Populations are Multisets – PLATO

Joaquim N. Aparício **Luís Correia** **Fernando Moura-Pires**
CENTRIA, DI - FCT, Universidade Nova de Lisboa
2825-114 Caparica, Portugal
fmp@di.fct.unl.pt lc@di.fct.unl.pt jna@di.fct.unl.pt

Abstract

There are presently many and seemingly different optimization algorithms, based on unrelated paradigms. Herein we propose a framework to encompass those heuristics, providing a common working structure and a basis for their comparison. A formal definition, based on the multiset formalism is presented. We also show how to express some common operators in our framework and we present some (partial) results on relations among them.

1 INTRODUCTION

There are presently many different optimization algorithms, based on a large amount of different paradigms. Especially in optimization heuristics ([Ree95] and [dJS93]), we notice many common features (type and sequence of operators, for instance) that lead us to search for a basic scheme. Therefore, we here introduce a general formal framework (PLATO) for population based optimization heuristics, using a multiset based representation. We show that the multiset formalism is a useful tool to describe these heuristics. It is not intended to provide analysis of properties, but we argue that a multiset based formalism may also be used as an implementation guide, providing new interesting features to a population based optimization algorithm. Some of the algorithms we try to cover with this model are not usually taken as population based (e.g. tabu search) but we show that they can be easily extended towards that direction.

In what follows, we consider, as the problem to solve, the optimization of a function $f(X)$, with X being the vector of the problem variables or of an appropriate codification of their values. There may be a transform converting $f(X)$ into a *fitness function* which we will generally use. The algorithms here considered do not need any other information about $f(X)$, such as derivatives, etc. In our approach the general representation of the population based optimization algorithms is depicted in Fig. 1. We consider $\breve{P}_0$ as the observable population in each iteration of the algorithm. For the sake of generality, there is also a *control* module, responsible for changes of the algorithm parameters or for alterations to the observable population between iterations. Due to space limitations this module will not be mentioned in this paper and we concentrate on presenting the *optimization algorithm* module.

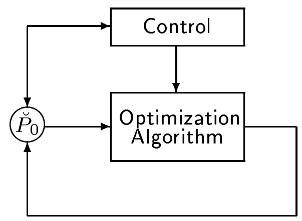

Figure 1: *Population based optimization algorithm*

The decomposition of the *optimization algorithm* module in its operators is presented in Fig. 2. Each rectangular box represents one operator over a population. The first operator in each iteration is *Selection for Reproduction* which selects the individuals to use in further operations. Next there is the *Reproduction* operator which generates new individuals, based on the ones provided by the previous operator. Following we have the *Mutation* operator. Its function is to provide random changes to the input individuals, on a one to one basis. Then the *Decimation* operator is applied, to remove individuals from the population, for instance those with low fitness or lying outside the domain of the fitness function. The last operator *Replacement* is also a form of selection. It takes the population produced in the current iteration cycle and the input observable population to the same iteration, and selects individuals from both to produce the next iteration's observable population.

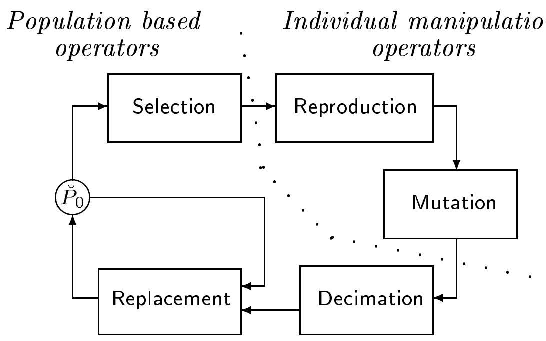

Figure 2: *Optimization Algorithm Operators*

In Fig. 2 we divided the modules into two different groups, corresponding to the basic families of operations in population based heuristics: Population based operations (*Selection for Reproduction*, *Decimation* and *Replacement*), which compare individuals in a population taking into account fitness function and retain only some of them for further use, and Individual based operations (*Reproduction* and *Mutation*) which manipulate individuals' representations.

2 BASIC FORMALISM

The basic concept in these algorithms is *Population*. Intuitively, it is a set of individuals (possibly empty). However, a population may contain a finite number of identical individuals but in a set, by definition, there are no identical elements. Therefore, we consider a population as a multiset of individuals, since it may have repeated elements (and we may refer to the number of times a given individual is present in it).

An individual is characterized by a chromosome (values of the domain variables, identifying one point in the search space) and a personal history (memory). Possible uses of memory are to record the generation number when the chromosome appeared in the population, or to record parenthood information. Some algorithms (Tabu Search [GL95]) definitely adopt this characterization and use this notion of memory associated to an individual.

Definition 1 (Individual) An individual is a pair $\langle c, m\rangle$ where c is a chromosome (a point of the search space) and m is its associated memory.

Definition 2 (Identity and cloning) Given individuals $i_1 = \langle c_1, m_1\rangle$ and $i_2 = \langle c_2, m_2\rangle$ we say that i_2 is a clone of i_1 iff $c_2 = c_1$. Moreover we say that any two individuals are identical if and only if they are two clones with the same memory.

Note that two individuals being identical means (among other things) they are clones, although the converse is not true. A simple case is when we consider memory to be just the individuals age. Two individuals with the same chromosome may appear in different generations. However, if age is considered, then, although being clones, those two individuals are not identical. Furthermore, it is easy to see that, for memoryless populations, cloning is equivalent to identity.

Definition 3 (Search Space) The search space Ω, is defined by a pair of values

$$\Omega = \{\langle c, m\rangle : c \in \mathcal{C}, m \in \mathcal{H}\} \tag{1}$$

where $\mathcal{C}$ is the chromosome space (a chromosome being the coding of the input variables) and $\mathcal{H}$ is the individuals memory space. Therefore, a population will be a collection of individuals with possible identicals. In order to accommodate this multitude of identicals we use the multiset formalism. If the reader is not familiar with this formalism we strongly recommend reading its description in the Annex.

Definition 4 (Population, Support) If Ω is the search space then a population is a mapping $\breve{P} : \Omega \to \mathbb{N}$ such that the set $\{\omega \in \Omega | \breve{P}(\omega) \neq \emptyset\}$ is finite. If $\breve{P}$ is a population we define its support as the set

$$P = \{\omega : \omega \in \breve{P}\}.$$

For population $\breve{P} = \{\{1, 1, 1, 3, 4, 4\}\}$ its support is $P = \{1, 3, 4\}$ and $\breve{P}(1) = 3$, $\breve{P}(3) = 1$ and $\breve{P}(4) = 2$.

Definition 5 (Population Cardinality) Let $\breve{P}$ be a multiset representing a population. The cardinality of the population ($|\breve{P}|$) is defined by

$$|\breve{P}| = \sum_{\omega \in P} \breve{P}(\omega).$$

The cardinality of its support is denoted by $|P|$. Clearly $|P| \leq |\breve{P}|$ always holds. In the previous example $|\breve{P}| = 6$ and $|P| = 3$. It should be stressed that the support cardinality provides a simple measure of population diversity.

3 POPULATION OPERATORS

Population based operators are usually known as selector operators and subdivide into three distinct operators: *Selection for Reproduction*, *Decimation* and *Replacement*. As a matter of fact, in the general scheme here presented, these are the only three operators where fitness function is used.

3.1 SELECTION FOR REPRODUCTION

The basic idea of Selection for Reproduction (σ) is to determine which individuals of the actual population will produce descendants, and it is the very first operator applied in each iteration (generation). Formally, the σ operator is a function that, from a population, produces another population (of reproducers), not necessarily of the same size:

$$\sigma : \mathcal{M}(\Omega) \to \mathcal{M}(\Omega).$$

We will designate by $\breve{I}$ the input population and by $\breve{O}$ the output population of this operator with supports I and O respectively.

Definition 6 (Selection for Reproduction) The Selection for Reproduction operator is defined by:

$$\breve{O} = \sigma(\breve{I}) = \bigsqcup_{i=1}^{|\breve{O}|} \{\{x_i\}\}$$

where x_i are values of a sequence of random variables X_i with values in I and a joint distribution $p(x_1, \ldots, x_{|\breve{O}|})$.

A sample of each variable X_i produces one element of the input population. Therefore, with generality, the output population will be obtained by a sequence of $|\breve{O}|$ samples of random variables (deterministic methods will have only one possible outcome). Support sets of the input and output populations of the σ operator verify $|O| \leq |I|$, meaning that the number of distinct elements in the output is less or equal than the number of distinct elements in the input population.

Thus, the net effect of this operator is to reduce the diversity of the population for reproduction. Normally it will retain, in a non-deterministic way, the more fit individuals, possibly with multiple identicals of some of them. Defining appropriately the N random variables X_i we obtain different σ functions.

Example 1 Simple roulette selection, σ_1. Variables are independent and identically distributed (i.i.d.). Probability mass function ($p(x)$), is defined by

$$p(x = \omega) = \frac{f(\omega)\breve{I}(\omega)}{\sum_{j=1}^{|\breve{I}|} f(\omega_j)\breve{I}(\omega_j)}$$

where $f(\omega)$ represents the fitness value of individual ω. Notice that, since a population is a multiset over its support, only the individuals in the support are distinguishable. We can not distinguish multiple identicals in a population, under the multiset formalism. Therefore, in the equation above, the fitness value for each element ω of the support I is weighted by the number of its identicals $\breve{I}(\omega)$ in the input population.

3.2 DECIMATION

Decimation is an operator that selects individuals based not only on their fitness values, but also based on their individual memory values (similar to Koza's [Koz92]). In fact, in this scheme, it is the only selection operator that takes into account the individual's memory. Population decimation can be built based on individual decimation which is defined as:

Definition 7 Individual decimation is a function $\delta_\theta : \Omega \to \Omega$ such that

$$\delta_\theta(\omega) = \begin{cases} \emptyset & if\ \theta(\omega) \\ \omega & otherwise \end{cases}$$

where θ is a boolean function $\theta : \Omega \to \{true, false\}$.

Using this, we define population decimation:

Definition 8 Population decimation is a function $\bar{\delta}_\theta : \mathcal{M}(\Omega) \to \mathcal{M}(\Omega)$ where

$$\begin{aligned} \bar{\delta}_\theta(\emptyset) &= \emptyset \\ \bar{\delta}_\theta(\{\{\omega\}\}) &= \{\{\delta_\theta(\omega)\}\} \\ \bar{\delta}_\theta(\breve{X} \sqcup \breve{Y}) &= \bar{\delta}_\theta(\breve{X}) \sqcup \bar{\delta}_\theta(\breve{Y}). \end{aligned}$$

Example 2 Annealing decimation[KG83]

$$\theta_1(\langle c, m\rangle) = \begin{cases} false & if\ rand(1) < \exp(-\frac{f(c)-m}{T}) \\ true & otherwise \end{cases}$$

where $f(c)$ is the fitness value of the chromosome c, m is the memory of the chromosome (the fitness value of the parent) and T is the annealing temperature.

Example 3 Tabu decimation[GL95]

$$\theta_2(\langle c, m\rangle) = \begin{cases} false & if\ not\ \ tabu(m) \\ true & otherwise \end{cases}$$

where $tabu(m)$ is true if the individual is in tabu.

3.3 REPLACEMENT

The *Replacement* operator (ρ) is responsible for the last phase of each iteration of the optimization algorithm. It selects members from the actual population ($\breve{P}_0$) and elements produced in the current iteration of the algorithm to form the population to the next generation (similar to Fogel's "Competition" [Fog95]).

Definition 9 (Replacement) Replacement is a two argument function on populations:

$$\rho : \mathcal{M}(\Omega) \times \mathcal{M}(\Omega) \rightarrow \mathcal{M}(\Omega).$$

The two input populations are not handled necessarily in the same way. Therefore the operator is further decomposed into two modules *Choice* and *Integration* (see Fig. 3) allowing for the necessary differentiation. Input $\breve{I}_1$ corresponds to the observable population $\breve{P}_0$ used in the beginning of the present iteration, while input $\breve{I}_2$ is the population resulting from the previous operator (Decimation) in the cycle (see Fig. 2).

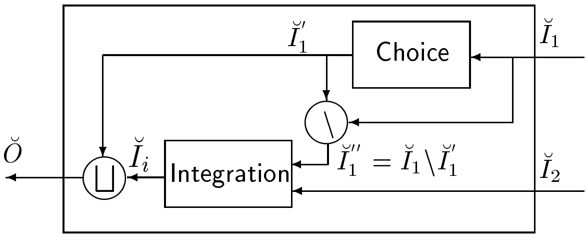

Figure 3: *Replacement :* $\breve{O} = \rho(\breve{I}_1, \breve{I}_2)$

The *Choice* operation selects some elements ($\breve{I}_1'$) from the actual population ($\breve{I}_1$), to go directly to the next generation. The remaining elements of the actual population($\breve{I}_1''$) are passed on to the *Integration* module by taking the result of the operation $\breve{I}_1 \backslash \breve{I}_1'$. This one produces population $\breve{I}_i$ to complete the next generation population. It selects individuals from population $\breve{I}_2$, produced in the current algorithm iteration, and from population $\breve{I}_1''$ received from the *Choice* submodule. Finally, the output of the *Replacement* operator ($\breve{O}$) will constitute next generation population obtained by the union of populations $\breve{I}_1'$ and $\breve{I}_i$.

With such a structure, the *Replacement* operator may preserve a defined percentage of the actual population and have its remaining elements competing with the ones produced during the current iteration, to go through to the next generation. In case the operator takes both input populations indistinctly we will have $\breve{I}_1' = \emptyset$ and thus $\breve{I}_1'' = \breve{I}_1$.

A final remark regarding population diversity: we have $|O| \leq |I_1 \cup I_2|$ although nothing can be said about the population diversity of two consecutive iterations ($|O|$ and $|I_1|$).

Example 4 (ρ_2) Evolution Strategies with a (μ, λ) model ([Sch95]). Notice that in this case $|\breve{I}_1| = \mu$ and $|\breve{I}_2| = \lambda$. The module operations are defined as:

$$\begin{aligned} Choice(\breve{I}_1) &= \emptyset \Rightarrow \breve{I}_1' = \emptyset, \breve{I}_1'' = \breve{I}_1 \\ Integration(\breve{I}_1'', \breve{I}_2) &= \mu\_best(\breve{I}_2), \end{aligned}$$

where *μ_best* is a function returning the μ best individuals of a population.

4 INDIVIDUAL OPERATORS

4.1 REPRODUCTION

Different sorts of reproduction have been considered in evolutionary algorithms. Existing reproduction operators may use one (sometimes referred to as neighborhood [GL95]), two (also known as crossover [Gol89]) or more parents (which has also been named orgy [EvKK95]).

We next show how to accommodate those approaches within this framework. We first introduce the general case of n–parental reproduction and then we show that it gracefully falls into the particular cases of one parent and two parents reproduction.

Definition 10 (n-parental reproduction) A single n-parental reproduction, is an operation $\tau_\pi^{\langle n \rangle} : \mathcal{M}(\Omega) \rightarrow \mathcal{M}(\Omega)$ involving $m \leq n$ individuals, which generates K descendants.

Let $\breve{P} = \{\{\omega_1, \ldots, \omega_m\}\}$ with $m \leq n$, then $\tau_\pi^{\langle n \rangle}(\breve{P})$ is defined by:

$$\tau_\pi^{\langle n \rangle}(\breve{P}) = \begin{cases} \breve{P} & \text{if } m < n \\ \breve{P} & \text{if } m = n \wedge \\ & \quad rand(1) > p_r \\ \pi(\omega_1, \ldots, \omega_{m=n}) & \text{otherwise,} \end{cases}$$

where $\pi : \Omega^n \rightarrow \mathcal{M}(\Omega)$ and p_r is a user defined reproduction probability. Function π will generate K descendants from n parents. Notice that the function π is the only one, in this framework, that may change the value of the individual memory.

Definition 11 (Population n-parental reprod.) Based on the previous definition, we define $n-parental$ reproduction over populations $\bar{\tau}_\pi^{\langle n \rangle} : \mathcal{M}(\Omega) \rightarrow \mathcal{M}(\Omega)$ as:

$$\begin{aligned} \bar{\tau}_\pi^{\langle n \rangle}(\emptyset) &= \emptyset && (2) \\ \bar{\tau}_\pi^{\langle n \rangle}(\breve{X}) &= \tau_\pi^{\langle n \rangle}(\breve{X}) \quad \text{if } |\breve{X}| < n && (3) \\ \bar{\tau}_\pi^{\langle n \rangle}(\breve{X} \sqcup \breve{Y}) &= \tau_\pi^{\langle n \rangle}(\breve{X}) \sqcup \bar{\tau}_\pi^{\langle n \rangle}(\breve{Y}) \text{ if } |\breve{X}| = n \end{aligned}$$

In the case of mono–parental reproduction, $n = 1$, equation 3 above reduces to equation 2.

Example 5 Mono–parental reproduction - the case of tabu search with a swapping neighborhood operator [GL95]. Then the reproduction function is

$$\pi_3(\langle c, m\rangle) = \bigsqcup_{i=1}^{K} \{\{XY\,|\langle c, m\rangle\}\}.$$

In this case we have a distribution of two conditional joint random variables:

$$(XY\,|\langle c, m\rangle\,, p\,(xy\,|\langle c, m\rangle\,)\,, Q), \tag{4}$$

where Q is a subset of Ω :

$$Q = \left\{ \begin{array}{ll} \langle x, y\rangle : & x_r = c_r, r = 1, \cdots, l \wedge \\ & \quad r \neq i \wedge r \neq j \wedge x_i = c_j \wedge x_j = c_i \\ & \wedge \\ & y = tabu\_update(m, i, j) \end{array} \right\}$$

and $p\,(xy|\,\langle c, m\rangle) = \frac{1}{\binom{l}{2}}$.

Example 6 Bi-parental reproduction without memory, $K = 2$ and one-point crossover. Given any two chromosomes it generates two chromosomes from the genetic material of the parents. Let ω_a and ω_b be two individuals, both with no memory: $\omega_a = \left\langle \left(c_{a_1}, \ldots, c_{a_i}, c_{a_{i+1}}, \ldots, c_{a_l}\right), \emptyset\right\rangle$ and $\omega_b = \left\langle \left(c_{b_1}, \ldots, c_{b_i}, c_{b_{i+1}}, \ldots, c_{b_l}\right), \emptyset\right\rangle$, with $1 \leq i < l$.

Then bi-parental reproduction is:

$$\begin{array}{ll} \pi_4(\omega_a, \omega_b) = \{\{ & \left\langle \left(c_{a_1}, \ldots, c_{a_i}, c_{b_{i+1}}, \ldots, c_{b_l}\right), \emptyset\right\rangle, \\ & \left\langle \left(c_{b_1}, \ldots, c_{b_i}, c_{a_{i+1}}, \ldots, c_{a_l}\right), \emptyset\right\rangle \;\}\}, \end{array}$$

with $i = int((l-1)rand(1)) + 1$.

4.2 MUTATION

Mutation randomly changes an individual's chromosomes. It does not affect memory. The population operator is based on the individual operator which outputs one individual for each input. This and the fact that it does not change memory are the aspects distinguishing *Mutation* from mono–parental reproduction.

Definition 12 Chromosome mutation is a non deterministic function defined by $\gamma : C \to C$, where this function is defined by the random variable

$$(X\,|c\,, p\,(x\,|c)\,, S_\gamma)$$

and the random variable $X|c$ takes values in S_γ (a subset of $\mathcal{C}$) with a probability distribution $p(x|c)$.

Definition 13 (Individual mutation) Individual mutation is a non deterministic function $\mu_\gamma : \Omega \to \Omega$ defined by

$$\mu_\gamma(\langle c, m\rangle) = \left\{ \begin{array}{lll} \langle c, m\rangle & if & rand(1) > p_m \\ \langle x, m\rangle : x = \gamma\,(c) & & otherwise \end{array} \right.$$

where $\langle c, m\rangle$ is an individual $\omega \in \Omega$, with chromosome c and memory m, p_m is mutation probability and $\gamma\,(c)$ is the chromosome mutation function.

Definition 14 (Population mutation) The population mutation is a function $\bar{\mu}_\gamma : \mathcal{M}(\Omega) \to \mathcal{M}(\Omega)$ defined by

$$\begin{array}{rcl} \bar{\mu}_\gamma\,(\emptyset) & = & \emptyset \\ \bar{\mu}_\gamma\{\{\omega\}\} & = & \{\{\mu_\gamma(\omega)\}\} \\ \bar{\mu}_\gamma(\check{X} \sqcup \check{Y}) & = & \bar{\mu}_\gamma(\check{X}) \sqcup \bar{\mu}_\gamma(\check{Y}). \end{array}$$

Example 7 Bitwise mutation, γ_2. It flips at least one bit in the chromosome. Let $c = (c_1, \cdots, c_l)$ be a string of bits

$$S_{\gamma_2} = \left\{ \begin{array}{ll} x : x = & (c_1, \cdots, c_l) \oplus (y_1, \cdots, y_l), \\ & y_i \in \{0, 1\}, \sum_{i=1}^{l} y_i > 0 \end{array} \right\}$$

and

$$p(x|c) = q^{\sum_{i=1}^{l} y_i}(1-q)^{l-\sum_{i=1}^{l} y_i},$$

where q is the bit flipping probability.

Example 8 Swap mutation, γ_3. Exchanges two randomly chosen distinct genes. Let $c = (c_1, \cdots, c_l)$ be any chromosome

$$S_{\gamma_3} = \left\{ \begin{array}{ll} x : x_r = & c_r, r = 1, \cdots, l \wedge i \neq j \wedge \\ & r \neq i \wedge r \neq j \wedge x_i = c_j \wedge x_j = c_i \end{array} \right\}$$

and $p(x|c) = \frac{1}{\binom{l}{2}}$.

5 CONCLUSIONS

We introduced PLATO as a formal framework for expressing optimization heuristics, and we showed how some well-known heuristics can be described within this framework. PLATO revealed to be general enough to cope with different paradigms. It also provides a clear way to construct hybrid algorithms by a mere combination of standard operators.

In the future, other heuristics will be expressed using PLATO. This will provide feedback for further refinements and possible improvements of the framework. Examples presented here are mainly memoryless algorithms and special attention will be given to the use of individual's memory in a near forthcoming report.

One interesting aspect to explore is to have the implementation working with the multiset representation. Populations will be represented by arrays of different individuals associated with their number of copies. In our opinion this will provide a larger population diversity along the iterations of the algorithm.

Acknowledgments

We thank João Moura–Pires for fruitful discussions in the early stages of this work. Luís Correia's work was partially done at École Polytechnique Fédérale de Lausanne (Switzerland), supported by Fundação Calouste Gulbenkian and by Fundação para a Ciência e Tecnologia - BSAB/51/98 - (Portugal).

ANNEX

Definitions below are taken from [Fer95].

Definition 15 (Multiset) Let S be any set. A finite multiset over S is a function $\alpha : S \to \mathbb{N}$ such that the set $\{s \in S \mid \alpha(s) \neq 0\}$ is finite. The set of all finite multisets over S is denoted by $\mathcal{M}(\mathcal{S})$.

A set-like notation $\{\{\}\}$ is used to denote a multiset. Operations similar to the ones applied on sets (e.g. $\in$, $\cup$, $\subseteq$, etc.) are also applied to multisets. Round symbols denote operations on sets (e.g. $\subseteq$) and similar square symbols mean the same operation on multisets (e.g. $\sqsubseteq$). Operator $\in$, is used both for sets and multisets. We also abbreviate finite multiset to multiset.

Definition 16 Let α, β be arbitrary multisets over S. The operations $\in, \sqcup, \sqsubseteq, \backslash, \sqcap$ on $\mathcal{M}(S)$, the set of finite multisets, are defined as follows:

- $\forall s \in S : s \in \alpha \iff \alpha(s) > 0$,
- $\alpha \sqcup \beta$ is the multiset defined by $(\alpha \sqcup \beta)(s) = \alpha(s) + \beta(s)$, for all $s \in S$,
- $\alpha \sqsubseteq \beta \iff \forall s \in S : \alpha(s) \leq \beta(s)$. If the last inequality is strict for all $s \in S$ the we have strict inclusion of multisets, i.e., $\alpha \sqsubset \beta$,
- $\alpha \sqcap \beta$ is the multiset defined by $(\alpha \sqcap \beta)(s) = \min(\alpha(s), \beta(s))$, for all $s \in S$,
- $\alpha \backslash \beta$ is the multiset defined by $(\alpha \backslash \beta)(s) = \max(\alpha(s) - \beta(s), 0)$, for all $s \in S$.

Definition 17 Let $\Phi : A \to \mathcal{M}(A)$ be a function. This function is extended to a function $\bar{\Phi}: \mathcal{M}(A) \to \mathcal{M}(A)$ as follows:

- $\bar{\Phi}(\emptyset) = \emptyset$,
- $\bar{\Phi}(\{\{a\}\}) = \Phi(a)$,
- $\bar{\Phi}(X \sqcup Y) = \bar{\Phi}(X) \sqcup \bar{\Phi}(Y)$.

References

[dJS93] Kenneth de Jong and William Spears. On the state of evolutionary computation. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993.

[EvKK95] A. E. Eiben, C. H. M. van Kemenade, and J. N. Kok. Orgy in the computer: Multi–parent reproduction in genetic algorithms. In F. Morán, J. J. Morelo, A. Morelo, and P. Chacón, editors, *Advances in Artificial Life – Third European Conference*, number 929 in Lecture Notes in Artificial Intelligence, pages 934–945. Springer Verlag, 1995.

[Fer95] M. C. Fernández Ferreira. *Termination of Term Rewrinting.* PhD thesis, Universiteit Utrecht, 1995.

[Fog95] David Fogel. *Evolutionary Computation: Towards a new Philosophy of Machine Inteligence.* IEEE Press, 1995.

[GL95] Fred Glover and Manuel Laguna. Tabu search. In Reeves [Ree95], chapter 3, pages 70–150.

[Gol89] David Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* John Wiley and Sons, 1989.

[KG83] S. Kirkpatrick and C. D. Gelatt. Optimization by simulated aneealing. *Science*, 220(4598):671–680, 1983.

[Koz92] John R. Koza. *Genetic Programming.* MIT Press, 1992.

[Ree95] Colin R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems.* Advanced Topics in Computer Science. McGraw-Hill, London, 1995.

[Sch95] Hans-Paul Schwefel. *Evolution and Optimum Seeking.* John Wiley and Sons, 1995.