Table of Contents

[1. Introduction 2](#_Toc22910603)

[2. Background 4](#_Toc22910604)

[2.1 (1+1)-ES 5](#_Toc22910605)

[2.1.2 (1 + λ) - ES & (1, λ) – ES 7](#_Toc22910606)

[2.1.3 (µ/ρ, κ, λ) – ES 7](#_Toc22910607)

[2.2 (µw, λ) - CMA-ES 9](#_Toc22910608)

[2.2.2 LS-CMA-ES 15](#_Toc22910609)

[2.2.4 LR-CMA-ES 17](#_Toc22910610)

[2.2.5 IPOP-CMA-ES 18](#_Toc22910611)

[2.2.6 (µ + λ) - CMA-ES 18](#_Toc22910612)

[2.2.7 (1+1)-Cholesky-CMA-ES 19](#_Toc22910613)

[2.2.8 Active-CMA-ES 21](#_Toc22910614)

[2.2.9 (µ, λ)-CMSA-ES 22](#_Toc22910615)

[2.2.10 sep-CMA-ES 23](#_Toc22910616)

[2.2.11 (1+1)-Active-CMA-ES 24](#_Toc22910617)

[2.2.12 SPO-CMA-ES 27](#_Toc22910618)

[2.3 Common Random Numbers (CRNs) 28](#_Toc22910619)

[3. Methodology 29](#_Toc22910620)

[3.1 Motivation for DC-CMA-ES 29](#_Toc22910621)

[References 31](#_Toc22910622)

# Introduction

Covariance Matrix Adaption – Evolution Strategies (CMA-ES) is one of the state-of-the-art algorithms in the field of Evolution Strategies. It was originally designed and praised for its performance in regard to solving unimodal functions and its ability maintain that performance with increasing problem dimensionality. However, the importance of solving multimodal problems became more prevalent as real-world optimization scenarios reflect multimodal problems rather than unimodal [1]. Unimodal functions have only one local minimum to reach, while multimodal functions have many similar or distinct areas of local minima, making it more difficult for an optimization algorithm to find the global optimum. As a result, CMA-ES was modified to solve multimodal problems, with the addition of the rank-µ update and the introduction of Increasing Population - CMA-ES (IPOP-CMA-ES).

Regardless of these modifications, at the heart of CMA-ES lies a single model. This model contains a single Estimation Distribution Algorithm (EDA), from which the sampling of all solutions in the search-space originate. The algorithm being discussed in this thesis, Dualcenter-CMA-ES (DC-CMA-ES), tackles multimodal problems with the addition of a differently formulated EDA that works with the original EDA. This additional EDA is inspired by well-known techniques used by other algorithms in Evolutionary Computation, specifically elitism, age and weighted-intermediate recombination, all of which have shown to be beneficial.

In DC-CMA-ES, the population is sampled using both EDAs (also known as centers) at fixed or differing ratios. Since these two centers sample solutions from different origins in the search space, the system is less likely to have both models trapped in the many local minima of multimodal functions, resulting in higher-likelihood of increased performance over IPOP-CMA-ES. Not only does DC-CMA-ES maintain this performance in increasing dimensionality, but the performance gap between itself and IPOP-CMA-ES grows as dimensionality increases.

# Background

Before diving into the world of Evolutionary Strategies, knowledge of multivariate normal distributions and covariance matrices is extremely necessary and will be discussed beforehand to aid in better understanding of the topics ahead. A multivariate normal distribution is an n-dimensional normal distribution which can be denoted as *N(m,C)*. Here, *m* is the mean vector of length *n* (the “favorite” point in space from which sampling of new solutions takes place) and *C* is the *n x n* symmetric positive-definite covariance matrix [4]. A close up of a womans face

Description automatically generated

Figure 1 – Image taken from a CMA-ES Tutorial by (Auger & Hansen, 2013)

The first image on left of Figure 1 shows a multivariate normal distribution with mean of 0 and standard deviation as the identity matrix. The identity matrix is responsible for the circular shape around the mean, while σ is a scalar which can shrink or grow the search space. The middle image shows another multivariate normal distribution this time with a standard deviation of *D2*, where *D* is a diagonal matrix. Using this diagonal matrix to transform a sample from *N(0,I)* can result in “stretching” or “flattening” of the search space. The last image on the right of Fig. 1 uses the covariance matrix as it’s standard deviation. Transforming *N(0,I)* using the covariance matrix can also result in stretching or flattening, but also changes the direction of the search space.

## (1+1)-ES

The (1+1) – ES was the first Evolutionary Strategies algorithm for solving black-box optimization problems. This algorithm was developed in the 1960s by three students of the Technical University of Berlin, Hans-Paul Schwefel, Ingo Rechenberg and Peter Bienert [1]. This algorithm is the simplest in the field of ES. This algorithm generates one offspring from one parent, by utilizing both the multivariate normal distribution N(0,I) and a scalar σ known as the step-size for mutation (Shown in Equation 2)

Equation 1

If the new offspring’s fitness is better than the parent’s then itself becomes the new parent for the next generation, otherwise the parent remains the same. If the parent was replaced, then that would count as a successful mutation, otherwise it would count as a failure. The algorithm archives or keeps track of the number of successes and failures. This archive is used in the self-adaptation of the step-size σ. This self-adaptation method is known as the 1/5th success rule. This self-adaptation method is based on the observation that the fastest convergence to the global optimum is achieved when approximately 1/5th of all mutations are successful, i.e. when the parent is replaced with the better offspring 1/5th of the time [1]. If the current success-rate is 1/5 then the value of σ stays the same. However, if the current success-rate is less than 1/5 then σ is increased by a constant factor, c. On the contrary, if the current success-rate is greater than 1/5 then σ is decreased by a constant factor, c. Therefore, to obtain the new step-size:

Equation 2

The value recommended for c in Equation 3 is and was recommended by [5] using theoretical methods. The probability of success, ps, is measured based the 10n most recent mutations, where n is the problem dimension. The equation used to calculate the probability of success can be shown via the pseudocode of the (1+1) – ES algorithm in Figure 2.

Figure 2 – (1+1)- ES taken from [1]

A screenshot of a cell phone

Description automatically generated

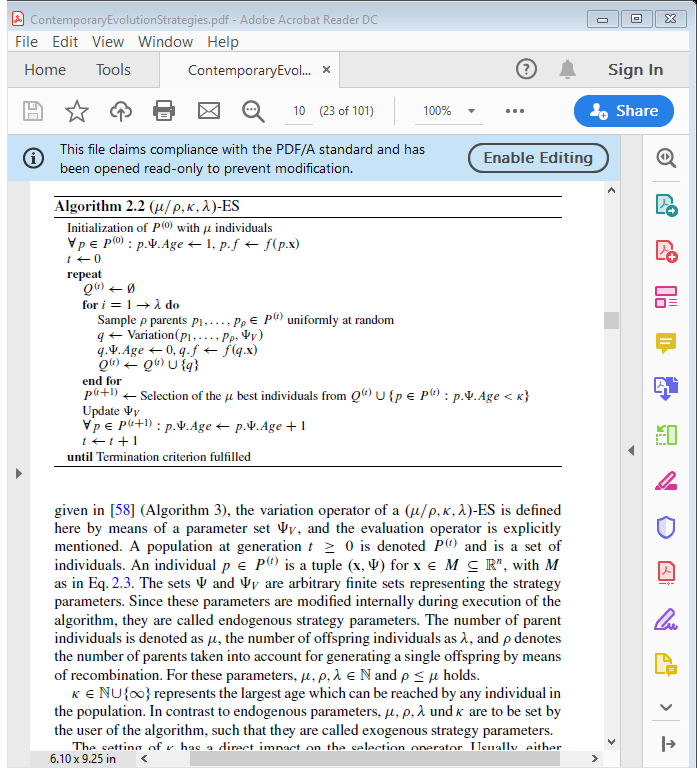
## 2.1.2 (1 + λ) - ES & (1, λ) – ES

The only difference between the (1 + λ) – ES and the (1 + 1) – ES is that instead of only generating one offspring from the parent, (1 + λ) – ES will generate λ offspring and replace the parent with the best offspring provided it is more fit than the current parent. However, in the (1, λ) – ES, λ offspring are also produced every generation, but the current parent is always replaced by the best offspring.

## 2.1.3 (µ/ρ, κ, λ) – ES

This algorithm is unique for introducing the concept that individual solutions have an ‘age’ property. The age property of a solution is incremented by 1 each time it is selected to be in the next generation.

Figure 3 – (µ/ρ, κ, λ) – ES taken from [1]



As shown in Figure 3 – (µ/ρ, κ, λ) – ES taken from [1]Figure 3, the variation operator uses a parameter set ѰV and each individual solution has a set of attributes denoted Ѱ. The population at a specific generation is denoted *P*(*t*) for . Each individual in a given population is represented as a tuple (x,Ѱ) where represents the chromosome and Ѱ are its corresponding attributes. As usual in ES, the number of parent individuals are denoted by *µ* and the population size *λ*. Additionally, *ρ* denotes the number of parents used to generate a single offspring using recombination. The maximum age that an individual solution can reach is denoted by *κ*. Since *µ*, *λ*, *ρ*, κ, are determined by the user of the algorithm they are referred to as exogenous strategy parameters. The domain of these parameters is and . The most common setting for a one generation maximum lifetime *κ* = 1 or an infinite maximum lifetime *κ* = [1]. A one generation maximum lifetime for each solution is also referred to as comma-selection and an infinite maximum lifetime is referred to as plus-selection or elitist selection. Thus, when *κ* = 1 the title of the algorithm is (µ/ρ, λ) – ES and for *κ* = , (µ / ρ + λ) – ES where [1].

## 2.2 (µw, λ) - CMA-ES

## 

Covariance Matrix Adaptation – Evolution Strategies is one of the most powerful black box optimization techniques in the field of Evolutionary Strategies. What makes CMA-ES unique is the sampling of a covariance matrix to determine the mutation distribution, and using a method called step-size cumulation to determine the scale of the mutation distribution. The covariance matrix determines the direction of the search while the step size determines “how far” the search would be in that direction [4].

However, in CMA-ES the covariance matrix *C* undergoes an eigen decomposition. This decomposes the matrix into eigenvectors (which determine the direction of the search space) and eigenvalues (which determine the scale of the eigenvectors). Therefore, the process of sampling new solutions is shown below in Equation 4 [1].

Equation 3

In Equation 4, the eigenvectors and eigenvalues BD are used to shape the spherical noise, z which is sampled from the multivariate normal distribution. ⟨x⟩ and ⟨y⟩ are updated using a method similar to an EDA, known as weighted-intermediate recombination. Weighted-intermediate recombination uses a set of µ weights w1 ≥ w2 ≥ . . . ≥ wµ with for generating the new ⟨x⟩ and new ⟨y⟩ [1]:

Equation 4

≡

Equation 5

Equation 5 shows how weighted intermediate recombination is used to determine the new point in space from which the next generation of sampling will take place. A weighted average was taken of the best µ solutions (xi’s) to create the new center. It’s important to note later that, the weighted average of the shaped noise, . In Equation 6, the best µ shaped-noise vectors (yi’s) associated with the best µ xi solutions from Equation 5, also under go weighted intermediate recombination to find the average of the best mutation steps. Having covered the concepts such as multivariate normal distributions, covariance matrices and weighted intermediate recombination, the algorithm will be discussed in more detail starting with the cumulation methods for step-size adaptation (updating σ) and updating the covariance matrix. These cumulation paths can be thought of as history vectors. They accumulate parameter information over several generations [1]. In Equation 7, the first term is the decay factor where the cumulation time parameter cσ<1 and approximately 1/ cσ [6]. This means as new parameter information accumulates, the older information is gradually weened out of the cumulation path.

Equation 6

The expression under the square root of Equation 7, is a normalization constant associated with the first term, . The variance effective selection mass µ*eff* is necessary for adapting the strategy parameters [1]. It is important to note that BD-1BT = C1/2 as this inverse matrix reverses the “shaping” of which is done in Equation 6 because ||pσ|| is then compared with the expected length of a normally distributed random vector, E(||N(0,I)||) in the step-size update in Equation 6.

Equation 7

The idea behind Equation 6 is that if the length of pσ is greater than the expected length of a normally distributed random vector then the value of σ will increase. For better understanding, consider Figure 4 below which illustrates the length of the center (mean vector) in the generation sequence. The black arrow heads represent the position of the center at a specific generation, the arrows with white arrow heads represent the distance between the original center and the best center so far, and the arrow tails represent how much the center has moved.

A screenshot of a social media post

Description automatically generated

Figure 4 - Image taken from a CMA-ES Tutorial by [4]

In the left segment of Figure 4, you will notice that center moves around a lot less than the other segments. In this case the step-size is decreased to continue a tighter search around the current center. However, in the right segment of Figure 4, the opposite happens. The step size is increased because the generation-steps are much longer and are relatively in the same direction. Therefore, in this case a bigger step-size would be more appropriate because it would be faster to make bigger steps in that direction rather than multiple small steps in the same direction.

Now we will move on to the steps of the covariance path update and the covariance matrix update.

Equation 8

⟨y⟩

Equation 9 represents the covariance path update. Once again, the first term in this equation is the decay factor where the cumulation time parameter cc<1 and approximately 1/ cc [6]. In the second term, hσ is either 1 or 0 based on Equation 10, where t is the generation number and n is the problem dimension [1].

Equation 9

Like the previous path update, the expression with the square root is a normalization constant associated with the first term, . The purpose of hσ is to choose whether to update pc based on information about the current generation t and the value of ||pc|| [1]. It’s also important to note that the length of a multivariate normally distributed random vector can be approximated by the gamma function shown in Equation 11.

Equation 10

The variable µeff is the variance effective selection mass which is defined in Equation 12.

Equation 11

After the covariance path update, the next step of the algorithm is the covariance matrix update which is shown in Equation 13.

Equation 12

The covariance matrix update is a weighted matrix addition with 3 terms. The first term is the decay factor of the current covariance matrix. The second term is known as the rank-one update, which uses the covariance path and its transpose to form another matrix. According to [4], “the rank-one update uses the evolution path and reduces the number of function evaluations to learn straight ridges from O(n2) to O(n). The third term is known as the rank-µ update, also known as the weighted empirical covariance matrix, was added to the update equation to accommodate large population sizes. In essence, the covariance matrix is updated to increases the likelihood of successful steps to occur in the next generation [4]. The following equations show how the strategy parameters from the algorithm are calculated:

cc is the parameter for the cumulation time of pc which is approximately 1/pc.

cσ is the parameter for the cumulation time of pσ which is approximately 1/pσ.

dσ is a dampening parameter that determines the change rate of the global step-size, σ during the step-size update.

c1 is the parameter that determines the change rate of the covariance matrix with respect to the rank-one update.

cµ is the parameter that determines the change rate of the covariance matrix with respect to the rank-one update.

wi:λ are the set of logarithmic weights used for weighted intermediate recombination of the sorted population to calculate the mean point. This method is also sometimes referred to as flattening.

Figure 5 – **(µw, λ) - CMA-ES Algorithm** [1]

A screenshot of a cell phone

Description automatically generated

## 2.2.2 LS-CMA-ES

The LS-CMA-ES algorithm is a CMA-ES variant that updates the covariance matrix based on an inverse Hessian matrix that is estimated via a solution to the appropriate least-squares estimation problem. To solve this least-squares estimation problem, the algorithm collects an archive of tuples (x, f(x)) where x is a candidate solution and f(x) is the fitness value of that candidate solution [1]. As a result, the least-squares estimation problem is defined by the following minimization task:

Equation 13

The result of this minimization in Equation 14 produces two estimators, for the gradient and for the Hessian matrix. The Taylor series expansion up to the quadratic term only provides an estimation of the true fitness landscape at x0­, it is of interest to obtain an error measure of the estimate to determine if should be used for the covariance matrix update. The error measure is shown in Equation 15 [1].

Equation 14

Since solving Equation 14 and computing H-1 uses numerical methods algorithms that have a time complexity of *O*(*n*6), executing this step at each generation is not ideal for performance. As a result, LS-CMA-ES has two different modes known as LS and CMA, used for performing the covariance matrix update [1].

In LS mode, Equation 15 is executed once every *nupd* generations. Additionally, if the error measure falls below a defined threshold, *Q*t, the covariance matrix until the new update after another *nupd* generations. If Q > Qt, the algorithm switches into CMA mode. In CMA mode the covariance matrix is updated using the rank-one update (the 2nd term in Equation 13). In this algorithm, sampling offspring from the parent is shown in Equation 16 below.

Equation 15

A global step size σ is used and updated via mutative step size adaptation. For example, if b is the index of the best offspring after selection, then the new global step size is updated as Finally, the covariance matrix is updated using rank-one update (the 2nd term in Equation 13). It is important to note that the evolution path pc is still updated regardless of which mode the algorithm is in so that the covariance matrix is always up-to-date before switching into, or staying in CMA mode [1]. The recommended exogenous strategy parameters are shown below:

## 2.2.4 LR-CMA-ES

LR-CMA-ES is an extension of (µw, λ) - CMA-ES that introduces local restart conditions. The idea of restarting the algorithm is to avoid wasting function evaluations when the algorithm is facing stagnation in its optimization process. This algorithm provides five restart conditions for identifying stagnation, in which a new run of regular (µw, λ) - CMA-ES will commence [1]. There are two tolerance values, T*x*= σ10-12 and T*f* = 10-12, used within the restart conditions. The first restart condition, *equalfunvalhist* is satisfied if either the best fitness values of the last generations are the same or the difference between the best fitness value and the weakest fitness values is small than T*x* [1]. The second restart condition, *TolX*, is satisfied if for a vector v = σpc , T*x* , where *i* ϵ {1,2,…,n}. The third restart condition, *noeffectaxis*, looks at the main coordinate axes formed by C. This condition is satisfied when, 0, where γi is the ith eigenvalue and ui is the ith eigenvector of C respectively, where . The fourth restart condition, *noeffectcoord*, also looks at the coordinate axis and is satisfied when . The fifth and final condition, *conditioncov*, is satisfied when the condition number of the covariance matrix exceeds 1014 [1].

## IPOP-CMA-ES

IPOP-CMA-ES is an extension of LR-CMA-ES with one small but very effective change. Should there be a restart of the (µw, λ) - CMA-ES, then the population size of the next run of (µw, λ) - CMA-ES is multiplied by a certain factor η, commonly set as η = 2 [1].

## 2.2.6 (µ + λ) - CMA-ES

(µ + λ) - CMA-ES is an extension of regular (µw, λ) - CMA-ES where the selection method is altered to perform elitism. Elitism is a well-known method used in the selection process of many Evolutionary Algorithms. What elitism does is it includes a certain number of the best selected offspring from the previous generation, in the selection processes of the current generation. In (µw, λ) - CMA-ES, we selection the best µ offspring from the population, λ. Therefore, with elitism we add the µ offspring from the previous generation to the λ offspring of the current generation, before performing a sort based on fitness and selecting the best µ offspring from the sorted µ + λ population. The main idea behind elitism in Evolutionary Algorithms is to ensure that there are always a set of the fittest solutions included in the population every generation.

## (1+1)-Cholesky-CMA-ES

The main property of this CMA-ES variant is that the covariance matrix is implicitly updated without using an eigen decomposition. As the name suggests, the algorithm uses a Cholesky decomposition instead. This method reduces the computational complexity in each generation from *O*(*n3*) to *O*(*n*2) [1]. [7] proves that the Cholesky factors A can be updated without explicit knowledge of the covariance matrix. The theorem will be discussed without the proof. However, consider this lemma which is required for the proof of the following theorem [1]. For any vector and , the following equation holds:

Equation 16

The theorem then states, let be a symmetric positive definite matrix with the Cholesky decomposition where . Additionally, let defined the update of C with . Finally, the update Cholesky factor is given below [1]:

Equation 17

An offspring is produced via the parent x with the following equation:

Equation 18

Using the previously discussed theorem, the Cholesky factor A is adapted with a constant exogenous strategy parameter denoted by ca.

Equation 19

The adaptation show in Equation 20 is executed if the value of (shown in Equation 21) is less than a threshold value, denoted *pt*.

Equation 20

The global step size adaptation in this algorithm possesses similarities with the 1/5th success rule of the (1+1)-ES shown earlier in Equation 3. Therefore, if the new offspring has a better fitness value than that of its parent, λs = 1, otherwise λs = 0. These measures of success are accumulated over generations by using the parameter *cp* as a learning rate as shown in Equation 21 [1]. Now, with the success rate the global step size is updated as follows:

Equation 21

where is the target value for the success rate [1].

The recommended exogenous strategy parameter settings are as follows:

## Active-CMA-ES

The idea behind Active CMA-ES is that the worst offspring should be considered for the adaptation of the covariance matrix by utilizing negative weights. As a result, Active-CMA-ES modifies the covariance matrix update of (µw, λ) - CMA-ES as follows [1]:

Equation 22

Equation 23

The parameter *cc* is also modified as . The parameter *β* has been tuned with empirical investigation, described in [8]. The parameter was chosen as a compromise between improving convergence velocity and ensuring the covariance matrix C is positive definite, in order to make the system more robust [1].

## 2.2.9 (µ, λ)-CMSA-ES

(µ, λ)-CMSA-ES aims to reduce the number of exogenous strategy parameters by reintroducing self-adaptation of the global step size σ. This simplifies the number of exogenous strategy parameters from five in (µw, λ)-CMA-ES down to two [1]. Offspring solutions each have their own individual step sizes which are calculated using the global step size σ as follows [1]:

The *µ* best offspring go through recombination with identical weights and is applied to the vectors and for to obtain the vectors and the new global step size σ. Therefore, the new parent is updated as . The vector is necessary for updating the covariance matrix C, using the learning rate as follows [1]:

Equation 24

The recommended settings of the exogenous strategy parameters are:

## sep-CMA-ES

This CMA-ES variant reduces the space and time complexity of the (µw, λ)-CMA-ES algorithm down to *O*(*n*) as compared to the original *O*(*n3*). The reason for this increase in performance is because simply uses a diagonal matrix D which only contains the square root of the main diagonal elements of the covariance matrix C, along its own diagonal. This means that no computationally expensive eigen decompositions need to be done. The drawback is that this matrix is unable to generate correlated mutations like a covariance matrix. Hence, the covariance matrix update is modified from (µw, λ)-CMA-ES as follows [1]:

Equation 25

The learning rate is increased because of the reduced complexity of the covariance matrix and is newly determined by:

Equation 26

All other parameters from (µw, λ)-CMA-ES remain unchanged.

## (1+1)-Active-CMA-ES

This algorithm is an extension of the (1+1)-Cholesky-CMA-ES that uses the Active-CMA-ES ideology of taking the worst offspring into account for the covariance matrix update. Rather than using the Cholesky decomposition for the covariance matrix , this algorithm uses the Cholesky factor A and its inverse A-1 [1]. Thus, the theorem discussed in 2.3.6 is modified. Let be a positive definite symmetric matrix with Cholesky decomposition . In addition, let be the update equation for C where . Let with and let be the Cholesky decomposition of the updated covariance matrix [1]. Then is updated as follows:

Equation 27

and is update as follows:

Equation 28

.

New offspring are generated the same way, . The success rate from (1+1)-Cholesky-CMA-ES is now modified with a condition as follows:

Equation 29

With this new success rate, a dampening parameter and the target success rate *pt*, the global step size is now:

Equation 30

The target success rate is set as , which is similar to the 1/5th success rule employed by the (1+1)-ES. If the fitness value of the offspring is better than that of the parent, then a positive Cholesky update is done. (1+1)-Active-CMA-ES relies on a search path s, which accumulates the successful mutation steps with a learning rate *c* [1]:

Equation 31

With the constant and vector , the positive update of A and A-1 are defined as follows:

Equation 32

Equation 33

The variables a and b in Equation 33 and Equation 34 are defined as follows:

In Active-CMA-ES, the λ - *µ* worst solutions are used in the covariance matrix update. With the strategy introduced in this algorithm, previous fitness values are stored and checked to see if the offspring is inferior to its parent. If the offspring is inferior to its parent, then the positive updates in Equation 33 and Equation 34 are done using *a* and *b*. However, the transformed search path , the vector z is used for the negative update [1]:

It is important that holds for the constant , in order to ensure that the covariance matrix is positive definite. Additionally, if the value of , the convergence of the algorithm can be come very unstable. To stop this effect, when , is given and upper bound of [1]. The recommended settings of the exogenous strategy parameters are:

## SPO-CMA-ES

This algorithm is restart-version of the (µw, λ)-CMA-ES that uses sequential parameter optimization (SPO) [9], to optimize the exogenous strategy parameters within the algorithm. SPO uses *design of experiments* (DoE) methods and *design and analysis of computer experiments* (DACE) [1]. This algorithm is concerned with the exogenous strategy parameters such as the number of offspring λ, the initial step size and the selection pressure . The first step of the algorithm is using *latin hypercube sampling* (LHS) [10] , an initial design of experiments (DoE) for the exogenous strategy parameters is created. In the second step, independent runs of (µw, λ)-CMA-ES are done using the parameter that were created in step one via the DoE plan. The best solution along with its fitness value from each independent run is stored in a set, *Y*. This first phase of the algorithm is referred to as the *exploration phase* and is followed by the *exploitation phase*. The exploitation phase is repeated until the function evaluation budget is reached. A performance measure *y* is computed for each of the results in *Y*. Based on the performance measure values the and the corresponding exogenous parameter sets from the DoE, a Kriging model *M* is trained. Subsequently, this model is used to find a new design point by running an optimization on the Kriging model and using the resulting point. This new design point *d* is then added to experimental plan *D* and the algorithm loops again [1].

## 2.3 Common Random Numbers (CRNs)

CRN is one of the most popular Variance Reduction Techniques (VRT). It is used when comparing similar systems with different configurations in hopes of investigating which system is better under the given circumstances [3]. The main idea is that two different configurations are compared while sharing the same experimental conditions, so that the results observed between the two system configurations are held with more integrity than if the two systems being compared did not share the same experimental conditions. For example, “experimental conditions” for computer simulations may be generated random variates or numbers that are used to feed both systems simultaneously as they carry out their separate operations. In other words, CRN is a VRT that attempts to induce positive correlation by using the same random numbers within simulation of both system configurations [3].

However, there is no guarantee that CRNs will always be a successful VRT. The success of CRN depends heavily on the models under comparison. The reason for this is that different models may generate very different responses to the CRNs. Therefore, it is most effective when conducting CRN simulations with similar models. On the contrary, there are some classes of models where CRN is guaranteed to be successful as a VRT [3]. These classes of models and conditions are discussed in [2].

# Methodology

## 3.1 DC-CMA-ES

## 3.1.1 Outline

In the Introduction, it was mentioned that DC-CMA-ES has a different method for computing the additional center (from now on referred to as the elite center) that is inspired by well-established techniques used in the field of EC namely, elitism, age and weighted-intermediate recombination.

Elitism is a technique where the best individual solutions from the previous generation are included with the current population to be considered in the selection process. The advantage of elitism in EC is that convergence is usually faster because good solutions are not lost unless better ones are produced.

They way elitism is used in DC-CMA-ES is inspired by the (1+1)-ES. The main idea was that the elite center should be replaced by the best solution of each generation. However, this method of updating the elite center was proven to be very unstable, especially in a multimodal environment.

This instability was solved with the introduction of a sliding window which accumulates the best solution from each generation and operates in a First-in-First-out fashion. When the window becomes full, if the new elite solution is better any of the solutions residing in the window then the eldest elite solution exits while the new elite solution enters. The functionality of the elite sliding window resembles the age property used by 2.1.3 (µ/ρ, κ, λ) – ES as the *life-span* of an elite solution strictly depends on the size of the sliding window. *Figure 6*, illustrates the functionality of an elite sliding window with a window size of 8.

functionality of the elite sliding window
Description automatically generated

Figure 6 - Functionality of the elite sliding window

The rationale behind this sliding window is that it would improve the stability of the elite center update if we average the elite solutions that currently exist in the sliding window. For this average, a conservative set of log weights were used which slightly favor the most recently added elite solutions. Performing a weighted-average of the elite solutions in the sliding window is analogous to the use of weighted-intermediate recombination. Equation 34 shows how the weights are calculated, where and n is the problem dimension. This equation produces log weights that are skewed so the first few terms are not as heavily weighted as usual. These weights take the most recent elite solutions into consideration to ensure convergence but not so much as to completely forget the eldest elite solutions in the window. The weights are ideal as they provide a steady progression of elite center movement and convergence.

Equation 34

This way of computing the elite-center leverages the history of elite solutions with the aim of obtaining an optimum position in the search space from which to sample solutions. The time period that elite solutions can stay in the window, paired with the fact that only better elite solutions can enter, ensures that the elite-center is less likely to be trapped in local minima as the contents of the window change very often and never get worse.

The job of sampling the new solutions can be split between the main center and the elite-center at any different ratio as long as they both sum to the desired population size, λ. Every solution sampled from the elite center is tied back to the main center, almost as if it were sampled from the main center originally. This allows the two centers to use the same covariance matrix for sampling because from the perspective of the covariance matrix, all of the mutation steps originated from the main center.

In the case where the sampling ratios of the main center and elite center are 1.0 and 0.0 respectively, DC-CMA-ES would be identical to IPOP-CMA-ES. It is because of this adjustable ratio that DC-CMA-ES can be thought of as a layer on top of IPOP-CMA-ES. However, the choice of these ratios is important as you must decide if one system will dominate the sampling process or evenly share it with the other.

It is imperative that you may want the currently best performing center to dominate the sampling process in order to maximize the overall convergence of the algorithm. In order to determine the best performing center, we naturally came up with two metrics. The first one is referred to as the *conditional* metric. The conditional metric checks to see if the elite center currently has a better fitness value than the main center after every generation. If this is the case, then an appropriate ratio is chosen for the next generation. For example, if the elite center is the currently the better one, you may want it to generate more samples than it was before. However, if the elite center is no longer the better one, the ratios will default to what they were originally. The drawback of the conditional metric is that it requires two extra function evaluations every generation.

The last metric is referred to as the rank-sum. As the name suggests, this metric performs a rank sum on the selected µ individual solutions to see which center is producing better solutions (the minimum rank-sum value). These µ solutions that are used to update the main center could be made up of any ratio of solutions from the two centers. Thus, the possibility exists that only solutions sampled from one of the centers are selected. Additionally, the ratio of solutions being sampled from the two centers cannot favor one center completely. This would negate the effect of the rank-sum condition because if one center generates all of the solutions, then the other center will not have a rank-sum. Having determined if the elite center produced a smaller rank-sum than the main center, the ratios would be changed appropriately. If the elite center has not produced the smaller rank-sum than the main center the default ratios would be reapplied.

The conditional and rank-sum metric allows the algorithm to determine a good opportunity for the elite center to contribute more to the sampling of solutions. They also help the algorithm to avoid generating bad solutions in the case where one of the centers may be unfit.

## 3.1.2 Pseudocode

Algorithm: *(µw, λ, rx, re)* – DC-CMA-ES

initialize

**repeat**

*\* λ*

**for**

**end**

**for**

**end**

dequeue()

**else**

enqueue()

**end**

⟨y⟩

# Experiments, Results and Discussion

Regarding experimentation, DC-CMA-ES was run simultaneously with IPOP-CMA-ES using a Variance Reduction Technique (VRT) known as Common Random Numbers (CRNs) [2] [3]. The CRN technique has been modified to handle ES algorithms, including those that perform restarts.

Table 1 – Mean of Systems with ANOVA and Tukey post-hoc correction

|  |  |  |
| --- | --- | --- |
| system | mean | group |
| dual02 | 0.85199657 | a |
| rs021505 | 0.21692952 | b |
| dual0515 | 0.16772734 | bc |
| rs0211 | 0.09733845 | bcd |
| rs0220 | 0.06048853 | bcde |
| IPOP-CMA-ES | 0.05221120 | cde |
| cond051511 | -0.05570279 | de |
| cond0211 | -0.06310690 | e |
| cond021505 | -0.23558759 | f |
| cond05151505 | -0.34789746 | fg |
| cond111505 | -0.35700565 | fg |
| rs051511 | -0.38902048 | fgh |
| rs05151505 | -0.39755906 | gh |
| dual11 | -0.46320213 | ghi |
| rs051520 | -0.48009344 | ghij |
| cond0220 | -0.48486517 | ghij |
| cond150520 | -0.52277937 | hij |
| cond051520 | -0.54270979 | hijk |
| cond1120 | -0.57850640 | ijkl |
| dual1505 | -0.62511317 | jkl |
| rs1120 | -0.68913184 | klm |
| rs111505 | -0.71257214 | lm |
| rs150520 | -0.83295515 | m |

# Bibliography

|  |  |
| --- | --- |
| [1] | T. Bäck, C. Foussette and P. Krause, Contemporary Evolution Strategies, Berlin: Springer Berlin, 2013. |
| [2] | P. Lecuyer, "Efficiency improvement and variance reduction," in *Proceedings of Winter Simulation Conference*, Montreal, 1994. |
| [3] | A. M. Law, Simulation Modeling & Analysis, 4th ed., New York: McGraw-Hill, 2007. |
| [4] | A. Auger and N. Hansen, "Tutorial CMA-ES - Evolutionary Strategies and Covariance Matrix," *Proceeding of the Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion - Gecco 13 Companion,* pp. 1-83, 2013. |
| [5] | H.-P. Schwefel, "Numberische Optimisrung von Computer-Modellen Mittles der Evolution-strategie," *Journal of Applied Mathematics and Mechanics,* vol. 60, no. 5, pp. 1-272, 1977. |
| [6] | N. Hansen and A. Ostermeier, "Completey Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation,* vol. 9, no. 2, pp. 1-195, 2001. |
| [7] | C. Igel, N. Suttorp and N. Hansen, "A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies," *Proceedings of the 8th annual conference on Genetic and Evolutionary Computation,* pp. 453-460, 2006. |
| [8] | G. A. Jastrebski, "Improving evolution strategies through active covariance matrix adaptation," *Master's thesis, Faculty of Computer Science, Dalhousie University,* 2005. |
| [9] | T. Bäck, D. B. Fogel and Z. Michalewicz, Evolutionary Computation 2: Advanced Algorithms and Operators, New York: Taylor & Francis, 2000. |
| [10] | B. Tang, "Orthogonal array-based latin hypercubes.," *J. Am. Stat. Assoc.,* vol. 88, no. 424, pp. 1392-1397, 1993. |
| [11] | H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: a comprehensive introduction," *Natural Computing,* vol. 1, no. 1, pp. 3-52, 2002. |