Journal of Computer Science

Original Research Paper

Performance Optimization of Physics Simulations Through Genetic Algorithms

[1,2]Oksana Shadura, [2]Federico Carminati and [1]Anatoliy Petrenko

[1]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kiev, Ukraine
[2]CERN, Geneve, Switzerland

Article history
Received: 21-09-2018
Revised: 08-12-2018
Accepted: 7-01-2019

Corresponding Author:
Oksana Shadura
CERN, Geneve, Switzerland
Email: oksana.shadura@cern.ch
ksu.shadura@gmail.com

Abstract: The GeantV R&D approach is revisiting the standard particle transport simulation approach to be able to benefit from "Single Instruction, Multiple Data" (SIMD) computational architectures or extremely parallel systems like coprocessors and GPUs. The goal of this work is to develop a mechanism for optimizing the programs used for High-Energy Physics (HEP) particle transport simulations using a "black-box" optimization approach. Taking in account that genetic algorithms are among the most widely used "black-box" optimization methods, we analyzed a simplified model that allows precise mathematical definition and description of the genetic algorithm. The work done in this article is focused on the studies of evolutionary algorithms and particularly on stochastic optimization algorithms and unsupervised machine learning methods for the optimization of the parameters of the GeantV applications.

Keywords: Genetic Algorithms, Multi-Objective Optimization, Black-Box Optimization, Simulation of Transport of Particles

Introduction

In the past years, simulation toolkit for transport particles through matter Geant4 (Agostinelli *et al*., 2003) has been the main application for full detector simulation in High Energy Physics (HEP). The complex design of Geant4, with deep class structure hierarchies and calling stacks, makes it less than optimally efficient when running on the latest computer architectures. In order to resolve this situation, in 2013 the simulation project GeantV (Amadio *et al*., 2016) was started. The connected R&D work included the implementation of instruction level parallelism for increased performance by leveraging Single Instruction Multiple Data (SIMD) code and improving data locality. For these reasons the new software applications was designed to be able to run on parallel as well as SIMD architectures and efficiently use system caches (Shadura and Carminati, 2016). GeantV expects to achieve significant improvement for event throughput to be ready for the larger simulated data samples needed by the High-Luminosity Large Hardron Collider (LHC).

General idea of work is about tuning of performance of complex High Energy Physics simulations such as GeantV. It depends on a large number of parameters that are complex to tune by hand. Stochastic optimization indicates a process of function minimization or maximization employing algorithms of random optimization. Even assuming that we use a set of efficient genetic algorithms for the optimization of a "black-box" Multi-Objective Problem (MOP) with a computationally expensive fitness function, we still face the problem of finding operators providing efficient convergence to the optimal Pareto Front. The genetic algorithms selected for this work are the Non-Dominant Sorting Genetic Algorithms - NSGA-II (Deb *et al*., 2002) and NSGA-III (Deb and Jain, 2014).

During research was identified a set of indicators that are relevant for the evaluation of GeantV performance. Our selection is: Run time, peak memory consumption during application run and other model-specific dependent indicators, such as the number of instructions or other performance measurements. Introducing extra operators for the genetic algorithm is an attempt to remove noise from the dataset and introduce a procedure to improve the genetic algorithm convergence to the "true Pareto front." Optimal individuals selected among the set of the interesting parameters populate this front. This set of parameters is used to apply an orthogonal transformation and maximize variance to discover strong patterns in data.

The objective of this work is to inquire whether unsupervised machine learning methods are useful as

injected "operators" in genetic algorithms and if, thanks to them, it is possible to speed up the process of finding a Pareto front.

Materials and Methods

During the work on optimization of computing performance of HEP simulation applications, we consider particle transport simulation as a heuristic parametric model with expensive evaluations in terms of consumption of computing resources and a complex fitness landscape that will be optimized by use of stochastic search algorithms.

As a result, this article is part of a research work on the optimization of high-energy physics simulations interpreted as multi-objective computing applications performance optimization problems. To test our ideas on a simpler application, we will try to optimize the performance of the Deb-Thiele-Laumanns-Zitzler (DTLZ) benchmarks (Deb *et al*., 2005) and of a simplified simulation benchmark. The goal is to improve convergence to the "true Pareto front" while using a combination of genetic algorithms and methods from unsupervised machine learning.

Considering that genetic algorithms are among the most widely used evolutionary algorithms, we tried to analyze a simplified model that allows precise mathematical definition and description of the genetic algorithm, i.e., the "Simple model of Genetic Algorithm" (SGA) (Vose, 1999) used for the investigation of the typical evolutionary system, describing the genetic algorithm as an example of dynamical system with a precise mathematical definitions.

In this approach, genetic algorithm is defined by Markov chains, which are represented by stochastic models with a sets of sequential events, where the probability of each new event depends on the state of the previous event. The states of the evolutionary system are populations of genetic individuals, while transitions between the various states are handled by a set of genetic operators: Selection, crossover and mutation (Rowe, 2007). These operators are operating in the space of:

$$\Lambda = \left(p_1, p_2, ..., p_m\right)^t \tag{1}$$

of the population's vectors.

The mutation operation is an interesting evolutionary-inspired mechanism that expresses the connection of the Markov chain (through fully connected matrix)and ensures that it has an exclusive equilibrium distribution over populations, which means that the Markov chain converges to the stationary distribution.

In case of Markov chains, the probability of generation of a particular population will depend only on the previous genetic generation and some extraneous affecting factors.

A Markov chain is defined by a transition matrix $T_{\vec{q},\vec{p}}$ from the population $\vec{p}$ to $\vec{q}$, where the vector describing the population is defined as:

$$\left\{\vec{p} = \left(p_1, p_2, \ldots, p_m\right)^t,\ \ 0 \le p_\alpha \le 1,\ \ \sum_{\alpha=1}^{m} p_\alpha = 1\right\}, \tag{2}$$

and the vector component p_α is the probability of the appearance of the α-th individual in the genetic population. In the sample space $\Omega = \{1,2,3..N\}$ we have a population of m different types of individuals. We will provide a full description of the genetic operators in terms of dynamic system in the next section together with the implementation of a new unsupervised machine learning genetic operator.

Here, we introduce the basic definitions used in the theory of genetic algorithms and dynamic systems. A dynamic system is a model in which a function describes the time dependence of the evolution of a point in a geometrical space. Dynamic systems can describe the evolution of individuals in the space of finite dimension, members of a populations of fixed size m, where m is number of measurements during the experiment. While defining genetic algorithms as a discrete dynamical system, we can discover mathematical entities such as fixed points, which are elements of the function's domain that are mapped to themselves by the function. These objects are not only relevant in the investigation of simple genetic algorithms, but in general for all optimization problems (Shadura and Carminati, 2016).

The investigation of the convergence properties of the SGA as evolution schema was explored in (Rudolph, 1997). Schmitt and Rothlauf (2001) it has been shown that the convergence rate of the genetic algorithm is determined by the second largest eigenvalue of the transition matrix $T_{\vec{q},\vec{p}}$.

For Markov chains, it is very complex to determine the process of evolution along an appropriate direction leading to a faster convergence to equilibrium. Principal Component Analysis (PCA) could be defined as a procedure that is able to inspect the genetic algorithm population's sensitivity and the correlations between parameters of the input data matrix and the generated population. For this reason, we want to introduce a PCA operator providing such functionality, using inverse PCA noise reduction.

Theory of PCA and UPCA-based Operator in GA

Before embarking in the optimization of the GeantV simulation, we need to identify a list of optimization parameters significant for the computing performance. These are user defined parameters such as the number of

threads used in simulation or the size of the vectors of transported particles etc. These can be described via a data matrix of size $m \times n$:

$$\hat{X} = \{X_{\alpha,i}\} = \{(\vec{x}_\alpha)_i\} = \{\vec{x}_\alpha\}, \quad (3)$$

where, $\vec{x}_\alpha = \{(\vec{x}_\alpha)_i\}(1 \le \alpha \le m, 1 \le i \le n)$ is the α-th individual of the population. In this matrix, the index i indicates the GeantV tuning parameters (i = 1,…,n) and the index α indicates the number of measurements of the fitness function for the given set of performance indicators (α = 1,…,m for m measurements).Comparing the definition in terms of GA, the matrix is described through m -samples of data in a n-dimensional space, where n is the size of the individual and m is the number of individuals in the generation.

Principle Component Analysis (PCA) is used for data analysis via covariance matrix to diminish complex data set to a lower dimensionality, applying PCA to a centered data matrix.

In this section, we apply PCA to an uncentered data matrix. We will reference to it as “Uncentred PCA” (UPCA) and we will prove that it is particularly useful in the case of transformations on constrained genetic algorithms populations, as it is the case for a set of GeantV parameters. A discussion of the relations between centered and uncentered data matrix can be found in the work of Cadima and Jolliffe (2009).

As ¨mentioned in the previous section, m-samples of data from an n-dimensional space are elements of the data matrix $\hat{X}$ of size $m \times n$ (m is the number of individuals in the generation and n is the size of the individual, which is equivalent to the dimension of a vector of genes $\vec{x} = \{x_i\}(1 \le i \le n)$.

Using the uncentered data matrix $\hat{X}$ size $m \times n$, defined in (3) we can write the matrix of non-central second moments for UPCA:

$$\hat{T} = \frac{1}{m}\hat{X}^t \cdot \hat{X}_t. \quad (4)$$

if $\vec{w}_j$ are eigenvectors of the matrix $\hat{T}$ with the corresponding eigenvalues t_j:

$$\hat{T} \cdot \vec{w}_j = t_j \vec{w}_j, 1 \le j \le n. \quad (5)$$

They satisfy the orthonormality condition:

$$\vec{w}_i^t \cdot \vec{w}_j = \delta_{i,j}, 1 \le i, j \le n. \quad (6)$$

Then:

$$\vec{w}_j^t \cdot \hat{T} \cdot \vec{w}_j = t_j, \quad (7)$$

and for the matrix $W_{i,j} = \{\vec{w}_j\} = (w_i)_j$ we have the orthogonality condition:

$$\hat{W}^t \cdot \hat{W} = \hat{I}. \quad (8)$$

From (7) we have:

$$\hat{W}^t \cdot \hat{T} \cdot \hat{W} = \hat{\Delta}, \Delta_{i,j} = t_i \delta_{i,j}. \quad (9)$$

$\vec{\theta}_j = \{(\theta_j)_\alpha\} = \{\vec{x}_\alpha^t \cdot \vec{w}_j\}$ is j-th uncentered principal component, here α = 1,…,m. If we define $\Theta_{\alpha,j} = \{\vec{\theta}_j\} = \{(\theta_j)_\alpha\}$ then:

$$\Theta_{\alpha,j} = X_{\alpha,i} W_{i,j}, 1 \le \alpha \le m, \quad (10)$$

which from (8) and (9) satisfies the condition:

$$\Theta_{i,\alpha}^t \Theta_{\alpha,j} = m\Delta_{i,j} = m t_i \delta_{i,j}. \quad (11)$$

We do not have a clear description of the relationship between the matrix eigenvalues t_j and the variance of the j-th uncentered principal component $(\sigma_{\theta,j})^2$ as for the PCA case. In our case this property is not fundamental for using the PCA algorithm in the GA. We apply instead the “eigenvalue control parameter” approximation (Cadima and Jolliffe, 2009). In simple terms, it is the ability to use the PCA algorithm for the Singular Value Decomposition (SVD) representation of the data matrix $\hat{X}$ (Amadio *et al*., 2017).

If we define the matrix $\tilde{\Theta}_{\alpha,i}$ as:

$$\Theta_{\alpha,j} = \sqrt{m}\tilde{\Theta}_{\alpha,i}\Delta_{i,j}^{1/2}, \Delta_{i,j}^{1/2} = t_i^{1/2}\delta_{i,j}. \quad (12)$$

from (11) and (12) we obtain:

$$\tilde{\Theta}_{i,\alpha}^t \tilde{\Theta}_{\alpha,j} = \delta_{i,j}.$$

Using (12), (11) and (7) we see that $\tilde{\Theta}_{\alpha,j} = \{\tilde{\vec{\theta}}_j\}$ is the matrix of the eigenvectors $(\tilde{\theta}_j)_\alpha$ of the matrix $\tilde{K} = \hat{X} \cdot \hat{X}^t$ of size $m \times m$:

$$\tilde{K}_{\alpha,\beta}(\tilde{\theta}_j)_\beta = X_{\alpha,k} X_{k,\beta}^t (\tilde{\theta}_j)_\beta = t_j (\tilde{\theta}_j)_\alpha.$$

From (10) we can derive the representation for the data matrix $\hat{X}$:

$$X_{\alpha,i} = \Theta_{\alpha,j} W_{j,i}^t, \quad (13)$$

and the SVD representation of the data matrix is then obtained:

$$X_{\alpha,i} = \sqrt{m}\tilde{\Theta}_{\alpha,k}\Delta_{k,j}^{1/2}W_{j,i}^{t}. \tag{14}$$

In the case when the matrix of non-central second moments $\hat{T}$ has the (n-q) smallest eigenvalues such that $t_j << 1$, $q+1 \le j \le n$, we can use the "eigenvalue control parameter" approximation of the data matrix and approximate it by the output data matrix $\tilde{X}_{\alpha,j}$ of rank q:

$$\begin{aligned}\tilde{X}_{\alpha,i} &= \sqrt{m}\tilde{\Theta}_{\alpha,k}\tilde{\Delta}_{k,j}^{1/2}W_{j,i}^{t} \\ &= \sqrt{m}\left(t_1^{1/2}\tilde{\Theta}_{\alpha,1}W_{1,i}^{t} + ... + t_q^{1/2}\tilde{\Theta}_{\alpha,q}W_{q,i}^{t}\right),\end{aligned} \tag{15}$$

where, the eigenvalue matrix $\tilde{\Delta}_{k,j}$ has rank q($t_{q+1} = t_{q+2} = \cdots = t_n = 0$). This approximation is the analog of the Hotelling transformation (Hotelling, 1936).

We can estimate the mean square error η_q for the approximation (15):

$$\begin{aligned}\eta_q &= \frac{1}{mn}\sum_{\alpha=1}^{m}\sum_{i=1}^{n}\left(X_{\alpha,i} - \tilde{X}_{\alpha,i}\right)^2 \\ &= \frac{1}{mn}\sum_{\alpha=1}^{m}\sum_{i=1}^{n}\left(\sqrt{m}\sum_{k=q+1}^{n}\sqrt{t_k}\tilde{\Theta}_{\alpha,k}W_{k,i}^{t}\right)^2 = \frac{1}{n}\sum_{k=q+1}^{n}t_k\end{aligned}$$

The minimum error is achieved if the matrix $\hat{T}$ has the (n-q) smallest eigenvalues such that $t_j << 1$, $q+1 \le j \le n$.

At this point having already defined the UPCA operator, we need to describe the genetic algorithm model for the injection of the operator we introduced earlier. Recalling ideas presented in (Vose, 1999) and described in the previous section, we will develop the optimization process (Amadio *et al*., 2017).

The GA operators are operating in the space $\Lambda = (p_1, p_2, \cdots, p_m)^t$ of the population's vectors.

The genetic operator $G_\alpha(\vec{p})$ is defined by the probability of generating an individual α (starting from the previous population $\vec{p}$). We define a map G: $\Lambda \to \Lambda$, where $G(\vec{p}) = \prod_{\alpha\in\Omega} G_\alpha(\vec{p})$ and $G(\vec{p}) \in \Lambda$ is an heuristic function and Ω is the sample space. The map G is equivalent to the composition of selection, mutation and crossover maps. The genetic selection operator:

$$\begin{aligned}&F: \Lambda \to \Lambda, \\ &F(\vec{p}) = \prod_{\alpha\in\Omega} F_\alpha(\vec{p})\end{aligned}$$

defines the probability of a genetic individual of type α (when the process of selection is applied to $\vec{p} \in \Lambda$). A selection operator selects genetic individuals from the current population via the performance indicators, $\vec{f} = \{f_\alpha\} \in R^m$, $f_\alpha = f(\alpha)$, $\alpha \in \Omega$:

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

where, $diag(\vec{f})$ is a diagonal matrix.

The mutation operator

$$U: \Lambda \to \Lambda,$$

is an $m \times m$ matrix with the (α,β)-th entry $u_{\alpha,\beta} > 0$ for all α, β and $u_{\alpha,\beta}$ indicates the probability of individual $\beta \in \Omega$ mutating into $\alpha \in \Omega$. $(U\cdot\vec{p})_\alpha$ is the probability for an individual (type α) to appear after the mutation process is applied to the population $\vec{p}$ (Amadio *et al*., 2017).

The crossover operator is defined as:

$$\begin{aligned}&C: \Lambda \to \Lambda, \\ &C(\vec{p}) = \left(\vec{p}^{t}\cdot\hat{C}_1\cdot\vec{p}, ..., \vec{p}^{t}\cdot\hat{C}_m\cdot\vec{p}\right)\end{aligned}$$

where, $\hat{C}_1, ..., \hat{C}_m$ is a sequence of symmetric non-negative $N \times N$ real-valued matrices. $C_\alpha(\vec{p})$ expresses probability that an individual α appears after the crossover process is applied to the population $\vec{p}$.

$$\begin{aligned}&G: \Lambda \to \Lambda, \\ &G(\vec{p}) = C \circ U \circ F(\vec{p}).\end{aligned} \tag{16}$$

Having defined G, we can express the stochastic transition matrix, based on the probability of transformation of the population $\vec{p}$ into $\vec{q}$:

$$T_{\vec{q},\vec{p}} = \bar{m}!\prod_{\alpha\in\Omega}\frac{\left(G_\alpha(\vec{p})\right)^{\bar{m}q_\alpha}}{(\bar{m}q_\alpha)!} \tag{17}$$

where, $G_\alpha(\vec{p})$ define probability of the appearance of individual α in the next generation and $\bar{m}q_\alpha$ is the number of individuals α in the population $\vec{q}$ with size $\bar{m}$.

We can improve the convergence rate of genetic algorithms (how fast genetic algorithms are converging to the optimum per generation) by adding a new genetic operator P based on UPCA procedure. We will call this procedure "UPCA noise cleanup operator and we will introduce it in the genetic algorithm's map $G_P(\vec{p}) = P \circ C \circ U \circ F(\vec{p})$.

Since we have a similar setup as it is done in the SGA case (Schmitt and Rothlauf, 2001), we can observe that a genetic algorithm convergence rate is defined by the eigenvalues that follow the highest one. That's why we had to apply the proposed operator P on the earlier eigenvalues.

An interesting observation is that the eigenvector with the largest eigenvalues defines a subspace of solutions for multi-objective problem populating Pareto front.

In the next section we will assess whether using an iterative procedure for an uncentered data matrix allows us to reach the subspace of optimal solutions faster than with a centered matrix. We will perform this test for the standard benchmarking problems used to validate performance of genetic algorithms.

Testing of the UPCA-Improved Genetic Algorithm on NSGA-II and DTLZ Benchmarks

We use NSGA-II (Allison *et al*., 2006) as a part of our testing setup since it is one of the most popular genetic algorithms. It is based on fast non-dominance sorting for generated populations and provides an impressive convergence rate to the optimal Pareto set.

The genetic algorithm population is initialized and it is afterwards sorted into a group of fronts using the non-domination principle. The first front is treated as a completely non-dominant set and the individuals of the first front are dominating the individuals of the second front only and so on until the last front. For each individual, in each front, a rank value is assigned based on the front to which they belong to. Individuals in the first front are given a rank value of one, while individuals in the second front are assigned a rank value of two and so on. In addition to the rank value, an additional parameter called crowding distance is calculated for each individual (Shadura, 2017). The crowding distance is a measure of how close an individual is to its neighbors. Larger average value of crowding distances indicates significant population diversity. NSGA-III (Deb *et al*., 2005) has a different algorithm schema, built on the idea of improved reference points selection and using an already defined set of reference points to assure diversity for the population.

The main problem of genetic algorithms is an absence of operators that enhance the convergence to a global model optimum. We present an improved algorithm (Fig. 1) based on the combination of NSGA-II and the UPCA operator defined in the previous section.

The DTLZ benchmarks (Vose, 1999) are a set of numerical multi-objective problems, used for comparison/validation of GA algorithms. We present the comparison of NSGA-II without and with the UPCA noise cleanup operator applied to DTLZ.

In Fig. 2 and 3 we present the tuning parameter distribution together with the mean and standard deviation values with and without the UPCA operator. In Fig. 3 and 5 the PCA-based noise-cleanup procedure is used, while in Fig. 2 and 4 it is not. We can observe a significant improvement of the convergence speed to the ideal values of the parameters in the first case when we use the UPCA operator. Figure 5 shows the first images of an early Pareto front.

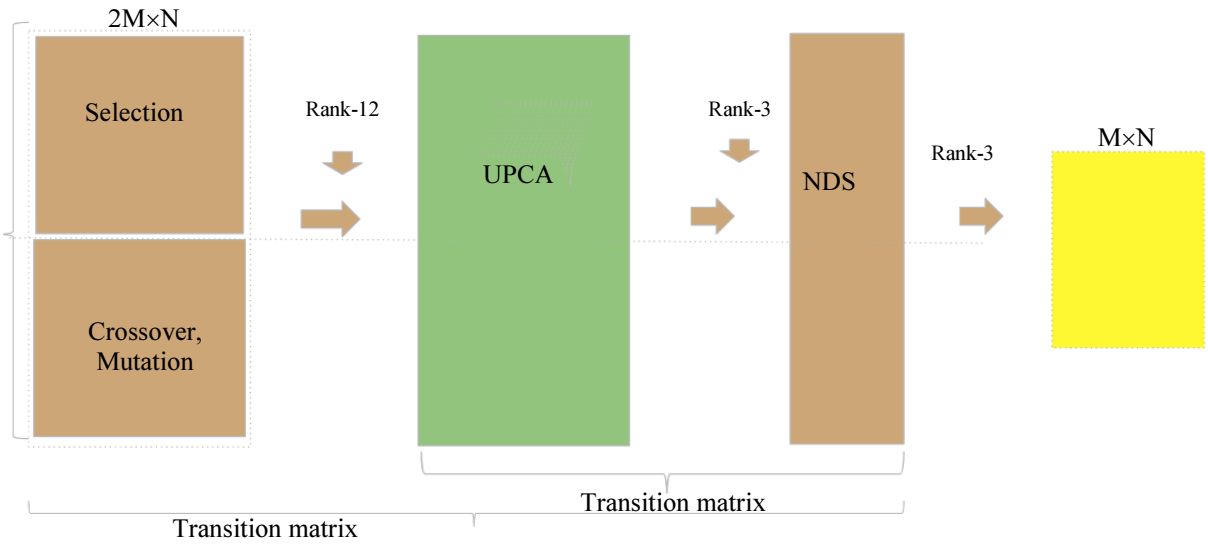

Fig. 1: Updated NSGA-II algorithm used in GeantV

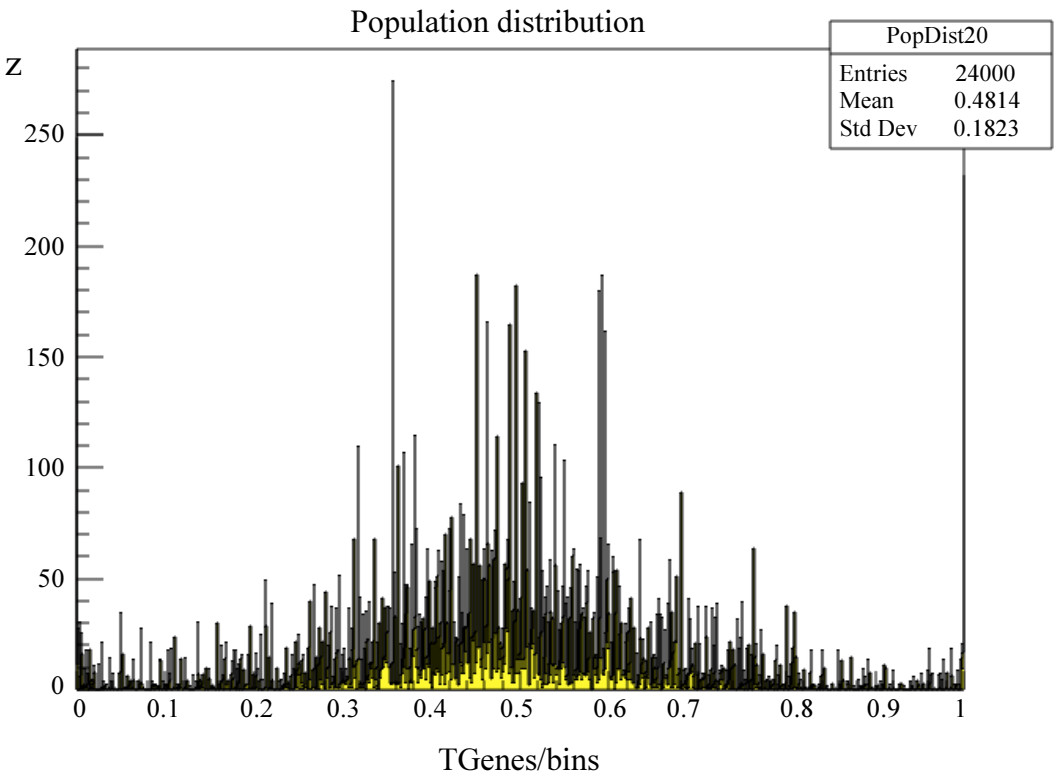

Fig. 2: Distribution of genetic population in the 20th generation of NSGA-II for DTLZ2 problem

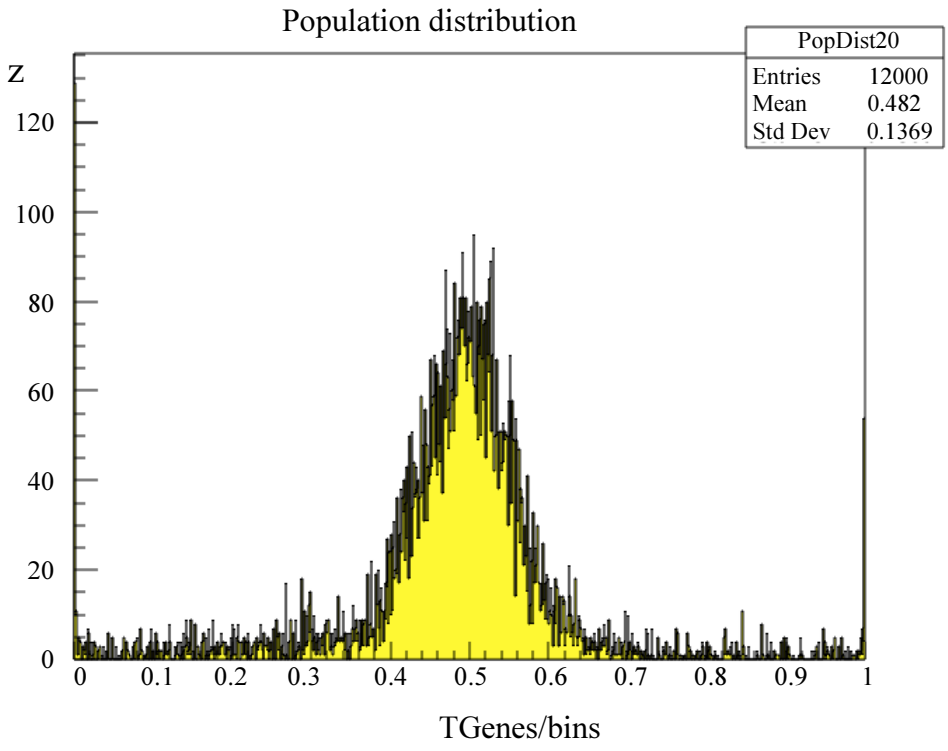

Fig. 3: Distribution of the genetic population at the 20th generation for NSGA-II with UPCA operator for DTLZ2 problem

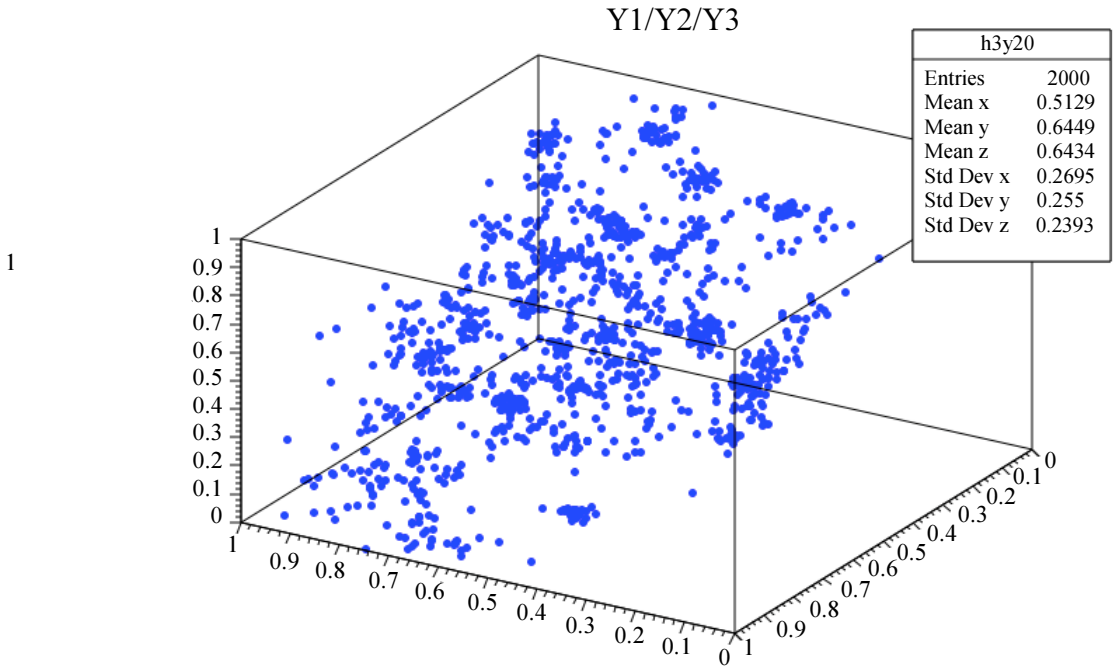

Fig. 4: Visualization of the Pareto Front for the 20th generation of NSGA-II for DTLZ2 problem

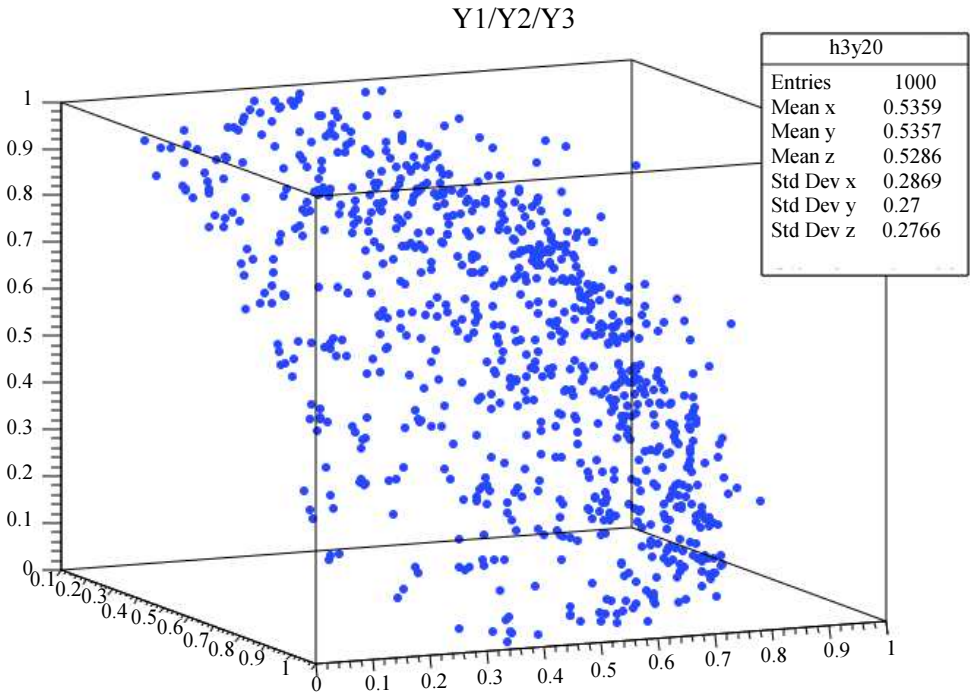

Fig. 5: Visualization of the Pareto Front at the 40th generation of NSGA-II with UPCA operator for DTLZ2 problem

Results with GeantV Simulation Code

An important step to evaluate if this kind of optimization is applicable to GeantV, is to have an efficient test suite, that can cover different use-cases, providing proper coverage of functionality and including interactions between the GeantV core and other sub modules: Physics, geometry and etc. We choose the

GeantV NoviceExampleN03 (https://gitlab.cern.ch/geant/GeantV.git) (Fig. 8.).

The first results for ExampleN03, which represent a sampling calorimeter with Pb absorber layers and liquid Ar detection gaps are shown in Fig. 6 and 7. All Electromagnetic (EM) processes and decay are simulated with specific production cuts for γ, e+, e- (used for shower studies).

The output of the simulation is the detector response that includes: Energy deposit and track length both in the detecting gaps and in the inert plates (absorber).

Implementing PCA operator in the genetic algorithm's library, used for tuning GeantV, helped to increase the speed of convergence of genetic algorithm by a factor two, allowing to quickly reach the "early" Pareto front. Early results prove that thanks to the stochastic tuning, the performance of the GeantV example on a Core i7-67000 machine, improves by 18%. On an Intel(R) Xeon(R) CPU E5-2695 the CPU time is reduced by 34%, providing a more stable memory consumption and reducing the overall run time of the batch by up to 27% (Shadura and Carminati, 2016).

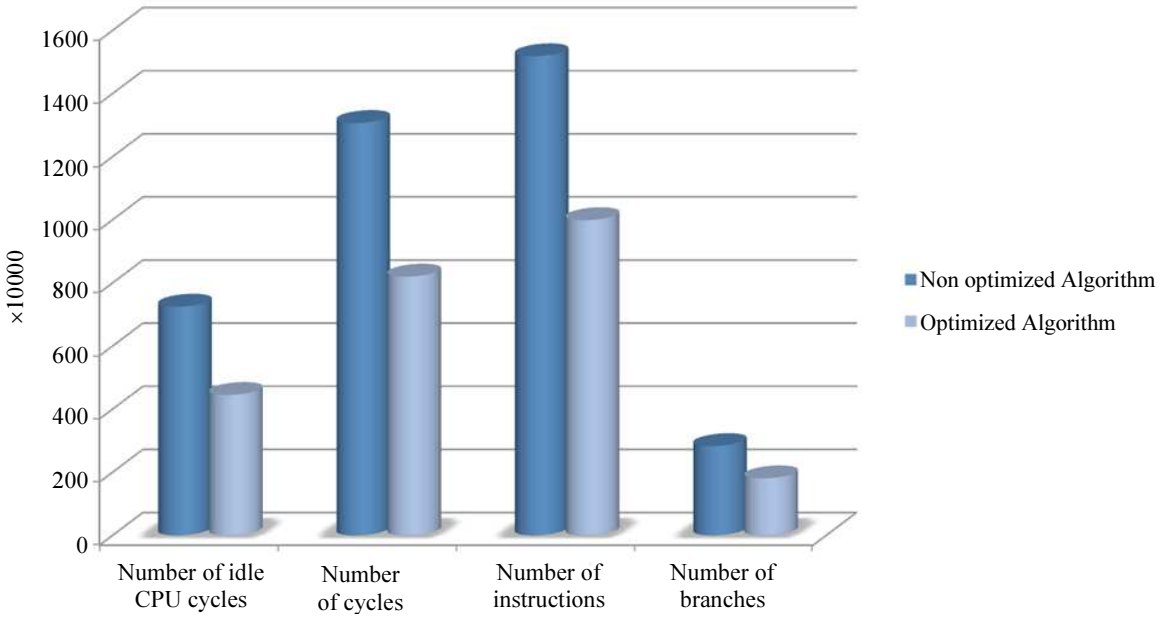

Fig. 6: Comparison of number of performance indicators for optimized and non optimized GeantV simulation

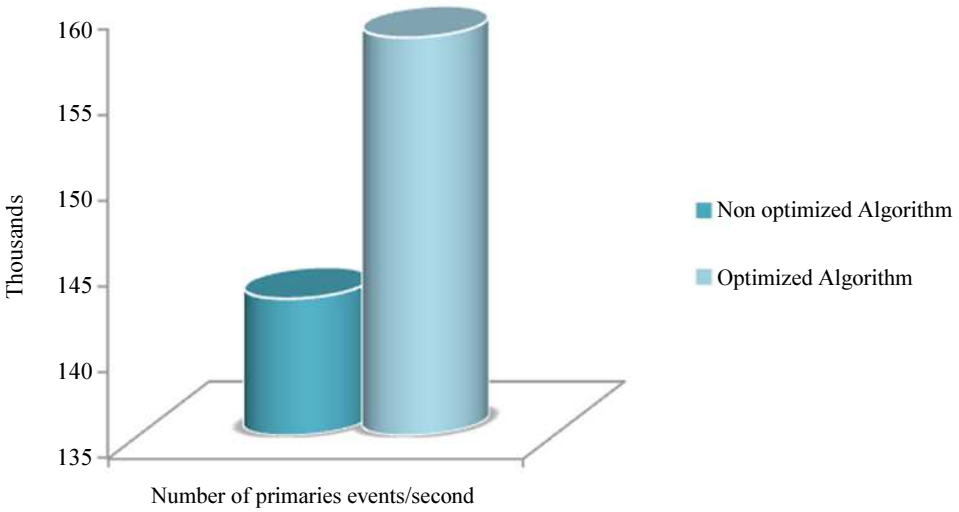

Fig. 7: Comparison of number primaries events for optimized and non optimized GeantV simulations

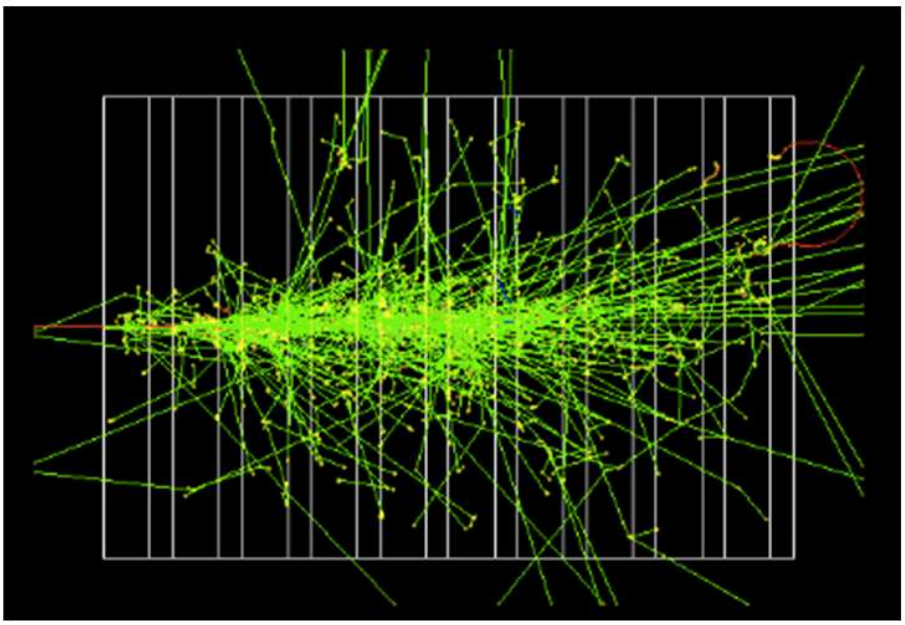

Fig. 8: Shower simulation example in the selected test case

We have of course carefully verified that the physics output does not change by applying stochastic tuning algorithms. Users will be able to collect data on the performance of their system and tune it while running simulation jobs.

In this study we demonstrated a proof of concept tuning of the performance of a complex multithreaded, highly vectorized application (GeantV) using evolutionary computation/genetic algorithms. This opens the possibility to use the same method to tune complex applications on supercomputers and High Performance Computing (HPC) clusters. It could also help to analyze the performance of compute-intensive jobs, running in heterogeneous environments, with the objective of reaching better scalability.

Conclusion

The work we presented is focused on the stochastic optimization of highly parallelized applications for simulation of radiation transport in complex detectors. During the work we introduced a new genetic operator able to speedup convergence of the algorithm to the true Pareto Front.

We have shown that the idea of merging the classical implementation of evolutionary algorithms with unsupervised machine learning methods can create a powerful symbiosis that can efficiently tune the performance of complex algorithms depending on a large number of correlated parameters and it could be easily implemented in any framework. The “mutation” of genetic algorithm coupled with principal component analysis helps us to get faster convergence rate using simple noise cleanup techniques based on an orthonormal transformation strategy. The performance enhancement reached with this tuning for particle transport simulations can significantly help to economize computing resources, improve scheduling strategies and provide energy economy for computing resources.

Acknowledgment

The authors would like to thank the CERN for their support in writing this article.

Author’s Contributions

All authors equally contributed in this work.

Ethics

This article is original and contains unpublished material. The authors confirm that are no conflict of interest involved.

References

Agostinelli, S., J. Allison, K.A. Amako, J. Apostolakis and H. Araujo *et al*., 2003. GEANT4 - a simulation toolkit. Nuclear Instruments Meth. Phys. Res., 506: 250-303. DOI: 10.1016/S0168-9002(03)01368-8

Allison, J., K. Amako, J. Apostolakis, H. Araujo and P. Arce Dubois *et al*., 2006. Geant4 developments and applications. IEEE Trans. Nuclear Sci., 53: 270-278. DOI: 10.1109/TNS.2006.869826

Amadio, G., A. Ananya, J. Apostolakis, A. Arora and M. Bandieramonte *et al*., 2016. GeantV: from CPU to accelerators. J. Phys., 762: 012019-012019.

Amadio, G., J. Apostolakis, M. Bandieramonte, S.P. Behera and R. Brun *et al*., 2017. Stochastic optimization of GeantV code by use of genetic algorithms. J. Phys.: Conf. Series, 898: 042026-042026. DOI: 10.1088/1742-6596/898/4/042026

Cadima, J. and I. Jolliffe, 2009. On relationships between uncentred and column-centred principal component analysis. Pak. J. Stat., 25: 473-503.

Deb, K. and H. Jain, 2014. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. IEEE Trans. Evolut. Comput., 18: 577-601. DOI: 10.1109/TEVC.2013.2281535

Deb, K., A. Pratap, S. Agarwal and T. Meyarivan, 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evolut. Comput., 6: 182-197. DOI: 10.1109/4235.996017

Deb, K., L. Thiele, M. Laumanns and E. Zitzler, 2005. Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Evolutionary Multiobjective Optimization, Abraham, A., L. Jain and R. Goldberg (Eds.), Springer, London, pp: 105-145.

Hotelling, H., 1936. Relations between two sets of variates. Biometrika, 28: 321-377. DOI: 10.2307/2333955

Rowe, J.E., 2007. Genetic algorithm theory. Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation, Jul. 07-11, ACM, London, UK, pp: 3585-3608. DOI: 10.1145/1274000.1274125

Rudolph, G., 1997. Convergence Properties of Evolutionary Algorithms. 1st Edn., Kovac, ISBN-0: 3860645544, pp: 286.

Schmitt, F. and F. Rothlauf, 2001. On the importance of the second largest eigenvalue on the convergence rate of genetic algorithms. Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, Jul. 07-11, Morgan Kaufmann Publishers Inc., pp: 559-564.

Shadura, O. and F. Carminati, 2016. Stochastic performance tuning of complex simulation applications using unsupervised machine learning. Proceedings of the IEEE Symposium Series on Computational Intelligence, Dec. 6-9, IEEE Xplore Press, Athens, Greece, pp: 1-8. DOI: 10.1109/SSCI.2016.7850200

Shadura, O., 2017. Performance tuning and monitoring using genetic optimization techniques. GeantV Spring Sprint.

Vose, M.D., 1999. The Simple Genetic Algorithm: Foundations and Theory. 1st Edn., MIT Press, Cambridge, ISBN-10: 026222058X, pp: 251.