manipulation of the underlying representation (binary strings). This turns the geometric description into a description of how to act on the parent binary strings to obtain the offspring. These two are different descriptions of the same operator. The operational description of the uniform convex hull recombination for the Hamming space is shown in Algorithm 2.

Algorithm 2 Binary Uniform Convex Hull Recombination

```
for each position in the binary strings do
    if all parents have 1 or 0 at that position then
        the offspring has 1 or 0 at that position, respectively
    else
        (if there is at least a 1 or at least a 0)
        the offspring has 1 or 0 at that position with probability 0.5
    end if
end for
```

The (abstract) Convex Search Algorithm can be formally specified to the Hamming space by replacing the Metric Uniform Convex Hull Recombination with the Binary Uniform Convex Hull Recombination, obtaining the Binary Convex Search Algorithm. This is a variant of Genetic Algorithm with Gene Pool Recombination, with a recombination which does not depend on the frequency of 0s and 1s at each position, and with a particular type of truncation selection.

3.2 Landscape

In section 2, we have considered two types of generalisations of concave landscapes, average-concave and quasi-concave, and their approximately concave extensions. These landscapes are "well-matched" with Convex Evolutionary Search *at an abstract level* because there the average/worst fitness of the offspring population is not worse than the average/worst fitness of the parent population *irrespective of the underlying specific metric and representation*. In this work, we focus on quasi-concave landscapes leaving the analysis of the other notions of concave landscapes to future work.

We define *(canonical) fitness-level sets* as follows. Assume that the codomain of the fitness function is finite, with values $a_0 < a_1 < \cdots < a_q$. Then for $0 \leq i \leq q$ we define the level set L_i as $\{x \in S \colon f(x) \geq a_i\}$ (slightly abusing the term *level set* as L_i includes higher levels). All these level sets form a nested chain of sets by construction: $L_0 \supsetneq L_1 \supsetneq \cdots \supsetneq L_q$. In particular, $L_0 = S$. The sets $(L_0 \setminus L_1), (L_1 \setminus L_2), \ldots, L_q$ form a *fitness-level partition* in the sense defined by Wegener [19].

The following properties hold for quasi-concave landscapes irrespective of the underlying metric and representation:

1. If f is quasi-concave: for all sets $C \subseteq S$, if $z \in \mathrm{co}(C)$ then $f(z) \geq \min\{f(x) \mid x \in C\}$.
2. A landscape f is quasi-concave iff all level sets are (metric) convex sets.

A direct consequence of the first property is that the convex hull recombination on quasi-concave landscapes returns offspring whose fitness is not worse than those of any of its parents, for any underlying metric and representation.

The second property allows us to characterise quasi-concave landscapes constructively: a landscape is quasi-concave iff it is a "Tower of Hanoi" of convex sets. This property is interesting when considered together with combinatorial spaces because they give rise to discrete sets of fitness values. This allows us to check quasi-concavity of a landscape by checking the convexity of all level sets one by one. Also, we can use this property to construct any quasi-concave function for any representation and metric space.

We illustrate this for the Hamming space on binary strings. The idea is to iteratively build a nested chain of convex level sets and subsequently increase fitness values for inner sets. We start from the largest level set and assign the same minimum fitness to all points. We then choose a convex subset of the current fitness level as next fitness level and increase the fitness of all points therein. The process is iterated by further dividing the next fitness level, and so on.

For example, for any x_0 matching the schema `******`, we initially assign, say, $f(x_0) := 0$. Consider the convex subset `0*****` of `******` and let us assign to any x_1 matching this schema fitness values of 1, i.e., $f(x_1) := 1$. So, the quasi-concave function we will obtain has fitness 0 for any string starting with 1 and fitness strictly larger than 0 for any string starting with 0. Now consider the fitness level

0*****. We choose the convex subset 01**1* of 0***** to have fitness value 2, and then choose a further convex subset in 01**1*, and so on. The procedure ends when a convex set with a single element is reached, or earlier, so obtaining a landscape with a plateau. As the LeadingOnes landscape can be built in this way, it is a quasi-concave landscape. It is possible to show that OneMax is not a quasi-concave landscape because its level sets are Hamming balls, which unlike Euclidean balls, are not geodetically convex.

Quasi-concave landscapes are a rather nice class of landscapes, potentially solvable efficiently. However, it is easy to see that the Needle landscape belongs to this class. This shows that this class, in the worst case, is intractable by any search algorithm that does not know a priori information about the position of the needle. What makes Needle intractable is that convex fitness levels are too few and shrink too fast. Therefore we restrict the class of quasi-concave landscapes as follows. A *polynomial quasi-concave landscape* is a quasi-concave landscape with:

1. the number $q+1$ of fitness levels is polynomial in n (problem size, $n = \log(|S|)$).
2. the rate between sizes of successive fitness levels is an inverse polynomial: $r := \min\{|L_{i+1}|/|L_i| \mid 0 \leq i < q\} \geq 1/\text{poly}(n)$.

A polynomial quasi-convex landscape has two characteristic parameters q and r. It is easy to verify that LeadingOnes is a polynomial quasi-concave landscape (it has n fitness levels, and for each level the area of the next level is half of the area of the previous level), whereas Needle is a quasi-concave landscape but not polynomial quasi-concave (it has only two fitness levels, but the area of the second level is exponentially smaller than that of the first level).

3.3 Performance Measures

We say that Convex Evolutionary Search has converged when the whole population contains copies of the same search point. As the algorithm does not always converge to the optimum, we are interested in estimating the probability that the algorithm converges to a global optimum (PC). We also look at the runtime conditional on convergence (RT). A lower bound on PC and a deterministic upper bound on RT give an upper bound on the expected time a global optimum is found when restarts are used. Restarting Convex Evolutionary Search after RT generations yields an upper bound of RT/PC on the expected total number of generations.

The number of function evaluations is by a factor of k larger than the number of generations. The former might be a more fair performance measure as it excludes the possibility that a polynomial number of generation is achieved by means of an exponentially large population. So we are most interested in settings where $k\cdot$RT/PC is polynomial.

The class of search algorithms and fitness landscapes considered are well-defined for any underlying metric space and associated representation. In their definitions, the underlying space appears as a parameter, so the class of search algorithms and fitness landscapes can be considered as functional forms that can be instantiated to a specific space by functional application to it. At an abstract level, we could imagine that an abstract search algorithm is "run" on an abstract fitness landscape *before* specifying the underlying specific space in both algorithm and landscape, obtaining a performance that is itself a functional form on the underlying space, i.e., in which the underlying space appears as a parameter. This functional way of interfacing general and specific results allows us to separate neatly the part of the analysis which is valid for any underlying space and associated representation (i.e., a general theory essentially resulting from matching the abstract notions of convexity of the search algorithm and the concavity of the landscape), and pinpoint which particular features of the underlying space and representation have an effect on the performance. The space-specific values of these features can then be measured and plugged in the general expression of the runtime result, as any other parameter, to obtain the specific runtime for the specific space. This way of shaping a theory has the benefit of systematising/automatising the runtime analysis for any new space and associated representation.

4. PURE ADAPTIVE SEARCH

Pure Adaptive Search (PAS) (see Algorithm 3) is an ideal algorithm, which is in general not implementable efficiently, that has been studied analytically since the 80s in the field of Global Optimisation. This search algorithm requires a search operator able to sample offspring uniformly at random in the level set corresponding to the fitness of the best solution found so far. As Pure Random Search (PRS), which at each iteration samples offspring uniformly at random in the entire search space, the performance of PAS does not depend on the structure of the space S but only on the frequency distribution of the values in the range of the objective function f. This makes PAS a representation-independent algorithm, well-defined on any metric space and across representations. PAS has a well-developed general theory [21], which essentially says that on almost all functions, PAS is exponentially better than PRS. This result is conceptually interesting because it shows that the two simple ingredients of (i) getting at each step better offspring, and (ii) getting them uniformly distributed in the improving set are *sufficient conditions* to produce an exponential speed up on random search performance, very generally, on any search space and virtually on any problem. Interestingly, the result above holds also for a number of relaxations of PAS which are closer to implementable algorithms, e.g., Hesitant Adaptive Search [21]. In this algorithm offspring can land in the improving fitness level only with a certain probability, and the probability distribution of the offspring may deviate from uniformity (in some restricted ways).

Algorithm 3 Pure Adaptive Search

1: Pick an initial point X_0 uniformly at random.
2: $i = 0$
3: **while** optimum not found **do**
4: Generate X_{i+1} uniformly at random on the level set $S_i = \{x : x \in S, f(x) \geq f(X_i)\}$ (improving set).
5: $i = i + 1$
6: **end while**

The big challenge about PAS is being able to find efficient implementations of it. Looking for general efficient implementations of PAS working on any problem is a chimera, as it would imply RP = NP. So, it makes sense to look for specific efficient implementations of PAS when applied to restricted classes of problems. This objective can be approached the other way around: starting from a randomised search heuristic and a class of problems and showing that

the search heuristic can be seen as an implementation of PAS for that class of problems. This would at the same time (i) give a general and fundamental explanation of why that specific search heuristic is efficient on the class of problems considered (i.e., because it can be reduced to PAS) (ii) show that PAS is implementable efficiently for a specific class of problems (i.e., because the specific search heuristic can be implemented efficiently). In this work, we will show that convex search on concave landscapes is essentially an implementation of PAS, linking for the first time the PAS framework to Evolutionary Algorithms.

The following theorem gives the runtime of PAS and PRS on the particular class of function considered in this study.

THEOREM 1. *Consider fitness levels $L_0, \dots, L_q$ and let $r := \min\{|L_{i+1}|/|L_i| \mid 0 \leq i < q\}$ as well as $s := \max\{|L_{i+1}|/|L_i| \mid 0 \leq i < q\}$.*

The expected running time of PAS is at most q/r. In particular, the expected running time of PAS on any polynomial quasi-concave landscape is polynomial.

The expected running time of PRS is at least s^{-q}. If $s = 1 - \Omega(1)$ and $q = n^{\Omega(1)}$ this time is exponential.

PROOF. The expected runtime of PAS can be upper bounded by $1/\Pr(L_1 \mid L_0) + 1/\Pr(L_2 \mid L_1) + \cdots + 1/\Pr(L_q \mid L_{q-1})$. As each term is at most $1/r$, we get an upper bound of q/r. For any polynomial quasi-concave landscape we have $q \leq \text{poly}(n)$ and $1/r \leq \text{poly}(n)$, hence we get a polynomial upper bound.

The hitting probability of PRS can be written as $\Pr(L_0) \cdot \Pr(L_1 \mid L_0) \cdot \cdots \cdot \Pr(L_q \mid L_{q-1})$, which is upper bounded by s^q. The expected hitting time of PRS is hence at least s^{-q}. □

5. RUNTIME ANALYSIS OF CONVEX EVOLUTIONARY SEARCH

In this section we first propose a general runtime result for convex search on quasi-concave landscapes that holds across spaces and representations. This result applies in principle to *any* metric space and representation. In order to illustrate its applicability, we show how the general result can be specialised to determine the runtime of convex search on quasi-concave landscapes for three specific spaces: Boolean spaces endowed with the Hamming distance, the space of integer vectors endowed with the Hamming distance, and the space of integer vectors endowed with Manhattan distance.

Before presenting the formal statement and proof of the general result, we give an informal description of the idea. The reasoning to determine the runtime of a successful convex search is as follows. The initial population comprises a number of points sampled uniformly at random from the search space (i.e., L_0). Then selection is applied which removes all points in the population with the worst fitness value. The remaining points are typically uniformly distributed at random on L_1 (by rejection sampling).

If the convex hull of these selected points covers L_1 completely, the convex hull recombination of these points generates the offspring population uniformly at random on L_1. The mentioned condition is met with high probability on combinatorial spaces provided the population size is sufficiently large[3]. Then the cycle is repeated, until the optimum has been found. Assuming that the convex hull of the selected points at a previous level always covers the next level (or, in rare cases, a further one), the algorithm always conquers a new fitness level at each iteration. Then, the runtime of the algorithm is bounded by the number of non-optimal fitness levels, q. As the algorithm performs uniform sampling on each traversed fitness-level set, in such a successful run, this makes the link to PAS explicit.

Note that the only space-specific element in this reasoning is the (worst-case) probability that a convex set in the specific space is covered by the convex hull of a number of points sampled uniformly at random in the convex set. This probability is important as it affects the required size of the population for the optimum to be reached with high probability, and it can be passed as the only required space-specific parameter to the general runtime expression.

5.1 A General Runtime Bound

We formalise these ideas. With regard to a given metric space, we say that a set P *covers* a set C if $\text{co}(P) \supseteq C$.

DEFINITION 2. *Given a metric space $M = (S, d)$, let $\mathbf{C_M}$ be the set of all geodetically convex sets on M. We define $\mathrm{P}_M^{\mathrm{Cov}}(m)$ as the worst-case probability that the convex hull of the set P of m points drawn uniformly at random from a geodetically convex set $C \in \mathbf{C_M}$ covers entirely the set C:*

$$\mathrm{P}_M^{\mathrm{Cov}}(m) = \min_{C \in \mathbf{C_M}} \Pr\left(\text{co}(P) = C \mid P = U_m(C)\right).$$

Note that the worst-case covering probability is monotone in the number of samples m as additional samples can only increase the convex hull:

$$\forall m' \geq m \colon \mathrm{P}_M^{\mathrm{Cov}}(m') \geq \mathrm{P}_M^{\mathrm{Cov}}(m).$$

The following lemma gives a lower bound on the probability that the next fitness level is covered completely.

LEMMA 3. *Assume fitness levels $L_0, \dots, L_q$ where $r = \min\{|L_{i+1}|/|L_i| \mid 0 \leq i < q\}$. Consider a (parent) population P'_t and suppose for some fitness level $0 \leq i < q$ we have $\text{co}(P'_t) = L_i$. Let the new population P_{t+1} result from k-times Metric Convex Hull Uniform Recombination of P'_t and let $P'_{t+1} = \text{sel}(P_{t+1})$ be the next parent population. The probability that $\text{co}(P'_{t+1}) = L_j$ for some $i < j \leq q$ is at least*

$$\mathrm{P}_M^{\mathrm{Cov}}\left(\frac{kr}{4}\right) - \exp\left(-\frac{9kr}{32}\right).$$

PROOF. Let j be the largest index such that $P_{t+1} \subseteq L_{j-1}$. (Typically $j = i + 1$.) Note that all offspring are still distributed uniformly in L_{j-1}. And as $P_{t+1} \cap (L_{j-1} \setminus L_j) \neq \emptyset$, selection will remove all points in $L_{j-1} \setminus L_j$, leading to $\text{sel}(P_{t+1}) \subseteq L_j$. The probability of covering L_j is at least $\mathrm{P}_M^{\mathrm{Cov}}\left(\frac{kr}{4}\right)$, provided that at least $\frac{kr}{4}$ offspring are in L_j.

The probability for this event can be estimated by standard Chernoff bounds [13]. The expected number of offspring in L_j is $k|L_j|/|L_{j-1}| \geq kr$. The probability that less than $kr/4$ offspring fall in L_j is at most

$$\exp\left(-kr \cdot (3/4)^2 \cdot 1/2\right) = \exp\left(-\frac{9kr}{32}\right).$$

The claim then follows by the union bound. □

[3]On continuous spaces, this condition is never met. However, this condition is sufficient and not necessary to guarantee that the optimum is reached. So, this does not necessarily imply that the convex search algorithm cannot be efficient on continuous spaces.

Lemma 3 easily generalises, using that all probabilities for covering different fitness levels are independent and taking the union bound for error probabilities $\exp\left(-\frac{9kr}{32}\right)$. We also take into account the probability of covering L_0 in the initialisation. It is at least $\mathrm{P}_M^{\mathrm{Cov}}(k) \geq \mathrm{P}_M^{\mathrm{Cov}}\left(\frac{kr}{4}\right)$.

THEOREM 4. *Assume a quasi-concave fitness function on any metric space M with fitness levels $L_0, \ldots, L_q$ where $r = \min\{|L_{i+1}|/|L_i| \mid 0 \leq i < q\}$. The Convex Search Algorithm with population size k finds a global optimum within q generations and kq fitness evaluations with probability at least*

$$\left(\mathrm{P}_M^{\mathrm{Cov}}\left(\frac{kr}{4}\right)\right)^{q+1} - q \cdot \exp\left(-\frac{9kr}{32}\right).$$

5.2 Boolean Spaces & Hamming Distance

We specialise the general runtime result to the space of binary strings endowed with the Hamming distance HD. The specialised result is the runtime of the convex search algorithm specialised to this space (the convex search algorithm with the binary uniform convex hull operator, Algorithm 2) on the quasi-concave landscape specialised to this space (the class of binary landscapes presented in section 3.2).

We first bound the probability of covering any set.

LEMMA 5. *For $M = (\{0,1\}^n, \mathrm{HD})$ and $m \in \mathbb{N}$ we have*

$$\mathrm{P}_M^{\mathrm{Cov}}(m) \geq (1 - 2^{-m+1})^n.$$

PROOF. For every geodetically convex set $C \in \mathbf{C_M}$ for the specific case of $M = (S, \mathrm{HD})$ a necessary requirement for the convex hull of m points uniformly drawn in C not covering C is that there is a bit that is fixed to the same value in all m samples. The probability for fixing a specific bit is 2^{-m+1} as the bit needs to be set to either 0 or 1 in all m uniform samples. The probability that no bit is fixed is $(1 - 2^{-m+1})^n$. □

Then the bound from Theorem 4 simplifies as follows.

THEOREM 6. *Let $M = (\{0,1\}^n, \mathrm{HD})$ and the assumptions for Theorem 4 hold. Then the Convex Search Algorithm finds a global optimum within at most q generations and kq fitness evaluations with probability at least*

$$\left(1 - 2^{-\frac{kr}{4}+1}\right)^{n(q+1)} - q \cdot \exp\left(-\frac{9kr}{32}\right)$$
$$\geq 1 - (q+2)n \cdot 2^{-\frac{kr}{4}}$$

where the second bound requires $2^{\frac{kr}{4}-1} \geq n(q+1)$.

PROOF. The first bound follows directly from Theorem 4 by plugging in Lemma 5. If $2^{\frac{kr}{4}-1} \geq n(q+1)$ we use $(1-x)^y \leq 1 - xy/2$ for $0 \leq xy \leq 1$ [7, Lemma 1] to estimate $(1-2^{-\frac{kr}{4}+1})^{n(q+1)} \leq (q+1)n \cdot 2^{-\frac{kr}{4}}$. We also have $2^{-kr/4} \geq e^{-9kr/32}$ since $2^{-1/4} \geq e^{-9/32}$. Further, $q \leq n$ as at least one bit is fixed with every fitness level. So $q+(q+1)n \leq (q+2)n$. Putting everything together gives the second bound. □

In particular, we can easily derive how large the population must be in order to guarantee that Convex Evolutionary Search with restarts finds a global optimum. If $k \geq 4\log(2(q+2)n)/r$, the probability bound from Theorem 4 is at least $1/2$. This gives the following.

COROLLARY 7. *If $k \geq 4\log(2(q+2)n)/r$ then Convex Evolutionary Search restarting after q generations finds a global optimum within $2q$ expected generations and $2kq$ expected fitness evaluations.*

On polynomial quasi-concave landscapes, as both q and $\frac{1}{r}$ are polynomial in n, the number of generations and the population size are polynomial. For LEADINGONES $q = n$ and $r = 1/2$, so with $k \geq 8\log(2(n+2)n) \approx 16\log n$ the expected number of generations with restarts is at most $2n$.

COROLLARY 8. *Convex Evolutionary Search with population size $k = 8\log(2(n+2)n)$ and restarts optimises* LEADINGONES *in $O(n \log n)$ function evaluations.*

This is remarkable as EAs using only mutation need at least $\Omega(n^2)$ function evaluations [17]. The same holds for all unary unbiased black-box algorithms [8], whereas binary black-box algorithms run in time $O(n \log n)$ [3].

5.3 Integer Vectors & Hamming and Manhattan Distances

We give two more examples of specialisation of the general runtime result. We consider two metric spaces on the same set: the Hamming distance HD and the Manhattan distance MD on integer vectors $S_d = \{0, 1, \ldots, d-1\}^n$, a natural space e. g. for colouring problems [18].

Before presenting the specialisation of the runtime, we explain the specialisation to these spaces in order to understand the space-specific forms of the convex search algorithm and the class of quasi-concave landscapes. Both metrics are product metric spaces; distances between two vectors can be written as a dimension-wise sum of (unidimensional) distances between their elements in each dimension. The unidimensional distance corresponding to the Manhattan distance is the absolute value of the difference between two integers, and for the Hamming distance is the "discrete" distance over a set of integers, which is 1 for any two different integers, and 0 otherwise. The product structural property of these spaces allows us to build geometric elements (e.g., segments) in these spaces by doing the cartesian product of uni-dimensional segments across all dimensions.

Space structure: For (S_d, HD), the neighbourhood graph associated with each dimension is a clique with d nodes as any two non-coinciding integers (i.e, nodes) are at distance one (i.e., they are direct neighbours and connected by an edge). The neighbourhood structure of (S_d, HD) is the cartesian product of clique graphs across dimensions. For (S_d, MD), the neighbourhood graph associated with each dimension is a line graph of d nodes (i.e, an integer line ranging from 0 to $d-1$). The neighbourhood structure of (S_d, MD) is the cartesian product of line graphs across dimensions (i.e., "discretised" hyper-boxes).

Segment: For (S_d, HD), unidimensional segments are edges, i.e., degenerate segments that are made of only the end-points of the segments. For (S_d, MD), unidimensional segments are integer intervals delimited by the end-points of the segments. The segments in (S_d, HD) and (S_d, HD) are the cartesian product of the unidimensional segments across dimensions. E.g., for (S_d, HD), the segment between $(0,2,1)$ and $(2,2,0)$ are the vectors obtained by $\{0,2\} \times \{2\} \times \{0,1\} = \{(0,2,0), (0,2,1), (2,2,0), (2,2,1)\}$. For (S_d, MD), the segment between the same vectors is obtained by $\{0,1,2\} \times \{2\} \times \{0,1\} = \{(0,2,0), (0,2,1), (1,2,0), (1,2,1), (2,2,0), (2,2,1)\}$.

Geometric crossover: A geometric crossover on product spaces can be seen as the vector-wise aggregation of geometric crossovers acting on each dimension, each one defined on the distance associated with each dimension [12]. So, for (S_d, HD), any mask-based crossover on integer vectors is a geometric crossover, as a segment in each dimension is an edge (i.e., it comprises only the end-points of the segment). For(S_d, MD), geometric crossovers are component-wise blend-crossovers in which each position in the offspring can take intermediate integer values between the values of parents at that position, as a segment in each dimension is the integer interval bounded by the values of the end-points.

Convex set: A convex set on a product space is the cartesian product of unidimensional convex sets across dimensions. For (S_d, HD), for a dimension, the set of all (unidimensional) convex sets are all sub-cliques of the neighbourhood graph of that dimension. Convex sets on the entire space can be represented as generalised schemata in which there are special symbols indicating all possible superpositions of values in $\{0, 1, \ldots, d-1\}^n$. E.g., on an alphabet with values in $\{0, 1, 2\}$ generalised schemata use the values $0, 1, 2$ and the special symbols $*_{01}, *_{02}, *_{12}, *_{012}$, each one denoting the superposition of the values appearing in its index. For example, the generalised schema $0 *_{01} *_{02} 2$ matches the vectors $0002, 0102, 0022, 0122$. For (S_d, MD), like for the Hamming space on binary strings, convex sets coincide with segments, i.e., they are integer hyper-boxes. Notice instead that in (S_d, HD) not all convex sets are segments.

Quasi-concave landscapes: We can derive the specific class of quasi-concave landscapes for any space by piling-up a nested chain of convex sets. For (S_d, HD), this translates to creating functions using a recursive construction analogous to the one in section 3.2 for the Hamming space on binary strings but using generalised schemata instead. Notice however, that unlike the case of binary strings, not every successive level fixes a '*' symbol to specific value, but it may simply reduce the degree of freedom of a '*' symbol, e.g., "fixing" $*_{012}$ to $*_{01}$. For (S_d, MD), piling-up a nested chain of convex sets translates into piling-up hyper-boxes from the largest to the smallest. E.g., on two-dimensional vectors, these landscapes are traditional quasi-concave functions with rectangular or squared levels, whose domain is restricted to integer values.

Convex hull: We can derive the convex hull from its definition and the notion of convex set. For (S_d, HD), the convex hull of a set of integer vectors is the most specific generalised schema matching all vectors. For (S_d, MD), the convex hull of a set of integer vectors (i.e., points) is the smallest hyper-box covering all points.

Uniform convex hull recombination: We can derive this operator from the notion of convex hull. For (S_d, HD), it is the operator that at each position returns, with the same probability, any value that occurs in that position in the parents at least once. This corresponds to sampling uniformly the most specific generalised schemata matching all parent vectors. For (S_d, MD), it is the operator that at each position (i.e., dimension) returns with the same probability any value in the interval between the minimum and maximum values of the parents in that position. This corresponds to sampling the hyper-box uniformly, as this can be achieved by sampling uniformly at random each coordinate (within the specified range that allows to cover all parents).

We now specify the runtime to the space (S_d, HD).

LEMMA 9. *For $M = (S_d, \mathrm{HD})$ and $m \in \mathbb{N}$ we have*

$$\mathrm{P}_M^{\mathrm{Cov}}(m) \geq 1 - dn \left(1 - \frac{1}{d}\right)^m \geq 1 - dn \cdot \exp\left(-\frac{m}{d}\right).$$

PROOF. The worst case is attained for covering the whole space S_d. We cover S_d if in every component of the integer vector every value is present in at least one sample. For each dimension we need all values because to cover a clique graph (i.e., a unidimensional subspace of S_d) we need all nodes (i.e., values) in the clique.

For a fixed component this corresponds to the well-known coupon collector's problem: we are drawing m coupons (integer values) uniformly at random and are interested in the probability that we have at least one coupon of each kind.

This gives rise to the following tail bound. The probability of one fixed value not appearing in m draws is $(1 - 1/d)^m$. The probability that this does not happen for any value is at most $d \cdot (1 - 1/d)^m$ by the union bound. Taking the union bound over n processes on all bits, we get a probability of at most $dn \cdot (1 - 1/d)^m$ that S_d is not covered. □

Then the bound from Theorem 4 simplifies as follows, the second bound being derived as in Theorem 6.

THEOREM 10. *Let $M = (\{0, 1, \ldots, d-1\}^n, \mathrm{HD})$ and the assumptions for Theorem 4 hold. Then the Convex Search Algorithm finds a global optimum within at most q generations and kq fitness evaluations with probability at least*

$$\left(1 - dn \cdot \exp\left(-\frac{kr}{4d}\right)\right)^{q+1} - q \cdot \exp\left(-\frac{9kr}{32}\right) \geq 1 - (q+2)dn \cdot \exp\left(-\frac{kr}{4d}\right)$$

where the second bound requires $\exp\left(\frac{kr}{4d}\right) \geq dn(q+1)$.

COROLLARY 11. *If $k \geq 4d \ln(2(q+2)dn)/r$ then Convex Evolutionary Search restarting after q generations finds a global optimum within $2q$ expected generations and $2kq$ expected fitness evaluations.*

On polynomial quasi-concave landscapes, if $d \leq \mathrm{poly}(n)$, we get polynomial expected numbers of generations and fitness evaluations for appropriate population sizes.

For the space (S_d, MD) we get similar results.

LEMMA 12. *For $M = (S_d, \mathrm{MD})$ and $m \in \mathbb{N}$ we have*

$$\mathrm{P}_M^{\mathrm{Cov}}(m) \geq 1 - 2n \left(1 - \frac{1}{d}\right)^m \geq 1 - 2n \cdot \exp\left(-\frac{m}{d}\right).$$

PROOF. The worst case is attained for covering S_d. We cover S_d if in every component of the integer vector both values 0 and $d-1$ are present in at least one sample. The probability of one fixed value not appearing in m samples is $(1 - 1/d)^m$. The probability that this does not happen for both values and n components is at most $2n \cdot (1 - 1/d)^m$. □

THEOREM 13. *Let $M = (\{0, 1, \ldots, d-1\}^n, \mathrm{MD})$ and the assumptions for Theorem 4 hold. Then the Convex Search Algorithm finds a global optimum within at most q generations and kq fitness evaluations with probability at least*

$$\left(1 - 2n \cdot \exp\left(-\frac{kr}{4d}\right)\right)^{q+1} - q \cdot \exp\left(-\frac{9kr}{32}\right) \geq 1 - 2(q+2)n \cdot \exp\left(-\frac{kr}{4d}\right)$$

where the second bound requires $\exp\left(\frac{kr}{4d}\right) \geq 2n(q+1)$.

COROLLARY 14. *If $k \geq 4d\ln(4(q+2)n)/r$ then Convex Evolutionary Search restarting after q generations finds a global optimum within $2q$ expected generations and $2kq$ expected fitness evaluations.*

6. SUMMARY AND FUTURE WORK

Two important open challenges in the Evolutionary Computation field are (i) finding out on what class of landscapes a certain search algorithm performs well and why, and (ii) devising a more systematical approach to runtime analysis to obtain results that hold for *classes* of search algorithms on *classes* of problems. In this work, we have proposed a novel framework that is a first step towards addressing jointly both challenges. The framework put together a geometric theory of representations and runtime analysis to determine systematically the performance *across, in principle, all spaces and representations* of a population-based evolutionary algorithm with a form of population-wide convex recombination on a class of concave landscapes. Runtime analysis at this level of generality is possible because at its core relies on abstract properties of convex search and of concave landscapes common to all underlying spaces and representations.

The general runtime expression is parametrised on a single space-dependent feature, the worst-case probability of covering convex sets. It can be determined on specific spaces and plugged in the general expression to determine the runtime of the space-specific pair of search algorithm and fitness landscape. We have illustrated this for three spaces and obtained polynomial runtime bounds for polynomial quasi-concave landscapes. We also showed that convex search can be regarded as pure adaptive search on concave landscapes. This is the ultimate cause behind the polynomial runtime of convex search on concave landscapes across representations.

Remarkably, for LEADINGONES convex search turned out to be faster than all unbiased unary black-box algorithms, including EAs using only mutation [8, 17]. This further demonstrates the potential of convex search and population-based EAs.

There is plenty of future work, in at least three directions of investigation. The first direction is to extend the theory to encompass evolutionary algorithms used in practice with standard forms of crossover, selection and mutation. This seems to be possible as EAs with standard operators approximate the convex search behaviour considered in this study. The second direction is to extend the theory to encompass a larger family of fitness landscapes, in particular approximately concave landscapes, and study how the degree of approximation to concavity affects the runtime. From preliminary study, it seems that performance could degrade gracefully as the approximation becomes less good. Furthermore, we would like to determine to what extent interesting non-toy problems fit this framework. There seems to be hope as fitness landscapes associated with well-known combinatorial problems have been shown empirically to have some form of "global concavity". We would like to develop an analytical methodology to recognise that a certain problem provably fits a certain class of concave landscapes. Clearly, NP-Hard problems can be expected to fit a polynomially solvable class of fitness landscapes only with respect to some relaxed target (e.g., approximation complexity, parametrized complexity). Finally, as third line of investigation, we would like to specialise the results to other representations (e.g., continuous and permutation spaces), and devise a general approach for easily determining the required space-specific convexity feature, using known results from convexity on graphs.

Acknowledgments: The authors thank Jon Rowe and a reviewer for helpful comments. This research was supported by EPSRC grants EP/I010297/1 and EP/D052785/1.

7. REFERENCES

[1] A. Auger and B. Doerr, editors. *Theory of Randomized Search Heuristics – Foundations and Recent Developments*. World Scientific, 2011.

[2] B. Doerr, E. Happ, and C. Klein. Crossover can provably be useful in evolutionary computation. *Theoretical Computer Science*, 425(0):17–33, 2012.

[3] B. Doerr, D. Johannsen, T. Kötzing, P. K. Lehre, M. Wagner, and C. Winzen. Faster black-box algorithms through higher arity operators. In *Proc. of FOGA '11*, pages 163–172. ACM Press, 2011.

[4] T. Jansen and I. Wegener. On the analysis of evolutionary algorithms—a proof that crossover really can help. *Algorithmica*, 34(1):47–66, 2002.

[5] T. Jansen and I. Wegener. Real royal road functions—where crossover provably is essential. *Discrete Applied Mathematics*, 149:111–125, 2005.

[6] T. Kötzing, D. Sudholt, and M. Theile. How crossover helps in pseudo-Boolean optimization. In *Proc. of GECCO '11*, pages 989–996. ACM Press, 2011.

[7] J. Lässig and D. Sudholt. General scheme for analyzing running times of parallel evolutionary algorithms. In *Parallel Problem Solving from Nature (PPSN 2010)*, pages 234–243. Springer, 2010.

[8] P. K. Lehre and C. Witt. Black box search by unbiased variation. In *Proc. of GECCO '10*, pages 1441–1448, 2010.

[9] A. Moraglio. *Towards a Geometric Unification of Evolutionary Algorithms*. PhD thesis, University of Essex, 2007.

[10] A. Moraglio. Abstract convex evolutionary search. In *Proc. of FOGA '11*, pages 151–162. ACM Press, 2011.

[11] A. Moraglio and R. Poli. Topological interpretation of crossover. In *Proc. of GECCO '04*, pages 1377–1388. Springer, 2004.

[12] A. Moraglio and R. Poli. Product geometric crossover. In *Parallel Problem Solving from Nature (PPSN 2006)*, pages 1018–1027. Springer, 2006.

[13] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[14] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer, 2010.

[15] N. R. Patel, R. L. Smith, and Z. B. Zabinsky. Pure adaptive search in Monte Carlo optimization. *Mathematical Programming*, 4:317–328, 1988.

[16] D. Sudholt. Crossover is provably essential for the Ising model on trees. In *Proc. of GECCO '05*, pages 1161–1167. ACM Press, 2005.

[17] D. Sudholt. General lower bounds for the running time of evolutionary algorithms. In *Parallel Problem Solving from Nature (PPSN 2010)*, pages 124–133. Springer, 2010.

[18] D. Sudholt and C. Zarges. Analysis of an iterated local search algorithm for vertex coloring. In *Proc. of ISAAC '10*, pages 340–352. Springer, 2010.

[19] I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In R. Sarker, X. Yao, and M. Mohammadian, editors, *Evolutionary Optimization*, pages 349–369. Kluwer, 2002.

[20] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transaction on Evolutionary Computation*, 1(1):67–82, 1996.

[21] Z. B. Zabinsky. *Stochastic Adaptive Search for Global Optimization*. Kluwer Academic, 2003.