Multivariate analysis: genetic algorithm and uncentered PCA

Oksana Shadura[1,2,3], Anatoliy Petrenko[2], and Sergey Svistunov[2,3]

[1] CERN, Geneve, Switzerland
[2] National Technical University of Ukraine "Igor Sikorsky Kyiv Politechnic Institute", Kiev, Ukraine
[3] Bogolyubov Institute for Theoretical Physics of the National Academy of Sciences of Ukraine, Kiev, Ukraine

Abstract. Machine learning for complex multi-objective problems (MOP) can substantially speedup the discovery of solutions belonging to Pareto landscapes and improve Pareto front accuracy. Studying convergence speedup of multi-objective search on well-known benchmarks is an important step in the development of algorithms to optimize complex problems such as High Energy Physics particle transport simulations. In this paper we will describe how we perform this optimization via a tuning based on genetic algorithms and machine learning for MOP. One of the approaches described is based on the introduction of a specific multivariate analysis operator that can be used in case of expensive fitness function evaluations, in order to speed-up the convergence of the "black-box" optimization problem.

1 Introduction

Modern fundamental science requires the development of complex experimental machines like the LHC in High Energy Physics or land and space based x-ray and gamma-ray telescopes in High Energy Astrophysics. Other examples can be found in the fields of protein synthesis, gene regulation research on genome evolution. All these activities generate large data sets and require the development of new approaches and methods for their efficient analysis on modern computer platforms.

In the point of the work on analyzing and optimizing the performance of the GeantV code [1], which is the prototype of the next-generation particle transport simulation software intended to succeed to Geant4 [2], which is the current golden standard in high energy physics (HEP) and beyond. Geant4 is a toolkit for simulation of the passage of particles through different kinds of matter, with application including high energy and nuclear physics, accelerator physics, medicine and space science. It is widely used in HEP experiments at the Large Hadron Collider (LHC) located at CERN (Geneva, Switzerland).

One of the objectives of the GeantV project is to achieve good performance on a wide range of modern computing architectures with good scalability for complex computations. This is important since Geant4 is the single program

consuming the largest share (50%) of the CPU cycles used for HEP. This code was developed in the 90s and it is now not well suited to take advantage from the latest CPU and accelerator architectures.

The GeantV project started in 2013, following an R&D phase focused on optimal exploitation of instruction level parallelism for particle transport simulation both on CPU and on accelerators such as GPUs and Intel Xeon Phi ®. Emphasis has been put on the optimization of cache usage by careful management of data locality [3]. GeantV is getting significant benefits via a specially developed computational solid geometry (CSG) modeler, which provides a set of optimized shape primitives and highly parallel geometry navigator. This provides GeantV with the necessary *ray-tracing* functionality for the efficient propagation of particles through the target geometry [4].

The GeantV project is recasting the simulation algorithms to get maximum benefit from SIMD/MIMD architectures on highly massive parallel systems [5]. This involves finding the optimal balance of several factors influencing computational performance (floating-point performance, off-chip memory bandwidth, usage of cache and memory hierarchy and etc.). As a consequence, a large number of parameters have to be optimized. This optimization task can be treated as a black-box problem, which requires searching the optimum set of parameters using only point-wise function evaluations.

In our optimization work, we consider particle transport simulation to be a complex heuristic parametric model with costly evaluations and unpredictable behavior of fitness landscape, that we intend to optimize using stochastic search algorithms. The objective of this work is to observe whether, by using unsupervised machine learning, we can accelerate the process of finding a Pareto front describing dominance relations between fitness functions.

Results described in this article is part of the research on the "black-box" optimization of GeantV as a multi-objective problem for performance measurements. Combining together genetic algorithm and machine learning approach we will try to discover special behaviors and fixed points of evolutionary systems, trying to accelerate convergence rate of algorithm for "black-box" optimization. Before going to optimize GeantV simulations, we will try to prototype algorithm's performance on a set of numerical DTLZ benchmarks [6] in order to accelerate their convergence to the true Pareto front via the integration of multi-objective search/optimisation (MOO) algorithms and unsupervised machine learning (PCA).

2 Genetic algorithms

Genetic algorithm is one of the widely used evolutionary algorithms for studies of various optimization problems. Theory of genetic algorithms (GA) was a subject of research for the last decades. Generally accepted model for studying GA is a simple model of genetic algorithm (SGA) [7] as a prototype of evolutionary system. This model is describing genetic algorithm (GA) as a dynamical system with accurate mathematical definitions and well studied in a literature.

In this model for description of GA a Markov chain is used, where states are populations and transition are operated by sets of genetic operators: selection, crossover and mutation [8]. Mutation ensures that the Markov chain is connected: therefore there is an unique equilibrium distribution over populations.

In this scenario, the probability to produce a particular population in one generation depends only on the previous generation external influencing factors. This randomized process is described by a Markov chain, characterized by a transition matrix $\Theta_{\boldsymbol{q},\boldsymbol{p}}$ from the population $\boldsymbol{p}$ to the population $\boldsymbol{q}$.

Dynamical systems describe the evolution of individuals in the finite space of possible populations of fixed size m, where m is number of measurements during the experiment. While rethinking the genetic algorithms as a discrete dynamical system, many interesting mathematical objects like fixed points could be found. These objects are apparently not only generic for simple genetic algorithms, but also general for optimization problems. Let's briefly recall the results presented in [7] and establish the possible links with the task of optimizing our parameters.

We have a population of N different types of individuals in search sample space Ω. Each element of Ω can be thought of as a "unique individual" with a given fitness value defined by the cost function.

A population consists of m-subsets ($m \ll N$) each of which contains v_{α_i} of the α_i-type individual where $i = 1, ..., m$ and defined by vector

$$\boldsymbol{b} = (b_{\alpha_1}, b_{\alpha_2}, ..., b_{\alpha_m})^t$$

where $\alpha_i \in \Omega$. The size of the population is $\bar{m} = \sum_{i=1}^{m} b_{\alpha_i}$.

We can redefine the population vector in the following form

$$\boldsymbol{p} = (p_1, p_2, ..., p_N)^t$$

where p_α ($p_{\alpha_i} = b_{\alpha_1}/\bar{m}$) is the probability of occurrence α-th individual in the population vector $\boldsymbol{b}$.

In this representation the repeated application of the genetic algorithm gives a sequence of vectors $\boldsymbol{p} \in \Lambda$ where

$$\Lambda = \{(p_1, p_2, ..., p_N)^t \in R^N \,|\, 0 \leq p_\alpha \leq 1, \ \sum_{\alpha=1}^{N} p_\alpha = 1\}.$$

Λ is a set of admissible states for the populations. We can consider Λ as a $(N-1)$-dimensional simplex (a hyper-tetrahedron).

Let $\mathcal{G}_\alpha(\boldsymbol{p})$ is a certain probability of producing individual α in the next generation if the previous population was $\boldsymbol{p}$ and define map $\mathcal{G} : \Lambda \to \Lambda$, where $\mathcal{G}(\boldsymbol{p}) = \prod_{\alpha\in\Omega} \mathcal{G}_\alpha(\boldsymbol{p})$, and $\mathcal{G}(\boldsymbol{p}) \in \Lambda$ could be considered as heuristic function. $\mathcal{G}(\boldsymbol{p})$ is GA procedure on $\boldsymbol{p} \in \Lambda$ and the map $\mathcal{G}$ is actually the composition of three different maps: selection, mutation and crossover.

Let define genetic selection operator $\hat{\mathcal{F}} : \Lambda \to \Lambda$, where $\mathcal{F}(\boldsymbol{p}) = \prod_{\alpha\in\Omega} \mathcal{F}_\alpha(\boldsymbol{p})$ and the α-th component, $\mathcal{F}_\alpha(\boldsymbol{p})$, represents the probability of the appearance of an individual of type α if the selection is applied to $\boldsymbol{p} \in \Lambda$. A selection operator chooses individuals from the current population using the cost function vector,

$\boldsymbol{f} = \{f_\alpha\} \in R^N$, where $f_\alpha = f(\alpha)$, $\alpha \in \Omega$. This generic type of selection collects elements with probability proportional to their fitness. This corresponds to a heuristic function

$$\mathcal{F}(\boldsymbol{p}) = \frac{\mathrm{diag}\,(f) \cdot \boldsymbol{p}}{\boldsymbol{f}^{\,t} \cdot \boldsymbol{p}},$$

where $\boldsymbol{p} \in \Lambda$ is the population vector, and $\mathrm{diag}\,(\boldsymbol{f})$ is the matrix with entries from $\boldsymbol{f}$ along the diagonal and zeros elsewhere.

The mutation operator $\hat{\mathcal{U}} : \Lambda \to \Lambda$ is an $N \times N$ real valued matrix with (α, β)-th entry $u_{\alpha,\beta} > 0$ for all α, β, and $u_{\alpha,\beta}$ represents the probability that individual $\beta \in \Omega$ mutates into $\alpha \in \Omega$. Then $(\hat{\mathcal{U}} \cdot \boldsymbol{p})_\alpha$ is the appearance of an individual of type α after applying a mutation to the population $\boldsymbol{p}$.

Lets define the crossover operator $\hat{\mathcal{C}} : \Lambda \to \Lambda$,

$$\mathcal{C}(\boldsymbol{p}) = (\boldsymbol{p}^t \cdot \hat{C}_1 \cdot \boldsymbol{p}, ..., \boldsymbol{p}^t \cdot \hat{C}_N \cdot \boldsymbol{p})$$

for $\boldsymbol{p} \in \Lambda$, where $\hat{C}_1, ..., \hat{C}_N$ is a sequence of symmetric non-negative $N \times N$ real-valued matrices. Here $\hat{C}_\alpha(\boldsymbol{p})$ represents the probability that an individual α is generated by applying the crossover to population $\boldsymbol{p}$.

Combining the selection, mutation and crossover maps we obtain the complete operator $\hat{\mathcal{G}}$ for the genetic algorithm (GA map)

$$\hat{\mathcal{G}} \; : \; \Lambda \to \Lambda, \quad \hat{\mathcal{G}}(\boldsymbol{p}) = \hat{\mathcal{C}} \circ \hat{\mathcal{U}} \circ \mathcal{F}(\boldsymbol{p}).$$

If we know the heuristic function $\mathcal{G}$, we can write the transition matrix which is stochastic and based on the probability of transforming the population $\boldsymbol{p}$ into the population $\boldsymbol{q}$:

$$\Theta_{\boldsymbol{q},\boldsymbol{p}} = \bar{m}! \prod_{\alpha \in \Omega} \frac{(\mathcal{G}_\alpha(\boldsymbol{p}))^{\bar{m} q_\alpha}}{(\bar{m} q_\alpha)!} \tag{1}$$

where $\mathcal{G}_\alpha(\boldsymbol{p})$ is probability of producing individual α in the next generation and $\bar{M} q_\alpha$ is the number of copies of individuals α in the population $\boldsymbol{q}$, $\bar{m}$ is the size of the population.

As a brief review, the convergence properties of the simple genetic algorithm evolution schema was properly explored in the work [9]. While [10] showed that the convergence rate of the genetic algorithm is determined by the second largest eigenvalue of the transition matrix (1). The details of the proof was performed for diagonalizable transition matrices and transferred to matrices in Jordan normal form.

Another remarkable feature of the SGA is the presence of a rich structure of fixed and metastable points (for a detailed discussion see [8]).

Describing GA model through Markov chain representation we try to discover "hotspots" and find algorithmic or data patterns that could be used for improvement of the GA.

For the optimization of the GeantV simulation, we identify a set of optimization parameters important for the performance of particle transport simulations (e.g. the size of vector of particles to be transported or other significant design

features) and build the data matrix $X_{\alpha,i} = \{(\boldsymbol{x}_\alpha)_i\} = \{\boldsymbol{x}_\alpha\}$ which contains the values of these parameters. In this matrix index i enumerates the tuning parameters ($i = 1, ..., n$) and index α enumerates the number of measurements of the parameters ($\alpha = 1, ..., M$ for M measurements), while in terms of GA index α enumerates M individuals and the population vector is constituted by $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M)$.

Recall the data and probabilistic sample representation. In the first case we can associate the vector based on the measurements of the i-th parameter $\boldsymbol{x}_i' = \{(\boldsymbol{x}_i')_\alpha\} = \{(\boldsymbol{x}_i')_1, (\boldsymbol{x}_i')_2, ..., (\boldsymbol{x}_i')_M\}$, where the component $(\boldsymbol{x}_i')_\alpha$ corresponds to the value of the i-th parameter in the α-th measurement with the population vector $(\boldsymbol{x}_1', \boldsymbol{x}_2', ..., \boldsymbol{x}_n')$.

In the second case $P_i(x)$ be the probability distribution function of the measurements of the i-th parameter, with normalization

$$\int_{-\infty}^{\infty} dx P_i(x) = 1.$$

Using the previous strategy we associate the population vector

$$(\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_n) \quad with \quad (\boldsymbol{x}_1', \boldsymbol{x}_2', ..., \boldsymbol{x}_n')$$

where

$$\boldsymbol{p}_i = \{(\boldsymbol{p}_i)_1, (\boldsymbol{p}_i)_2, ..., (\boldsymbol{p}_i)_M\},$$

and the component $(\boldsymbol{p}_i)_\alpha$ is the probability to measure of the i-th parameter value $(\boldsymbol{x}_i')_\alpha$ in the α-th measurement.

One of the challenges of a Markov chain is to determine the evolution of the components along an appropriate direction for faster convergence to equilibrium. Using Principal Component Analysis (PCA) allows to check the genetic algorithm parameter sensitivity and the possible correlation between parameters. For this we introduce a operator that will be based on PCA and inverse PCA noise reduction operation for a genetic algorithm's optimisation of set of parameters.

We considered a possibility to improve the convergence rate by adding to a standard set of GA operator's (selection, mutation, crossing), a new operator $\hat{P}$ performing uncentered PCA on the GA populations. We will analyze the result of the implementation of the operator on the uncentered data matrix on standard GA performance benchmarks. From the experimental output we see that as in the SGA case [10], the convergence rate of genetic algorithm depends on the eigenvalues following the highest one, and for this reason the proposed operator $\hat{P}$ was applied on them.

3 Pre-processing and post-processing of data: centered and uncentered PCA and noise cleanup procedure

Usually PCA is used to analyze the covariance matrix in order to reduce a complex data set (in our case performance measurements) to a lower dimensional

set. In this case PCA is traditionally applied to the centered data matrix. In this subsection we present a way that sort of PCA could be implemented on an uncentered data matrix. This is particularly convenient in the case of transformations of constrained data measurements using genetic algorithms, which are in our case highly constrained and multi-scaled performance parameters. As a basis of ideas about the connection between the centered and uncentered data matrix was used ideas from [11, 12].

As a data matrix we have a set of performance parameters, while objective functions are represented in a set of the cost functions evaluating performance efficiency of GeantV simulations.

1) PCA for the centered data matrix. The elements of the data matrix $\hat{X}$ of size $M \times n$ are described through M-samples of data from an n-dimensional space. In our case M is the number of individuals in the generation and n is the size of the individual (n is the dimension of vector of genes $\boldsymbol{x} = \{x_i\}, 1 \leq i \leq n$).

Let $\boldsymbol{x}_\alpha = \{(\boldsymbol{x}_\alpha)_i\} (1 \leq \alpha \leq M, 1 \leq i \leq n)$ is α-th individual of the population and

$$\hat{X}^{(u)} = \{X^{(u)}_{\alpha,i}\} = \{(\boldsymbol{x}_\alpha)_i\}, \tag{2}$$

be a uncentered data matrix, size $M \times n$. Let us define the centered data matrix $\hat{X}^{(c)}$:

$$\hat{X}^{(c)} = \{X^{(c)}_{\alpha,i}\} = \{X^{(u)}_{\alpha,i} - \mu_i\}, \tag{3}$$

where μ_i is mean over M-individuals of i-th component of the gene:

$$\mu_i = \frac{1}{M} \sum_{\alpha=1}^{M} X^{(u)}_{\alpha,i}, \quad 1 \leq i \leq n, \quad \boldsymbol{\mu} = \{\mu_i\}. \tag{4}$$

The centered data matrix defines the covariance matrix $\hat{\Sigma}$

$$\hat{\Sigma} = \frac{1}{M} \hat{X}^{(c)\,t} \cdot \hat{X}^{(c)} = \{\Sigma_{i,j}\} = \frac{1}{M} X^{(c)\,t}_{i,\alpha} X^{(c)}_{\alpha,j} \tag{5}$$

with the matrix multiplication repeated induces imply summation. Similarly for the uncentered data matrix we obtain the matrix of non-central second moments,

$$\hat{T} = \frac{1}{M} \hat{X}^{(u)\,t} \cdot \hat{X}^{(u)} = \{T_{i,j}\} = \frac{1}{M} X^{(u)\,t}_{i,\alpha} X^{(u)}_{\alpha,j} \tag{6}$$

In standard PCA terms lets define an orthonormal vector $\boldsymbol{u}_1$ $(u_{i,1}, i = 1, ..., n)$ for which the projection $x^{(c)}_{u1,\alpha}$ of the vector $\boldsymbol{x}^{(c)}_\alpha$ on $\boldsymbol{u}_1$ has the largest variance σ^2_{u1}:

$$x^{(c)}_{u1,\alpha} = \boldsymbol{x}^{(c)\,t}_\alpha \cdot \boldsymbol{u}_1 = \sum_{i=1}^{n} X^{(c)}_{\alpha,i} u_{i,1}, \quad \boldsymbol{u}^t_1 \cdot \boldsymbol{u}_1 = 1,$$

and

$$\sigma^2_{u1} = \frac{1}{M} \sum_{\alpha=1}^{M} \left[\sum_{i=1}^{n} X^{(c)}_{\alpha,i} u_{i,1} \right]^2$$

Then the first principal component (PC) $\boldsymbol{v}_1^{(c)}$ is the linear combination with the largest variance: $v_{\alpha,1}^{(c)} = \boldsymbol{x}_\alpha^{(c)\,t} \cdot \boldsymbol{u}_1 = X_{\alpha,i}^{(c)} u_{i,1}$, where the n-dimensional vector $\boldsymbol{u}_1 = (u_{1,1}, ..., u_{n,1})^T$ solves

$$\boldsymbol{u}_1 = \arg\max_u \mathrm{Var}(\boldsymbol{x}_\alpha^{(c)\,t} \cdot \boldsymbol{u}), \quad \boldsymbol{u}^T \cdot \boldsymbol{u} = 1.$$

The second principal component is the linear combination with the second largest variance and orthogonal to the first principal component, and so on.

To calculate PC it is convenient to consider the variational problem. For $\boldsymbol{v}^{(c)} = \{v_\alpha^{(c)}\} = \{X_{\alpha,i}^{(c)} u_i\}$ we have

$$\mathrm{Var}(\boldsymbol{v}^{(c)}) = \frac{1}{M}\, \boldsymbol{u}^{\,t} \cdot \hat{X}^{(c)\,t} \cdot \hat{X}^{(c)} \cdot \boldsymbol{u} = \boldsymbol{u}^{\,t} \cdot \hat{\Sigma} \cdot \boldsymbol{u} \tag{7}$$

and with the Lagrangian

$$\mathrm{L} = \boldsymbol{u}^{\,t} \cdot \hat{\Sigma} \cdot \boldsymbol{u} + \lambda(\boldsymbol{u}^{\,t}\boldsymbol{u} - 1).$$

The stationary condition is

$$\frac{\partial L}{\partial \boldsymbol{u}} = 2\hat{\Sigma} \cdot \boldsymbol{u} - 2\lambda \boldsymbol{u} = 0, \quad \hat{\Sigma} \cdot \boldsymbol{u} = \lambda \boldsymbol{u}.$$

This matrix equation has n solutions

$$\hat{\Sigma} \cdot \boldsymbol{u}_j = \lambda_j^{(c)} \boldsymbol{u}_j, \quad 1 \le j \le n,$$

where $\boldsymbol{u}_j$ are eigenvectors of $\hat{\Sigma}$ with the eigenvalue $\lambda_j^{(c)}$ and $\boldsymbol{u}_j$ satisfy the orthonormality condition

$$\boldsymbol{u}_i^{\,t} \cdot \boldsymbol{u}_j = \delta_{i,j}, \quad 1 \le i, j \le n, \tag{8}$$

and

$$\boldsymbol{u}_j^{\,t} \cdot \hat{\Sigma} \cdot \boldsymbol{u}_j = \lambda_j^{(c)}. \tag{9}$$

Then the direction with maximum variance is the eigenvector with the largest eigenvalue. This procedure can be iterated to get the second largest variance projection (orthogonal to the first one), and so on.

From (7) we get:

a) for the variance of the i-th centered principal component

$$\mathrm{Var}(\boldsymbol{v}_i^{(c)}) = \boldsymbol{u}_i^{\,t} \cdot \hat{\Sigma} \cdot \boldsymbol{u}_i = \lambda_i^{(c)} \tag{10}$$

b) for the covariance of the i-th and j-th centered principal components $(i \neq j)$

$$\mathrm{Cov}(\boldsymbol{v}_i^{(c)}, \boldsymbol{v}_j^{(c)}) = \boldsymbol{u}_i^{\,t} \cdot \hat{\Sigma} \cdot \boldsymbol{u}_j = 0. \tag{11}$$

Let define $U_{i,j} = \boldsymbol{u}_j = (u_i)_j$, from (8) this matrix satisfies the orthonormality condition

$$U_{i,i'}^t U_{i',j} = \delta_{i,j}. \tag{12}$$

Then in matrix form we have

$$\hat{U}^{\,t} \cdot \hat{\Sigma} \cdot \hat{U} = \hat{\Lambda}^{(c)}, \quad \Lambda_{i,j}^{(c)} = \lambda_i^{(c)} \delta_{i,j}, \tag{13}$$

Let define $V_{\alpha,j}^{(c)} = \{\boldsymbol{v}_j^{(c)}\} = \{(v_\alpha^{(c)})_j\}$, where $\boldsymbol{v}_j^{(c)}$ – j-th centered principal component. Then

$$V_{\alpha,j}^{(c)} = X_{\alpha,i}^{(c)} U_{i,j}, \quad 1 \le \alpha \le M, \tag{14}$$

and the first principal component $\boldsymbol{v}_1^{(c)}$

$$v_{\alpha,1}^{(c)} = V_{\alpha,1}^{(c)} = X_{\alpha,i}^{(c)} U_{i,1} = \boldsymbol{x}_c^{(\alpha)\,t} \cdot \boldsymbol{u}_1$$

if $\lambda_1^{(c)}$ is the largest eigenvalue. From (12), (13) we have

$$V_{i,\alpha}^{(c)\,t} V_{\alpha,j}^{(c)} = M \Lambda_{i,j}^{(c)} = M \lambda_i^{(c)} \delta_{i,j}. \tag{15}$$

It is convenient to define the new matrix $\bar{V}_{\alpha,j}$

$$V_{\alpha,j}^{(c)} = \sqrt{M} \bar{V}_{\alpha,i}^{(c)} \Lambda_{i,j}^{(c)\,1/2}, \tag{16}$$

where

$$\Lambda_{i,j}^{(c)\,1/2} = \left[\lambda_i^{(c)}\right]^{\frac{1}{2}} \delta_{i,j},$$

Then from (15) and (16) we obtain:

$$\bar{V}_{i,\alpha,}^{(c)\,t} \bar{V}_{\alpha,j}^{(c)} = \delta_{i,j},$$

Using (16), (14) and (13) we have

$$\bar{V}_{i,\alpha}^{(c)\,t} X_{\alpha,k}^{(c)} X_{k,\beta}^{(c)\,t} \bar{V}_{\beta,j}^{(c)} = \Lambda_{i,j}^{(c)} = \lambda_i^{(c)} \delta_{i,j},$$

then $\bar{V}_{\alpha,j}^{(c)}$ is the matrix of eigenvectors $(\bar{v}_j^{(c)})_\alpha$ of the matrix $\hat{K}^{(c)} = \hat{X}^{(c)} \cdot X^{(c)\,t}$ of the size $M \times M$

$$K_{\alpha,\beta}^{(c)} (\bar{v}_j^{(c)})_\beta = X_{\alpha,k}^{(c)} X_{k,\beta}^{(c)\,t} (\bar{v}_j^{(c)})_\beta = \lambda_j^{(c)} (\bar{v}_j^{(c)})_\alpha.$$

From (14) we have

$$X_{\alpha,i}^{(c)} = V_{\alpha,j}^{(c)} U_{j,i}^{\,t},$$

This relation allows to obtain the Singular Value Decomposition (SVD) [13] for the centered data matrix

$$X_{\alpha,i}^{(c)} = \sqrt{M} \bar{V}_{\alpha,i}^{(c)}\, \Lambda_{i,j}^{(c)\,1/2} U_{j,i}^{\,t}. \tag{17}$$

After dimension reduction the reverse PCA gives the output data matrix $\tilde{X}_{\alpha,i}^{(c)}$:

$$\tilde{X}_{\alpha,i}^{(c)} = \sqrt{M} \bar{V}_{\alpha,i}^{(c)}\, \tilde{\Lambda}_{i,j}^{(c)\,1/2} U_{j,i}^{\,t} = \tag{18}$$

$$= \sqrt{M}\left(\left[\lambda_1^{(c)}\right]^{\frac{1}{2}} \bar{V}_{\alpha,1}^{(c)} U_{1,i}^t + ... + \left[\lambda_m^{(c)}\right]^{\frac{1}{2}} \bar{V}_{\alpha,m}^{(c)} U_{m,i}^t\right).$$

if we retain m the principal components in the optimization problem. The approximation of matrix $X_{\alpha,i}^{(c)}$ is the matrix $\tilde{X}_{\alpha,i}^{(c)}$ of reduced rank $m < n$. This transformation is also known as the discrete Karhunen-Loéve or the Hotelling transformation [16].

Using the SVD representation (17) and (18) for the centered data matrix we calculate the mean square error (the standard error)

$$\eta_m = \frac{1}{nM}\sum_{\alpha=1}^{M}\sum_{i=1}^{n}(X_{\alpha,i}^{(c)} - \tilde{X}_{\alpha,i}^{(c)})^2 =$$

$$= \frac{1}{nM}\sum_{\alpha=1}^{M}\sum_{i=1}^{n}\left(\sqrt{M}\sum_{k=m+1}^{n}\sqrt{\lambda_k^{(c)}}\bar{V}_{\alpha,k}^{(c)}U_{k,i}^t\right)^2 =$$

$$= \frac{1}{n}\sum_{k=m+1}^{n}\lambda_k^{(c)}.$$

Thus the minimum error is obtained if the covariance matrix $\hat{\Sigma}$ has $(n-m)$ smallest eigenvalues $\lambda_j^{(c)}$, $m+1 \le j \le n$ and the Hotelling transformation can be considered as the "eigenvalue control parameter" approximation.

2)Uncentered PCA (uncentered data matrix case). In a similar way, we can apply the PCA method for the uncentered data matrix $\hat{X}^{(u)}$. Let $\boldsymbol{w}_j$ eigenvectors of the matrix of non-central second moments

$$\hat{T} = \frac{1}{M}\hat{X}^{(u)\,t} \cdot \hat{X}^{(u)} \tag{19}$$

with the eigenvalue $\lambda_j^{(u)}$

$$\hat{T} \cdot \boldsymbol{w}_j = \lambda_j^{(u)}\boldsymbol{w}_j, \quad 1 \le j \le n,$$

and satisfy the orthonormality condition

$$\boldsymbol{w}_i^t \cdot \boldsymbol{w}_j = \delta_{i,j}, \quad 1 \le i,j \le n,$$

then

$$\boldsymbol{w}_j^t \cdot \hat{T} \cdot \boldsymbol{w}_j = \lambda_j^{(u)}, \tag{20}$$

We define matrix $W_{i,j} = \boldsymbol{w}_j = (w_i)_j$ that satisfies the orthogonality condition

$$\hat{W}^{\,t} \cdot \hat{W} = \hat{I}.$$

From (20) we have

$$\hat{W}^{\,t} \cdot \hat{T} \cdot \hat{W} = \hat{\Lambda}^{(u)}, \quad \Lambda_{i,j}^{(u)} = \lambda_i^{(u)}\delta_{i,j}, \tag{21}$$

$\boldsymbol{v}_j^{(u)}$ is j-th uncentered principal component

$$v_{\alpha,j}^{(u)} = X_{\alpha,i}^{(u)} W_{i,j} = \boldsymbol{x}_\alpha^{(u)\,t} \cdot \boldsymbol{w}_j$$

For the variance of j-th uncentered principal component we obtain

$$\mathrm{Var}(\boldsymbol{v}_j^{(u)}) = \sigma_{w,j}^2 = \frac{1}{M} \sum_{\alpha=1}^{M} \left[\sum_{i=1}^{n} (X_{\alpha,i}^{(u)} - \mu_i) W_{i,j} \right]^2 =$$

$$= \boldsymbol{w} j^{\,t} \cdot \hat{T} \cdot \boldsymbol{w}_j - \left(\boldsymbol{\mu}^{\,t} \cdot \boldsymbol{w}_j \right)^2 =$$

$$= (\boldsymbol{\mu})^2 \left(\frac{\lambda_j^{(u)}}{(\boldsymbol{\mu})^2} - \cos^2(\boldsymbol{\mu}, \boldsymbol{w}_j) \right),$$

where

$$\mu_j = \frac{1}{M} \sum_{\alpha=1}^{M} X_{\alpha,j}^{(u)}, \quad 1 \le j \le n,$$

Similarly it can obtain [11]

$$\mathrm{Cov}(\boldsymbol{v}_i^{(u)}, \boldsymbol{v}_j^{(u)}) = -(\boldsymbol{\mu})^2 \cos(\boldsymbol{\mu}, \boldsymbol{w}_i) \cos(\boldsymbol{\mu}, \boldsymbol{w}_j),$$

and the cosine of the angle between the i-th column-centered PC and the j-th uncentered PC is

$$\cos(\boldsymbol{v}_i^{(c)}, \boldsymbol{v}_j^{(u)}) = \sqrt{\frac{\lambda_i^{(c)}}{\lambda_i^{(u)}}} \cos(\boldsymbol{u}_i, \boldsymbol{w}_j). \tag{22}$$

Hence that for case of uncentered matrix we do not have a simple relationship between the eigenvalues $\lambda_j^{(u)}$ and the variance j-th uncentered principal component $(\sigma_{w,j})^2$ as for the centered data matrix. However, this property is not essential for the usage of the PCA method for the GA and in this case it is convenient to apply the "eigenvalue control parameter" approximation. The idea is to use the PCA method for the SVD representation of the uncentered data matrix.

Let $V_{\alpha,j}^{(u)} = \boldsymbol{v}_j^{(u)} = (v_j^{(u)})_\alpha$. Then

$$V_{\alpha,j}^{(u)} = X_{\alpha,i}^{(u)} W_{i,j}, \quad 1 \le \alpha \le M, \tag{23}$$

and

$$V_{i,\alpha}^{(u)\,t} V_{\alpha,j}^{(u)} = M \Lambda_{i,j}^{(u)} = M \lambda_i^{(u)} \delta_{i,j}, , \tag{24}$$

Let us define the matrix $\bar{V}_{\alpha,j}$

$$V_{\alpha,j}^{(u)} = \sqrt{M} \bar{V}_{\alpha,i}^{(u)} \Lambda_{i,j}^{(u)\,1/2}, \tag{25}$$

where

$$\Lambda_{i,j}^{(u)\,1/2} = \left[\lambda_i^{(u)}\right]^{\frac{1}{2}} \delta_{i,j},$$

From (24) and (25) we obtain:

$$\bar{V}_{i,\alpha,}^{(u)\,t} \bar{V}_{\alpha,j}^{(u)} = \delta_{i,j}.$$

Again using (25), (23) and (20) we have

$$\bar{V}_{i,\alpha}^{(u)\,t} X_{\alpha,k}^{(u)} X_{k,\beta}^{(u)\,t} \bar{V}_{\beta,j}^{(u)} = \Lambda_{i,j}^{(u)} = \lambda_i^{(u)} \delta_{i,j},$$

and then $\bar{V}_{\alpha,j}^{(u)}$ is the matrix of eigenvectors $(\bar{v}_j^{(u)})_\alpha$ of the matrix $\hat{K}^{(u)} = \hat{X}^{(u)} \cdot X^{(u)\,t}$ of size $M \times M$

$$K_{\alpha,\beta}^{(u)} (\bar{v}_j^{(u)})_\beta = X_{\alpha,k}^{(u)} X_{k,\beta}^{(u)\,t} (\bar{v}_j^{(u)})_\beta = \lambda_j^{(u)} (\bar{v}_j^{(u)})_\alpha.$$

From (23) we obtain the representation for the uncentered data matrix

$$X_{\alpha,i}^{(u)} = V_{\alpha,j}^{(u)} W_{j,i}^t, \tag{26}$$

from which we get the SVD representation for the uncentered data matrix

$$X_{\alpha,i}^{(u)} = \sqrt{M} \bar{V}_{\alpha,k}^{(u)} \, \Lambda_{k,j}^{(u)\,1/2} W_{j,i}^t. \tag{27}$$

Then using the "eigenvalue control parameter" approximation we get for the output data matrix $\tilde{X}_{\alpha,j}^{(u)}$ of rang p

$$\tilde{X}_{\alpha,i}^{(u)} = \sqrt{M} \bar{V}_{\alpha,k}^{(u)} \, \tilde{\Lambda}_{k,j}^{(u)\,1/2} W_{j,i}^t = \tag{28}$$

$$= \sqrt{M} \left(\left[\lambda_1^{(u)}\right]^{\frac{1}{2}} \bar{V}_{\alpha,1}^{(u)} W_{1,i}^t + ... + \left[\lambda_p^{(u)}\right]^{\frac{1}{2}} \bar{V}_{\alpha,p}^{(u)} W_{p,i}^t \right),$$

where the eigenvalue matrix $\tilde{\Lambda}_{i,j}^{(u)}$ has rang p ($\lambda_{p+1}^{(u)} = \lambda_{p+2}^{(u)} = ... = \lambda_n^{(u)} = 0$).

We approximate $X_{\alpha,i}^{(u)}$ with rank n with the matrix $\tilde{X}_{\alpha,i}^{(u)}$ which has rank p. This is the analog of the Hotelling transformation.

Using the SVD representation we can estimate the mean square error η_p for this approximation:

$$\eta_p = \frac{1}{Mn} \sum_{\alpha=1}^{M} \sum_{i=1}^{n} (X_{\alpha,i}^{(u)} - \tilde{X}_{\alpha,i}^{(u)})^2 =$$

$$= \frac{1}{Mn} \sum_{\alpha=1}^{M} \sum_{i=1}^{n} \left(\sqrt{M} \sum_{k=p+1}^{n} \sqrt{\lambda_k^{(u)}} \bar{V}_{\alpha,k}^{(u)} W_{k,i}^t \right)^2 =$$

$$= \frac{1}{n} \sum_{k=p+1}^{n} \lambda_k^{(u)}$$

Again the minimum error is obtained if the matrix of non-central second moments $\hat{T}$ has $(n-p)$ smallest eigenvalues $\lambda_j^{(u)}$, $p+1 \leq j \leq n$.

Analysis of eigenvalues in SVD representation of the uncentered input data matrix $X_{\alpha,i}^{(u)}$ used as population in GA can significantly accelerate the processes of finding the Pareto front for the MOP. We verified this hypothesis for the standard GA test problems [6].

Eigenvectors with the largest eigenvalues likely determine the subspace of solutions of the MOP in which lies the Pareto front. Using an iterative procedure for uncentered data matrix from MOP we can faster converge to the optimal solution subspace.

PCA-based genetic operator $G_P(\boldsymbol{p}) = \hat{P} \circ \hat{C} \circ \hat{U} \circ F(\boldsymbol{p})$ allows to check the genetic algorithms parameter sensitivity and the possible correlation between parameters. We introduced a new algorithmic step applied to generation modification step that performs data transformation based on PCA and inverse PCA noise reduction operation the set of parameters used for GA.

4 Evolutionary schema performance improvement on an NSGA-II example

NSGA-II [14] is considered to be one of the most common GAs and it features fast non-dominance sorting procedure of population and preservation of a good convergence rate to the optimal Pareto set. The spread of best individuals is obtained through a diversity preservation operation called crowding distance and non-dominated ranking procedure.

NSGA-III [15] as a evolution of NSGA-II has more specific algorithm schema based on reference point's selection procedure. Suppose that we are using a decent set of GA suitable for black-box MOP with a computationally-expensive fitness function (for example constrained or unconstrained NSGA-II or NSGA-III algorithms). An important issue is the lack of additional operators that could provide higher convergence to the set of global optimum points. Adding specific operator that can be regarded as a denoising factor for faster approximation and convergence to the true Pareto front consisting of ideal individuals, we can apply orthogonal transformation to be able to discover strong patterns in data set.

5 Results and benchmarking

The DTLZ problems [6] are a set of numerical MOP benchmarks that are used for comparing and validating results from different GA algorithms. We present results of the DTLZ benchmarks [6] for NSGA-II and NSGA-II with PCA noise cleanup operator. We recognized that currently NSGA-III is outperforming NSGA-II but here results are provided as a proof of concept. On Figure 1 – Figure 7 are presented the parameter distribution (mean and standard deviation values) and cost function values behavior depending on used algorithms. Comparing Figure 6 and Figure 7 where was applied noise-removing procedure

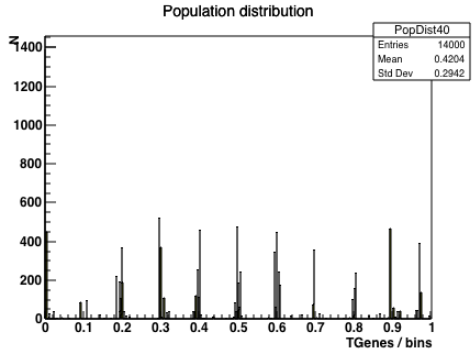

Fig. 1. Population distribution on 40th generation - NSGA-II - DTLZ3

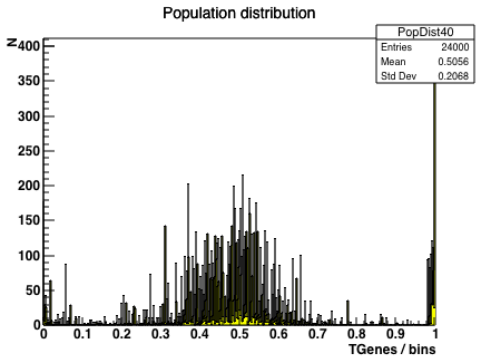

Fig. 2. Population distribution on 40th generation - NSGA-II - DTLZ4

and Figure 3 and Figure 4 where was not, we can observe faster convergence to the ideal values of the parameters in the first case. Figure 7 and Figure 8 show the first approach to Pareto front in combination with correct set of parameters. In Figure 9, we represent results of the first simple transport particle simulations, with promising results of benefit already 20 % of run time of simulation comparing to initial generations.

In next table you can find behavior of system on 40th population (we are using variance to define how fast we are providing noise cleanup procedure):

Benchmark	Old alg.	New alg.	PF conv.
DTLZ1	0.3032	0.3034	Not converged
DTLZ2	0.1624	0.1344	Converged for both
DTLZ3	0.2942	0.1357	Converged for NSGAIIPCA
DTLZ4	0.2068	0.1357	Converged for NSGAIIPCA
DTLZ5	0.3076	0.2048	Converged for NSGAIIPCA

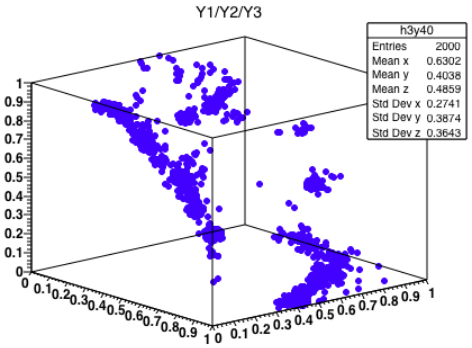

Fig. 3. Pareto Front on 40th generation of NSGA-II - DTLZ4

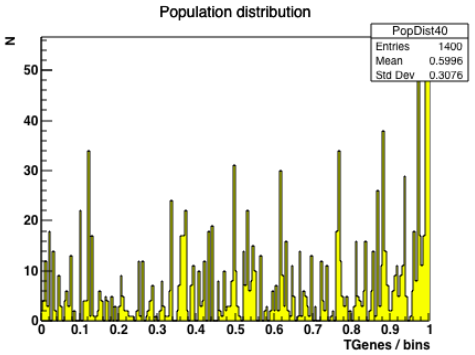

Fig. 4. Pareto Front on 40th generation of NSGA-II - DTLZ5

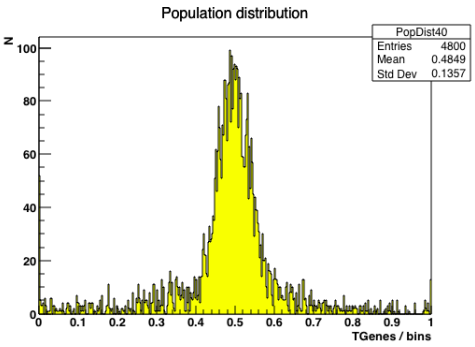

Fig. 5. Population distribution on 40th generation of NSGA-II with preprocessing of data - DTLZ4

The next steps of our work will be to agree our concept with the existence of fixed points in dynamical systems, to re-evaluate a possible speedup comparing to other algorithms together with the "black-box" benchmarks [17] and port

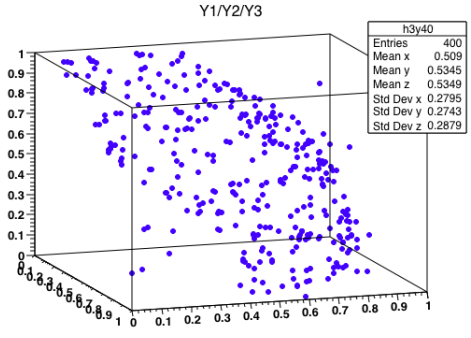

Fig. 6. Population distribution on 40th generation - NSGA-II with preprocessing of data - DTLZ4

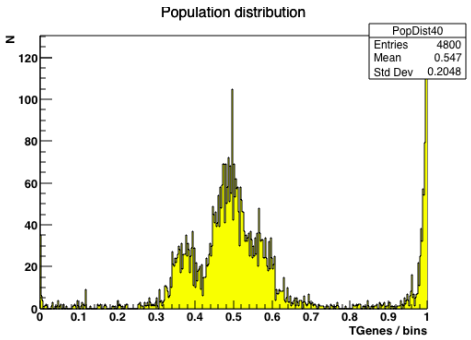

Fig. 7. Population distribution on 40th generation - NSGA-II with preprocessing of dataI - DTLZ5

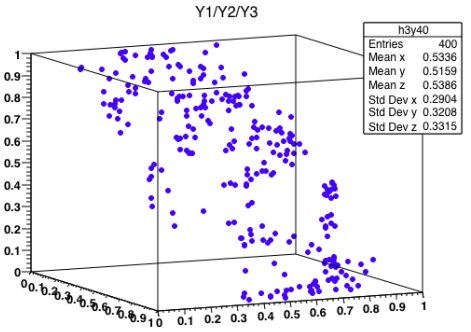

Fig. 8. Pareto Front on 40th generation of NSGA-II with preprocessing of data - DTLZ5

a new algorithm as a part of the optimization framework for GeantV particle transport simulations code.

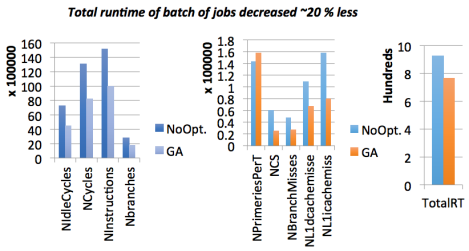

Fig. 9. Parameter benefit after tuning simplified particle transport simulation application with NSGA-II and preprocessing of data

6 Conclusion

We have explored the possibility to combine stochastic optimization methods and unsupervised machine learning to obtain a powerful combination that speedup existing GA algorithms. The combination of genetic algorithm and principal component analysis helps to get better convergence rate performing data set noise cleanup procedure based on orthonormal transformations. Performance optimization based on of tuning parameters of particle transport simulations using the approach outlined here will help to free up computational resources, change the algorithmic approaches used for job scheduling and provide energy-efficient solution granting the same work proficiency.

References

1. GeantV Collaboration., http://geant.web.cern.ch
2. Geant4 Collaboration., http://geant4.web.cern.ch/geant4/
3. Amadio, G and Apostolakis, J and Bandieramonte, M and Bhattacharyya, A and Bianchini, C and Brun, R and Canal, Ph and Carminati, F and Duhem, L and Elvira, D and others,*The GeantV project: preparing the future of simulation*, Journal of Physics: Conference Series, v.664, n.7, 072006, (2015).
4. Apostolakis, J and Bandieramonte, M and Bitzes, G and Brun, R and Canal, P and Carminati, F and Cosmo, G and Licht, J C De Fine and Duhem, L and Elvira, V D and et al., *Towards a high performance geometry library for particle-detector simulations*, J. Phys.: Conf. Ser. Journal of Physics: Conference Series, v.608, 012023 (2015) DOI:10.1088/1742-6596/608/1/012023,
5. Apostolakis, J and Bandieramonte, M and Bitzes, G and Brun, R and Canal, P and Carminati, F and Licht, JC De Fine and Duhem, L and Elvira, VD and Gheata, A and others *Adaptive track scheduling to optimize concurrency and vectorization in GeantV*, Journal of Physics: Conference Series, v.608, n.1, p.012003, (2015),
6. K. Deb and L. Thiele and M. Laumanns and E. Zitzler, Scalable Test Problems for Evolutionary Multi-Objective Optimization, Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, Springer, 2005
7. Vose, Michael D., The Simple Genetic Algorithm: Foundations and Theory, MIT Press, Cambridge, USA, 1999, 251p.

8. J. E. Rowe. Genetic algorithm theory, Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO12, pages 917-940, New York, NY, USA, 2012. ACM.
9. G. Rudolph. Convergence properties of evolutionary algorithms. Kovac, Hamburg, 1997.
10. Florian Schmitt and Franz Rothlauf. On the Importance of the Second Largest Eigenvalue on the Convergence Rate of Genetic Algorithms, Proceedings of the 14th Symposium on Reliable Distributed Systems, 2001
11. J.Cadima and I.Jolliffe, "On Relationships between Uncentered and Column-centered Principal Component Analysis", Pak.J.Statist., 2009, Vol.25(4), p.473-503,
12. Paul Honeine, "An eigenanalysis of data centering in machine learning", preprint ArXiV ID: 1407.2904, 14p., 2014 (License: http://arxiv.org/licenses/nonexclusive-distrib/1.0/).
13. Baker, Kirk. "Singular value decomposition tutorial." The Ohio State University 24 (2005).
14. K. Deb and A. Pratap and S. Agarwal and T. Meyarivan,A fast and elitist multi-objective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 2002, Vol.6, p.182-197, Doi:10.1109/4235.996017
15. Seada, Haitham, and Kalyanmoy Deb. "U-NSGA-III: A unified evolutionary algorithm for single, multiple, and many-objective optimization." COIN Report 2014022.
16. Annadurai, S. Fundamentals of digital image processing. Pearson Education India, 2007.
17. N. Hansen, A. Auger, O. Mersmann, T. Tuar, and D. Brockhoff (2016). COCO: A platform for Comparing Continuous Optimizers in a Black-Box Setting. ArXiv e-prints, arXiv:1603.08785.