

**APPENDIX A**

**GENETIC ALGORITHMS: FURTHER DETAILS**

A.1 Synthesis of Evolutionary Computation

Scholars, at the end of 1980s and beginning of 1990s, started meeting irregularly at some conferences to deliver and present their researches and communicate ideas. Such conferences may be mentioned as an example of that, but not limited to, are the International Conference on Genetic Algorithms (ICGA) [103] and Parallel Problem Solving from Nature (PPSN) [218]. These two conferences were established in 1985 and 1990 correspondingly as biannual conferences. Though, the IEEE Conference on Evolutionary Computation (ICEC) was started in 1994 as an annual one. However, at the end of 1999s, the two conferences ICGA and ICEC were combined with GP and EP conferences respectively generating the annual GECCO [26] and CEC [1] conferences.

The term of *Evolutionary Computation* was intended at the begging of PPSN conference to include all EA research dialects. In addition, the term *Evolutionary Algorithm* was used as the generic form of the group of algorithms. The following definition gives an example of them main processes in the Evolutionary Algorithm. This formulation can be fit on the general Evolutionary Algorithms which have not been integrated with other algorithms.

Evolutionary Algorithm:

1. INITIALIZE population with random candidate solutions.
2. EVALUATE each candidate.
3. REPEAT UNTIL TERMINATION CONDITION:
   a. SELECT parent candidates. (Reproductive selection)
   b. RECOMBINE AND/OR MUTATE parent candidates into child candidates.
   c. EVALUATE child candidates.
   d. SELECT candidates for next generation. (Survival selection)
A.2 Population Size

In literature there are also lots of empirical works on population sizing. Arabas and his co-researcher [12] introduced “Genetic Algorithm with Variable Population Size (GAVaPS)”, which considers the size of population as an obvious parameter through adding age and highest level of lifetime attribute for each individual in any given population. By that and upon the individual’s birth, the maximum level of lifetimes is assigned to it depending on its newborn fitness, although the age set to zero at birth. The individual’s age is increased by one for each generation, and when individual’s age reaches its limit of predefined level of lifetime it is marked as expired and then removed from the current population. However, this mechanism works towards making population size recognizable with dynamic adjusting and not just a parameter, as well as it makes survivor selection strategy unnecessary. In contrast of that, Bäck et al. [21] introduced “The Adaptive Population size Genetic Algorithm (APGA)” which uses a steady-state GA. The APGA works in order to maintain the best individual’s lifetime invariable through processing time when population individuals become older.

Harik and Lobo [113] and Lobo [87, 152] pointed a “parameter-less GA (PLGA)” which simultaneously develops a list of different sizes of populations. In this instance, PLGA allow more function evaluations for smaller populations than the bigger ones, for example, population ($P_i$) is evolved and create generations four times more than the population ($P_{i+1}$). However, the algorithm will drop any smaller population once it converges. Costa et al [44] introduced “Random Variation of the Population Size GA (RVPS)”. Through this algorithm, they compared GAs with fixed and varying population sizes, for a given N they aimed to make population size varied every N fitness evaluations. They found out, the GA’s performance with changeable population size, in the worst case, is doing the same effort of the simple GA, regarding less the population size is changed randomly throughout the process.

Hinterding, et al. [121] illustrated an adaptive GA technique. They mainly used three groups of populations along with different population size each. The idea behind this technique was to make the size of these populations to be adapted at regular periods (epochs) on the basis of the current state of the search, leading algorithm to make the performance of the group in the maximum with the mid-most size. However, fitness diversity was used for this algorithm as criterion to varying the population sizes.

Schlierkamp-Voosen and Mühlenbein [213] work used a competition scheme between sub-populations. They aimed to modify all sub populations’ size along with
changing the size of total population. There is a quality and gain criteria attached to
each group, which control the change’s level in the size of groups. However, the
algorithm mechanism was built on the way to allow only the size of the fittest group to
increase. Another empirical study based on Goldberg’s research [91] was introduced by
Robert Smith [227] which uses a dynamic mechanism to adapt the size of population
under controlling of the selection error’s probability. He concluded that by
automatically adapting population size, Genetic Algorithms could control their search
process and would effectively adjust and contract to control variations in problem
complexity and in available computational resources.

A.3 Selection Operators

Unlike linear ranking selection, the exponential ranking selection schema [169]
assigns probabilities of the ranked individuals through exponential weighting.
Parameter \( c \) is used as the base of the exponent in which \( c \in [0, 1] \) of the selection
method. Thus, the probability of the each individual can be illustrated as follow:

\[
P_i = \frac{c^{\mu-i}}{\sum_{j=1}^{\mu} c^{\mu-j}}; \quad i \in \{1, \ldots, \mu\}
\]

The sum \( \sum_{i=1}^{\mu} c^{\mu-i} \) normalizes the probabilities to make sure that \( \sum_{i=1}^{\mu} P_i = 1 \)

Tournament selection as it was stated by Bäck [14] and Blickle [29], may be
considered as the second most well-known reproductive selection strategy. A group of
size \( t \in [2, \mu] \) of individuals from the population are randomly chosen. These
individuals according to their fitness are dynamically ranked and then the top two
chosen as parents for the next generation individuals which latter will participate within
the genetic operations like mutation and crossover. This process will be repeated until
producing new population of the next generation.

Boltzmann schemes is an important selection technique in EA as it has theoretical
properties which help in theoretical analysis, it’s derived from Simulated Annealing
(SA) [144]. SA is employing such search scheme so called generate-and-test which
derived from a physical analogy, instead of a biological one. Theoretically it’s formed
on a similarity of annealing techniques employed to reduce the internal stresses in the material sample. Annealing mechanism achieved by first increasing material temperature to raise the size of its crystals and release the stresses, and then the temperature slowly decreased with control to allow the stresses to become stable throughout the material.

Boltzmann selection schemes have many commonalities to the temperature control of simulated annealing, in which the selection pressure can be managed analogously to temperature. This gives the advantage instead of using constant selection pressure, it can be controlled. In the initial set of generations, the selection pressure can be slowly decreased (like SA temperature), which will initiate a more explorative mode. Then, after a while of the process and a better area of the search space have been allocated, the selection pressure may be increased towards highly-fit individuals which is similar to temperature dropping in Simulated Annealing in order to reach zero degree for more rapidly towards the optimum. For more details about Boltzmann selection see good text of Mahfoud & Goldberg [160] and de la Maza & Tidor [56].

A.4 Mutation Operator

Hessner and Männner [119] derived theoretically optimal schedules for deterministically changing $P_m$ for the counting-ones function. They suggest

\[
P_m = \sqrt{\frac{\alpha}{\beta}} \cdot \frac{\exp \left( -\frac{\gamma t}{2} \right)}{\lambda \sqrt{t}}
\]

Where $\alpha$, $\beta$ and $\gamma$ are constant, and $t$ (time) is the generation number. In contrast of that, another research of Bäck [15] illustrated an optimal mutation rate decrease method of $P_m$ in opposed to a function of time as a function of distance, like:

\[
P_m (f(x')) \approx \frac{1}{2(f(x') + 1) - t}
\]

While the function to control decrease schedule of $P_m$ illustrated by Bäck and Schütz [23] used time ($t$) instead of distance which constrained $P_m(t)$ so that the initial
mutation rate value \( P_m(t) = 0 \) and for later stages \( P_m(T) = \frac{1}{T} \) where \( T \) is the maximum evaluation number, which can be clearly noticed from following formula:

\[
P_m(t) = (2 + \frac{l - 2}{T} \cdot t)^{-1}, \quad 0 \leq t \leq T
\]

Janikow and Michalewicz [130] through their experimental study tested a non-uniform mutation (NUM) where for a selected individual \( S^t = \{v_1, v_2, \ldots, v_k, \ldots, v_l\} \) of current generation \( t \) and \( v_k \in [l_k, u_k] \) where \( l_k \) and \( u_k \) are the lower and upper bounds of \( v_k \) respectively, be the element to be mutated for the next generation \( t+1 \), the resulting individual \( S^{t+1} \) which can be represented as \( S^{t+1} = \{v'_1, v'_2, \ldots, v'_k, \ldots, v'_l\} \) with \( k \in [1, l] \), where:

\[
v'_k = \begin{cases} v_k + \Delta(t, u_k - v_k), & \text{if random digit is 0} \\ v_k - \Delta(t, v_k - l_k), & \text{if random digit is 1} \end{cases}
\]

The function \( \Delta(t, y) \) returns a value in the range \([0, y]\), such that the probability of \( \Delta(t, y) \) being close to 0 increases as \( t \) increases (i.e. generation number). This property causes this operator to search the space uniformly initially (when is small), and very locally at later stages. In the experiments reported in [130], the following function \( \Delta(t, y) \) was used:

\[
\Delta(t, y) = y \cdot r \cdot (1 - \frac{t}{T})^b
\]

Where \((r)\) is a random number from the range \([0, 1]\), \((T)\) is the maximal generation number and \((b)\) is a system parameter determining the non-uniformity degree.

Michalewicz [167] proposed random (uniform) mutation. In this type of mutation, a selected gene \( v_k \) is substituted with a random value \( r_k \) between its lower and upper bounds \( r_k \in [l_k, u_k] \). A special type of uniform mutation is “boundary mutation” [168] in which a gene \( v_k \) with equal probability is substituted by either its lower bound \( l_k \) or
upper bound $u_k$. The boundary mutation usefully applied to type of problems in which their results on the boundary or very close to the boundary.

$$v'_k = \begin{cases} l_k & \text{if random digit is 0} \\ u_k & \text{if random digit is 1} \end{cases}$$

Mäkinen with his co-researcher [161] first suggested a mutation strategy named “Exponential Mutation” to solve some multidisciplinary shape optimization problems using GA in aerodynamics and electromagnetics. Meittinen and his co-researchers [165] later used this strategy to solve the large set of constrained optimization problem. While on 2007, Deep and Thakur through their work used this mutation and named it Mäkinen, Periaux and Toivanen Mutation (MPTM). In this mutation if the element $v_k$ of a selected individual $S = \{v_1, v_2, \ldots, v_k, \ldots, v_l\}$ for the current generation with $v_k \in [l_k, u_k]$ the resulting individual for the next generation will be a vector $S' = \{v_1, v_2, \ldots, v'_k, \ldots, v_l\}$ with $k \in [1, l]$ and $v'_k = (1 - \hat{t})l_k + \hat{t}u_k$ where:

$$\hat{t} = \begin{cases} t - t\left(\frac{t - r}{t}\right)^b & \text{if } r < t \\ t & \text{if } r = t \\ t + (1 - t)\left(\frac{r - t}{1 - t}\right)^b & \text{if } r > t \end{cases}$$

Where $t = \frac{v - l_k}{u_k - v}$ and $r$ be a uniformly distributed random number between 0 and 1.

Munteanu and Lazarescu [179] suggested PCA-mutation. This mutation strategy is based on principal component analysis (PCA) which is familiar statistical technique that has been widely used in compression and data analysis. PCA-mutation is observed to achieve a higher level of population diversity than classical ones like uniform mutation and non-uniform mutation on IIR Filter design problem by Deczky Method [185] that requires a global search strategy.

Deb and Agrawal [58] introduced polynomial mutation (PM) operator which is one of the most widely used mutation operator that successfully applied in solving single and multi-objective optimization problems [57]. Polynomial mutation is based on
polynomial probability distribution instead of normal distribution to produce new offspring closer to their parents and therefore enhance the real-coded genetic algorithm.

In polynomial mutation if the element $v_k$ from individual $S = \{v_1, v_2, ..., v_k, ..., v_l\}$ has been selected from current population to be mutated using PM introduces the element $v'_k$ of the individual $S' = \{v_1, v_2, ..., v'_k, ..., v_l\}$ for the population of the next generation where:

$$
v'_k = v_k + (u_k - l_k) \cdot \delta_k
$$

the parameter $\delta_k$ can be obtained by the polynomial probability distribution:

$$
P(\delta) = 0.5 \left( \eta_m + 1 \right) \cdot (1 - \delta^\eta_m)
$$

where $\eta_m$ is the polynomial distribution index (non-negative real number), and $r_k$ is a random number in $[0, 1]$.

$$
\delta_k = \begin{cases} 
-1 + \left(2 \cdot r_k \right)^{1/\eta_m+1} & \text{if } r_k < 0.5 \\
1 - \left[2 \cdot (1 - r_k) \right]^{1/\eta_m+1} & \text{if } r_k \geq 0.5
\end{cases}
$$

However, the shape of probability distribution is controlled by the parameter $\eta_m$ and distribution is not dynamically changed with generations.

Wavelet mutation was first introduced by Ling and Leung [150] which is mainly based on wavelet theory. Wavelet mutation (WV) uses Morlet wavelet as the mother wavelet [51]. The following equation can be used as an example of mother wavelet, and figure 2.4-1 shows the Morlet wavelet:

$$
\psi(x) = e^{-x^2/2} \cos(5x)
$$
The wavelet mutation operation shows a fine-tuning ability therefore the GA performance used WM is illustrated better than some other real coded GAs performance on a group of benchmark test functions as well as economic load dispatch and tuning an associative-memory neural network. In wavelet mutation, every element of population individuals will have a chance to mutate which is controlled by a mutation probability $P_m \in [0, 1]$ which is defined by the user. For current iterations $(i)$ each element $v_k$ from individual $S = \{v_1, v_2, ..., v_k, ..., v_l\}$ where $1 \leq k \leq l$, a random number $r \in [0, 1]$ will be generated such that if $r \leq P_m$ a mutation operation will take place on that $v_k$ as follow:

$$
v'_k = \begin{cases} 
  v_k + \sigma \cdot (u_k - v_k) & \text{if } \sigma > 0 \\
  v_k + \sigma \cdot (v_k - l_k) & \text{if } \sigma \leq 0
\end{cases}
$$

As been mentioned before $l_k$ and $u_k$ are the lower and upper bounds of $v_k$ respectively and if Morlet wavelet has been selected as the mother wavelet, then:

$$
\sigma = \frac{1}{\sqrt{a}} e^{-\frac{(\varphi/a)^2}{2}} \cos(5 \frac{\varphi}{a})
$$

the parameter $\varphi$ is a random number in domain $[-2.5a, 2.5a]$, while parameter $(a)$ can be calculated using the following equation:
\[ a = e^{-\ln(g) \cdot \left(1 - \frac{t}{t_{\max}}\right)^{\xi_{wm} + \ln(g)}} \]

where \( \xi_{wm} \) is the shape parameter of the monotonic increasing function, and \( g \) is the upper limit of the parameter \( a \), \( t_{\max} \) is the total number of iterations.

**A.5 Crossover Operator**

The main idea of crossover is to allow that the genetic strings to be mutual from at least two parents in some way to simulate the reproduction action in natural populations. The following is the main and common types of crossover operator.

**A.5.1 One-Point Crossover**

Holland [122] was the first one who suggested the standard and original formulation of recombination which is so called one-point crossover which was previously analysed by DeJong [52]. The one-point crossover mechanism can be described as follow:

1. Two parents \( A, B \) are selected from population.
2. A random position \( P \in [0, l - 1] \) is chosen where \( l \) is individual's length.
3. The genes string of parent \( A \) is copied from location 0 to \( P \) to the offspring genome.
4. The offspring genome is completed with the values from genes string of parent \( B \) from location \( P+1 \) to \( l-1 \).

The original form by Holland was to generate two offspring where the rest of parent \( A \) and \( B \) will form the second child so that by reversing the previous scheme. Figure 2.4-2 illustrates a default form of one-point crossover.
A.5.2 Two-Points Crossover

There is advantage to have more points of crossover operator which is make the problem searching space probably be searched more deeply. As alternative form of one-point crossover, the two-point crossover schema can be described as follows (see figure2.4-3a):

1. Two parents A, B are selected from population.
2. Two locations \( P_1 \) and \( P_2 \) are chosen randomly from range \( [0, l - 1] \) with each number equally likely.
3. Two new children \( C_1, C_2 \) are formed from exchanging the alleles from each parent to another, where child \( C_1 \) will has alleles from parent A and B so that \( C_1 = \{A_{P_1}^0, B_{P_1+1}^P, A_{P_1+1}^{l-1}\} \), while child \( C_2 \) will be formed from parent A and B by taking the rest of alleles so that \( C_2 = \{B_{P_1}^P, A_{P_1+1}^P, B_{P_1+1}^{l-1}\} \)

In fact, some studies represent the two-point crossover schema in different way using ring shape. DeJong [52] first represents each population member is imagined as ring rather than linear strings, it is build up by joining the individual's ends together. Then, a two cut points \( P_1, P_2 \) will be chosen to make a segment of alleles to be exchanged between parents forming two offspring. Figure2.4-3b illustrates this process. Here, the two-point crossover can be seen as one-point crossover with one of the cut points is considered at the start of the individual string, with the intention that the two-point crossover can make the same performance of one-point crossover. This way of representing the individuals (ring format) in order to apply the two-point crossover may produce extra variety of building blocks (segments) since they can be able to "wrap
around" at the end of the string. Generally, the two-point crossover is classified better than one-point crossover.

![Two-Point Crossover](image)

**Figure A-3: Two-Point Crossover; a) Normal representing, b) Ring representing**

### A.5.3 Multi-Points Crossover

The Multi-Point (M-point) Crossover uses (at least) two parents as input producing two offspring. This type of crossover used more than two cut points (m-points) in its schema to generate offspring. The crossover mechanism to produce children and after determining the number of cutting points, can be described as follow:

1. Select the crossover (cutting) points' positions $p_i : i = 1 ... m$ and $m \in [0, l - 1]$.
2. The first child gets the even segments from the first parent, with the odd segments from the second parent.
3. The second child gets the odd segments from the first parent, along with the even segments from the second parent.

The m-point crossover can be seen as two-point crossover if number of cutting points equal to two i.e. $m=2$. Such that, the first and third segments from parent A with second segment from parent B will used to form child C1, while the first and third segments from parent B with second segment from parent A will used to form child C2. Figure2.4-3 shows an example for $m=4$. 

166
A.5.4 Uniform Crossover:

A number of empirical studies have projected the advantage of increasing the crossover points [228]. The uniform crossover can be understood as a logical extension of the multi-point crossover is pointed. The formulation of uniform crossover drastically differs to the other types of crossovers schema. Uniform crossover employees a pre-defined template (crossover mask) to process its form instead of randomly selecting the cut points for crossover. The uniform crossover mask (bit-mask) consists of a series of zeros and ones with length equal to problem dimension \( l \). However, this bit-mask is usually built up through an exact 50/50 probability of choosing each single bit to be zero or one. Each allele value in offspring is generated by copying the corresponding allele from one or the other parent chosen according to the generated crossover bit-mask. The crossover formulation to produce new offspring can be described as follow:

1. Two parents \( A, B \) are chosen from population.
2. For each index of the bit-mask, if the bit value is one then the offspring will get allele value from parent \( A \).
3. If the bit value is zero, allele value will be copied from parent \( B \) to the offspring.

In fact, it is not so important to generate such a bit-mask in real coding. The simple way is to use a ‘coin flip’ at each location can do the job. However, to understand and analyse the uniform crossover behaviour in theory, the bit-mask explanation is
necessary. Figure 2.4-5 illustrates uniform crossover. This type of crossover allows using recombination operator with more than two parents. The defined schema is vary where a simple method uses bit-mask to state the selection between parent A against B, this process is extended to make selection between a group parents.

\[
\begin{array}{c}
\text{Parent A} & 1010010101011101101 \\
\text{Parent B} & 01101001001110011011 \\
\text{Bit-Mask} & 11100011001011001010 \\
\text{Child C1} & 10101001001101101001 \\
\text{Child C2} & 01100010101010111111 \\
\end{array}
\]

Figure A-5: Uniform Crossover with bit-mask

**A.5.5 Adaptive Crossover:**

A different type of crossover it can be found in literature is Adaptive crossover. This crossover in evolutionary algorithm used, as it runs, to enhance process performance by making a decision which form of crossover operator or operators will be the optimal to be used for current generation. This concept can be applied by two different ways, either locally or globally regarding to level of applying crossover operator.

**Locally:** adaptive crossover operator can be processed after selecting two parents A, B from population then generating one or two children to be added to population for next generation. This child is build up by looking at the parents' allele value for each locus. If the allele values on both parents are equal to one then uniform crossover is used. If they are zero on both side then two-point crossover is chosen to be applied. However, in case of each parent has different allele value, a random crossover then will be chosen to be applied.

**Globally:** adaptive crossover operator can be globally applied by looking at the entire population of current generation. That will be done through a statistical calculating the average of ones and zeros in all individuals. In case of population
members have more zeros then the two-point crossover operator will be used on the entire population. While in the contrary, the uniform crossover operator will be used for the entire population.

In fact, all genetic operators including crossover and mutation have the capability to control these generated members in global and local manner [228].

A.5.6 Guided Crossover

Guided Crossover (GC) [197] is a different form of the normal crossover with some advantages. The guided crossover is an enhanced type of operator used in genetic algorithms to replace some standard crossover and mutation operation in the direction of improving and guiding the convergence headed to the end of optimization. It is generally designed to enhance the result's value so that the genetic algorithm has a better chance to become closer to the global optima. The guided crossover as originally defined by Rasheed [197] can be described as follow:

1. Select two individuals A, B as parents from the population.
2. The first member A is selected from current population via the normal selection schema.
3. The second member B is selected from current population using the following way: for each individual I in the current population rather than A, the quantity function Mutual_fitness(A, I) is calculated such that:

\[
Mutual\_Fitness(A, I) = \frac{(fitness(A) - fitness(I))^2}{Euclidean\_distance(A, I)^2}
\]

Where B will be considered as follow:

\[
B = \max(Mutual\_Fitness(A, I_i)) : i \in [0, \mu]
\]

4. After selecting two parents, the nominated members A and B may be swapped in order to make A with the highest fitness among A and B.
5. The guided crossover then will randomly generate the offspring C taken from the line joining the two candidates from the small region closest the highest fitness parent A such that:

\[ C = L \ast A + (1 - L) \ast B \]

where \( L \) is a uniformly distributed random number taken in the range of \([1 - 0.2x, 1 + x]\) and \( x \) is a function of the number of elapsed iterations \( I_e \) and the total allowed number of iterations \( I_t \) such that:

\[ x = 0.75 \frac{(I_t - I_e)}{I_e} + 0.25 \]

However, this crossover operator can simply be described as it observes all points in searching space that can be generated by joining the randomly selected one parent with all others in the current population. For more details about the guided crossover operator please see [197] and [136].

\textbf{A.5.7 Gene Pool Crossover}

The gene-pool crossover is alternative type of recombination operator was examined in EA literature. Mühlenbein and Voigt [178] introduced a new recombination operator and named it Gene-Pool crossover (GPC). In this formulation, the gene value of the single offspring for a particular location is formed from the allele \((\text{multinomial})\) distribution of the whole population at the same location rather than from two selected parents.

In fact, there are some extra studied recombination operators in evolutionary algorithm literature can be found such that " Ordered Crossover (OC) [93], Partially-Mapped Crossover (PMC) [263], Uniform Partially-Mapped Crossover (UPMC) [43], Cycle Crossover (CC) [183] and Non-Wrapping Ordered Crossover (NWOX) [92].
A.5.8 Fuzzy connective-based crossover operators

This crossover allows establishing different exploration and exploitation degrees and Offspring Selecting Method (OSM) inducing different diversity levels in the population. In order to do so, it needs to develop the following steps.

1. Define genes combination functions.
2. Use these functions to define crossover operators between two chromosomes.
3. Apply the crossover operators to the individuals in the population, establishing the number and type of operators along with the selection mechanism to be used.

**Genes Combination Functions:**

Consider $c_i^1, c_i^2 \in [a_i, b_i]$ two genes to be combined and $\alpha_i = \min\{c_i^1, c_i^2\}$ and $\beta_i = \max\{c_i^1, c_i^2\}$. The action interval $[a_i, b_i]$ of these genes can be divided into three intervals $[a_i, \alpha_i], [\alpha_i, \beta_i]$ and $[\beta_i, b_i]$. These intervals bound three regions to which the resultant genes of some combination of the former may belong. Moreover, considering a region $[\alpha_i', \beta_i']$ where $\alpha_i' \leq \alpha_i$ and $\beta_i' \geq \beta_i$ would seem reasonable. Figure A-6 illustrates it graphically as follow:

![Figure A-6: Action interval for a gene](image)

The intervals described above could be classified as exploration or exploitation zones. The interval with both genes being the extreme is an exploitation zone, the two intervals that remain on both sides are exploration zones and the region with extremes $\alpha_i'$ and $\beta_i'$ could be considered as a relaxed exploitation zone. The following table shows these considerations which are related to Figure A-6.
Table A-1: Interval Properties

<table>
<thead>
<tr>
<th>Interval</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>[a, a_i]</td>
<td>exploration</td>
</tr>
<tr>
<td>[a_i, b_i]</td>
<td>exploitation</td>
</tr>
<tr>
<td>[b_i, b]</td>
<td>exploration</td>
</tr>
<tr>
<td>[a'_i, b'_i]</td>
<td>relaxed exploitation</td>
</tr>
</tbody>
</table>

With regards to these intervals, it proposes four functions: F, S, M and L defined from \([a, b] \times [a, b]\) in \([a, b], a, b \in \mathbb{R}\), and which fulfil the following properties:

P1. \(\forall c, c' \in [a, b] \text{ then } F(c, c') \leq \min\{c, c'\}\)

P2. \(\forall c, c' \in [a, b] \text{ then } S(c, c') \geq \max\{c, c'\}\)

P3. \(\forall c, c' \in [a, b] \text{ then } \min\{c, c'\} \leq M(c, c') \leq \max\{c, c'\}\)

P4. \(\forall c, c' \in [a, b] \text{ then } F(c, c') \leq L(c, c') \leq S(c, c')\)

P5. F, S, M, and L are monotone and non-decreasing.

Each one of these functions allows combining two genes giving results belonging to each one of the aforementioned intervals. This is shown in the following Figure. Where \(a'_i = F(a_i, b_i)\) and \(b'_i = S(a_i, b_i)\)

```
L
     F   M   S
 a_i  \(a'_i\)  \(a_i\)  \(b_i\)  \(b'_i\)  \(b_i\)
```

Figure A-7: Genes generated using F, S, M and L

These functions will have different exploration or exploitation properties depending on their range. Although the only zone that seems natural to be considered for obtaining offspring is the exploitation zone, the other two zones may be shown to be
appropriate for introducing diversity in the population. The four functions presented allow all these zones to be covered.

**F, S, M and L Functions Using Fuzzy Connectives**

Using t-norms, t-conorms, averaging functions and generalized compensation operators; by associating a t-norm to \( F \), a t-conorm to \( S \), an averaging operator to \( M \) and a generalized compensation operator to \( L \). In order to do so, it needs to transform the genes, that will be combined, wherefrom the interval \([a, b]\) into \([0, 1]\) and later, the results into \([a, b]\). Complying with a set of fuzzy connectives, \( \{T, G, P, Q\} \) set of functions \( \{F, S, M, L\} \), associated with it, is built as described below:

If \( c, c' \in [a, b] \) then

\[
F(c, c') = a + (b-a)T(s, s')
\]

\[
S(c, c') = a + (b-a)G(s, s')
\]

\[
M(c, c') = a + (b-a)P(s, s')
\]

\[
L(c, c') = a + (b-a)Q(s, s')
\]

Where \( s = \frac{c-a}{b-a} \) \quad \text{and} \quad s' = \frac{c'-a}{b-a}

These operators have the properties of being continuous and non-decreasing, and satisfy the respective properties (P1 to P5). Table A-2 details families of fuzzy connectives as follow:

**Table A-2: Families of Fuzzy Connectives**

<table>
<thead>
<tr>
<th>Family</th>
<th>t-norms</th>
<th>t-conorms</th>
<th>Avg. functions</th>
<th>Gen. compensation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>( T_1(x,y) = \min{x,y} )</td>
<td>( G_1(x,y) = \max{x,y} )</td>
<td>( P_1(x,y) = (1-\lambda)x + \lambda y )</td>
<td>( Q = T_1^{1-\lambda} \cdot G_1^\lambda )</td>
</tr>
<tr>
<td>Hamacher</td>
<td>( T_2(x,y) = \frac{xy}{1+(1-x)(1-y)} )</td>
<td>( G_2(x,y) = \frac{x+y-2xy}{1-xy} )</td>
<td>( P_2(x,y) = \frac{1}{y-\lambda x+\lambda y+1} )</td>
<td>( Q_2(x,y) = P_2(T_2,G_2) )</td>
</tr>
<tr>
<td>Algebraic</td>
<td>( T_3(x,y) = xy )</td>
<td>( G_3(x,y) = x + y - xy )</td>
<td>( P_3(x,y) = x^{1-\lambda}y^\lambda )</td>
<td>( Q_3(x,y) = P_3(T_3,G_3) )</td>
</tr>
<tr>
<td></td>
<td>Einstein</td>
<td>$T(x,y) = \frac{xy}{1+(1-x)(1-y)}$</td>
<td>$G_4(x,y) = \frac{x+y}{1+xy}$</td>
<td>$P_4(x,y) = \frac{2}{1+(\frac{2-x-1}{x})^{1/y}}$</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>----------------------------------</td>
<td>-----------------------------</td>
<td>-----------------------------</td>
</tr>
</tbody>
</table>

**FCB-crossovers: F-crossover, S-crossover, M-crossover and L-crossover:**

Let us assume that $Q \in \{F, S, M, L\}$, $C_1 = (c_1^1, ..., c_n^1)$ and $C_2 = (c_1^2, ..., c_n^2)$ are two chromosomes that have been selected to apply the FCB crossover operator to them. We can generate the chromosome $H = (h_1, ..., h_n)$ as:

$$h_i = Q(c_i^1, c_i^2), \quad i = 1, \ldots, n$$

(6)

This operator applies the same F, S, M or L function for all the genes in the chromosomes to crossover. For this reason, they will be called F-crossover, S-crossover, M-crossover and L-crossover when the F, S, M and L functions are applied respectively. It should be emphasized how these crossover operators have different properties: the F-crossover and S-crossover operators show exploration, the M-crossover operators show exploitation and the L-crossover operator shows relaxed exploitation.

Using the families of fuzzy connectives in the Table A-3 we can build four families of crossover operators. Each one of them shall be called the same as the related fuzzy connective family:

**Table A-3: Set of crossover operators**

<table>
<thead>
<tr>
<th>Interval</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_1$, $S_1$, $M_1$, $L_1$</td>
<td>Logical</td>
</tr>
<tr>
<td>$F_2$, $S_2$, $M_2$, $L_2$</td>
<td>Hamacher</td>
</tr>
<tr>
<td>$F_3$, $S_3$, $M_3$, $L_3$</td>
<td>Algebraic</td>
</tr>
<tr>
<td>$F_4$, $S_4$, $M_4$, $L_4$</td>
<td>Einstein</td>
</tr>
</tbody>
</table>

174
In fact, the implementation of genetic algorithm more optimum crossovers can be found such as leading crossover [249], order crossover [158], annular crossover [186], DPS [240] and DSO crossover [72].

A.6 Crossover vs. Mutation

In 1990, Fogel and his college Atmar [76] studied EA and used it in solving problems of linear equations. They reached a thought that crossover operator can be considered as a small value in importance, while mutation operator alone has a big chance in finding good solutions. Latter in 1993, Eshelman and Schafer [71] by the use of schema processing methods disagreed this idea, that crossover has a possibility to be valuable and effective to solve some problems.

A.6.1 Mutation Only EAs Challenge

The following discussion will underline some researches on fitness functions which can be recognised hard to optimize with mutation only, while by adding crossover operator will increase the optimization performance towards finding solutions. The first problem Royal Road (RR) were projected by Mitchell et al. [170], a well-studied problem as set of fitness functions. This set was designed to express that the crossover scheme is very critical in genetic algorithms GA to find optimized solution. Latter in 1994, Jones [137] proposed a full definition of the Royal Roads problem with its elements.

Mitchell with his co-researchers mentioned that for a typical GA without taking hitchhiking effects, the optimization complexity time is expected as \( O(2^k \log \left( \frac{n}{k} \right)) \). Forrest and Mitchell [79, 80], throughout their experimental studies show, somewhat surprisingly, that some random mutation only hill-climber algorithms performance are overcome other GAs with crossover. This result encouraged them to propose an open problem in [171]; "Finding a set of function optimization and verify that the genetic algorithms GAs are initially outperform the evolutionary algorithms EAs without using crossover operator".

Prügel-Bennett [195] tied to a good answer to this theory by proposing a fitness gradient with a series of barriers. His study showed with asymptotic approximations and
experiments that a hill-climber performance is a slightly better by using GAs with no applying of crossover, while it much better when using full GA with crossover operator. The expected optimization time found by Jansen and Wegener [134] of the well-studied problem (1+1) EA on the classic Royal Road function is $O \left( 2^k \left( \frac{n}{k} \right) \log \left( \frac{n}{k} \right) \right)$ where $n$ is the individual size and $k$ is the sub elements string size, with mutation rate equal to $1/n$.

Deb and Goldberg research [59] proposed the concatenated trap functions. This type of fitness function formed from a series of minor fully deceptive functions. The proposed fitness functions make all members in the search space, excluding the optima ones, showing local guidance to make process follows a wrong direction in search space towards the optimal solutions. The authors used a function build up from a number of connected non-overlapping deceptive functions.

However, in reasonable time GAs with mutation only ware unable to find optimized solution of the compound functions, while using the crossover operator allows generating sequences with short-order forming high fitness individuals and used these sequences combined to successfully lead process to find optimize solution of the function. This building block scheme [59] is historically fundamental of claiming crossover with highly performances on some fitness functions. Note in case of the trap size is fixed or equal to $n^\epsilon$ where $\epsilon \in (0,1)$ then it has been reached a suggestion that the mentioned conclusions are inapplicable.

Although the Royal Road and the concatenated trap functions were both proposed to show and prove the crossover operator is necessary for GAs, the successive proof of theory has failed for the Royal Road functions resulted in experiments used to explain that the RR problems were separable and non-deceptive. These particular issues endorsed the use of EAs with non-crossover and RMHCs to locate optimized solution of RR. Focusing on this concern, Watson and Pollack [250, 251] generated a fitness function with hierarchical scheme and named it HIFF (hierarchical if and only if). The authors build this function in way making all sub-blocks are interrelated, non-separable and being deceptive to the mutation process. EAs with mutation only most likely need exponential time to find an optimized solution for HIFF. Later Dietzfelbinger et al. [61] asymptotically analyzed a recombinative hill-climber on the HIFF function and demonstrated an expected complexity time is $O(n \log(n))$. 
Jansen and Wegener study [133] proposed JUMPrn a fitness function utilizing a mutation with special structure. The main attribute is to make the fitness canyon sorting out the global optimal solutions from the fake optimal ones using m-bit gap of low fitness (m < n). The steady state GA (employing uniform crossover) complexity time to locate optimized solution of JUMPrn fitness function is expected as O(n² log(n)). However, to achieve a good optimization, a hill-climber steady state GA which allows no fitness decreases, such that (μ+1)-EA, is required to apply at same time m bits mutilation to pass the canyon. The expected complexity time to for such process to find optimized solution is estimated as O(nᵐ), which lead to a clear difference in the time of optimization.

In [134] Jansen and Wegener in their study proposed that the (1+1)-EA which is well known evolutionary algorithm, can successfully optimize the classic Royal Road problem with expected optimization time \( O \left( 2^k \left( \frac{n}{k} \right) \log \left( \frac{n}{k} \right) \right) \), where n represents the string size, k represents the sub-elements string size with mutation rate equal to 1/n. This contrasts with the order n/k speed up of the optimized GA's expected time of \( O \left( 2^k \log \left( \frac{n}{k} \right) \right) \). As been previously mentioned, experimental results also showed that different random mutation hill climber algorithms (RMHCs) along with the (1+1) EA, can perform better than the normal genetic algorithms GAs.

Furthermore, in [134] Jansen and Wegener proposed the Real Royal Road (RRR) fitness functions by utilizing two types of crossover operator which are one-point crossover as well as uniform crossover. In the work of RRR, two methods were analysed the steady state GA along with equivalent mutation only EA, in intention to find accurate bounds of expected complexity time. The authors' results showed that the steady state GA successfully optimized the Real Royal Road functions in polynomial complexity time, whereas the mutation only EAs take to find optimum solution exponential expected time. Their work employed a various population size \( S_n \) which gives a chance to be different at a rate rather than the constant individual length n.

Other researchers latter, Storch and Wegener [233], stated a new form of Real Royal Roads fitness function using both one-point and uniform crossover but for fixed population size. The population has been used of size two that in order to allow a crossover operator performs better with small population. They reach a result that expected complexity time to optimize of the steady state (2 + 1) GA was polynomial time, whereas the (2 + 1) EA expected optimization time was exponential.
The Real Royal Road functions which are formed from a set of artificial functions are fall in a valid critique as they were constructed to work on a narrow challenge. More researches have been made on other natural fitness functions. Fischer and Wegener [73] stated that an advanced GA can perform much better for one-point and uniform crossover than the EA. However, the experimental results are mixed for a standard GA. If \( \lambda \) is carefully chosen, the \((1 + \lambda)\) EA can perform better than the typical GAs. In [234], Sudholt practical results on the Ising problem shows for GA the expected complexity time was polynomial while it expected exponential time for the EA.

Horn et al. [125] suggested a Hamming Path with one-bit mutation operator, which latter named as the Long Path. Horn et al. [125] and latter Rudolph [208] experimental results showed that the expected time to optimize the well known hill-climbing genetic algorithms which searching a one-bit neighbourhood using the long path of length \( n \) is exponential. Afterwards, Rudolph confirmed that the expected waiting time of \((1 + 1)\) EA which changes multiple bits values in one search step to optimize long paths is \( O(n^3) \) time. In [85] Garnier and Kallel statistical study showed that the first hitting of Rudolph's method expected times are differ in the long path of size \( k \). The expected waiting time of simple non-crossover \((1 + 1)\) EA for any chosen \( k \) that is various relative to the size of bit string \( n \) is exponential.

In [206], Rosenblueth with his co-researcher Stephens have explained an analysis scheme to help in justifying the effectiveness of recombination operator for a certain fitness function. This scheme mainly depends on two types of methods the Walsh basis [248] methods and the building block basis (BBB) methods of Stephen [231, 232]. The authors in [206] have tested a pair of functions, the first one is counting-ones (CO) with no epistasis while the second one is the needle-in-a-haystack (NEEDLE) with maximal epistasis. Their work explained that the crossover operator can help to optimize the CO function, while it was not for the NEEDLE function. In fact, any comparison work in terms of genotype distribution will reach the result that it very difficult to notice benefit or harm of the crossover operator. However, the distinction illuminates clearly when using the methodology of building block basis or the Walsh one. Latter, Burjorjee [33] proposed a new type of problems (the biological kind!) which can be significant to computational genetics. He explained the genetic algorithm efficiency on these problems and demonstrated an alternative view of describing the GAs performance and efficiency on them.
A.6.2 Crossover Only EAs Challenge

The problems which are hard to be optimized using crossover operator may be optimized with mutation only algorithms. However they are not widely common in literature of evolutionary and genetic algorithms, they are not totally new. In [193] the authors designed a new fitness function called OneMix (see figure 2.5-1) and they experimentally proved that it is deceptive to crossover operator. The function landscape has a very special saw-tooth pattern accompanied with a distinctive level of frequency oscillation in its shape making it alternately moves from high to low fitness, on the contrary, on first side of the unitation landscape to reach the global optimum. While the second side of the searching space involves a smooth hill climbing towards a local optimum. The saw-tooth side is similar to a long 2 path function and needs two-bits mutation flipping to switch to the low fitness side on the landscape. Following is the formal definition of OneMix function:

\[
OneMix(x) = \begin{cases} 
(1 + a) \left( \frac{l}{2} - x \right) + \frac{l}{2} & \text{if } x \text{ is even and } x < \frac{l}{2} \\
 x & \text{otherwise}
\end{cases}
\]

Poli et al. in [193] proposed a new GAs theory, the low-pass filter, and discussed that the crossover only has no chance to distinguish the fitness changing with high frequency. Consequently they defined that the crossover operator over a large number of populations works like a low-pass filter. The low-pass filter has the capability to distinguish the diversities between low fitness area and high fitness one of the Poli et al. pattern, therefore GA deception through distinguishing a slope with a extra lower gradient than the gradient formed just from the high-fitness long 2 path.

The authors studied the infinite population model function and evaluate this model through run-time tests. The results showed that the effects of low-pass filter for all genetic operators are also existed with crossover dominating of most effects. In fact, Poli with his co-researchers in [193] did no attempt to use any run-time analysis methods to show any clear boundaries between the GA and EA on the OneMix fitness function.

The author in [203] state that the OneMix results detailed in Poli et al. [193] have a great chance to be truly proven by means of run-time analysis method. Moreover he considers that the fitness space of the saw-tooth pattern will not successfully deceives a
GA with small population size, for instance the (2+1) GA. Additionally he clarifies that depending Poli et al. believes that if a trivial diversity methods were added with guaranteeing that all population's members in the fitness landscape have a good distance between each other, then a small population GA will optimize the OneMix fitness function in polynomial expected time with an exponentially low failure rate.

![Figure A-8: Poli et al. OneMix fitness function with low pass filter, b=100, a=0.6](image)

**A.7 Operators & Attributes Control**

A main key issue for researchers is how to tune and control a range of EA's parameters and operators. Some basic ways suggest to previously setting all parameters, while the majority of researchers prefer to carry out a set of runs to statically attempt to setup good settings of the tuned parameters. A modern way more willingly than those have projected and employed adaptive scheme or parameter-less EAs.

A simple and common example of using such a mechanism is mutation adaptation. A basic technique is to adjust the mutation rate regarding to a deterministic schedule. A further group of techniques applying adaptation on more than one parameter a time such the process mutation, population size, and crossover rates by mining algorithm progress or heuristic of quality. The studies are widely existed and a reasonable set of schemes are assessed [65, 66, 226]. However, some practitioners
employing a different technique using a fuzzy logic rule systems, this technique is also reviewed via Richter and Peak study in [204].

One more common studied method in literature is diversity control of the population size. These schemes maintained through two attributes, explicit and implicit. These methods try to utilize reproduction wisdom along with genetic knowledge on the meaning of healthy populations with importance of genetic diversity. The Implicit methods covering some schemes like incest prohibition [45] and some other prohibited mating techniques.

A further formulation in literature is to employ sub-populations to mate evolution process within an isolated population with controlled migration scheme of population members. This will work towards making population diversity included through delaying any probable population dominance via a set of good individuals and allow using a semi-independent evolution direction. The cellular genetic algorithms by Whitley [257] can be projected as a typical example of combining a restrictive mating method together with sub-populations. In such case, all members are distributed on a wide net of virtual points isolated from the search space. These members will only be able to mate among neighbouring solutions. This process will replace one parent in the net by new offspring solutions. Once more, desired target is to retard the highly fitted members spreading and as a result implicitly guaranteeing diversity.

The diversity manipulating schemes aims straight to manage population diversity in active way. Basic mechanisms contain phenotype and genotype duplicate inhibition. Genotype duplicate inhibition is applying a rule on population preventing it from holding any multiple copy of any individual. This restriction is very close to some kinds of bit-string of the genomes similarity behaviour. From other side, the phenotype behaviour is very similar to restriction used through a similarity metric across the fitness function value or any other solution member.

However, fitness sharing and deterministic crowding [159] can be consider more complex than diversity manipulating scheme. The author stated that, a genotype similarity measuring function is utilized to assign the reproductive fitness of new individuals by reducing individual fitness according to similarity assessment with other members. The philosophy behind this scheme is to reduce the chance of individuals being recognised genetically similar of being selected for reproduction, whilst increase it for distinct member of similar fitness. The deterministic crowding scheme as described in [159] suggested a kind of competition (the fitter win) between parents and
their offspring. If the offspring if fitter than its parent, it will replaces that parent in next population.

However, the above summarized successful measuring schemes may be poor in measuring a-priori for a certain fitness landscape. They may be generally distinct in such landscapes which manipulating several dispersed sub-optima or others suffering from various 'barriers and basins' straight the way though out global optima. While some are intuitive to a certain extent which make no possibility to firmly say that they must be used all the time. Yet, as a matter of course, a various forms of algorithm's improvements and developments the researchers should attempt to always work towards enhance the quality of solution. From other side, they must be very careful to make a big chunk of developing or modifications of any EA key preferences, where the changing of problem nature being solved may conquer the assumptions applied by the improvements and optional operators.

In fact, several studies have been done trying to find appropriate selection of EA operators and their associated parameters which will process across a variety type of problems. DeJong [52] introduce the early key study which characterized a collection of test functions and projected a group of parameters which it was wished in order to work well over a wide range of problem types. Though, further researches like Grefenstette [102] used a “meta-ga” in trying to identify a typical values or using comprehensive testing such as Schaffer et al. [212] found dissimilar conclusions. Meanwhile, the theoretical analysis of Goldberg [94] work on finding the optimal population sizes established to formalise the clear point of conclusion that the value of population sizes could be dependably determined relies on the size of the used search space.

However, letter studies introduced a collection of new operators proposed, some of them, like Uniform Crossover by Syswerda [238], used the Schema Theory revaluation and led to the concentrating on two considerable concepts. The first one is Crossover Bias introduced by Eshelman [70], which illustrates the variety ways in that the p.d.f. can arise from different crossover operators upholding hyper planes of high-level of estimated fitness throughout recombination processes, as a determination of their order and length. This reflects the performance and usability of the p.d.f. on the landscape caused by the recombination operator which’s generated by the problem encoding. DeJong and Spears [54, 55, 230] researches help to approve these findings by
using extra proper analysis on the relative properties of different recombination mechanisms.

The second concept, the Safety Ratios (ratio of probability), was introduced by Schaffer and Eshelman [211] in which that the use of reproductive operators produce a new point that would be healthier than its parent(s) to the probability that it is worse. Practically, the Safety Ratios were shown to be changed for the different reproductive operators, and changed over time as well. Once more these can demonstrate harmonise between the p.d.f.s caused by reproductive operators on the current population to the searching landscape’s fitness outlines.

This can be illustrated by employing a simple case like the OneMax function as a thought experiment:

\[ f(a) = \sum_{i=0}^{1} a_i \]

In the first initial step and after population has randomly generated, a proportion of fifty percent of the genes will be with the non-optimal fitness value, and so on almost fifty percent of all mutations will be doing well. Though, during the algorithm search progresses, the population fitness average becomes better and the amount of genes with the optimal fitness value will be increased, and that will make the chance of a randomly distributed mutation be beneficially decreased. Bäck work [16, 17] validated this issue, in where for a mutation probability \( P_m \), a correct form of the Safety Ratio was produced. Unfortunately, this issue could not be solved analytically, but it was optimised numerically and noticed to be fitted by a curve of the following form:

\[ P_{opt} \approx \frac{1}{2 (f(a) - 1) - 1} \]

This can be obviously seen as a time-dependant process with the existence of any fitness dependent selection pressure.

In interest to combine interactions between other Evolutionary Algorithm societies that were previously working on adaptive operators, such as Evolutionary Strategies of Rechenberg (1+1) [198], and Schwefel (\( \lambda \ mu \)) [216, 217], gave a better
awareness in the opportunity of developing algorithms which can make one or more of their parameters or operators be able to adapt in order to equate the p.d.f. caused by the algorithm to the searching landscape.

In order to formalise what was mentioned above, this describes the possibility of allowing time variation of the GA functions: M, R and U, the parameter set δ, and the variables m, and I. It is very necessary in this case to use a function or updating form to state the rule of transformation $X_t \to X_{t+1}$ where X is the algorithm’s main aspect that being adapted.

Once algorithm prepared to be adaptive, it will face an increased range of jobs, in which include searching the problem space as well as the space of all available variants of the essential algorithm. In account of the range of the possible changes and the learning algorithm nature, this searching job may be restricted, as it may takes variant forms. It may differ from the simple time dependant decrease in the value of a single parameter according to some fixed rule (e.g Fogarty [75]) to a complex path which potentially occupies any position in the hyper space defined by variations in R, M and $\delta$, and which is wholly governed by self-adaptation and the updating process (e.g. Smith and Fogarty [225])

In case of knowing the accurate nature of the searching landscape, it can be probable to track the optimal route through algorithmic process for a given problem. Unfortunately this may not normally to be the accurate situation, since practically it is essential to supply form of learning technique to steer the algorithm trajectory. Certainly it will not be a trouble free, even it aimed to be though, but the search process becomes more powerful than another process maintaining a certain group of operators and parameters.

Wolpert and Macready [258] stated that almost of all problem types, the entire non-revisiting algorithms will show identical effort. Regarding for certain set of parameter & operator, there can be some problems for which they are optimal, but of course there must be other types of problems for which the performance is so weak. The purpose of the exploiting of adaptive operators and parameters within genetic algorithms is to develop a range of algorithms which demonstrate high performance so that they will be trustable to be used as optimisers on other problems.

A.8 Metaheuristic Algorithms & EAs
A metaheuristic can be defined as a group of concepts designed to allow utilizing a set of heuristic methods to tackle a large set of complex problems. This is particularly applied for the several real-world problems that are naturally combinatorial where other optimization schemes show a failure to be either efficient or effective. In other words, metaheuristic algorithms have come to be classified as a general optimization scheme to be used on different types of deterministic combinatorial problems for optimization, taking into account the few amendments and adaptations to be applicable for a different class of problems [11, 81, 182]. The metaheuristic algorithms are mainly applicable to locate good heuristic solutions for difficult optimization problems which holding several local optima and weak inherent structure to direct the search process. The scheme approach to find good solution of such problems is done by initially obtaining one solution or a group of solutions, and subsequently working towards enhancing the search lighted by certain principle or a set of principles.

The evolutionary algorithms EAs and genetic algorithms GAs can be classified as certain type of metaheuristic optimization algorithms. The metaheuristic algorithms family contains type of algorithms such that, Simulated Annealing SA, Ant Colony algorithms ACA, Metropolis-Hastings algorithm MHA, differential evolution DE, Tabu Search TS and other invented algorithms. An excellent introduction and general reference provided by Glover and Kochenberger [89] and Luke [155] to many of the most popular metaheuristic algorithms.

A.9 Simulated Annealing and Metropolis Algorithms

In [64] Droste et al. demonstrated that the algorithm \((1 + 1)\) EA has the capability be amended to look like Metropolis algorithms and showed deferent convergence and complexity results. Following is the \((1 + 1)\) EA combined with selection scheme derived from Metropolis algorithm. The selection scheme is represented by the function \(\alpha(t) : \mathbb{N} \rightarrow [1, \infty)\). In case function \(\alpha(t)\) is chosen to be constant, the \((1 + 1)\) EA algorithm becomes exactly same to the Metropolis one. However, once this function selected to be non-constant then the \((1 + 1)\) EA algorithm becomes identical to the Simulated Annealing one. Note that the \((1 + 1)\) EA utilizes a Hamming distance in its mutation operator which flips only one bit. The algorithm:

1. Assign \(t = 1\),
2. Select individual $x \in \{0,1\}^n$ uniformly at random.
3. Generate offspring $y$ by flipping one bit of $x$ randomly.
4. Use probability $P_m = \min\{1, \alpha(t)^{(f(y)-f(x))}\}$ to assign $x = y$.
5. Increase $t$ by 1.
6. Repeat from line 2.

The study and analysis was limited to symmetric fitness functions and a Markov chain with $n + 1$ status. The symmetric functions which some time named “unitation functions” can be defined as a group of functions which deal only with the number of ones ($\|x\|_1$) on the individuals’ bits string. This representation help in reducing the different state space of the function. In [64], The function Valley is defined as an instance of symmetric fitness function where Valley($x$) $\in \{0,1\}^n \Rightarrow \mathbb{R}$ : $n$ is even, the function graph comes as a result of previous algorithm.

$$
\text{Valley}(x) = \begin{cases} 
\frac{n}{2} - \|x\|_1 & \text{if } \|x\|_1 \leq \frac{n}{2} \\
7n^2 \ln(n) - \frac{n}{2} + \|x\|_1 & \text{if } \|x\|_1 > \frac{n}{2}
\end{cases}
$$

The author in [203] has studied this function and found that the expected time of previous algorithm to optimize Valley fitness function for the first time with constant $\alpha(t)$ is given by:

$$
\Omega \left( \left( \sqrt[4]{\frac{\alpha}{4}} \right)^n + \left( \frac{1}{\alpha} + 1 \right)^n \right) = \Omega(1.179^n) \text{ where } \exists \alpha \in [1,\infty)
$$

Furthermore, he stated that with probability of $P = 1 - O(n^{-n})$ with selection schedule $\alpha(t) = 1 + \frac{1}{s(n)}$ the expected number of steps required to make the algorithms find the global optimum (maximum) of Valley function for the first time is $1 - O(n s(n))$ for any polynomial $s$ with $s(n) \geq 2en^4 \log n$. 

186
Droste with his co-researchers [64] continued in presenting another copy of the (1 + 1) EA algorithm. In this instance they used a cyclic mutation rate scheme then they success to verify that the algorithm can be able to search and locate the optimized solution for a particular function’s type in expected polynomial complexity time. In addition, they managed to demonstrate that the evolutionary algorithm with standard version may take highly polynomial time to find global optimum, and they illustrated in general the best worst-case behaviour for selected problems. Following is (1 + 1) EA algorithm using Dynamic-Cyclic Mutation operator.

1. Select individual \( x \in \{0, 1\}^n \) uniformly at random.
2. Set mutation rate \( P_m = 1/n \).
3. Generate offspring \( y \) by independently mutate each bit string of \( x \) with \( P_m \).
4. If \( f(y) \geq f(x) \) assign \( x = y \).
5. Set \( P_m = 2 P_m \), if \( P_m > 0.5 \) assign \( P_m = 0.5 \)
6. Repeat from number 3.

In [339] Jansen and Wegener study, demonstrated another vision the (1 + 1) EA algorithm with employing a dynamic mutation scheme. In order to ease the algorithm analysis, the authors introduced a reasonably hard fitness function with jumping fitness barrier and utilizing a particular Hamming path in the binary space. They proved that the expected time to optimize the standard algorithm with an optimal mutation rate is \( O(n^{2.361}) \), while the standard mutation rate of \( P_m = 1/n \) expected to be a super polynomial step. The algorithm with dynamic mutation rate requires \( O(n^2 \ln n) \) time. This study and its approach toward designing fitness functions encourages for more
researches and papers for constructing new functions to examine the changes may occur in evolutionary algorithm with crossover only and mutation only.

A further related work, Jansen and Wegener [131] examined Metropolis (MA) and simulated annealing (SA) along with the (1+1) EA algorithm. The authors demonstrated that the expected time for SA and MA to optimize a type of fitness function that affected with the number of ones in bit string representation is $O(n^2 \ln n)$ steps. In addition, they stated that the SA and EA expected time to optimize fitness functions may differ exponentially and constructed formulations on the cooling methods for different time barriers. They gave some example in their paper [131] showing that the EA and SA/MA performance is vary regarding to the outcome of the constraint that allowing to flip one bit only of string for any generation. In case the fitness function deals with more than one bit string will make the SA/MA algorithms expect an aggravation status towards making progress traverse the barrier. However, EA has ability to flip more than one bit a time and consequently traverse the barrier earlier.

In [132] Jansen and Wegener they treat the question of fitness highlands, they noted that the classic version of (1+1)EA algorithm will only deal with offspring of fitness value are better than or, in the worst case, equal to their parents. What of the slight changes requiring a strict fitness superiority? This will easily show that it may cause a fitness functions problem which named fitness plateaus. These are Hamming neighbour region that at certainty fitness values the algorithm be required to navigate to locate a new area with better fitness value. The authors considered the different direction and tack and characterized a new type of fitness functions and named SPTn. in this family of fitness function, the EA with its classic version expected an exponential time to find the global optimum, but with algorithm does not allowing neutral moves can be optimized in expected $O(n^3 \ln n)$ time. Once more, these fitness functions include path to control and guide the algorithms and involve traps which make the classic version fall in. The reached result was developed to deal with more than one path and trap among the same differentiation holding.

Wegener and Witt [253] and through using of quadratic pseudo-boolean functions, they demonstrated a new analysis of the (1+1)EA with multistart variants. These functions are polynomials of degree 2, which can be defined as a type of fitness functions which include NP-hard problems. A multistart variant is a scheme can be defined as populations of isolated members where every single member generates offspring can be only replaced with this offspring. The authors in [253] presented some
instances of this functions which give a good overview of failure and success of the two algorithms. A subclass of quadratic functions which only using positive operators, the (1+1) EA can find the global optimum of them in expected polynomial $O(n^3)$ time, while it will takes expected exponential time to optimize the functions with the negative operators. On the other hand, the multi-start variant algorithm was be able to locate the global optimum of this subclass with expected time of $\mathcal{O}(P(n)^n \ln n)$ where $P(n) = \omega(n \ln n)$. Another type of fitness function was presented as a part of quadratic group was the TRAPn function. The reached result showed that the (1+1)EA and multistart variant will require exponential time to optimize TRAPn fitness function or exponential restarts to achieve the global optimum. As a final result of the authors stated that the (1+1) EA will not require exponential time to successfully optimize if the separable quadratic function which can be defined as a function which can be formed as a summation of quadratic functions on smaller domains. The (1+1) EA algorithm requires a population with the basic manner which make some researchers consider it belongs to hill-climber algorithms than of the GA. Note that EA not including a population $p > 1$ will not be able employ any form of the crossover operator.