Investigation of step-size adaptation methods for CMA-ES based on population midpoint fitness

Jarosław Arabas and Eryk Warchulski

Warsaw University of Technology, Institute of Computer Science, Poland
jarabas@elka.pw.edu.pl, ewarchul@mion.elka.pw.edu.pl

Abstract. In this paper, we introduce and investigate three different new methods to control step size multiplier for the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The methods are intended to substitute the Cumulative Step Adaptation rule (CSA) that is a standard method to control step-size in CMA-ES. All investigated methods are based on the comparison of fitness values between the population midpoint and the offspring. All methods reveal linear convergence for the quadratic fitness function.
In the first two methods, σ is increased when the success ratio exceeds 1/5 and is decreased when the success ratio falls below 1/5. The success ratio in each iteration is defined as a fraction of the number of points generated in that iteration whose fitness value exceeds the fitness of the weighted mean (for the first method) or the simple arithmetic mean (for the second method) of points from the previous iteration. In the third approach, σ is increased when the fitness of the current iteration midpoint is located below a certain percentile of the fitness values in that population.
According to the results of tests with the use of CEC'2013 and CEC'2017 benchmark sets, two introduced methods based on the population midpoint fitness analysis yielded very good results. The computational overhead was significantly smaller than in the case of CSA.

Keywords: CMA-ES, 1/5 success rate rule, CSA

1 Introduction

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [**?**] has been among the leading methods in black-box global optimization competitions. In each iteration, the method generates a population of random points using a multivariate normal distribution whose parameters — the expectation vector and the covariance matrix — are maintained by the method. In each iteration, the parameters are modified by analyzing the distribution of fitness values of generated points, according to the adaptation formula called CMA (Covariance Matrix Adaptation). In effect, the probability density function of the distribution to generate points tends to roughly approximate the fitness function shape in the area covered by the populations.

Dynamics of the probability distribution adaptation by CMA is improved by introducing the third adaptive parameter — the step-size multiplier — that is used to multiply the random values generated using the aforementioned normal distribution. The canonical version of CMA-ES adapts the step-size multiplier by comparing the actual trajectory of the expectation vector with a hypothetical trajectory that would be observed when the fitness would be a constant function. This adaptation mechanism is called the Cumulative Step Adaptation (CSA).

CMA-ES has proved efficient in global optimization in real-space. It has been the basic algorithm for a family of methods that won black-box optimization competitions that have been regularly organized at evolutionary computation conferences like GECCO or CEC. Yet the very impressive effectiveness of CMA-ES is occupied by a considerable computational effort that is needed to perform matrix operations. Therefore researchers present various simplifications of CMA-ES, usually at the expense of the algorithm efficiency. Researchers often change the CMA procedure, usually by reducing the accuracy of the matrix update. Some representative papers on that issue are [**?**,**?**,**?**]. Much fewer papers are devoted to the CSA rule.

In this paper, we focus on the procedure to control the step-size. The research was motivated by the results reported in [**?**,**?**]. It was shown there that the efficiency of CMA-ES could be improved by computing fitness of the population midpoint. Here we demonstrate that the information about midpoint fitness can facilitate the step-size adaptation rule. We introduce and examine three example methods to control the step-size that use the information about midpoint fitness.

The first method takes into account the expectation vector of the distribution used to generate points. If its fitness is smaller than 20% of generated points then the steps-size is decreased, otherwise, it is increased. The second method works similarly, but, instead of the expectation vector, it uses the arithmetic mean of all points generated in the previous iteration. The third method looks at the arithmetic mean of points from the current iteration. It increases or decreases the step-size multiplier value when the fitness of the current population mean is smaller or greater than a certain fraction of the current population.

The paper is composed as follows. Section 1 briefly introduces the research problem. In Section 2, we outline the considered version of CMA-ES. We discuss the standard CSA rule to control the value of the step-size multiplier. We present two alternative rules of step-size control taken from the literature. Section 3 introduces three methods to control the step-size which are based on a comparison of fitness values of points generated by CMA-ES and their midpoint. Section 4 presents the results of simulations that were obtained by CMA-ES coupled with various methods to control step-size, for linear and quadratic fitness functions. We discuss the convergence rate for the compared methods as well as the computational overhead that is introduced by the use of step-size control procedures. In Section 5 we present the results of testing CMA-ES, which was coupled with different step-size control rules. We use two benchmark sets that were introduced for CEC'2013 and CEC'2017 black-box optimization competition. Section 6 concludes the paper.

2 Step-size adaptation methods for CMA-ES

Algorithm 1 presents an outline of the considered version of CMA-ES. The method maintains three parameters: the expectation vector $\mathbf{m}^t$, the covariance matrix $\mathbf{C}^t$ and the step-size multiplier σ^t, which are the parameters of the multinomial normal distribution used to generate points; the upper index t stands for the iteration number. In each iteration the algorithm uses the multinomial normal distribution with zero expectation and the covariance matrix $\mathbf{C}^t$ to generate the set of vectors $\{\mathbf{d}_1^t, ..., \mathbf{d}_\lambda^t\}$ (line 6).Then the populations of vectors $\mathbf{d}_i^t$ is sorted according to their fitness values (line 9) such that it holds:

$$q(\mathbf{m}^t + \sigma^t \mathbf{d}_i^t) \leq q(\mathbf{m}^t + \sigma^t \mathbf{d}_{i+1}^t)$$

where $q : \mathbb{R}^n \to R$ stands for the fitness function. For brevity, we shall denote $\mathbf{x}_i^t = \mathbf{m}^t + \sigma^t \mathbf{d}_i^t$.

Out of λ generated vectors, a subset of μ elements with the lowest fitness value is selected to perform an adaptation of the algorithm parameters. The expectation vector $\mathbf{m}^t$ is modified by adding the weighted mean of selected vectors $\mathbf{\Delta}^t$ (lines 10,11).

The covariance matrix $\mathbf{C}^t$ is updated by formula (line 13) that takes into account two summands. The first summand is a rank-μ matrix $\mathbf{C}_\mu^t$ which is a weighted sum of the outer products of μ selected vectors $\mathbf{d}_i^t$. The second summand is a rank-1 matrix $\mathbf{C}_1^t$ being the outer product of the vector $\mathbf{p}_c^t$. This vector accumulates values of $\mathbf{\Delta}^t$ from previous iterations (line 12).

The step-size multiplier σ^t is updated using the Cumulative Step Adaptation rule which is discussed in more detail in the next subsection.

2.1 Cumulative Step-size Adaptation

Consider the case when the fitness function is a constant function. Then the indices of vectors $\mathbf{d}_i^t$ have no correlation with their location in space. Consequently, since the selected vectors $\mathbf{d}_i^t$ are independent and normally distributed with zero mean and the covariance $\mathbf{C}^t$, the vectors $(\mathbf{C}^t)^{-1/2}\mathbf{d}_i^t$ are distributed with zero mean and the identity covariance matrix. Then the vector $(\mathbf{C}^t)^{-1/2}\mathbf{\Delta}^t$ will be distributed with zero mean and the covariance matrix equal to:

$$\Sigma((\mathbf{C}^t)^{-1/2}\mathbf{\Delta}^t) = \sum_{i=1}^{\mu}(w_i)^2\mathbf{I} = \frac{1}{\mu_{\text{eff}}}\mathbf{I} \tag{1}$$

The vectors $(\mathbf{C}^t)^{-1/2}\mathbf{\Delta}^t$ will be also mutually independent.

The CSA method (see Procedure 1) takes advantage from this observation. Vectors $(\mathbf{C}^t)^{-1/2}\mathbf{\Delta}^t$ from consecutive iterations are accumulated into the vector $\mathbf{p}_\sigma^t$. In the case of constant fitness function, the vector $\mathbf{p}_\sigma^t$ will behave in consecutive iteration as if it were randomly generated with zero mean and identity covariance, since its covariance matrix is

$$\Sigma(\mathbf{p}_\sigma^{t+1}) = (1 - c_s)^2\Sigma(\mathbf{p}_\sigma^t) + \mu_{\text{eff}}c_s(2 - c_s)\Sigma((\mathbf{C}^t)^{-1/2}\mathbf{\Delta}^t) \tag{2}$$

Taking into account (1) and assuming that $\Sigma(\mathbf{p}_\sigma^t) = \mathbf{I}$ we obtain:

$$\Sigma(\mathbf{p}_\sigma^{t+1}) = (1-c_s)^2\mathbf{I} + \mu_{\text{eff}} c_s(2-c_s)\left(\frac{1}{\mu_{\text{eff}}}\mathbf{I}\right) = \mathbf{I} \tag{3}$$

Hence, if the fitness function is a constant function then the vector $\mathbf{p}_\sigma^t$ can be treated as a random variable with zero mean and the identity covariance matrix. Then it will hold

$$E\|\mathbf{p}_\sigma^t\| = E\|N(\mathbf{0},\mathbf{I})\|$$

and

$$E\|\sigma^{t+1}\| = E\|\sigma^t\|$$

For an arbitrary fitness function, the length of $(\mathbf{C}^t)^{-\frac{1}{2}}\boldsymbol{\Delta}^t$ will differ from the case of the constant fitness. Two factors will influence the change of $E\|\mathbf{p}_\sigma^t\|$:

1. The average length of $(\mathbf{C}^t)^{-\frac{1}{2}}\boldsymbol{\Delta}^t$ is smaller when the selection process prefers shorter vectors $\mathbf{d}_i^t$, which typically takes place when the algorithm is nearby the local optimum. Then σ^t will be reduced. Contrarily, if longer vectors are selected, then this may indicate a saddle and will result in increasing σ^t.
2. If the consecutive vectors $\boldsymbol{\Delta}^t$ are positively correlated then the length of $\mathbf{p}_\sigma^t$ will increase along with the iteration number. Such a situation may appear when the populations are located on a slope towards optimum. In effect of increasing length of $\mathbf{p}_\sigma^t$ the step-size will increase. Due to the same mechanism, a negative correlation of consecutive vectors $\boldsymbol{\Delta}^t$ will result in decreasing σ^t.

2.2 Step-size control inspired by the one-fifth success rule

The one-fifth success rule [**?**] is a method to control the step-size that was originally applied for the (1+1)-ES algorithm. It has been verified that the optimization progress of (1+1)-ES is highest when the ratio of successful mutation is kept around 1/5. Therefore, a record of successful mutation should be kept for (1+1)-ES. If the ratio of successful mutations falls below 1/5 then the step-size should decrease, and if it exceeds 1/5 then the step size should be reduced.

The method inspired by the one-fifth rule was defined to control the step-size for (1+1)-CMA-ES [**?**]. This algorithm maintains a single point $\mathbf{m}^t$, a covariance matrix $\mathbf{C}^t$, and a step size multiplier σ^t. In each iteration one vector $\mathbf{d}^t$ is generated from the multinomial normal distribution with covariance matrix $\mathbf{C}^t$. An offspring point is defined as $\mathbf{x}^t = \mathbf{m}^t + \sigma^t\mathbf{d}^t$. If the fitness value of $\mathbf{x}^t$ is smaller than $\mathbf{m}^t$, the offspring supersedes its parent, and the covariance matrix $\mathbf{C}^t$ is made in a fashion similar to CMA-ES. The step-size adaptation is defined in a way inspired by the one-fifth rule — see Procedure 2.

The value p_{succ}^t accumulates the ratio of succesful mutations. The smoothing factor c_p, the damping factor d and the target success rate p_{target} are set by the user. The settings suggested in [**?**] are $c_p = 1/12, d = 1 + n/2, p_{\text{target}} = 2/11$.

The CSA rule adapts the value of σ^t in CMA-ES by looking on distribution of λ points generated in each iteration, while the one-fifth rule needs a number

that estimates the success rate. The Median Success Rule (MSR), which was introduced in [**?**], looks at percentiles of fitness values observed in points generated in consecutive iterations.

With $P_i(X)$ denote the i-th percentile of values conatined in the set X. In the MSR method, the median fitness of the current population $P_{0.5}(\{q(\mathbf{x}_i^t)\})$ is compared with a certain fitness percentile $P_k(\{q(\mathbf{x}_i^{t-1})\})$ from the previous population. It is argued in [**?**] that a desired situation is when it holds

$$Prob\{P_{0.5}(\{q(\mathbf{x}_i^t)\}) < P_k(\{q(\mathbf{x}_i^{t-1})\})\} \approx 1/2 \tag{4}$$

Hence, the desired success ratio is expected to be around one-half instead of one-fifth. According to the results of experiments provided in [**?**], the following settings are advisable: $k = 0.3, c_s = 0.3, d = 2(n-1)/n$.

3 Step-size adaptation based on the midpoint fitness

Generalization of the one-fifth rule to work for the algorithm that generates $\lambda > 1$ points, such as CMA-ES, is neither straightforward nor unique. Here we suggest three possible generalizations. All of them need to evaluate the fitness of a population midpoint. This causes some additional effort in terms of the number of fitness evaluations, which might be a drawback when a limited budget of the number of fitness evaluations is assumed. However, as it was shown in [**?**], computing of the population midpoint may improve a precise location of local optima and may serve as a basis to improve the overall efficiency of CMA-ES [**?**]. Therefore, we present three methods that take into account the values of fitness of population midpoints.

The first method, which is outlined in Procedure 4, is inspired by the one-fifth rule for (1+1)-CMA-ES. It looks at $\overline{\mathbf{x}}^{t-1}$ — the midpoint of points from the previous iteration. The success ratio p_s is defined as the number of points from the current iteration whose fitness is smaller than the previous iteration midpoint. The method does not use any smoothing procedure for p_s. Suggested values of parameters are $p_{\text{target}} = 1/5, d = 3$.

The second method is outlined in Procedure 5. It is another attempt to directly apply the philosophy of the one-fifth rule. In (1+1)-ES, one parent generates one offspring by adding a normally distributed difference vector. In CMA-ES, normally distributed difference vectors $\mathbf{d}_i^t$ are used to generate new points around the point $\mathbf{m}^t$. Therefore, the point $\mathbf{m}^t$ acts as a parent, and its fitness value should be taken for comparison with all newly generated points. Suggested values of parameters are $p_{\text{target}} = 1/5, d = 8$.

In contrast to the aforementioned methods based on the midpoint from the previous iteration, the third considered method looks at the current iteration. Note however that the current population midpoint $\overline{\mathbf{x}}^t$ is an estimator of $\mathbf{m}^t$ since the average of independent samples of a random variable estimates its mean. The method outline is presented in Procedure 6. If the fitness of $\overline{\mathbf{x}}^t$ is better than the best point of $\mathbf{x}_i^t$ then the population has probably approached to the vicinity of a local optimum. Then the precision can be improved by reducing

σ^t. Contrarily, if the population is located on a slope and the fitness function can be locally approximated as a linear function, then the population mean fitness $q(\overline{\mathbf{x}}^t)$ will be roughly equal the median fitness value of the population.

In the considered method we compare $q(\overline{\mathbf{x}}^t)$ with the k percentile of the population fitness. In further text we report the results obtained for $k = 0.09$, and the step-size multiplier was set to $\alpha = 0.83$, according to [**?**].

4 Convergence of CMA-ES with various step-size adaptations

4.1 Compared methods

In this section we investigated rates of convergence and computing overhead of proposed methods. As mentioned in previous sections we considered five methods using different σ adaptation rules:

1. CMA-ES-CSA
2. CMA-ES-MSR
3. CMA-ES-EXPTH
4. CMA-ES-JA
5. CMA-ES-QUANT

Each method used default settings for basic parameters suggested by authors execpt for population size λ. All methods generated $\lambda = 4N$ points in each iteration. For CMA-ES-MSR to count K_{succ} of better points in current population than j-th best point of the previous population we set j as:

$$j = 0.3\lambda.$$

4.2 Fitness functions: linear, quadratic, gutter

To illustrate and compare rates of convergence we used three different fitness landscapes i.e. linear (5), quadratic (6) and gutter function (7).

$$f_l(\mathbf{x}) = x_1 \tag{5}$$

$$f_q(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 \tag{6}$$

$$f_g(\mathbf{x}) = x_1 + \sum_{i=2}^{D} x_i^2,\ \mathbf{x} \in \mathbb{R}^D \tag{7}$$

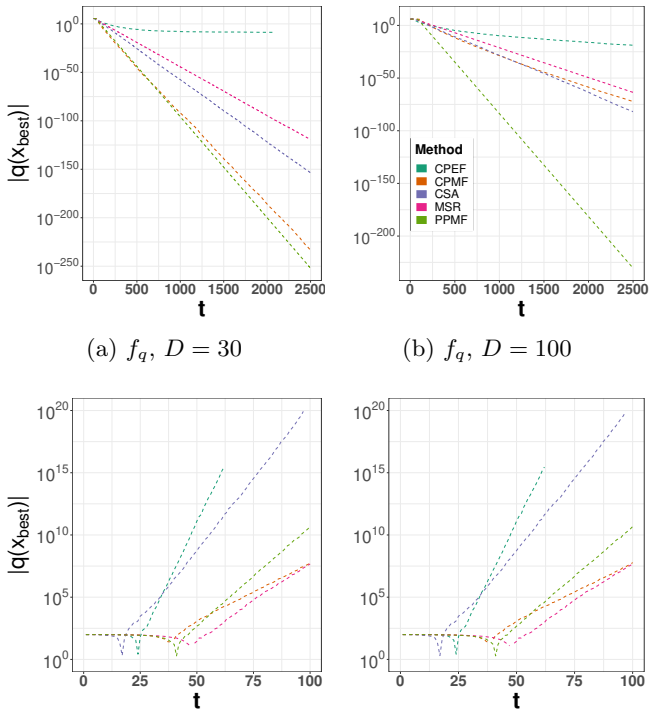
To assess the rates of convergence, each function was treated in a minimalization manner and we recorded the fitness of the arthimetic mean and the best point in the population.
The curves depicted in the figures below for iteration t show the fitness of the

best point and midpoint in the tth generation.
For each problem and algorithm, the mean point of population is initialized as:

$$\mathbf{m}_i = [100, \ldots, 100]^D.$$

4.3 Conclusions: convergence rate, benefits from using midpoint, search space dimension

Results of simulations reveal that performance of non-CSA variants is competitive with CSA and MSR on both functions. For n = 30 one can see that PPMF and CPEF have the biggest convergence rate among all considered methods. Behavior of CPEF seems to be ambiguous due to significantly different rate of convergence depending on dimensionality — it may suggest potential inapplicability to problems with higher number of dimensions. On the other hand, constant convergence rate of PPMF and outperforming behavior on sphere function regardless of dimensionality encourage to formulate two contrary hypothesis: such improvement will be observed further or PPMF variant will be affected by premature convergence. Possibility of such improvement or deterioration is verified in section **??**



(a) f_q, $D = 30$ (b) f_q, $D = 100$

(c) f_l, $D = 30$ (d) f_l, $D = 100$

Fig. 1: Convergence curves for different fitness landscapes.

5 Benchmarking step-size adaptation methods on CEC'2013 and CEC'2017

5.1 Overview of CEC'2013 and CEC'2017

We tested the performance of proposed methods using standardized sets of single objective optimization problems i.e. CEC2017 [**?**] and CEC2013 [**?**]. Both of them are based on the benchmark set created in 2005 [**?**].

Each problem in set is defined as follows:

$$\min_{\mathbf{x}\in[-100,100]^D} f(\mathbf{x})$$

and should be considered as a black-box problem. One knows only dimensionality D, evaluation budget and value of global optimum. Note that the number of benchmark functions and functions by itself in CEC2013 and CEC2017 vary.

CEC2013 has 28 functions which are dived into three groups: unimodal, multimodal and composition functions. Structure of CEC2017 is similar with but

with the addition of hybrid as a fourth group of function. Hybrid and composition functions are composed of several multimodal functions which are defined in such a way that in various regions of the domain the dynamics of the objective function is dominated by different components of that composition. Thus, the optimization algorithm must be able to capture these regularities over the whole run. Structures of benchmarks are presented in the table below (**??**).

Benchmark	Unimodal	Multimodal	Hybrid	Composition
CEC2017	$f_1, \ldots, f_3$	$f_4, \ldots, f_{10}$	$f_{11}, \ldots, f_{20}$	$f_{21}, \ldots, f_{30}$
CEC2013	$f_1, \ldots, f_5$	$f_6, \ldots, f_{20}$		$f_{21}, \ldots, f_{28}$

Table 1: Structure of used benchmarks.

We evaluated each algorithm with default settings for CEC benchmarks i.e. 51 independent repetitions for function, $10^4 \cdot D$ objective function evaluations and dimensionality $D = 10, 30, 50$.

5.2 Presentation of results using ECDF curves

Instead of using tabular form of results presentation suggested by authors of CEC competitions, we decided to use ECDF (empirical cumulative distribution function) curve. ECDF curve was proposed by Hansen [**?**] as a convenient form of results aggregation. A single curve defines the average dynamics of the algorithm. More formally, let us suppose that one of the tested algorithms in k-th repetition of D-dimensional benchmark function $f \in \mathcal{F}$ recorded in generation t value $Q^f_{k,D,t}$ which is the difference between the best-so-far objective function value and the global optimum.
Next, consider mapping from $Q^f_{k,D,t}$ to unit interval $[0, 1]$:

$$q^f_{k,D,t} = \frac{\log\left(\frac{Q^f_{k,D,t}}{m^f_D}\right)}{\log\left(\frac{M^f_D}{m^f_D}\right)} \tag{8}$$

where m^f_D and M^f_D are respectively minimal and maximal values achieved among all tested algorithms, repetitions and generations for the given set of problems.

To obtain ECDF points for functions from $\mathcal{F}$ one has to aggregate above values as follows:

$$q^F_{D,t} = \frac{1}{K \cdot |F|} \sum_{f \in F} \sum_{k=1}^{K} q^f_{k,D,t} \tag{9}$$

where K is the number of repetitions.
The curve is constructed by plotting values of $q^F_{D,t}$ against the set of fitness evaluation milestones based on fraction of given budget.

5.3 Results

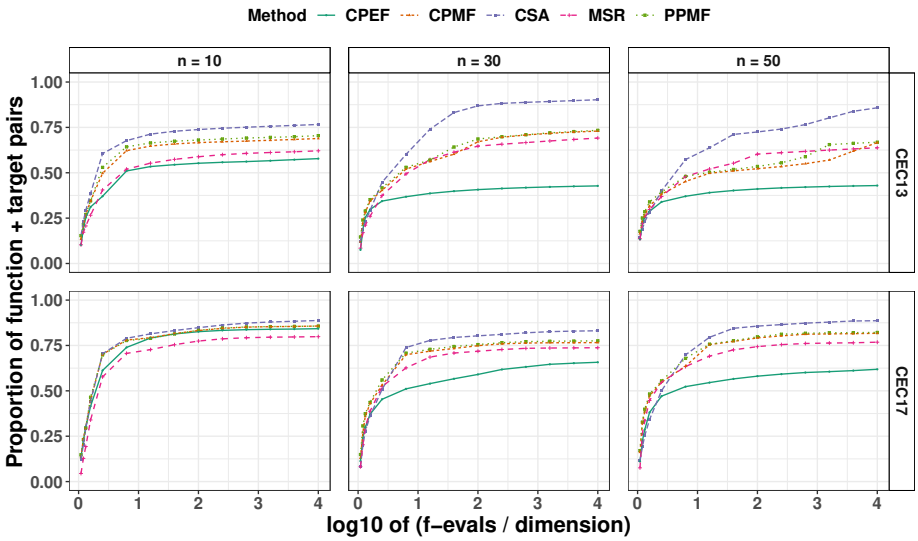


Fig. 2: ECDF curves for $D = 10, 30, 50$

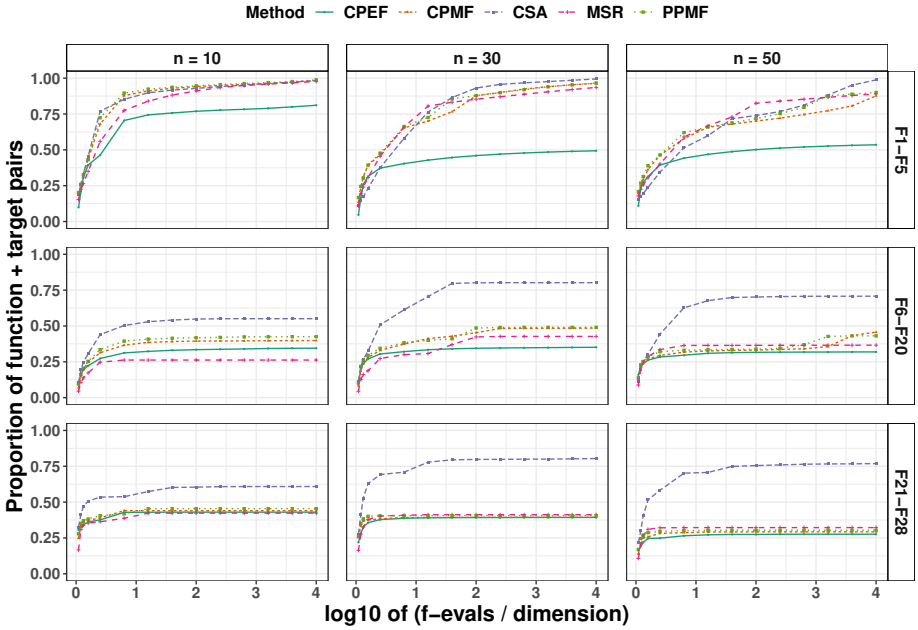


Fig. 3: ECDF curves for $D = 10, 30, 50$

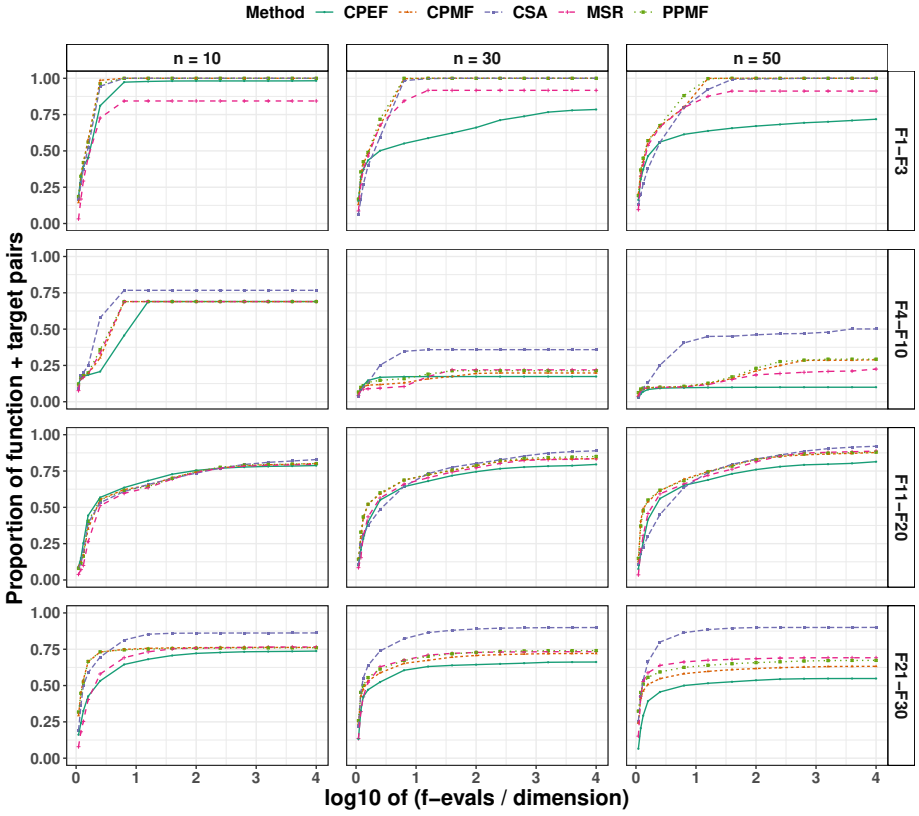


Fig. 4: ECDF curves for $D = 10, 30, 50$

6 Conclusions

1. which method is advisable **JA+EW**
2. how much computing we can spare by avoiding matrix operations **JA+EW**
3. future research directions **JA+EW**