NP-Completeness of Deciding Binary Genetic Encodability

Andreas Blass* and Boris Mitavskiy

[1] Department of Mathematics, University of Michigan, Ann Arbor, MI, 48109, United States
ablass@umich.edu
[2] School of Computer Science, University of Birmingham, Birmingham B15 2TT, Great Britain
B.S.Mitavskiy@cs.bham.ac.uk

Abstract. In previous work of the second author a rigorous mathematical foundation for re-encoding one evolutionary search algorithm by another has been developed. A natural issue to consider then is the complexity of deciding whether or not a given evolutionary algorithm can be re-encoded by one of the standard classical evolutionary algorithms such as a binary genetic algorithm. In the current paper we prove that, in general, this decision problem is NP-complete.

1 Introduction

In recent years evolutionary algorithms have been widely exploited to solve various complex optimization problems. In order to apply an evolutionary algorithm to attack a specific optimization problem, one needs to model the algorithm in a suitable manner. The importance of finding appropriate models is emphasized in much of the research literature: see, for instance, the introduction to chapter 17 of [13], [14], [12] and [11]. The general methodology for how to construct the search space and the appropriate recombination operators with the aim of applying the classical genetic algorithm first appeared in [9]. However, there is a variety of different types of EAs which people use. This might be, for example, nonlinear GP with homologous crossover introduced by Poli (see [8] for a detailed description of how this algorithm works), or, even more so, it might be a special type of an algorithm used to attack a specific problem. It is in general interesting to know if it is possible to re-encode a given algorithm by a binary genetic algorithm. In [6] and in [7] a rigorous mathematical framework was introduced, allowing one to re-encode one evolutionary algorithm by another.

In particular, necessary and sufficient conditions for a given evolutionary search algorithm to be embeddable into a binary genetic (or semi-genetic) algorithm have been established. The aim of the current paper is to investigate the computational complexity of deciding if a given evolutionary search algorithm can be re-encoded by another (probably a more commonly used) evolutionary algorithm. The main results of the current paper demonstrate that deciding if a given evolutionary algorithm can be embedded into a binary semi-genetic algorithm can be done in polynomial time, while the more useful, analogous decision problem pertaining to the classical genetic algorithm is, unfortunately, NP-complete.

* Partially supported by NSF grant DMS–0070723.

A.H. Wright et al. (Eds.): FOGA 2005, LNCS 3469, pp. 58–74, 2005.

The paper is organized as follows: In the next two sections we introduce the basic notation and framework used throughout the paper. Next, in Section 4 we demonstrate how the classical types of algorithms fit into this framework. Section 5 is devoted to a summary of the previous work which sets the foundation for the results of the current paper. The main results are then presented in the final Section 6.

2 Notation

Ω is a finite set, called a *search space*.

$f : \Omega \to (0, \infty)$ is a function, called a *fitness function*. The goal is to find a maximum of the function f.

$\mathcal{F}$ is a collection of binary operations on Ω. Intuitively $\mathcal{F}$ can be thought of as the collection of *reproduction transformations*: two parents produce one offspring. The family of asexual reproductions or *mutations* (these are unary operations on Ω, i. e. functions from Ω into itself) will be denoted by $\mathcal{M}$. By a *search system* we mean a search space Ω together with families $\mathcal{F}$ and $\mathcal{M}$ of reproduction transformations and mutations. We shall denote a search system either by $(\Omega, \mathcal{F}, \mathcal{M})$ or simply by Ω; the latter notation follows the convention, common in many parts of mathematics, of using the same symbol, in this case Ω, for a mathematical structure, in this case a search system, and for its underlying set.

Remark: In general there is no reason to assume that a child has exactly two parents. All of the results in this paper are valid for families of q-ary operations on Ω. The only reason $\mathcal{F}$ is assumed to be a family of binary transformations is to alleviate the complexity of notation. In the general case, the definition of search system should, of course, include all the reproduction transformations, regardless of the number of arguments. Search systems were called "heuristic tuples" in earlier work of the second author. They are almost the same as what are called "algebras" in universal algebra, but the morphisms of search systems, defined in Section 5, are different from homomorphisms of algebras.

3 How Does an Evolutionary Algorithm Work?

A typical evolutionary algorithm works as follows: A population $P = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2m} \end{pmatrix}$ with $x_i \in \Omega$ is selected randomly. The algorithm cycles through the following stages:

Evaluation

Individuals of P are evaluated:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2m} \end{pmatrix} \begin{matrix} \to \\ \to \\ \vdots \\ \to \end{matrix} \begin{matrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{2m}) \end{matrix}$$

Selection
A new population

$$P' = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{2m} \end{pmatrix}$$

is obtained by choosing each y_i independently by the following random process. Choose at random j in the range $1 \leq j \leq 2m$, the probability of any j being $\frac{f(x_j)}{\Sigma_{l=1}^{2m} f(x_l)}$. Then set $y_i = x_j$.

Thus, all of the individuals of P' are among those of P, and the expectation of the number of occurrences of any individual of P in P' is proportional to the number of occurrences of that individual in P times the individual's fitness value. In particular, the fitter the individual is, the more copies of that individual are likely to be present in P'. On the other hand, the individuals having relatively small fitness value are not likely to enter into P' at all. This is designed to imitate the natural "survival of the fittest" principle.

Partition
The individuals of P' are partitioned into m pairwise disjoint couples for mating according to some probabilistic rule: For instance the couples could be

$$Q_1 = \begin{pmatrix} y_{i_1^1} \\ y_{i_2^1} \end{pmatrix} \; Q_2 = \begin{pmatrix} y_{i_1^2} \\ y_{i_2^2} \end{pmatrix} \; \dots \; Q_j = \begin{pmatrix} y_{i_1^j} \\ y_{i_2^j} \end{pmatrix} \; \dots \; Q_m = \begin{pmatrix} y_{i_1^m} \\ y_{i_2^m} \end{pmatrix}$$

Reproduction

Replace every one of the selected couples $Q_j = \begin{pmatrix} y_{i_1^j} \\ y_{i_2^j} \end{pmatrix}$ with the couple

$$Q' = \begin{pmatrix} T_1(y_{i_1^j}, y_{i_2^j}) \\ T_2(y_{i_1^j}, y_{i_2^j}) \end{pmatrix}$$

for some couple of transformations $(T_1, T_2) \in \mathcal{F}^2$. The couple (T_1, T_2) is selected according to a fixed probability distribution on $\mathcal{F}^2$. This gives us a new population

$$P'' = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{2m} \end{pmatrix}$$

Mutation
Finally, with small probability we replace z_i with $F(z_i)$ for some randomly chosen $F \in \mathcal{M}$. The choices for different i's are independent. This, once again, gives us a new population $P''' = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{2m} \end{pmatrix}$

Upon completion of mutation start all over with P''' as the initial population. The cycle is repeated a certain number of times depending on the problem. A more general and extensive description is given in [13]. The importance of choosing a reasonable representation for a specific problem is emphasized in some of the modern research. See, for instance, [10]. A few special types of evolutionary algorithms are introduced in the next section.

4 Special Evolutionary Algorithms

Classical Genetic Algorithm with Masked Crossover: Let $\Omega = \prod_{i=1}^{n} A_i$. For every subset $M \subseteq \{1, 2, \ldots, n\}$, define a binary operation L_M on Ω as follows:

$$L_M(\mathbf{a}, \mathbf{b}) = (x_1, x_2, \ldots, x_i, \ldots, x_n)$$

where $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n) \in \Omega$ and $x_i = \begin{cases} a_i & \text{if } i \in M \\ b_i & \text{otherwise.} \end{cases}$

This L_M is a masked crossover operator with mask M. Let $\mathcal{F} = \{L_M \mid M \subseteq \{1, 2, \ldots, n\}\}$. That is, $\mathcal{F}$ in this example is simply the family of masked crossover transformations. The probability distribution on the set $\mathcal{F}^2$ is concentrated on the pairs of the form $(L_M, L_{\bar{M}})$ where $\bar{M}$ denotes the complement of the set M in $\{1, 2, \ldots, n\}$.

Example: Let $n = 5$ and $A_i = \{0, 1, \ldots, i+1\}$. Suppose a given population P consists of 6 individuals which are the rows of the matrix

$$\begin{pmatrix} 2\,3\,4\,5\,6 \\ 0\,1\,2\,3\,4 \\ 1\,2\,3\,4\,5 \\ 0\,0\,1\,2\,3 \\ 1\,1\,0\,1\,2 \\ 1\,2\,1\,5\,4 \end{pmatrix}$$

Say, after the selection stage is complete one obtains the following population

$$P' = \begin{pmatrix} 2\,3\,4\,5\,6 \\ 2\,3\,4\,5\,6 \\ 1\,2\,3\,4\,5 \\ 0\,0\,1\,2\,3 \\ 0\,1\,2\,3\,4 \\ 1\,2\,3\,4\,5 \end{pmatrix}$$

Now the following individuals are paired for mating (masked crossover in this case):

$$Q_1 = \begin{pmatrix} 2\,3\,4\,5\,6 \\ 0\,0\,1\,2\,3 \end{pmatrix},\ Q_2 = \begin{pmatrix} 2\,3\,4\,5\,6 \\ 1\,2\,3\,4\,5 \end{pmatrix},\ \text{and } Q_3 = \begin{pmatrix} 0\,1\,2\,3\,4 \\ 1\,2\,3\,4\,5 \end{pmatrix}$$

Suppose we have chosen the masks $M_1 = \{1, 4, 5\}$, $M_2 = \{1, 2\}$ and $M_3 = \{3, 4\}$ for the crossover of pairs Q_1, Q_2 and Q_3 respectively. In the language of this paper it

means we have chosen the pairs of transformations $(L_{M_1}, L_{\bar{M}_1})$ for Q_1, $(L_{M_2}, L_{\bar{M}_2})$ for Q_2 and $(L_{M_3}, L_{\bar{M}_3})$ for Q_3 respectively. Upon applying these we obtain

$$Q_1 \to \begin{pmatrix} L_{M_1}((2,\,3,\,4,\,5,\,6),(0,\,0,\,1,\,2,\,3)) \\ L_{\bar{M}_1}((2,\,3,\,4,\,5,\,6),(0,\,0,\,1,\,2,\,3)) \end{pmatrix} = \begin{pmatrix} 2\,0\,1\,5\,6 \\ 0\,3\,4\,2\,3 \end{pmatrix},$$

likewise

$$Q_2 \to \begin{pmatrix} L_{M_2}((2,\,3,\,4,\,5,\,6),(1,\,2,\,3,\,4,\,5)) \\ L_{\bar{M}_2}((2,\,3,\,4,\,5,\,6),(1,\,2,\,3,\,4,\,5)) \end{pmatrix} = \begin{pmatrix} 2\,3\,3\,4\,5 \\ 1\,2\,4\,5\,6 \end{pmatrix},$$

and, finally,

$$Q_3 \to \begin{pmatrix} L_{M_3}((0,\,1,\,2,\,3,\,4),(1,\,2,\,3,\,4,\,5)) \\ L_{\bar{M}_2}((0,\,1,\,2,\,3,\,4),(1,\,2,\,3,\,4,\,5)) \end{pmatrix} = \begin{pmatrix} 1\,2\,2\,3\,5 \\ 0\,1\,3\,4\,4 \end{pmatrix}.$$

The family $\mathcal{M}$ of mutation transformations in this example (and in all of the following ones) consists of the transformations $M_{\boldsymbol{a}} : \Omega \to \Omega$, where $\boldsymbol{a} \in \prod_{i\in S} A_i$ and $S \subseteq \{1, 2, \ldots, n\}$. The transformation $M_{\boldsymbol{a}}$ sends any $\mathbf{x} = (x_1,\, x_2,\, \ldots, x_n) \in \Omega$ to the $\mathbf{y} = (y_1,\, y_2\, \ldots, y_n) \in \Omega$ whose components are $y_q = \begin{cases} a_q & \text{if } q \in S \\ x_q & \text{otherwise.} \end{cases}$ In other words, $M_{\boldsymbol{a}}$ simply replaces the q^{th} coordinate of its argument with $a_q \in A_q$ whenever $q \in S$.

Binary Genetic Algorithm with Masked Crossover
When every $A_i = \{0,\, 1\}$ (which means that $\Omega = \{0,\, 1\}^n$) in the example above, one obtains the classical binary genetic algorithm.

Random Respectful Recombination
Random Respectful Recombination first appeared in [9]. Here the search space Ω and the family of mutation transformations, $\mathcal{M}$, are exactly the same as in the example of classical genetic algorithm, and the family of mating transformations is described below. As in [5], we call these mating transformations Holland transformations because their corresponding family of fixed subsets is precisely the collection of subsets of Ω determined by the classical Holland schemata together with the empty set. (See examples following Corollary 8 in the next section.) For every given point $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in \Omega$ define a Holland transformation $T_{\mathbf{u}} : \Omega^2 \to \Omega$ as follows: for every $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n) \in \Omega$

$$T_{\mathbf{u}}(\mathbf{a}, \mathbf{b}) = (x_1, x_2, \ldots, x_n)$$

where

$$x_i = \begin{cases} a_i \text{ if } a_i = b_i \\ u_i \text{ otherwise} \end{cases}$$

In other words, if the i^{th} coordinates of $\mathbf{a}$ and $\mathbf{b}$ coincide, then the i^{th} coordinate of $T_{\mathbf{u}}(\mathbf{a}, \mathbf{b})$ also coincides with them. If, on the other hand, the i^{th} coordinates of $\mathbf{a}$ and $\mathbf{b}$ differ, then the i^{th} coordinate of $T_{\mathbf{u}}(\mathbf{a}, \mathbf{b})$ is that of $\mathbf{u}$, namely, u_i. Let $\mathcal{F} = \{T_{\mathbf{u}} \mid \mathbf{u} \in$

$\Omega\}$ be the family of all Holland transformations. The probability distribution on $\mathcal{F}$ may be chosen in different ways depending on the circumstances, but this is not relevant to the objective of the current paper.

Every transformation in the pair $(T_{\mathbf{u}}, T_{\mathbf{v}})$ is chosen independently.

Binary Random Respectful Recombination

The search space Ω and the family of mating transformations $\mathcal{F}$ and the family of mutations $\mathcal{M}$ are exactly the same as these for the binary genetic algorithm with masked crossover described above. The only difference is that the probability distribution on $\mathcal{F}^2$ is now completely uniform (rather than being concentrated on the diagonal-like subset described in the classical genetic algorithm example). For instance, if $n = 5$, $M_1 = \{2,\ 3,\ 4\}$, $M_2 = \{1,\ 3,\ 5\}$ and the pair (T_{M_1}, T_{M_2}) is selected for mating, we have, for instance,

$$\begin{pmatrix} 1\ 0\ 0\ 1\ 1 \\ 1\ 1\ 0\ 0\ 1 \end{pmatrix} \longmapsto \begin{pmatrix} T_{M_1}((1,\ 0,\ 0,\ 1,\ 1),\ (1,\ 1,\ 0,\ 0,\ 1)) \\ T_{M_2}((1,\ 0,\ 0,\ 1,\ 1),\ (1,\ 1,\ 0,\ 0,\ 1)) \end{pmatrix} = \begin{pmatrix} 1\ 0\ 0\ 1\ 1 \\ 1\ 1\ 0\ 0\ 1 \end{pmatrix}$$

The same definition could be applied in the non-binary case, but then it would not agree with the random respectful recombination of [9] (as described above). The difference is that, when the two parents have different alleles of a certain gene, then random respectful recombination allows the offspring to have any allele of that gene, while the present definition only allows the offspring to have either of the two alleles present in the parents. The two notions are equivalent just when there are only two possible alleles.

The following type of algorithm may seem useless at first. Its importance will become clear in the next section when we present the binary embedding theorem which shows that the binary semi-genetic algorithm possesses an interesting universal property.

Binary Semi-genetic Algorithm

The search space $\Omega = \{0,\ 1\}^n$, just as in the case of the binary genetic algorithm. The family of mating transformations $\mathcal{F}$ is defined as follows: Fix an individual $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in \Omega$. Define a semi-crossover transformation $F_{\mathbf{u}} : \Omega^2 \to \Omega$ as follows: For any given pair $(\mathbf{x},\ \mathbf{y}) \in \Omega^2$ with $\mathbf{x} = (x_1,\ x_2, \ldots, x_n)$ and $\mathbf{y} = (y_1,\ y_2, \ldots,\ y_n)$ we have $F_{\mathbf{u}}(\mathbf{x},\ \mathbf{y}) = \mathbf{z} = (z_1, z_2, \ldots z_n) \in \Omega$ where

$$z_i = \begin{cases} 1 & \text{if } x_i = y_i = 1 \\ u_i & \text{otherwise} \end{cases}$$

In other words, $F_{\mathbf{u}}$ preserves the i^{th} gene if it is equal to 1 in both of the parents and replaces it with u_i otherwise. Let $\mathcal{F} = \{F_{\mathbf{u}} \mid \mathbf{u} \in \Omega\}$ be the family of all semi-crossover transformations. The family of mutation transformations $\mathcal{M}$ is exactly the same as in the examples above.

Example: With $n = 5$ and $\mathbf{u}_1 = (0,\ 1,\ 1,\ 0,\ 1)$, $\mathbf{u}_2 = (0,\ 1,\ 0,\ 0,\ 1)$ we have

$$\begin{pmatrix} 1\ 0\ 0\ 1\ 1 \\ 1\ 1\ 0\ 0\ 1 \end{pmatrix} \longmapsto \begin{pmatrix} F_{\mathbf{u_1}}((1,\ 0,\ 0,\ 1,\ 1),\ (1,\ 1,\ 0,\ 0,\ 1)) \\ F_{\mathbf{u_2}}((1,\ 0,\ 0,\ 1,\ 1),\ (1,\ 1,\ 0,\ 0,\ 1)) \end{pmatrix} = \begin{pmatrix} 1\ 1\ 1\ 0\ 1 \\ 1\ 1\ 0\ 0\ 1 \end{pmatrix}$$

Notice that, if 1 is present in the i^{th} position of both parents, then it remains in the i^{th} position of both offspring. There are absolutely no other restrictions, though.

In practice the choice of the search space Ω is primarily determined by the specific problem and related circumstances. The general methodology for the construction of the search spaces first appeared in the work of Radcliffe (see, for instance, [9]). Radcliffe introduced the notion of a forma which captures the essential properties of the Holland schemata in a representation independent setting. A forma is simply a partition of the search space into equivalence classes. A given collection of forma with suitable properties (see [9]) is, in a sense, no different from the collection of the classical Holland schemata provided that one encodes the search space using the "genetic representation function" which is also introduced in [9]. The connection between all of the possible families of mating transformations on a given search space Ω and the corresponding families of invariant subsets established in [5] has been exploited in [6] and in [7] to extend Radcliffe's notion of the genetic representation function to compare various evolutionary algorithms via possible encodings of their search spaces (see corollary 12 of [6]). In particular, necessary and sufficient conditions, stated in terms of the internal structure of the search space, for re-encoding a given algorithm by a search system corresponding to a binary genetic algorithm have been established (see Theorem 14 of [6] or, more generally, Theorem 3.7 of [7]). These ideas will be summarized in the next section.

5 Summary of Previous Work

As we have seen in the previous sections, a given evolutionary search algorithm is determined primarily by the ordered 4-tuple $(\Omega, \mathcal{F}, \mathcal{M}, f)$. In the current paper we shall be primarily concerned with the search space Ω and the family of mating transformations $\mathcal{F}$. The family of mutations $\mathcal{M}$ is of less importance because it is ergodic, meaning that the only invariant subsets under $\mathcal{M}$ are $\emptyset$ and the entire search space Ω. The notion of an invariant subset is defined below. The reason why invariant subsets play a significant role in the current paper is Theorem 8. We give the definition for a family Γ of operations of any number of arguments, but we shall use it in this paper only for the family $\mathcal{F}$ of binary operations of a search system.

Definition 1 For a given family of m-ary operations Γ on a set Ω (that is, functions from Ω^m into Ω) a subset $S \subseteq \Omega$ is *invariant* under Γ if and only if for all $T \in \Gamma$ we have $T(S^m) \subseteq S$. We shall denote by Λ_Γ the family of all invariant subsets of Ω under Γ. In other words, $\Lambda_\Gamma = \{S \mid S \subseteq \Omega,\ T(S^m) \subseteq S\ \forall\ T \in \Gamma\}$.

Below we list the families of invariant subsets for each of the examples of Section 4:

Classical Genetic Algorithm. In this case, the family of invariant subsets $\Lambda_{\mathcal{F}}$ is $\{\prod_{i=1}^{n} T_i \mid T_i \subseteq A_i\}$. This is precisely the family of subsets determined by Antonisse's schemata (see corollary 2.4 of [5]).

Random Respectful Recombination. $\Lambda_{\mathcal{F}} = \{\prod_{i=1}^{n} T_i \mid T_i = \{a\}$ for some $a \in A_i$ or $T_i = A_i\} \cup \{\emptyset\}$. This is precisely the family of subsets determined by the Holland schemata together with the empty set (see corollary 3.5 of [5]).

Binary Semi-genetic Algorithm. It is not hard to verify that $\Lambda_{\mathcal{F}} = \{\prod_{i=1}^{n} T_i \mid T_i = \{1\} \text{ or } T_i = \{0, 1\}\} \cup \{\emptyset\}$. This is precisely the family of subsets determined by Holland schemata whose fixed positions can only equal 1 (can't equal 0).

The mathematical properties of the family of invariant subsets, Λ_Γ have been described in detail in [5]. In the current presentation we just mention a few facts and notions which will be of particular importance here.

It is easy to verify (see Proposition A1 of [5]) that the family Λ_Γ is closed under arbitrary intersections and contains Ω. It then follows that for every element $x \in \Omega$ there is a unique smallest element of Λ_Γ containing x (namely the intersection of all the members of Λ_Γ containing x).

Definition 2 Given a search system $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$, denote by S_x^Ω the smallest element of $\Lambda_{\mathcal{F}}$ containing x. When the search system Ω is clear from the context we shall just write S_x instead of S_x^Ω.

The following definition is a natural extension of the notion of a genetic representation function introduced in [9].

Definition 3 Given two search systems $\Omega_1 = (\Omega_1, \mathcal{F}_1, \mathcal{M}_1)$ and $\Omega_2 = (\Omega_2, \mathcal{F}_2, \mathcal{M}_2)$, a *morphism* $\delta : \Omega_1 \to \Omega_2$ is just a function $\delta : \Omega_1 \to \Omega_2$ which respects the reproduction transformations in the following sense: for each $T \in \mathcal{F}_1$ and each $\mathbf{x} = (x, y) \in \Omega_1^2$ there exists $F_{(x,y)} \in \mathcal{F}_2$ such that $\delta(T(x, y)) = F_{(x,y)}(\delta(x), \delta(y))$ (see Figure 1). Analogously, we must have, for each $M \in \mathcal{M}_1$ and each $x \in \Omega$ some $H_x \in \mathcal{M}_2$ such that $\delta(M(x)) = H_x(\delta(x))$.

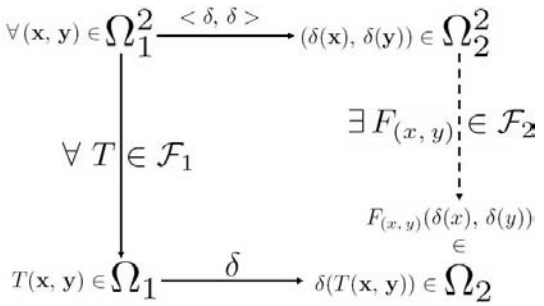

Fig. 1. The morphism $\delta : \Omega_1 \to \Omega_2$

A morphism $\delta : \Omega_1 \to \Omega_2$ provides the means for encoding the search system Ω_1 by the search system Ω_2. Unless the underlying function δ is one to one, there is some non-trivial coarse graining involved. We therefore give a special name to those morphisms whose underlying functions are injective.

Definition 4 We say that a morphism $\delta : \Omega_1 \hookrightarrow \Omega_2$ is an *embedding* if the underlying function $\delta : \Omega_1 \to \Omega_2$ is one-to-one.

search systems and the morphisms between them form a well-defined category (see [3] for details about the notion of category). Some universal constructions on this category have been studied in [7]. The central result of [6] is a connection between the family of invariant subsets and the family of all possible re-encoding morphisms between two given search systems. This connection is analogous to the corresponding connection between the family of open subsets and continuous maps in topology and between sigma-algebras and measurable functions in analysis:

Proposition 5 *Let $\delta : \Omega_1 \to \Omega_2$ be a morphism of search systems. Then $S \in \Lambda_{\mathcal{F}_2} \Longrightarrow \delta^{-1}(S) \in \Lambda_{\mathcal{F}_1}$. In words, a preimage of an invariant set under a morphism is invariant.*

The converse of Proposition 5 holds under the following technical requirement:

Definition 6 We say that a given family of m-ary operations Γ on a set Ω (that is a family of functions from Ω^m to Ω) is *composition closed* if the following two conditions hold:

1. For all T_0, T_1, $T_2, \ldots, T_m \in \Gamma$, the composite operation $T : \Omega^m \to \Omega$ defined by $T(\mathbf{x}) = T_0(T_1(\mathbf{x}), T_2(\mathbf{x}), \ldots, T_m(\mathbf{x}))$ is also a member of Γ.
2. For all $S \subseteq \Omega$, we have $\bigcup_{T \in \Gamma} T(S^m) \supseteq S$.

It is fairly straightforward to verify that every one of the families of mating transformations involved in the examples of Section 4 is composition closed. In fact, it was already shown in [5] that the families of masked crossover transformations and Holland transformations (those which are used for modelling random respectful recombination) are composition closed (see Proposition 2.1 and Theorem 3.6 of [5]). It was also shown (see proposition 9 of [6]) that the family of binary semi-crossover transformations is composition closed.

As noted before, for any family of m-ary operations on Ω the corresponding family of invariant subsets Λ_Γ is closed under arbitrary intersections. Moreover, for any function $\delta : \Omega_1 \to \Omega_2$, the inverse image of the intersection of two subsets of Ω_2 is the intersection of the inverse images of these subsets: $\delta^{-1}(U \cap V) = \delta^{-1}(U) \cap \delta^{-1}(V)$. This motivates the following definition:

Definition 7 Given a family of m-ary operations Γ on Ω, we say that a family of subsets $\widetilde{\Lambda_\Gamma} \subseteq \Lambda_\Gamma$ is a *base* of Λ_Γ if every set $K \in \Lambda_\Gamma$ is the intersection of some sets in $\widetilde{\Lambda_\Gamma}$. (Equivalently, $K = \bigcap_{S \in \widetilde{\Lambda_\Gamma},\, S \supseteq K} S$).

We now continue with the examples following Definition 1 and list bases for each of them:

Classical Genetic Algorithm. In this case a base for $\Lambda_{\mathcal{F}}$ is the family $\widetilde{\Lambda_{\mathcal{F}}} = \{\prod_{i=1}^n T_i \mid T_i = A_i \text{ for all but one } i\}$. The reader can see that $|\widetilde{\Lambda_{\mathcal{F}}}| = (\sum_{i=1}^n 2^{|A_i|}) - n + 1$. Every element of $\widetilde{\Lambda_{\mathcal{F}}}$ can be thought of as a union of subsets determined by Holland schemata having exactly one fixed position at the same gene.

Random Respectful Recombination. In this case a base for $\Lambda_{\mathcal{F}}$ is the family $\widetilde{\Lambda_{\mathcal{F}}}$ consisting of all products $\prod_{i=1}^n T_i$ where T_i is a one-element set for one value of i, and $T_i = A_i$ for all other values of i. This is precisely the family of subsets determined by Holland schemata having exactly one fixed position.

Binary Semi-genetic Algorithm. It is not hard to verify that in this case a base for $\Lambda_{\mathcal{F}}$ is the family $\widetilde{\Lambda_{\mathcal{F}}}$ consisting of all products $\prod_{i=1}^{n} T_i$ where $T_i = \{1\}$ for one value of i, and $T_i = A_i$ for all other values of i. Notice that $\widetilde{\Lambda_{\mathcal{F}}}$ is precisely the family of subsets determined by Holland schemata having exactly one fixed position, and that fixed position is equal to 1.

The following fact is the central result of [6]:

Theorem 8 *Let $\Omega_1 = (\Omega_1, \mathcal{F}_1, \mathcal{M}_1)$ and $\Omega_2 = (\Omega_2, \mathcal{F}_2, \mathcal{M}_2)$ denote search systems with $\mathcal{F}_2$ and $\mathcal{M}_2$ being composition closed, and let $\delta : \Omega_1 \to \Omega_2$ be a function. Then the following are equivalent:*

1. $S \in \widetilde{\Lambda_{\mathcal{F}_2}} \implies \delta^{-1}(S) \in \Lambda_{\mathcal{F}_1}$.
2. $S \in \Lambda_{\mathcal{F}_2} \implies \delta^{-1}(S) \in \Lambda_{\mathcal{F}_1}$.
3. $\delta : \Omega_1 \to \Omega_2$ is a morphism of search systems.

In the current paper we shall be primarily concerned with the issue of whether or not a given search system can be embedded into a search system representing the binary semi-genetic or the binary genetic algorithm in the sense of Definition 4. We therefore introduce the following definition.

Definition 9 We say that a given search system $\Omega = (\Omega, \mathcal{F}, \mathcal{M}_1)$ is *semi-genetic (genetic)* if it can be embedded, in the sense of Definition 4, into the search system $(\{0, 1\}^n, \mathcal{F}_M, \mathcal{M})$ for some n where $\mathcal{F}_M$ is the family of all semi-crossover transformations (the family of all masked crossover transformations) and $\mathcal{M}$ is the family of mutations (the same in all of the examples) as introduced in Section 4.

In [6] necessary and sufficient conditions for a given search system to be semi-genetic have been established[1]:

Theorem 10 *Given a search system $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$, the following are equivalent:*

1. *Ω can be embedded into an n-dimensional semi-genetic search system for some n.*
2. *$\forall\, x, y \in \Omega$ with $x \neq y$ we have either $x \notin S_y^{\Omega}$ (see Definition 2) or vice versa: $y \notin S_x^{\Omega}$.*
3. *$\forall\, x, y \in \Omega$ with $x \neq y$ we have $S_x^{\Omega} \neq S_y^{\Omega}$. (Another way to say this is that the map sending x to S_x^{Ω} is one-to-one.)*

Moreover, if an embedding exists for some n, then there exists one for $n = |\Omega|$. We also must have $n \geq \lceil \log_2 |\Omega| \rceil$.

Once we are equipped with Theorem 8, it is not hard to establish a criterion analogous to Theorem 10 for a given search system to be genetic:

Theorem 11 *A given search system $\Omega_1 = (\Omega_1, \mathcal{F}_1, \mathcal{M}_1)$ is genetic if and only if for each $x \neq y$ in Ω_1 there exists a complementary pair of invariant subsets A and B ($A \cap B = \emptyset$ and $A \cup B = \Omega_1$, and $A, B \in \Lambda_{\mathcal{F}}$) with $x \in A$ and $y \in B$.*

[1] In fact, a lot more has been accomplished in [6] and in [7]. A one-to-one correspondence between all possible morphisms (re-encodings) of the search space by a binary semi-genetic (genetic) algorithm and certain corresponding collections of ordered tuples of invariant subsets of the search space has been established (see Theorems 14 and 20 of [6])

Proof. Suppose first that Ω_1 is genetic. Let $\delta : \Omega_1 \to \Omega$ be an embedding where $\Omega = (\{0, 1\}^n, \mathcal{F}_M, \mathcal{M})$ is the search system describing the binary genetic algorithm with masked crossover as in Section 4. Fix $x \neq y \in \Omega_1$. Then, since δ is an embedding, $\delta(x) = (x_1, x_2, \ldots, x_n) \neq \delta(y) = (y_1, y_2, \ldots, y_n) \in \{0, 1\}^n$. But this means that $x_i \neq y_i$ for some i. Consider the Holland schemata $H_1 = \{0, 1\}^{i-1} \times \{x_i\} \times \{0, 1\}^{n-i}$ and $H_2 = \{0, 1\}^{i-1} \times \{y_i\} \times \{0, 1\}^{n-i}$. Notice that $\delta(x) \in H_1$ and $\delta(y) \in H_2$, $H_1 \cap H_2 = \emptyset$ and $H_1 \cup H_2 = \{0, 1\}^n$. Now simply let $A = \delta^{-1}(H_1) \in \Lambda_{\mathcal{F}}$ and $B = \delta^{-1}(H_2) \in \Lambda_{\mathcal{F}}$. Since δ is a morphism, A and B are as required.

Now suppose that for all $x \neq y \in \Omega_1$ there exists a complementary pair of invariant subsets A and B with $x \in A$ and $y \in B$. Then we can choose a sequence of invariant sets $A_1, A_2, \ldots, A_n$ with invariant complements A_i^c such that each pair $x \neq y \in \Omega_1$ is separated by the chosen sets, i.e., there is i with $x \in A_i$ and $y \in A_i^c$ or vice versa. Now consider the map $\delta : \Omega_1 \to \{0, 1\}^n$ defined as follows for all $x \in \Omega_1$: $\delta(x) = (x_1, x_2, \ldots, x_n)$ where $x_i = \begin{cases} 1 & \text{if } x \in A_i \\ 0 & \text{if } x \in A_i^c \end{cases}$. We observe that δ is a morphism: Indeed, according to examples following Theorem 8, Holland schemata with one fixed position (subsets of the form $H_i^1 = \{0, 1\}^{i-1} \times \{1\} \times \{0, 1\}^{n-i}$ and $H_i^0 = \{0, 1\}^{i-1} \times \{0\} \times \{0, 1\}^{n-i}$) form a base of $\Lambda_{\mathcal{F}_M}$ and we have $\delta^{-1}(H_i^1) = A_i \in \Lambda_{\mathcal{F}_1}$ and $\delta^{-1}(H_i^0) = A_i^c \in \Lambda_{\mathcal{F}_1}$ so that $\delta : \Omega_1 \to \Omega$ is a morphism of search systems thanks to Theorem 8. It remains to show that δ is one-to-one: Fix $x \neq y \in \Omega_1$. Then for at least one i we have $x \in A_i$ and $y \in A_i^c$ or vice versa. But then we have $x_i \neq y_i$ and so, according to the definition of δ, $\delta(x) \neq \delta(y)$, and the desired conclusion follows. We deduce now that δ is an embedding so that Ω_1 is, indeed, genetic. □

6 Complexity of Deciding if a Given Search System Is Genetic

In the previous section we have summarized the results which establish some conditions to tell us when a given algorithm can be re-encoded by a binary semi-genetic or by a binary genetic algorithm. In the current section we shall investigate the complexity of deciding whether or not condition 2 of Theorem 10 and the condition of Theorem 11 are satisfied. We shall see below that deciding whether or not a given algorithm is semi-genetic (in the sense of Definition 9) can be done in polynomial time, while deciding if a given algorithm is genetic (also in the sense of Definition 9) is an NP-complete problem. Here both "polynomial time" and "NP" are with respect to the size of the representation, i.e., $|\Omega| + |\mathcal{F}|$.

Theorem 12 *The following problem can be solved in polynomial time with respect to the size of the input provided that there exists a constant q such that for every $F \in \mathcal{F}$ the computation of $F(x, y)$ is done in $O(n^q)$ steps*[2].

Instance of the problem: A search system Ω=$(\Omega, \mathcal{F}, \mathcal{M})$ and individuals $x, y \in \Omega$.

Question: Is it true that $y \notin S_x$?

[2] A very reasonable assumption since there would be little point in running such an evolutionary search algorithm otherwise

Proof. We construct a polynomial time procedure to answer this question: According to proposition A.4 of [5] we have a nested chain of inclusions $\{x\} \subseteq \mathcal{F}(\{x\}) \subsetneq \mathcal{F}(\mathcal{F}(\{x\})) \subsetneq \ldots \subsetneq \mathcal{F}^k(\{x\}) = \mathcal{F}^{k+1}(\{x\}) = S_x$. Moreover, it is clear that $k \leq |\Omega|$. The following algorithm will therefore answer the question: "Is it true that $y \notin S_x$?"

Step 1: Set $K := \{x\}$ and $l = 1$

Step 2: For each $(u, v) \in K^2$ and each $T \in \mathcal{F}$ compute $T(u, v)$. If $T(u, v) = y$ then stop and return "no". If not, let $K := K \cup \{T(u, v) \,|\, (u, v) \in K \text{ and } T \in \mathcal{F}\}$ and $l := l + 1$. If $l = |\Omega| + 1$ then stop and return "yes". Otherwise repeat step 2.

It remains to show that the algorithm above solves the problem in $O(n^m)$ steps where m is a fixed integer and $n = |\Omega| + |\mathcal{F}|$. But notice that the computational part of step 2 takes no longer than $|\Omega|^2 \cdot |\mathcal{F}| \cdot O(n^q)$ steps for the integer q such that for every $F \in \mathcal{F}$ the computation of $F(x, y)$ is done in $O(n^q)$ steps (see the assumption). Moreover, step 2 is repeated at most $|\Omega| \leq n$ times so that the total amount of time it takes the algorithm to run is $|\Omega| \cdot |\Omega|^2 \cdot |\mathcal{F}| \cdot O(n^q) \leq O(n^{q+4})$ steps and the argument is complete. □

Since the number of pairs of elements in a search space is quadratic (we only need that it is bounded by a polynomial) with respect to the size of the search space itself, Theorem 12 together with Theorem 10 immediately implies:

Corollary 13 *Given a search system $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$ the decision whether or not Ω is semi-genetic can be made in polynomial time with respect to the size of the input provided that computation of $F(x, y)$ for all x, y can be done in polynomial time uniformly (meaning that the time bound is independent of x and y).*

The situation turns out to be less pleasant in the case of deciding whether or not a given search system is genetic, i. e. whether the condition of Theorem 11 is satisfied, as the following theorem shows:

Theorem 14 *The following problem is NP-complete.*

Instance of the problem: A search system $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$ and individuals $x, y \in \Omega$. We assume given a polynomial time algorithm for evaluating the functions in $\mathcal{F}$ on arguments in Ω.

Question: Does there exist a subset $A \subseteq \Omega$ such that $x \in A$ while $y \in A^c$ and both A and $A^c \in \Lambda_{\mathcal{F}}$?

Proof. It is easy to see that the problem is in NP; just guess an appropriate A and check that it works. The challenging part is to build an appropriate polynomial time reduction. We reduce from the "Not All Equal 3-SAT" problem (see section A9.1, page 259 of [2]) In order to state the "Not All Equal 3-SAT" problem, we need the notion of a literal and of a clause:

Definition 15 Given a set U of boolean variables, a literal over U is either a variable, say a, from U or the negation of the variable $a \in U$, denoted by $\bar{a}$. We sometimes use a bar over a literal that is not a variable; then $\bar{\bar{a}}$ means simply a.

Definition 16 A 3-clause over a set U of boolean variables is a disjunction of some three literals over U. If the literals involved in C are a, b and c then we shall write $C = a \vee b \vee c$.

The "Not All Equal 3-SAT" problem is given by:

Instance: A collection $\mathcal{C}$ of 3-clauses over a set U of variables.

Question: Does there exist a truth assignment such that every clause in $\mathcal{C}$ contains a literal whose value is true and a literal whose value is false?

The "Not All Equal 3-SAT" problem is known to be NP-complete (see [2]). We now proceed building the reduction. So fix an instance of the "Not All Equal 3-SAT", i. e. a collection $\mathcal{C}$ of 3-clauses over a set of variables U. Let the search space Ω consist of all the literals involved in the clauses of $\mathcal{C}$. We still have to define the family of reproduction transformations $\mathcal{F}$ and the distinguished pair of points x and y. ($\mathcal{M}$ does not play any role here since it is always assumed to be ergodic.) We define one such transformation for every clause $C \in \mathcal{C}$, say $C = a \vee b \vee c$, where a, b and c denote arbitrary literals. We define the transformation $T_C : \Omega^2 \to \Omega$ as follows:

$$T_C(a,\, b) = T_C(b,\, a) = \bar{c}$$
$$T_C(a,\, c) = T_C(c,\, a) = \bar{b}$$
$$T_C(b,\, c) = T_C(c,\, b) = \bar{a}.$$

For all other pairs $(u,\, v) \in \Omega^2$ we define $T_C(u,\, v) = u$. Now, for each $w \in \Omega$ define a transformation $F_w : \Omega^2 \to \Omega$ as follows: For pairs of the form $(a,\, \bar{a})$ let $F_w(a,\, \bar{a}) = w$. For all other pairs let $F_w(u,\, v) = u$. Finally let $\mathcal{F} = \{T_C \,|\, C \in \mathcal{C}\} \cup \{F_w \,|\, w \in \Omega\}$. To define the distinguished pair of points in Ω, fix a literal $d \in \Omega$ and let $x = d$ and $y = \bar{d}$. We have now constructed an instance of the problem we are interested in. Clearly the construction above is done in polynomial time with respect to the input size. It remains to show that there exists a truth assignment such that every clause contains a literal whose value is true and a literal whose value is false $\Longleftrightarrow$ there exists a subset $A \subseteq \Omega$ such that $x \in A$ while $y \in A^{\mathrm{c}}$ and both A and $A^{\mathrm{c}} \in \Lambda_{\mathcal{F}}$.

Proof (of the $\Longrightarrow$ direction). Fix a truth assignment f as in the assumption. Let $A = \{u \,|\, f(u) = T\}$. Then $A^{\mathrm{c}} = \{u \,|\, f(u) = F\}$. Since $x = d$ and $y = \bar{d}$, it must be the case that either $x \in A$ and $y \in A^{\mathrm{c}}$ or vice versa. It only remains to show that both A and A^{c} are invariant under $\mathcal{F}$. So, fix individuals (literals) u and $v \in A$ (or u and $v \in A^{\mathrm{c}}$). Choose any transformation $G \in \mathcal{F}$. Then $G = F_w$ for some literal $w \in \Omega$ or $G = T_C$ for some clause $C = a \vee b \vee c$. Now observe that we can't have $(u,\, v) = (a,\, \bar{a})$ for any $a \in \Omega$ since both u and v have the same truth value. Therefore, every transformation of the form F_w maps $(u,\, v)$ into u and, hence, leaves both A and A^{c} invariant. Every T_C does the same thing unless both u and v appear in some clause C. Let z denote the remaining literal in the clause C. Since $f(u) = f(v)$, we must have $f(z) \neq f(u)$ (this was the assumption about the truth assignment). But then $f(T_C(u,\, v)) = f(\bar{z}) = f(u) = f(v)$ and so T_C leaves both A and A^{c} invariant. This shows that both A and A^{c} are invariant under $\mathcal{F}$. □

Proof (of the $\Longleftarrow$ direction). Now suppose there exist a complementary pair of subsets of Ω invariant under $\mathcal{F}$, say A and A^{c}, with $x \in A$ and $y \in A^{\mathrm{c}}$. We have to produce a truth assignment f on Ω such that, for each clause $C \in \mathcal{C}$, not all the literals have the same truth value. Now for all literals $u \in A$ let $f(u) = T$ and for $u \in A^{\mathrm{c}}$ let $f(u) = F$. We still have to show that f is a well-defined truth assignment, meaning that literals u and $\bar{u}$ are never assigned the same truth value, or, equivalently, it is not the case that both

u and $\bar{u} \in A$ or both u and $\bar{u} \in A^c$. This is precisely the purpose of the transformations F_w: if, say, u and $\bar{u} \in A$, then for all $z \in \Omega$ we have $F_z(u, \bar{u}) = z \in A$ and therefore $A = \Omega$, contradicting the fact that $A^c \neq \emptyset$ since $y \in A^c$. Analogously, the assumption that u and $\bar{u} \notin A$ leads to a contradiction. We conclude now that f is, indeed, a well-defined truth assignment. It remains to show that, for each clause $C \in \mathcal{C}$, not all the literals have the same truth value. So fix a clause $C \in \mathcal{C}$. Say $C = a \vee b \vee c$. Since there are exactly three literals in C, some two of them must have the same truth value. Without loss of generality assume these are a and b. Whichever of A and A^c contains a and b must, since it is invariant, also contain $T_C(a, b) = \bar{c}$. By what we have already proved, while checking that f is well-defined, this set cannot contain c. Thus, c does not have the same truth value as a and b. The desired conclusion is now established. □

We have shown now that "Not All Equal 3-SAT" problem can be reduced in polynomial time to the problem in the statement of the theorem, and "Not All Equal 3-SAT" is known to be NP-complete. The desired conclusion now follows. □

Theorem 14 gives us the NP-completeness of testing whether, for a given Ω and $\mathcal{F}$, a given pair $x \neq y$ can be separated by complementary, invariant sets. To determine whether $(\Omega, \mathcal{F}, \mathcal{M})$ is genetic, we would have to test whether *every* pair $x \neq y$ in Ω can be separated. This problem is clearly also in NP, since it amounts to checking $|\Omega|^2 - |\Omega|$ instances of the question in Theorem 14. (In fact, by symmetry, it suffices to check only $\frac{|\Omega|(|\Omega| - 1)}{2}$ instances.) But we cannot immediately conclude that it is NP-complete. It seems harder than the problem in Theorem 14, since it involves many instances of that problem, but it is conceivable that there could be a way to determine whether all pairs are separable without checking each (or any) one individually. The following theorem settles this question. For brevity, we say that two elements can be separated (in a given search system) if there is a complementary pair of invariant sets, each containing one of the two elements.

Theorem 17 *The following problem is NP-complete.*

Instance of the problem: A search system $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$ with a polynomial time algorithm for evaluating the functions in $\mathcal{F}$ on arguments in Ω.

Question: Can every pair of distinct elements of Ω be separated?

Proof. We have already observed that the problem in the theorem is in NP. To prove completeness, we reduce the problem from Theorem 14 to the problem in the present theorem. This will suffice, since the former problem is already known to be NP-complete.

So let Ω, $\mathcal{F}$, and x, y constitute an instance of the problem from Theorem 14. We must convert it, by a polynomial time computation, into Ω' and $\mathcal{F}'$ such that x and y can be separated in Ω if and only if all pairs of distinct elements can be separated in Ω'. Here we have simplified notation by ignoring the $\mathcal{M}$ component of search systems, since it is irrelevant to the problem.

We may assume, when constructing Ω' and $\mathcal{F}'$, that $x \neq y$. Indeed, if $x = y$ then they obviously cannot be separated, so we need only produce some Ω' and $\mathcal{F}'$ in which not all pairs of distinct elements can be separated, and this task is trivial. We may further

assume that $\mathcal{F}$ is nonempty. Indeed, if $\mathcal{F} = \emptyset$, then all subsets of Ω are invariant, so x and y (being distinct) can be separated, and we need only produce Ω' and $\mathcal{F}'$ in which all pairs can be separated; this too is trivial.

Henceforth, we therefore assume that $\mathcal{F}$ is nonempty and that x and y are distinct. We define Ω' to be the following set, consisting of four copies of Ω plus two additional elements.

$$\Omega' = (\Omega \times \{1,2,3,4\}) \cup \{5,6\}.$$

We abbreviate the ordered pairs $\langle a, 1\rangle$ in Ω' as $a1$, and similarly with 2, 3, or 4 in place of 1. The family $\mathcal{F}'$ contains, for each $f \in \mathcal{F}$, an associated function $f' : (\Omega')^2 \to \Omega'$ defined as follows.

$$\begin{aligned}
f'(a1, b2) &= f(a,b)3 \\
f'(i, aj) &= a4 \text{ for } i \in \{5,6\} \text{ and } j \in \{1,2,3\} \\
f'(x4, y4) &= 5 \\
f'(y4, x4) &= 6 \\
f'(u,v) &= u \text{ for all } u, v \text{ not covered by the previous lines.}
\end{aligned}$$

We have used here the assumption that $x \neq y$ because otherwise the third and fourth lines of the definition of f' would contradict each other. It is clear that Ω' and $\mathcal{F}'$ can be computed from Ω and $\mathcal{F}$ in polynomial time. We must verify that they have the required separation properties, and we break this verification into two lemmas.

Lemma 18 *In $(\Omega', \mathcal{F}')$, every two distinct elements, except possibly 5 and 6, can be separated.*

Proof. By inspecting the definition of f', we find that the following sets and their complements (in Ω') are invariant under $\mathcal{F}'$.

1. $(\Omega \times \{4\}) \cup \{5,6\}$
2. $(\Omega \times \{1,4\}) \cup \{5,6\}$
3. $(\Omega \times \{2,4\}) \cup \{5,6\}$
4. $((\Omega - \{a\}) \times \{4\}) \cup \{5,6\}$ for any $a \in \Omega$.

These sets suffice to separate any two elements of Ω' that come from different sets in the following list, which partitions Ω':

$$\Omega \times \{1\}, \quad \Omega \times \{2\}, \quad \Omega \times \{3\}, \quad \Omega \times \{4\}, \quad \{5,6\}$$

It remains to separate any two distinct elements from the same block of this partition except for 5 and 6.

If the two points come from $\Omega \times \{4\}$, then they can be separated by a set as in item 4 of the list above. If they come from $\Omega \times \{1\}$, then, calling them $a1$ and $b1$, we can separate them because the singleton $\{a1\}$ and its complement are invariant. The same argument applies if they come from $\Omega \times \{2\}$. Finally, if they are $a3$ and $b3$, then they are separated by

$$(\Omega \times \{1,4\}) \cup \{a3, 5, 6\}$$

and its complement, both of which are invariant. □

Lemma 19 *5 and 6 can be separated in* $(\Omega', \mathcal{F}')$ *if and only if* x *and* y *can be separated in* $(\Omega, \mathcal{F})$.

Proof. Suppose first that A and A^c are invariant and separate x and y in Ω. Then $(A \times \{1,2,3,4\}) \cup \{5\}$ and $(A^c \times \{1,2,3,4\}) \cup \{6\}$ are invariant and separate 5 and 6 in Ω'.

Conversely, suppose B and B^c are invariant in Ω' and separate 5 and 6; say $5 \in B$ and $6 \in B^c$. By virtue of the second equation in the definition of f' (applied with $i = 5$) and the invariance of B, we know that if B contains $a1$, $a2$, or $a3$, then it must also contain $a4$ (for the same $a \in \Omega$). (We have used here the assumption that $\mathcal{F} \neq \emptyset$, because we need an f' to use in this invariance argument.) The same argument applies to B^c if we use $i = 6$ instead of 5. As a result, for each $a \in \Omega$, all four of $a1$, $a2$, $a3$, and $a4$ lie in the same one of B and B^c. That is, there is an $A \subseteq \Omega$ such that

$$B = (A \times \{1,2,3,4\}) \cup \{5\} \quad \text{and} \quad B^c = (A^c \times \{1,2,3,4\}) \cup \{6\}.$$

By virtue of the first equation in the definition of f', the invariance of B and B^c under f' implies the invariance of A and A^c under f. Furthermore, if x and y were in the same one of A and A^c, then the last two equations in the definition of f' would force 5 and 6 into the same one of B and B^c, contrary to our assumption. Thus, A and A^c are invariant subsets of Ω separating x and y, as required. □

The two lemmas together tell us that we have a reduction of the problem in the present theorem to the one in Theorem 14. □

7 Conclusions

In the current paper it has been rigorously established that deciding whether or not a given search algorithm can be re-encoded by a binary genetic algorithm is, in the very general case, a complicated (NP-complete) problem. It should be pointed out though, that many well-known types of algorithms, such as the non-linear genetic programming with homologous crossover, can be easily embedded into a binary genetic algorithm. The situation is somewhat analogous to that in analysis: according to the Brownian motion model, the path of a particle is continuous and nowhere differentiable with probability 1, while most continuous functions used in calculus (various combinations of elementary functions, their integrals...) are at least piecewise differentiable.

References

1. Davis, L. (1985) Applying Adoptive Algorithms to Epistatic Domains, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 162-164.
2. Garey, M. and Johnson, D. (1979) Computers and Intractability. A Guide to the Theory of NP-Completeness. W. H. Freeman and Company.
3. Mac Lane, S. (1971) Categories for the Working Mathematician. *Graduate Texts in Mathematics 5*, Springer-Verlag.
4. Michalewicz, Z. (1996). Genetic algorithms + data structures = evolution programs. Springer-Verlag.

5. Mitavskiy B. (2004). Crossover invariant subsets of the search space for evolutionary algorithms. *Evolutionary Computation*, 12(1): 19-46.
http://www.math.lsa.umich.edu/∼bmitavsk/
6. Mitavskiy B. (2003). Comparing evolutionary computation techniques via their representation. In E. Cantú-Paz *et al*. editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Vol. 1, pages 1196-1209, Springer-Verlag.
7. Mitavskiy, B. (2003). A category theoretic method for comparing evolutionary computation techniques via their representation. In B. ten-Cate, editor, *Proceedings of the Eighth ESSLLI Student Session*, pages 201 - 210.
8. Poli, R. (2000). Hyperschema Theory for GP with One-Point Crossover, Building Blocks, and Some New Results in GA Theory. In R. Poli, W. Banzhaf, and *et al*., editors, *Genetic Programming, Proceedings of EuroGP'2000*, Springer-Verlag
9. Radcliffe, N. (1994). The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 10:339-384. http://users.breathemail.net/njr/papers/amai94.pdf
10. Rothlauf F., Goldberg D., Heinzl A. (2002). Network random keys - a tree representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10(1): 75 - 97.
11. Rowe, J., Vose, M., and Wright, A. (2002). Group properties of crossover and mutation. *Evolutionary Computation*, 10(2): 151-184.
12. Stephens, C. (2001). Some exact results from a coarse grained formulation of genetic dynamics. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 631-638, Morgan Kaufmann.
13. Vose, M. (1999). The Simple Genetic Algorithm: Foundations and Theory. *MIT Press*.
14. Vose, M. and Wright, A. (2001). Form invariance and implicit parallelism. *Evolutionary Computation*, 9(3):355-370.