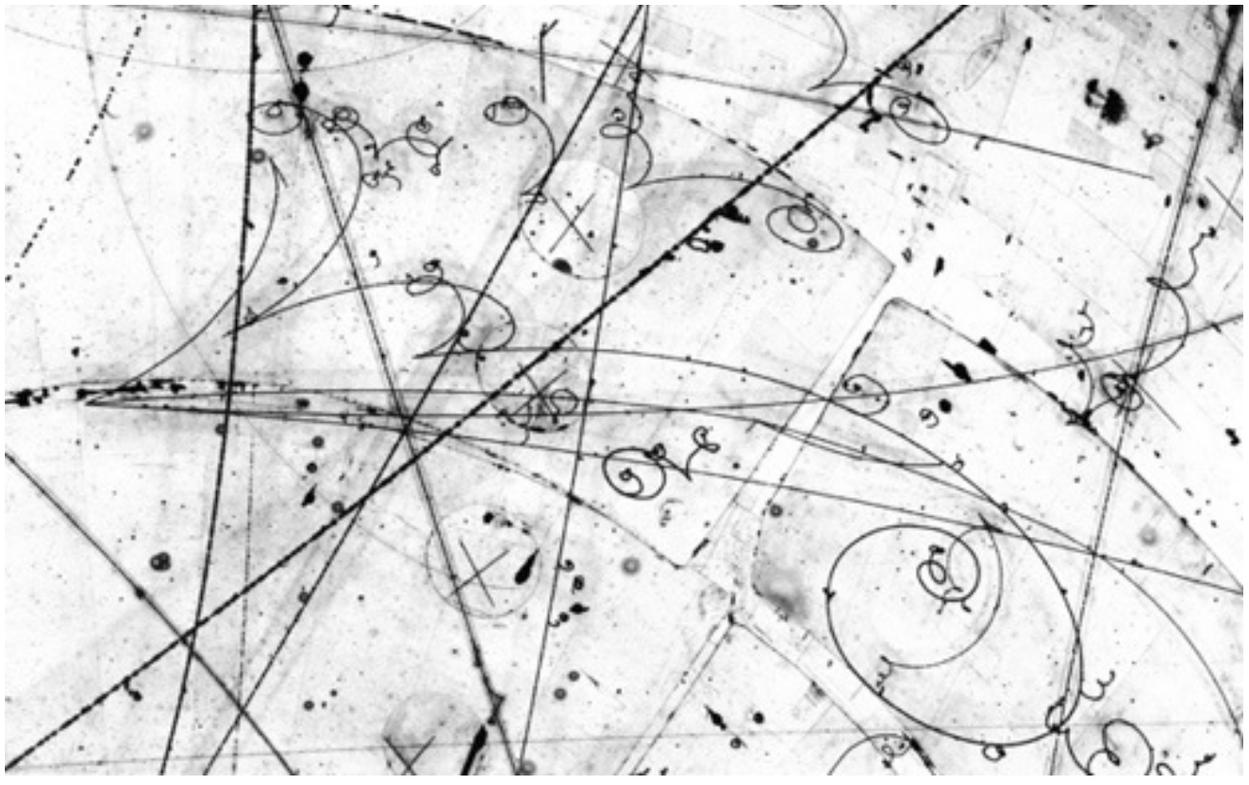
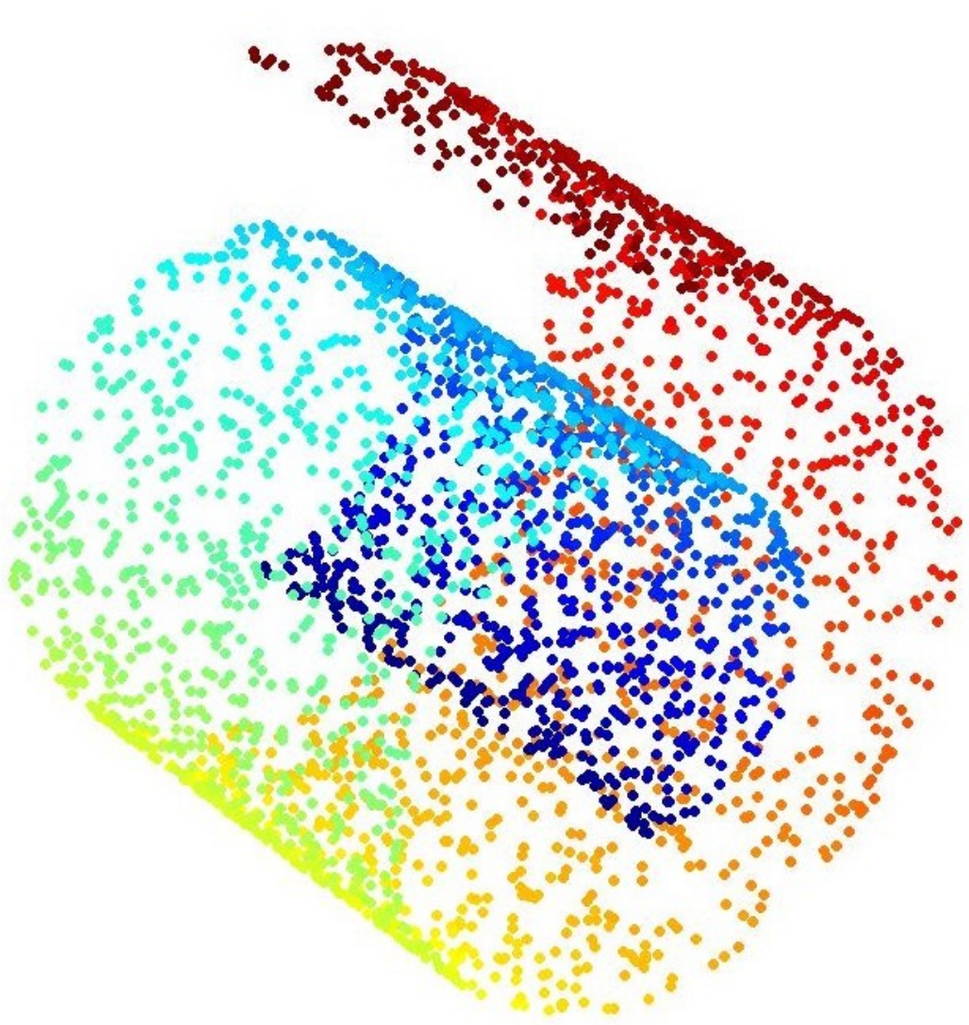
Unsupervised learning and GeantV

Oksana Shadura, CERN





Outline

- Introduction
- Unsupervised learning
- Stochastic optimisation
- GeantV optimisation

Introduction: Machine learning

- **Supervised learning** - generates a function that maps labeled inputs to desired outputs.

- **Semi-supervised learning** - combines both labeled and unlabeled examples to generate an appropriate function or classifier.

- **Unsupervised learning** - models a set of inputs: labeled examples are not available.

Unsupervised machine learning

Unsupervised learning

- In machine learning, unsupervised learning refers to the problem of trying to find hidden structure in **unlabeled data.** Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning and reinforcement learning.

- The goal of unsupervised learning is to create general systems that can be trained with **little data.**

- Unsupervised learning is closely related to the problem of density estimation in statistics.

- Methods:

  - **blind signal separation using feature extraction techniques for dimensionality reduction** (e.g., Principal component analysis, Independent component analysis, Non-negative matrix factorization, Singular value decomposition).

  - **clustering** (e.g., k-means, mixture models, k-nearest neighbors, hierarchical clustering)

  - **kernel methods** - *not covered in this presentation*

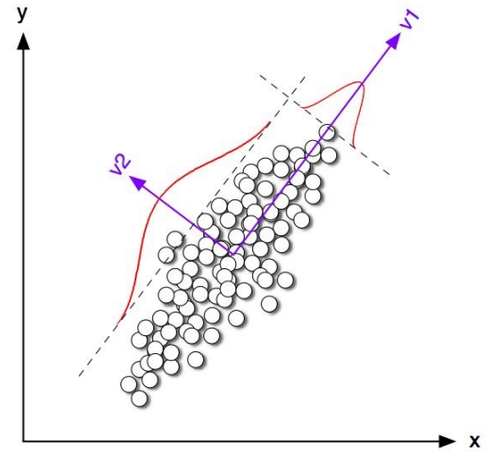
Principal Component Analysis(PCA), Singular Value Decomposition (SVD) and Uncentered PCA

Noise suppression with PCA

- Three uses of PCA:
  - pre-processing for empirical modelling;
  - data compression;
  - noise suppression.

The basic idea is that the variance captured by the **least important principal components** is noise which should be suppressed. Assuming that the variables bear a **linear relationship**, they will lie in a line (plane, hyperplane) and noise items will lift them away from the line. Dropping the last principal components means flattening the data in a geometric sense and (hopefully) eliminating some of the noise.

PCA



Centered data matrix of size $M \times n$

$$\hat{X}^{(c)} = \{X^{(c)}_{\alpha,i}\} = \{X^{(u)}_{\alpha,i} - \mu_i\}, \quad \mu_i = \frac{1}{M}\sum_{\alpha=1}^{M} X^{(u)}_{\alpha,i},$$

where α numerates individuals, $1 \leq \alpha \leq M$ and i – components of the gene $1 \leq i \leq n$.

The important objects for centered PCA are the matrix $U_{i,j} = (u_i)_j$ of the eigenvectors $\{\vec{u}_j\}$ and the matrix $\Lambda^{(c)}_{i,j} = \lambda^{(c)}_i \delta_{i,j}$ eigenvalues $\{\lambda^{(c)}_i\}$ of the covariance matrix

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

and

$$\hat{\Sigma} \cdot \vec{u}_j = \lambda^{(c)}_j \vec{u}_j, \quad 1 \leq j \leq n,$$

PCA

Let define the matrix $V^{(c)}_{\alpha,j} = \{(v^{(c)}_{\alpha})_j\}$ of the centered principal components $\{\vec{v}^{(c)}_j\}$

$$V^{(c)}_{\alpha,j} = X^{(c)}_{\alpha,i} U_{i,j}, \quad U^t_{i,i'} U_{i',j} = \delta_{i,j},$$

The variance of the j-th centered principal component

$$\mathrm{Var}(\vec{v}^{(c)}_j) = \vec{u}^t_j \cdot \hat{\Sigma} \cdot \vec{u}_j = \lambda^{(c)}_j$$

The first principal component $\vec{v}^{(c)}_1$ ($\lambda^{(c)}_1$ is the largest eigenvalue)

$$v^{(c)}_{\alpha,1} = V^{(c)}_{\alpha,1} = X^{(c)}_{\alpha,i} U_{i,1} = \vec{x}^{(c)\,t}_{\alpha} \cdot \vec{u}_1$$

The PCA representation is convenient for the approximation of the centered data matrix:

1) **the "principal component control parameter" approximation**

– if $\mathrm{Var}(\vec{v}^{(c)}_k) \ll 1$ for $k = m+1, m+2, ..., n$ we can put $\vec{v}^{(c)}_{m+1} = \vec{v}^{(c)}_{m+2} = ... = \vec{v}^{(c)}_n = 0$ in $V^{(c)}_{\alpha,j}$ and obtain new matrix $\tilde{V}^{(c)}_{\alpha,j}$ of the rang m.

2) **the "eigenvalue control parameter" approximation**

– if $\lambda^{(c)}_k \ll 1$ for $k = m+1, m+2, ..., n$ we can put $\lambda^{(c)}_{m+1} = \lambda^{(c)}_{m+2} = ... = \lambda^{(c)}_n = 0$ in $\Lambda^{(c)}_{i,j}$ and get new eigenvalue matrix $\tilde{\Lambda}^{(c)}_{i,j}$ of the rang m.

For the centered PCA these approximations coincide.

Applying the "principal component control parameter" approximation we get

$$\tilde{X}^{(c)}_{\alpha,i} = \tilde{V}^{(c)}_{\alpha,j} U^t_{j,i} = V^{(c)}_{\alpha,1} U^t_{1,i} + ... + V^{(c)}_{\alpha,m} U^t_{m,i},$$

KPCA

- **KPCA -** transform existing dataset to another high-dimensional space and then perform PCA on the data in that space.

- **Centralization**

  For a given data set, subtracting the mean for all the observation to achieve the centralized data in R^N.

- **Finding principal components**

  Compute the matrix $K_{ij} = (\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j))$ using kernel function, find eigenvectors α^k and eigenvalues $\lambda_k, \forall k = 1, 2, \ldots, N.$

- **Encoding training/testing data**

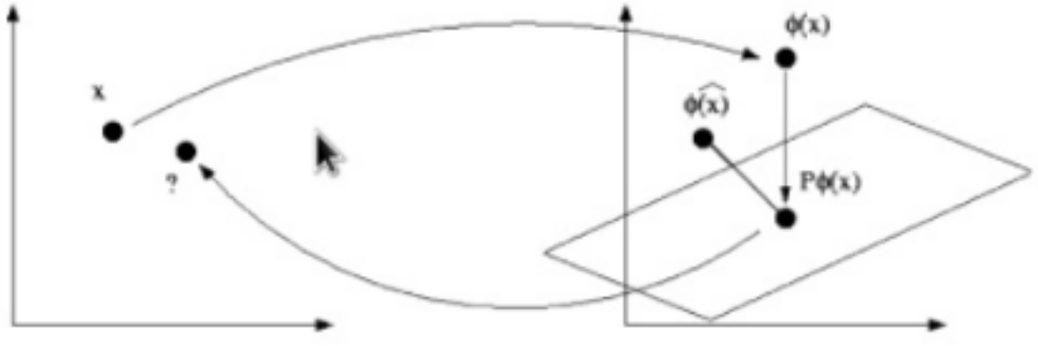
  $\mathring{\mathbf{x}} = \Sigma_{i=1}^{N} \alpha_i^k k(\mathbf{x}_i, \mathbf{x})$ where ***x*** is a vector that encodes the training data. This can be done since we calculated eigenvalues and eigenvectors.

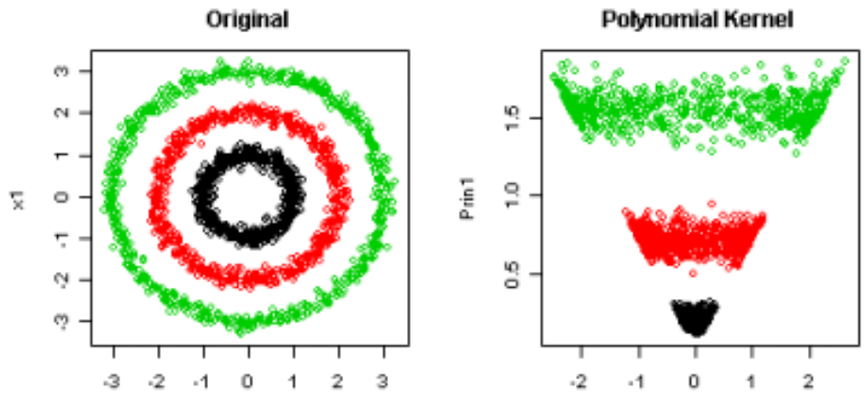
- **Reconstructing training data**

  The operation **cannot be done** because eigenvectors do not have a pre-images in the original dimension.

- **Reconstructing test data point**

  The operation **cannot be done** because eigenvectors do not have a pre-images in the original dimension.





Uncentered PCA

The important objects for uncentered PCA are the matrix $W_{i,j} = (w_i)_j$ of the eigenvectors $\{\vec{w}_j\}$ and the matrix $\Lambda^{(u)}_{i,j} = \lambda^{(u)}_i \delta_{i,j}$ eigenvalues $\{\lambda^{(u)}_i\}$ of the matrix of non-central second moments,

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

and

$$\hat{T} \cdot \vec{w}_j = \lambda^{(u)}_j \vec{w}_j, \quad \vec{w}^{\,t}_i \cdot \vec{w}_j = \delta_{i,j}, \quad 1 \leq i, j \leq n,$$

Let define the matrix $V^{(u)}_{\alpha,j} = \{(v^{(u)}_\alpha)_j\}$ of the uncentered principal components $\{\vec{v}^{\,(u)}_j\}$ where

$$V^{(u)}_{\alpha,j} = X^{(u)}_{\alpha,i} W_{i,j}, \quad W^t_{i,i'} W_{i',j} = \delta_{i,j},$$

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

$$\mathrm{Var}(\vec{v}^{\,(u)}_j) = \lambda^{(u)}_j - (\vec{\mu})^2 \cos^2(\vec{\mu}, \vec{w}_j)$$

Uncentered PCA

Using the representation

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

where

$$\Lambda^{(u)\,1/2}_{i,j} = \left[\lambda^{(u)}_i\right]^{\frac{1}{2}}\delta_{i,j}, \quad \bar{V}^{(u)\,t}_{i,\alpha,}\bar{V}^{(u)}_{\alpha,j} = \delta_{i,j}.$$

we obtain the Singular Value Decomposition (SVD) of the uncentered data matrix

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

If $\lambda^{(u)}_k \ll 1$ for $k = p+1, p+2, ..., n$ we can apply the **"eigenvalue control parameter"** approximation to get for the output data matrix $\tilde{X}^{(u)}_{\alpha,j}$ of rang p

$$\tilde{X}^{(u)}_{\alpha,i} = \sqrt{M}\left(\left[\lambda^{(u)}_1\right]^{\frac{1}{2}}\bar{V}^{(u)}_{\alpha,1}W^{t}_{1,i} + ... + \left[\lambda^{(u)}_p\right]^{\frac{1}{2}}\bar{V}^{(u)}_{\alpha,p}W^{t}_{p,i}\right).$$

The mean square error η_p for this approximation:

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

Clustering

“Clustering is the unsupervised learning task of organizing or partitioning data into meaningful groupings.”

Clustering methods

- **Partitioning algorithms**: Construct various partitions and then evaluate them by some criterion.

- **Hierarchy algorithms**: Create a hierarchical decomposition of the set of data (or objects) using some criterion.

- **Density-based**: based on connectivity and density functions

- **Grid-based**: based on a multiple-level granularity structure

- **Model-based**: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other.

- ***Main question: how to select correct clustering strategy?***

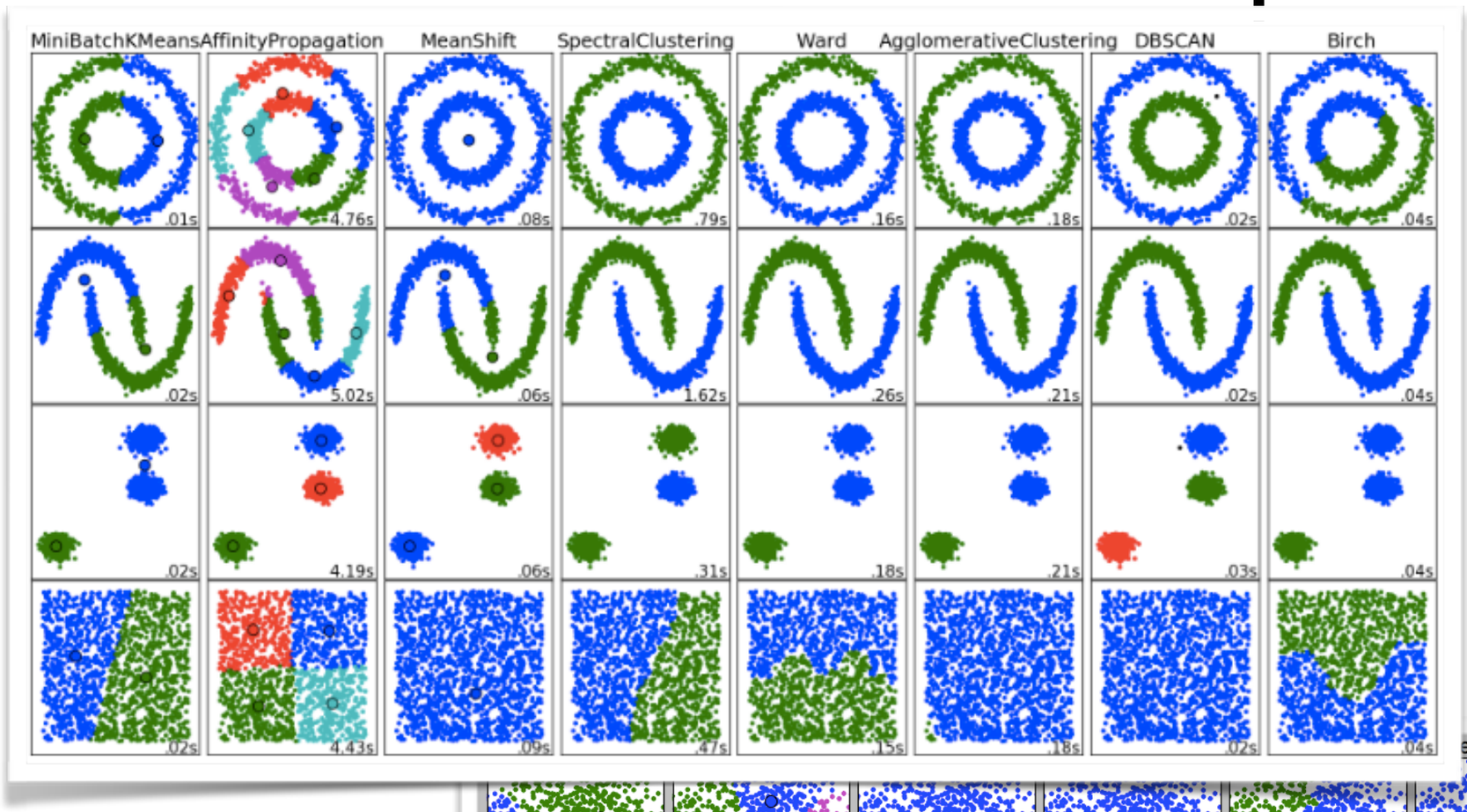
Clustering methods

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
☆ Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
☆ Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
☆ DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
☆ Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

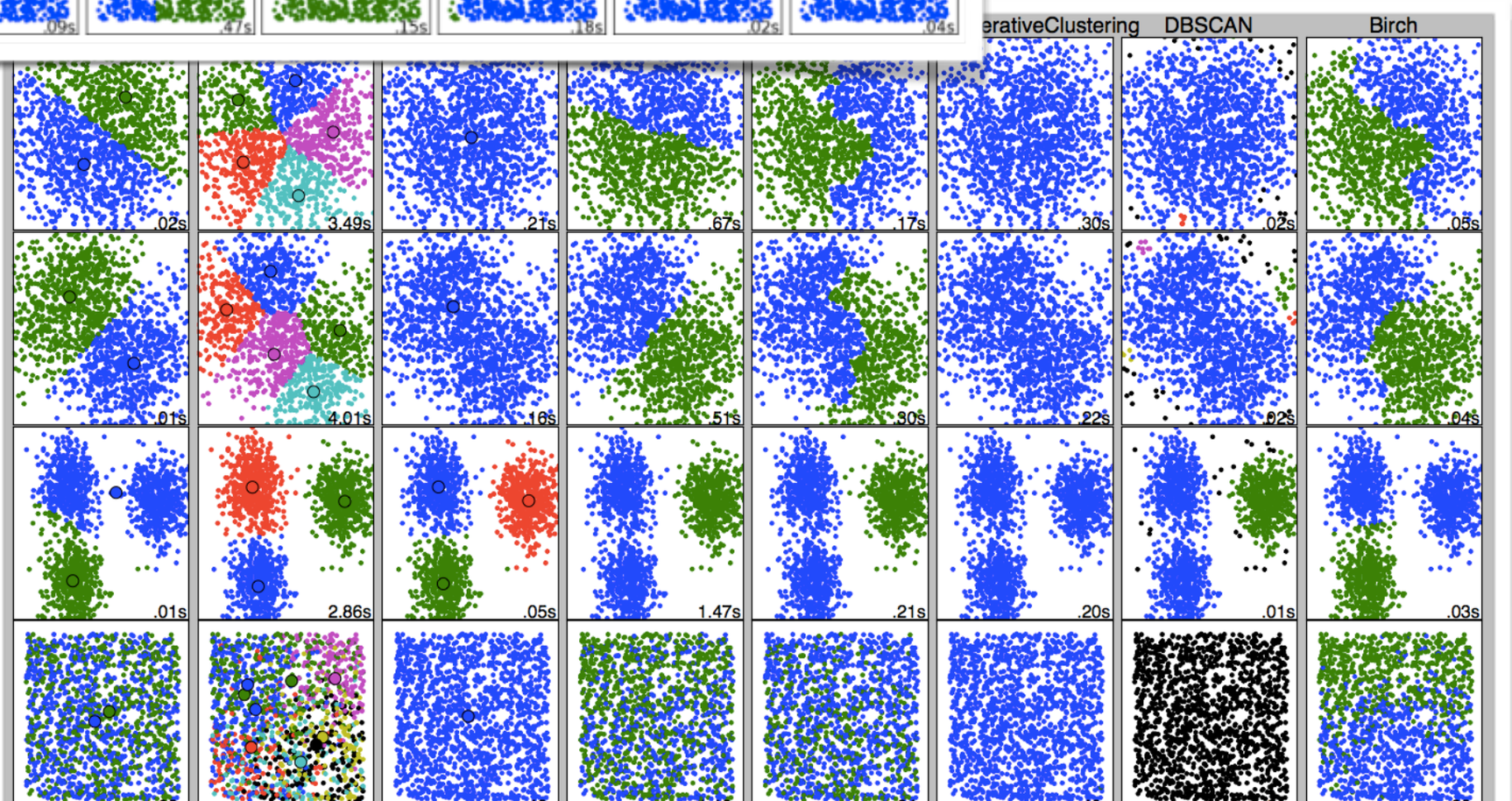
☆ Clustering methods without presetuped number of clusters

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Data examples



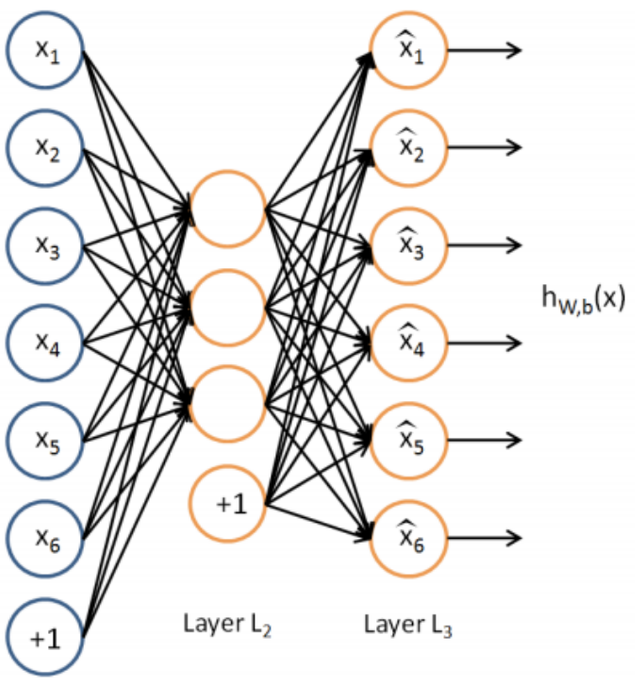
Noisy dataset

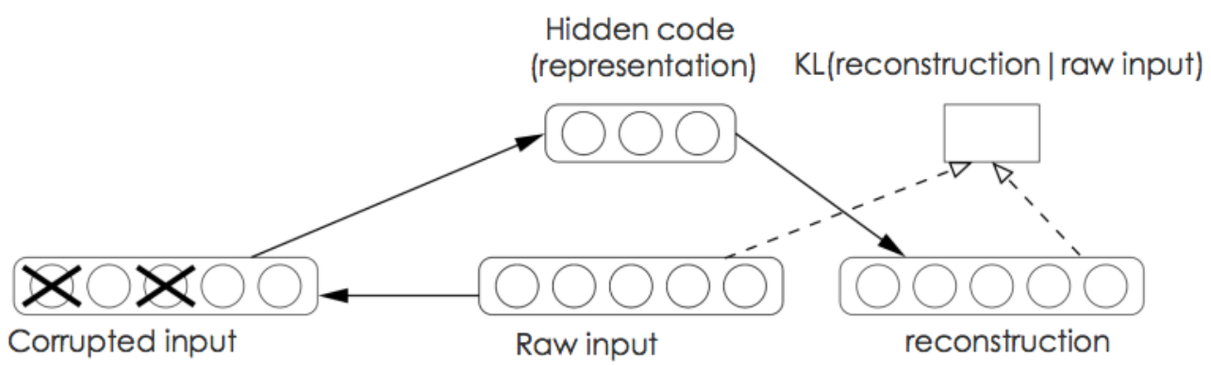


Other similar unsupervised techniques

Denoising autoencoder

- Autoencoders are used to **capture structure in data, using unsupervised learning**;
- **Data is provided as input** and the output of network tries to reconstruct input;
- Learning is performed using **back propagation** or related methods;
- The network captures a representation of inputs;
- Denoising autoencoders are used to avoid overfitting and to encourage learning structure instead (adding noise to input)
- Practical applications of autoencoders are data denoising, and dimensionality reduction for data visualization. With appropriate dimensionality and sparsity constraints, **autoencoders can learn data projections that are more interesting than PCA** or other basic techniques.





Vincent, Pascal, et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." Journal of Machine Learning Research 11.Dec (2010): 3371-3408.

Stochastic optimisation

Evolutionary algorithms (EA):

- *Genetic Algorithms - GA*
- *Evolutionary strategies - ES*

Scope of optimisation

Parameter optimisation for optimal performance

Optimal design and manufacturing

Inverse problem

System modeling

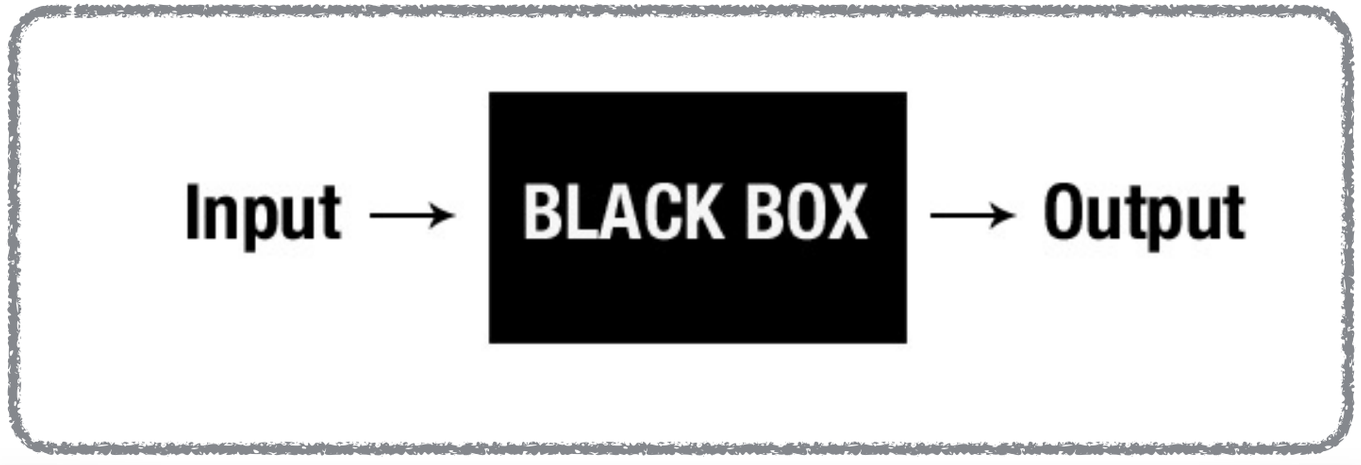
Machine learning

Data mining

Bioinformatics

Prediction

Black-box Optimisation



- A black box search task is to minimise function $f(\vec{x}) : R^n \rightarrow R$ for which the analytical form is not known.

- Black-box can be evaluated to obtain:
  - value
  - approximate gradient (huge number of evaluations)

$$\frac{d\,f(x)}{d\,x} : \Delta_k f \approx \frac{f(x_k + \epsilon_k) - f(x_k)}{\epsilon_k} \Delta x_k$$

- Only evaluation of the function value, no derivatives or gradients → no gradient methods available

Pareto Dominance

Definition

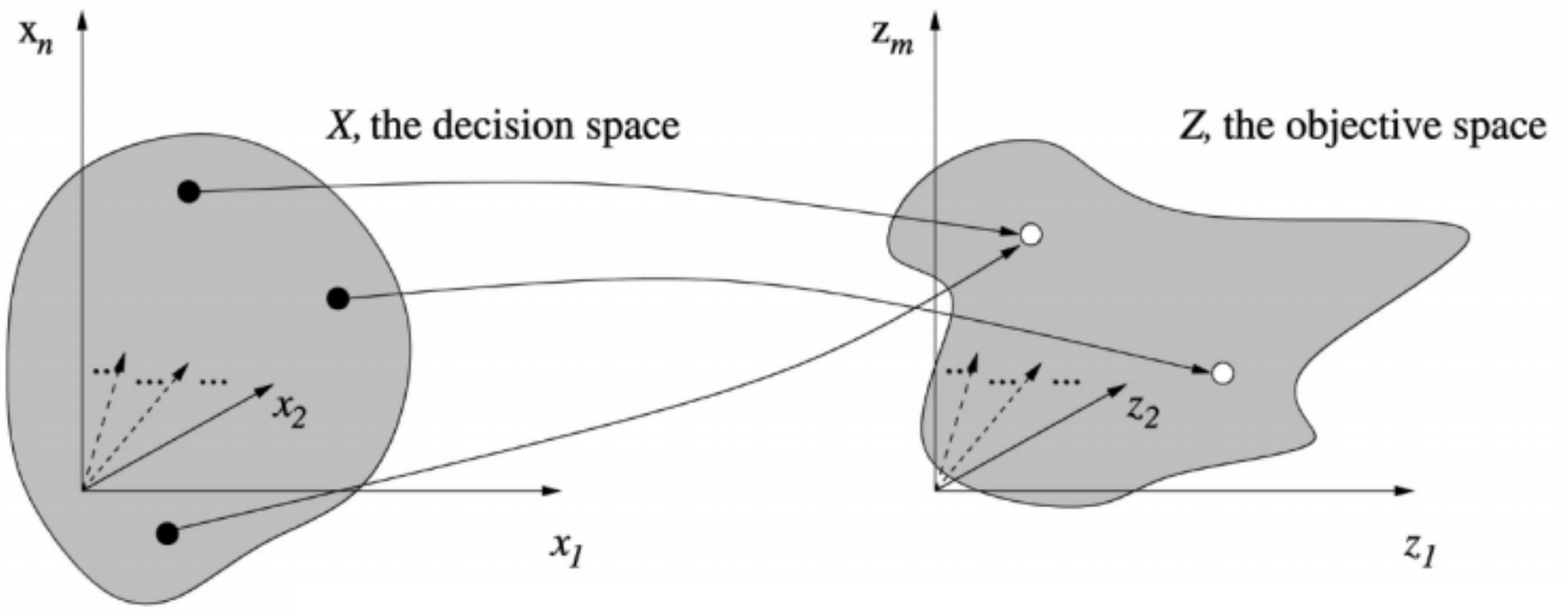
A solution $\mathbf{x_1}$ is said to dominate another solution $\mathbf{x_2}$, and we write $\mathbf{x_1} \preceq \mathbf{x_2}$, if both the following conditions are true:

1. *The solution $\mathbf{x_1}$ is no worse than $\mathbf{x_2}$ in all objectives.*
2. *The solution $\mathbf{x_1}$ is strictly better than $\mathbf{x_2}$ in at least one objective.*

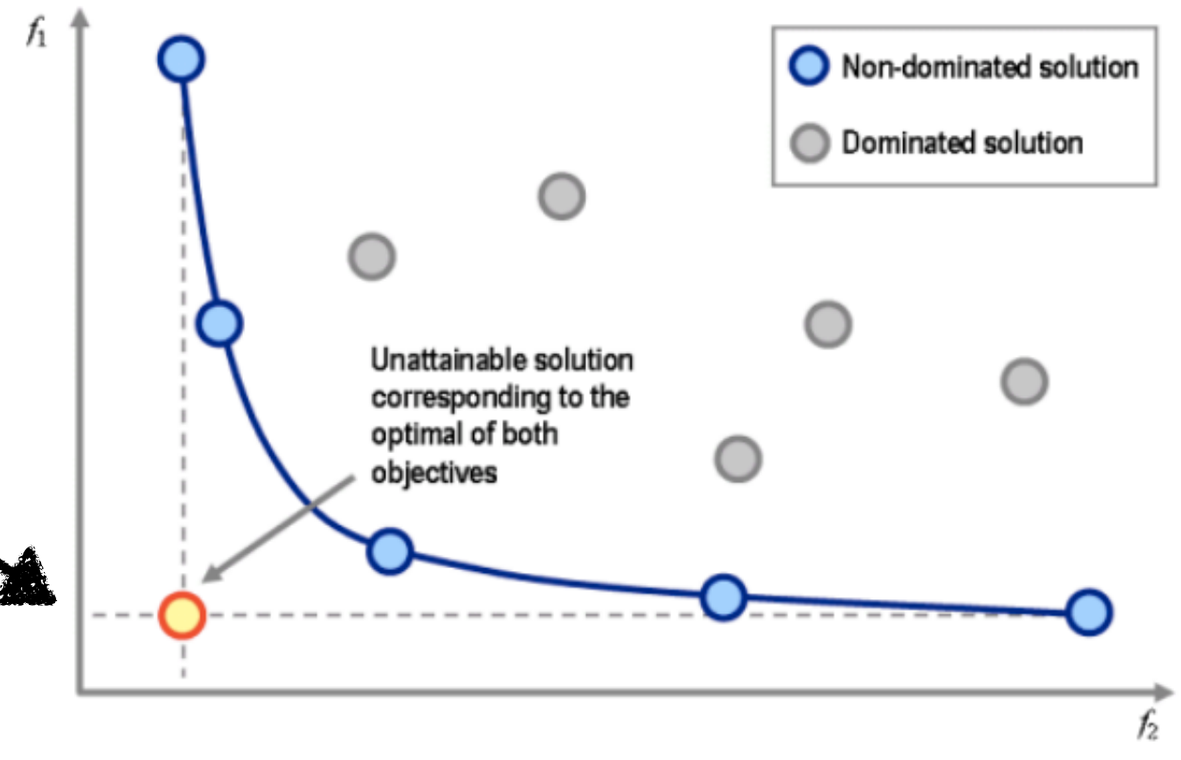
$$\mathbf{x_1} \preceq \mathbf{x_2} \text{ iff } \begin{cases} f_i(\mathbf{x_1}) \leq f_i(\mathbf{x_2}) & \forall i \in 1, \ldots, M \\ \exists j \in 1, \ldots, M & f_j(\mathbf{x_1}) < f_j(\mathbf{x_2}) \end{cases} \tag{7}$$

if any of the above conditions is violated the solution $\mathbf{x_1}$ does not dominate the solution $\mathbf{x_2}$.

Pareto points & front



"Fixed point?"



Advantages of evolutionary algorithms

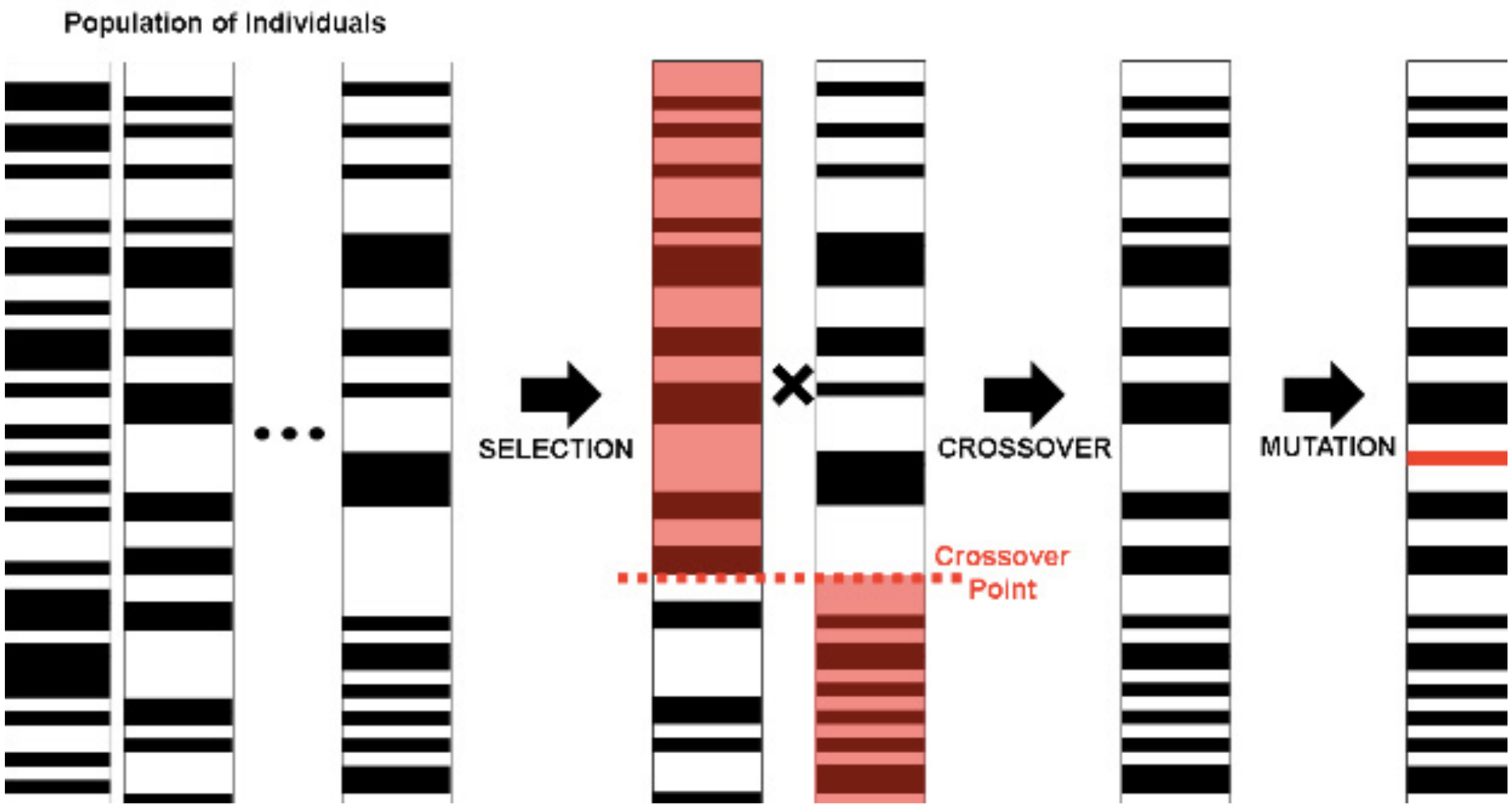
- Evolutionary algorithms could be used for any type of problem

- Useful in large search spaces and highly complex non-convex functions

- Evolutionary algorithms are ideal for “brute force” search problem

- Evolutionary algorithms are most effective as a strategy for dealing NP complete problems

Genetic algorithm schema



GA as a Markov chain (discrete dynamic system approach)

We define the population vector in the following form

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

where p_α ($p_{\alpha_i} = v_{\alpha_1}/\bar{M}$) is the probability of occurrence α-th individual in the population vector $\vec{p}'$.

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

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

Selection operator

$$F : \Lambda \to \Lambda$$

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

Mutation operator

$$\hat{U} : \Lambda \to \Lambda$$

$$(\hat{U} \cdot \vec{p})_\alpha$$

Convergency acceleration operator

?

Crossover operator

$$\hat{C} : \Lambda \to \Lambda$$

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

Transition matrix and GA concept

$$G \; : \; \Lambda \to \Lambda$$

$$G(\vec{p}) = \hat{C} \circ \hat{U} \circ F(\vec{p})$$

$$Q_{\vec{q},\vec{p}} = \bar{M}! \prod_{\alpha \in \Omega} \frac{(G(\vec{p})_\alpha)^{\bar{M} q_\alpha}}{(\bar{M} q_\alpha)!}$$

Statements

- **Convergence rate of the genetic algorithm** is determined by the **second largest eigenvalue of the transition matrix**.

- Another remarkable feature of the genetic algorithm is the presence of **a rich structure of fixed and metastable points.**

- Pareto front is located in environments of **fixed point.**

- 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.

Famous genetic algorithms for multi-objective optimisation

- GA: **NSGA-II*, NSGA-III**, MOEAD/D*****
- ES: **MO-CMAES******

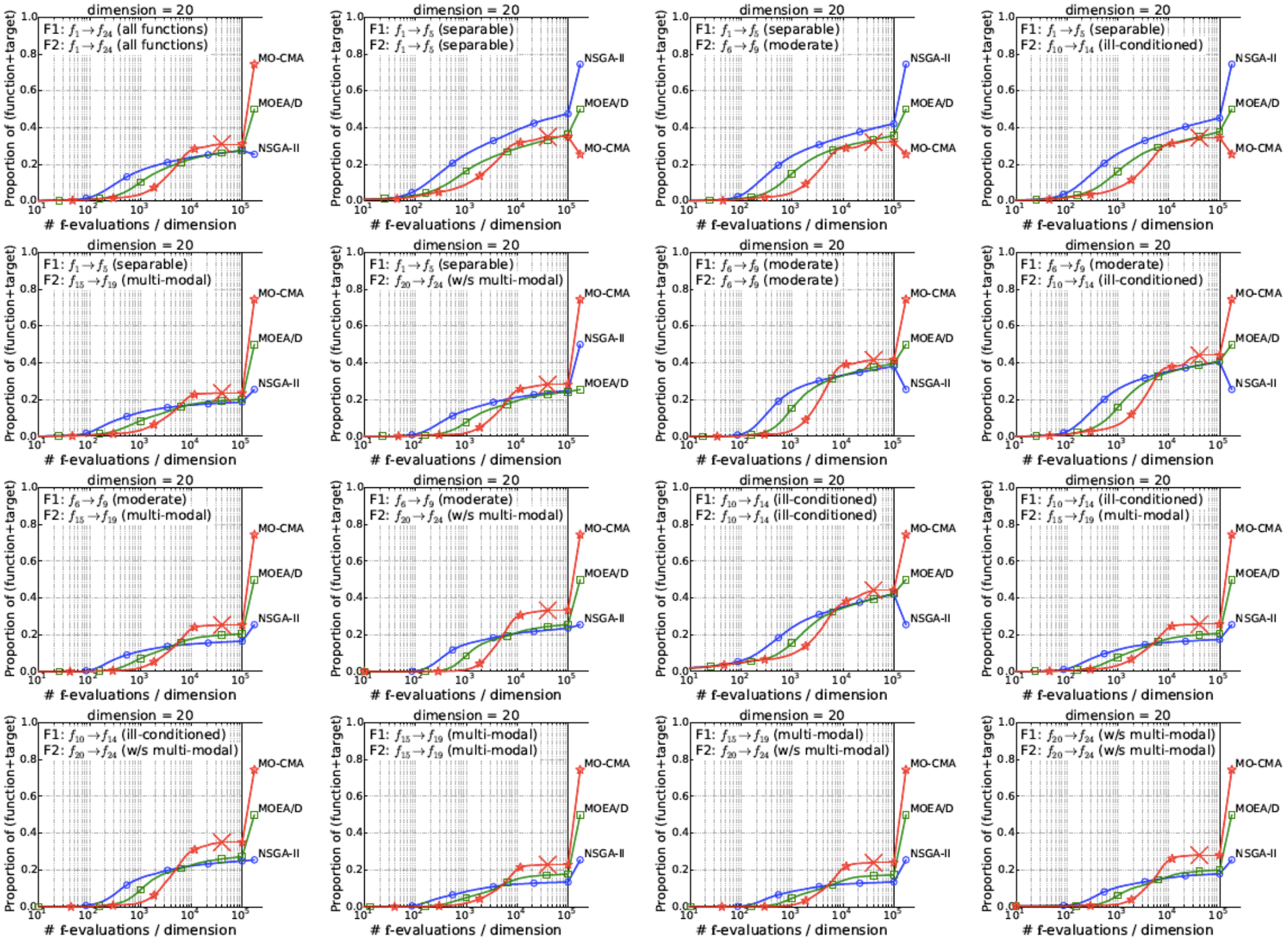
*K. Deb and A. Pratap and S. Agarwal and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, 2002, Vol.6, p.182-197, Doi:10.1109/4235.996017

**Seada, Haitham, and Kalyanmoy Deb. "U-NSGA-III: A unified evolutionary algorithm for single, multiple, and many-objective optimization." COIN Report 2014022.

***Q. Zhang and H. Li, MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition, IEEE Trans. on Evolutionary Computation, vol.11, no. 6, pp712-731 2007

****Hansen and Ostermeier (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, pp. 312-317.

Performance of algorithms



No unique recipe for “function-algorithm”configuration!

[1] Dimo Brockhoff, Thanh-Do Tran, and Nikolaus Hansen. Benchmarking Numerical Multiobjective Optimizers Revisited. In Genetic and Evolutionary Computation Conference (GECCO 2015), pages 639–646. ACM, 2015.

Genetic algorithm operator evolution

New GA operator

- Noise reduction genetic operator

$$G_P(\vec{p}) = \hat{P} \circ \hat{C} \circ \hat{U} \circ F(\vec{p})$$

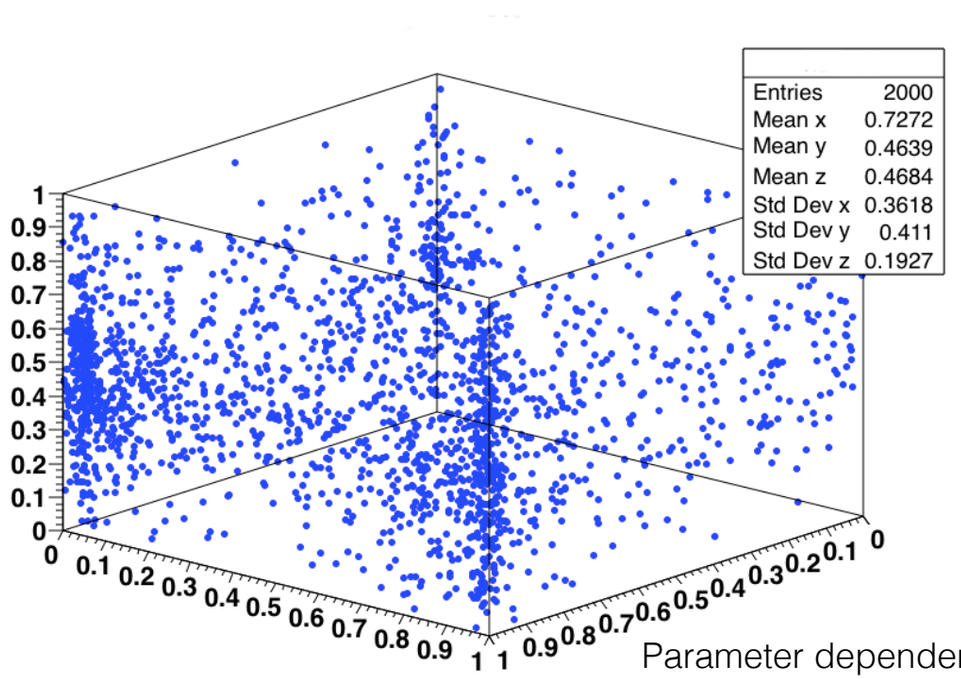
allows to check the genetic algorithms parameter sensitivity and the possible correlation between parameters.

- New algorithmic operator **P** applied to perform data transformation based on **PCA** and **inverse PCA** noise reduction operation the set of parameters used for GA.

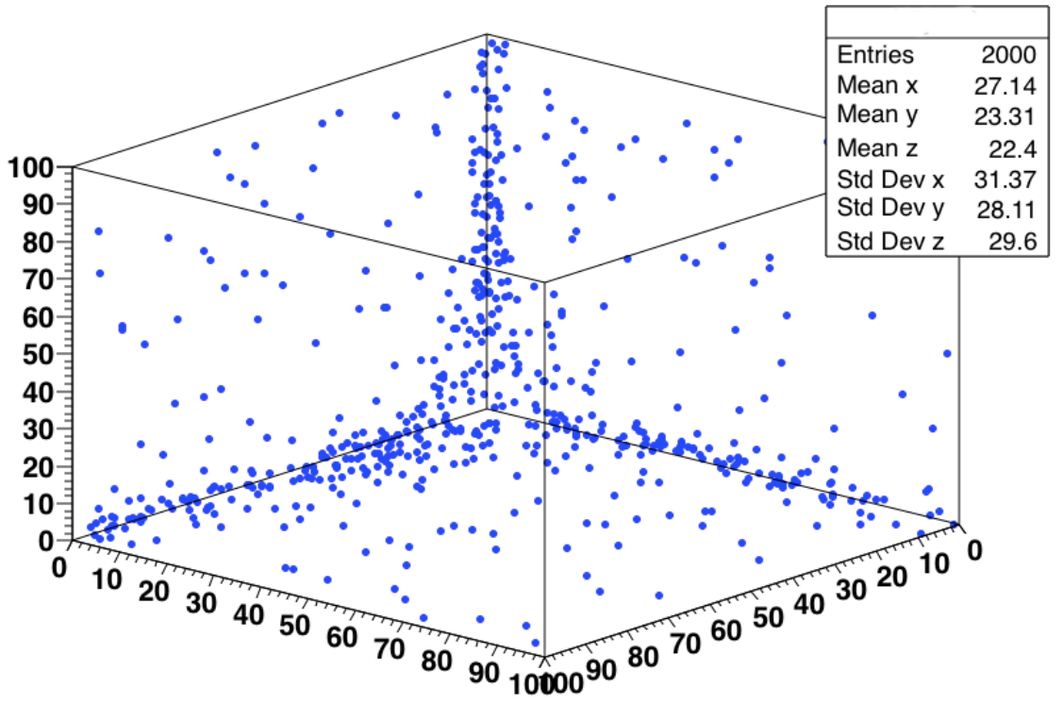
- We are using uncentered PCA in order to inverse properly back to highly constrained data parameter matrix.

Genetic algorithm benchmarking

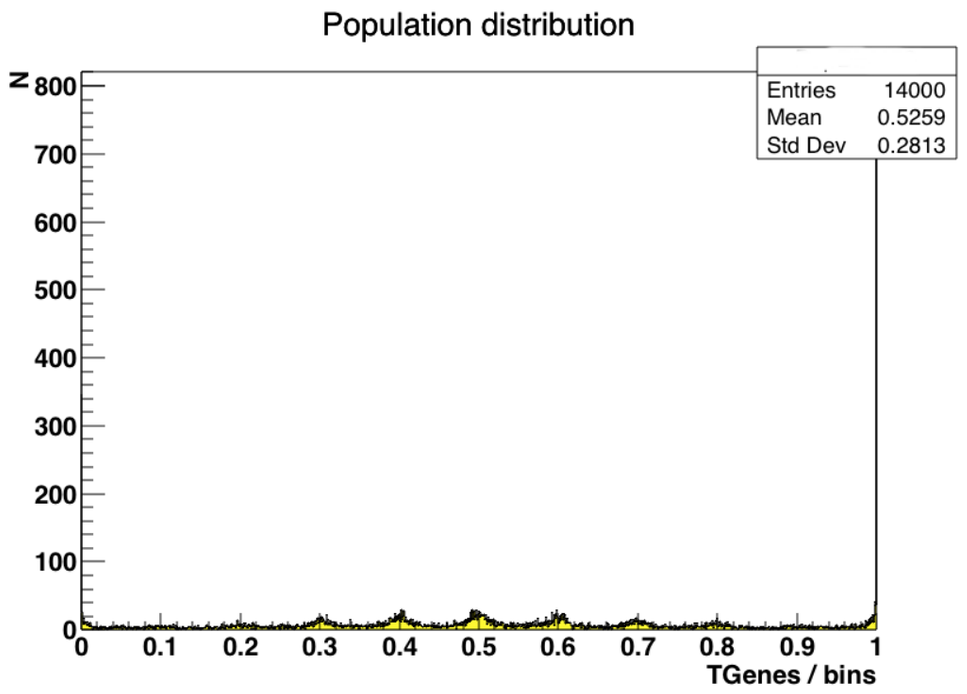
DTLZ1-NSGA-II



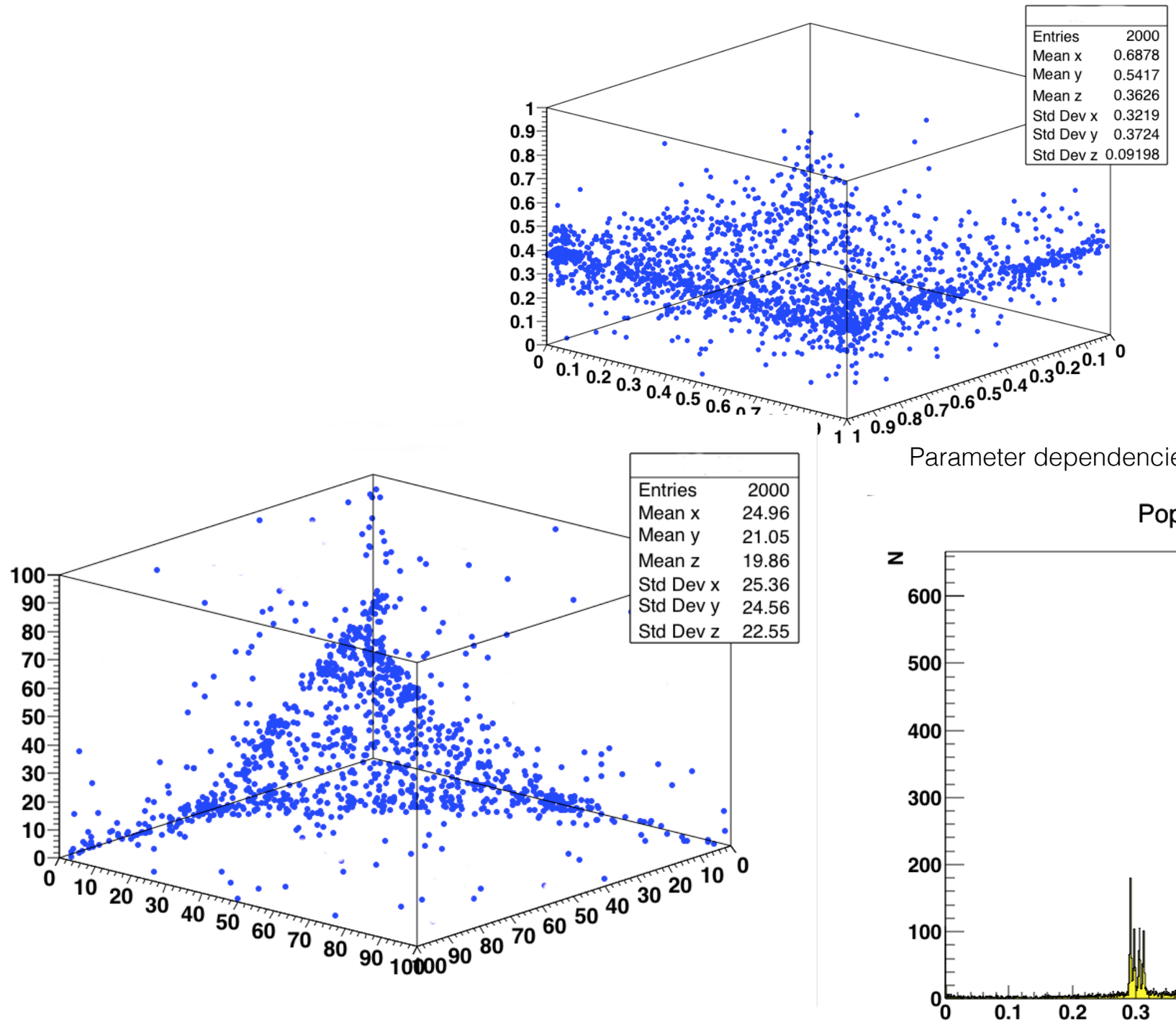
Parameter dependencies on 100th generation - NSGA-II - DTLZ1



Cost function values on 100th generation - NSGA-II - DTLZ1

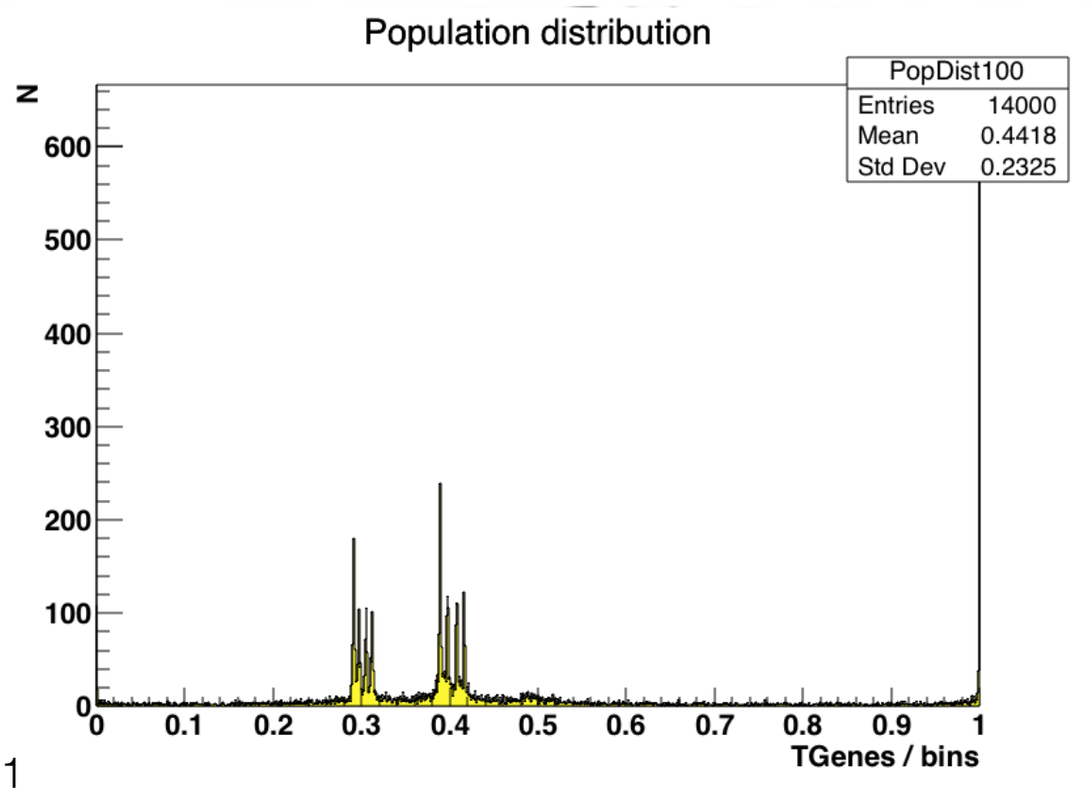


DTLZ1-NSGA+UPCA-II

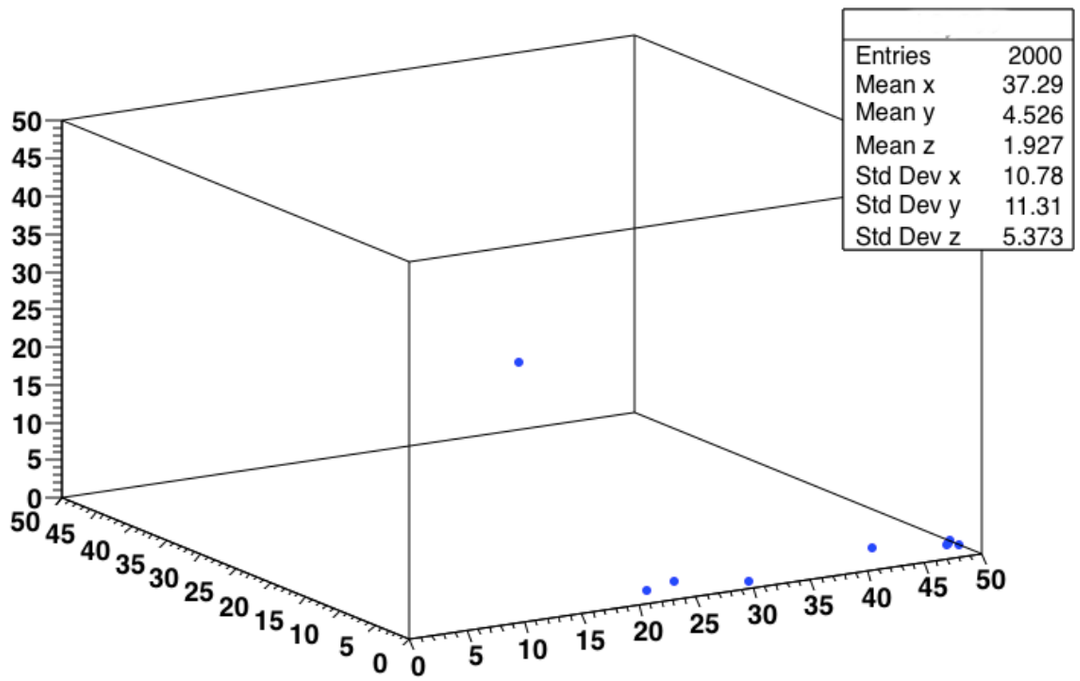


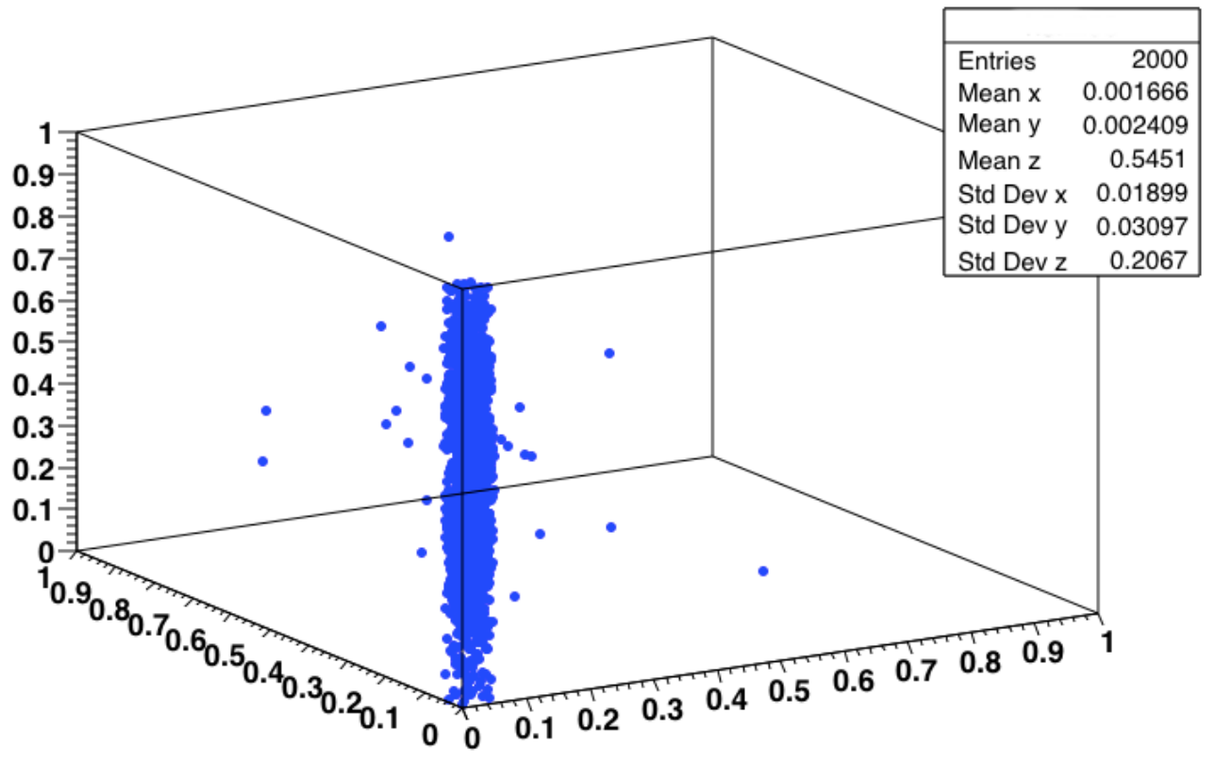
Cost function values on 100th generation - NSGA-II +UPCA- DTLZ1

Parameter dependencies on 100th generation - NSGA-II + UPCA- DTLZ1



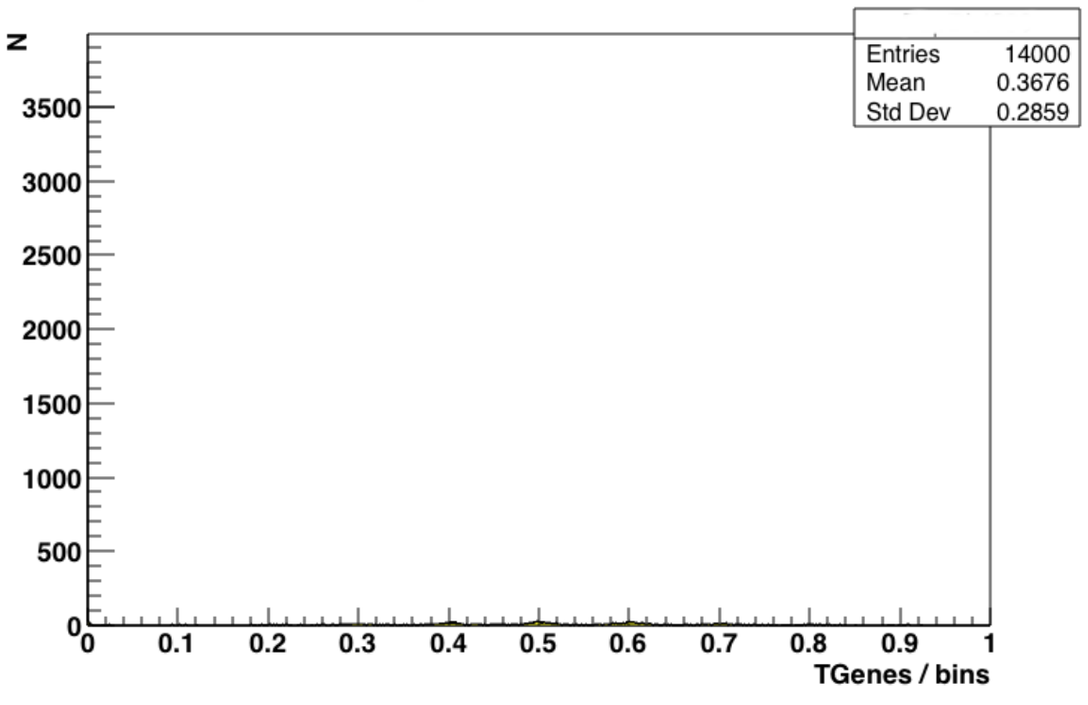
DTLZ3-NSGA-II



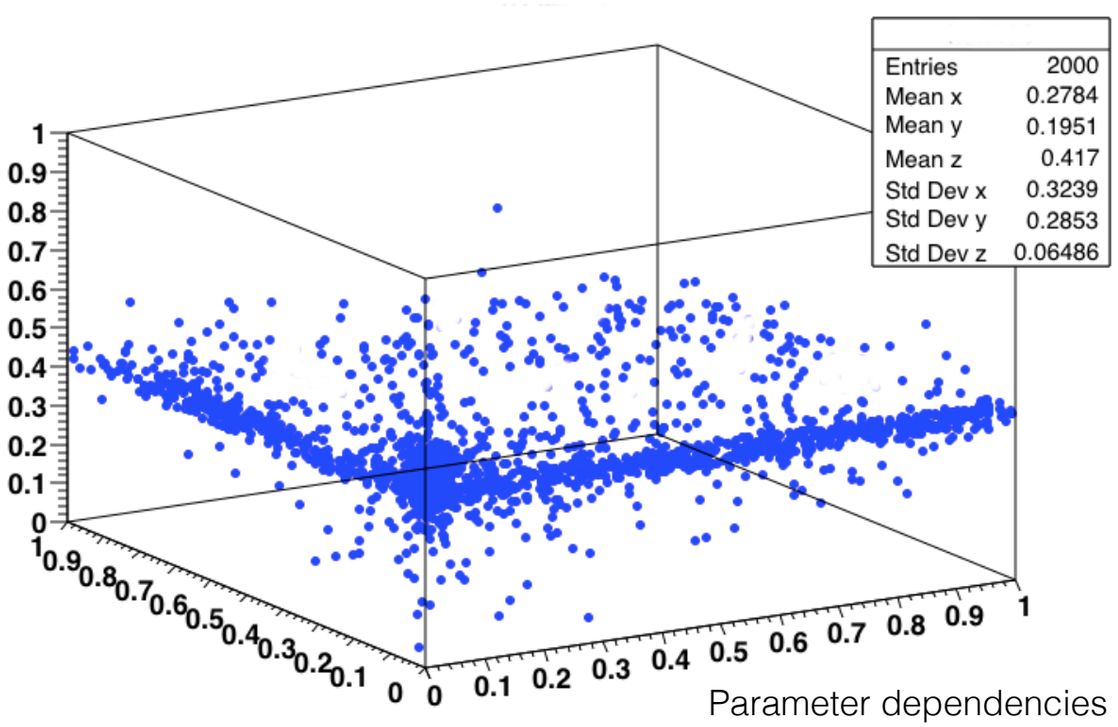


Cost function values on 200th generation - NSGA-II - DTLZ3

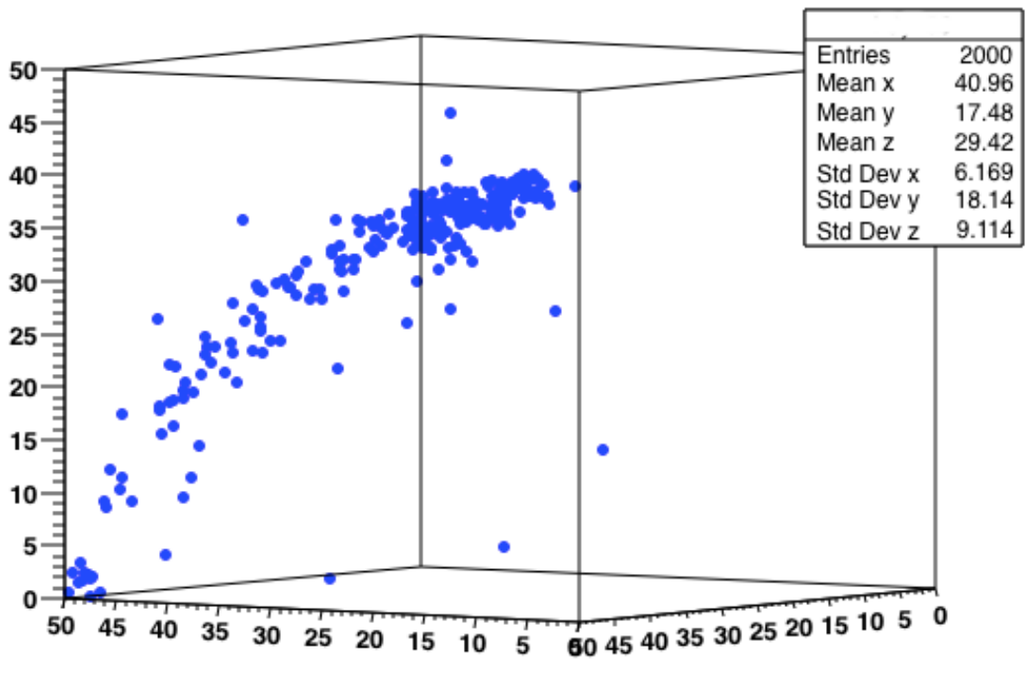
Parameter dependencies on 200th generation - NSGA-II - DTLZ3



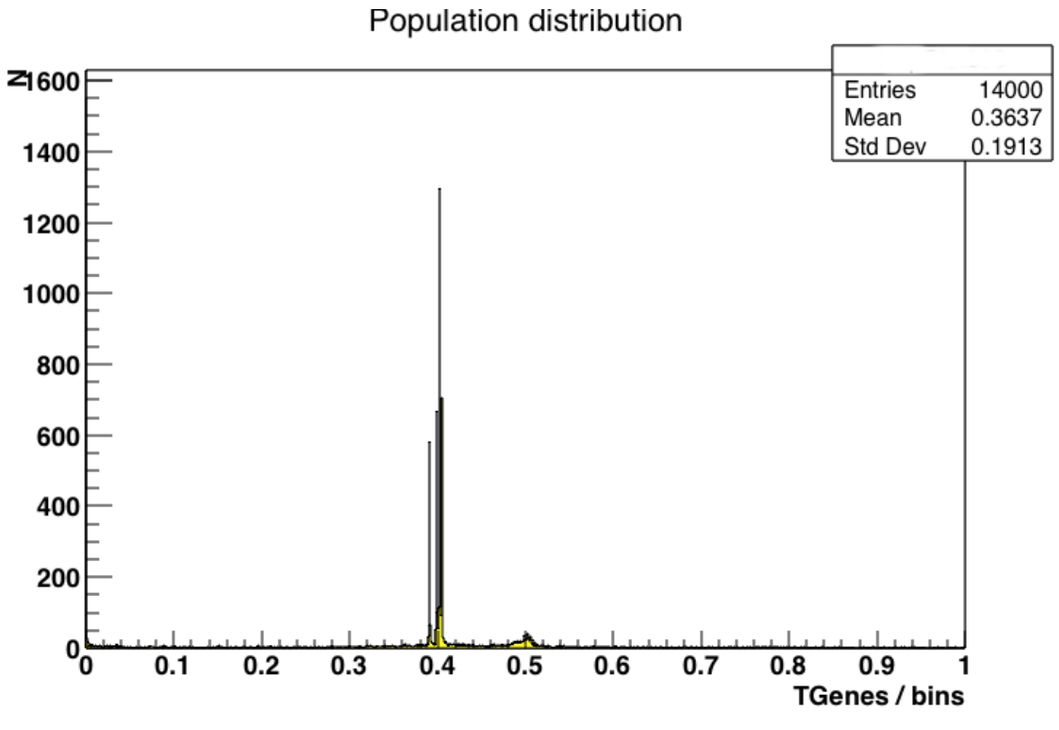
DTLZ3-NSGA+UPCA-II

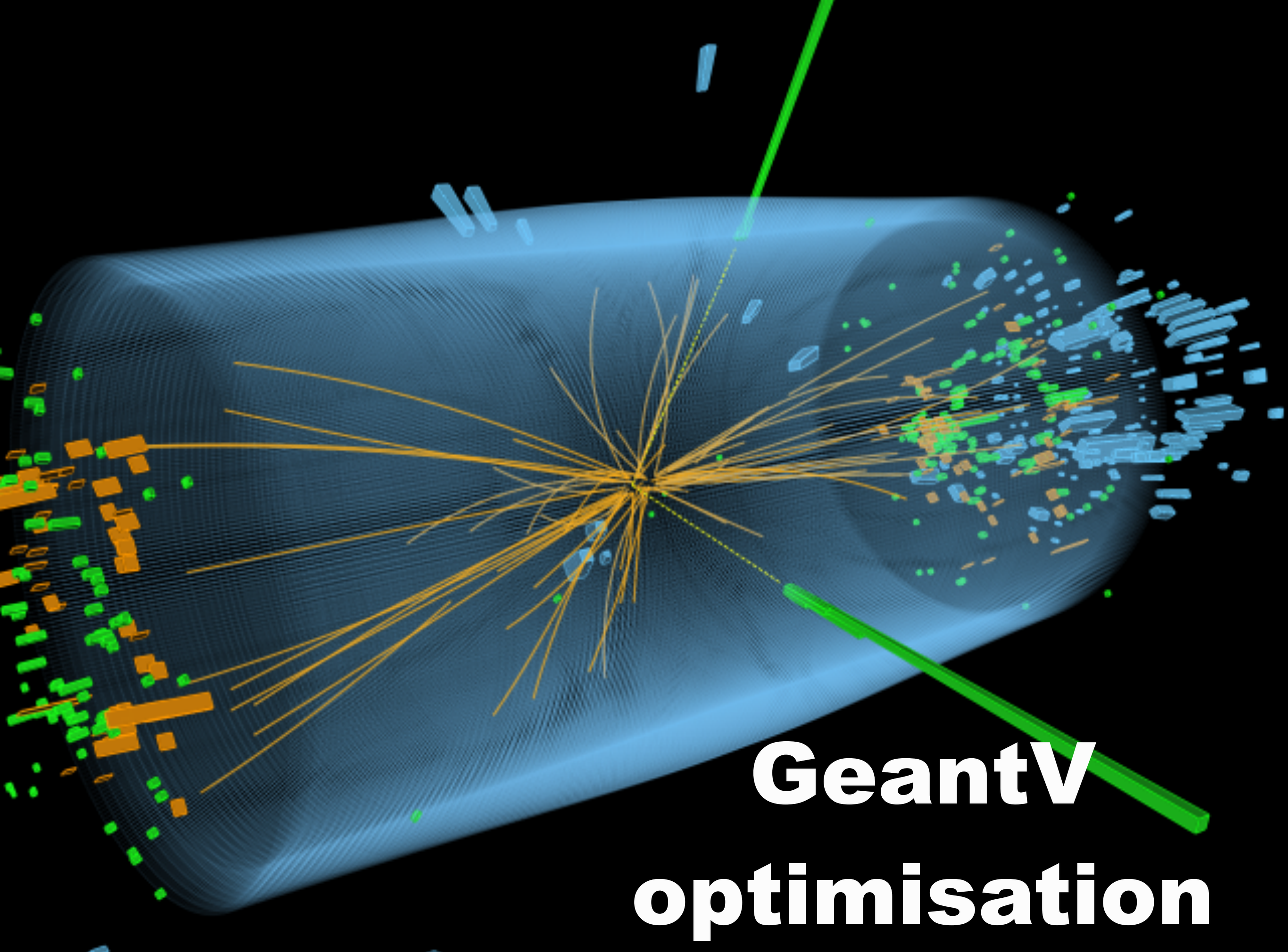


Parameter dependencies on 200th generation - NSGA-II + UPCA - DTLZ3



Cost function values on 200th generation - NSGA-II +UPCA+DTLZ3

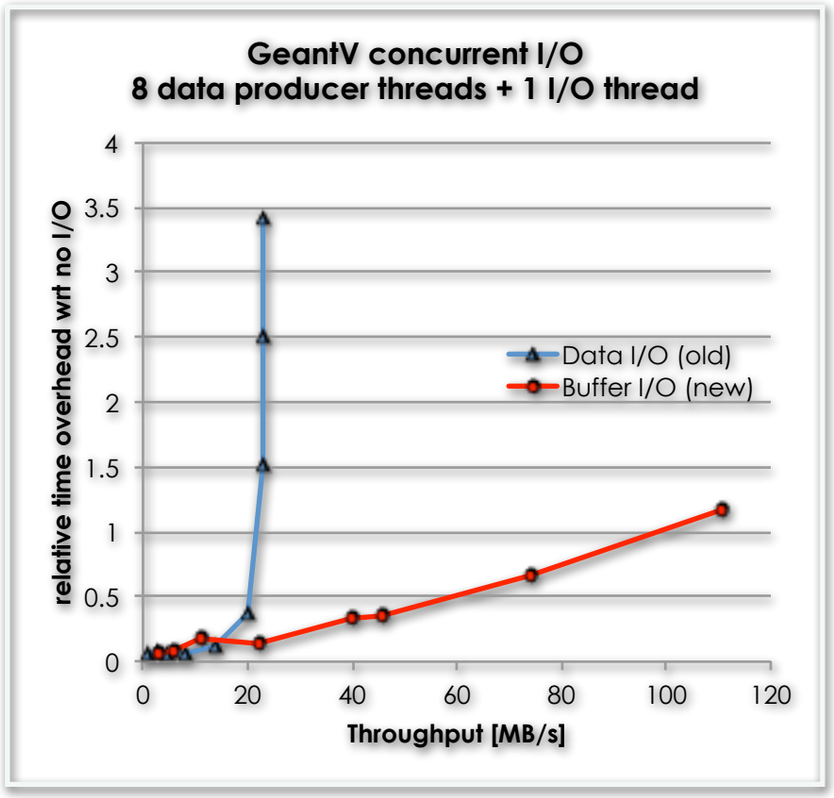



GeantV
optimisation

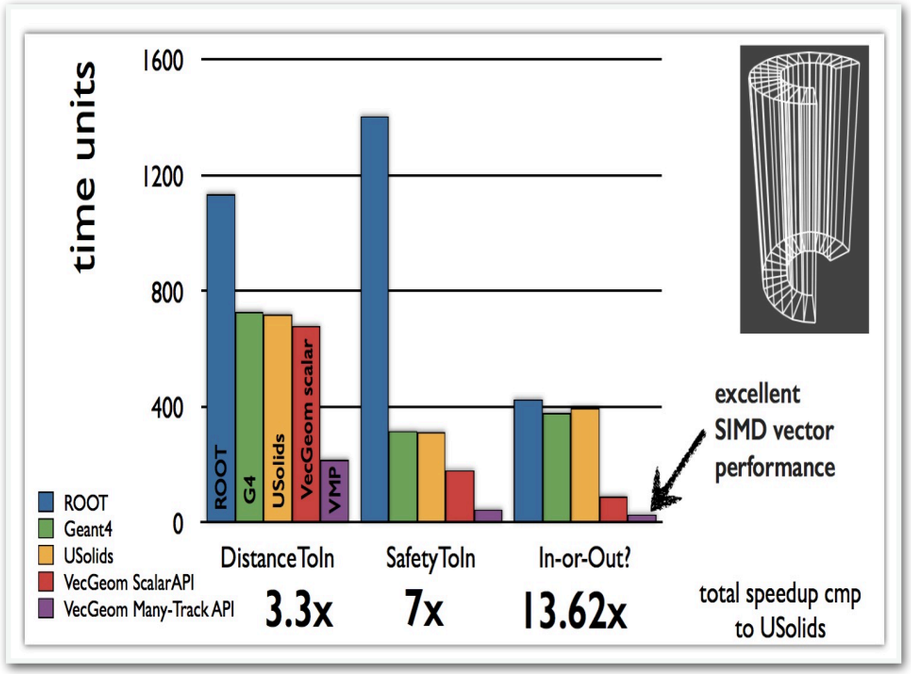
GeantV - next generation of simulation software

- Idea of GeantV is to develop an all-particle transport simulation software with:
  - Geant4 or new improved (where possible) physics models;
  - A performance between 2 and 5 times greater than Geant4;
  - Full simulation and various options for fast simulation;
  - Portable on different architectures, including accelerators (GPUs and Xeon Phi's)

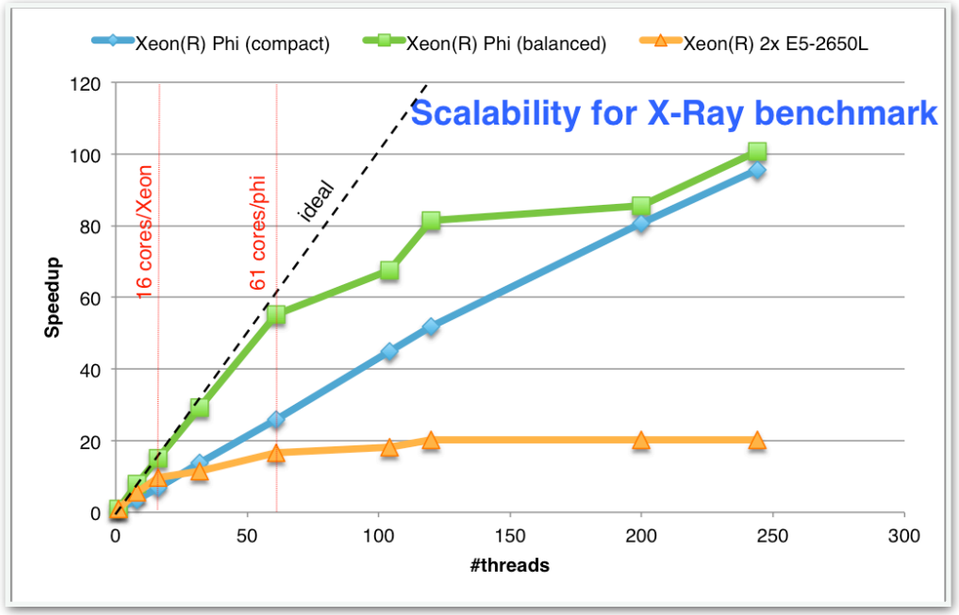
GeantV benchmarking



I/O benchmarks



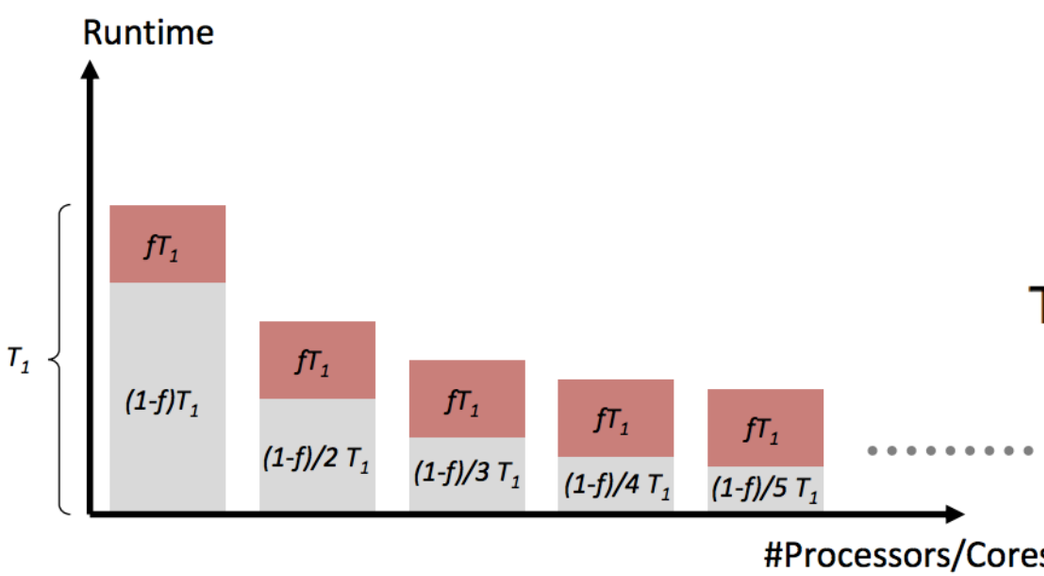
Geometry benchmarks

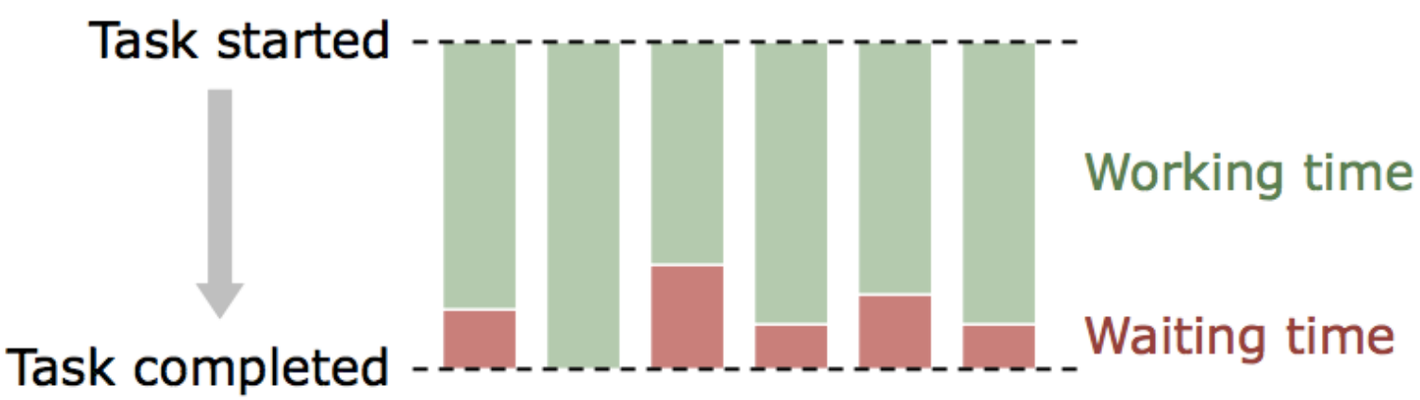


X-Ray benchmark and other benchmarks

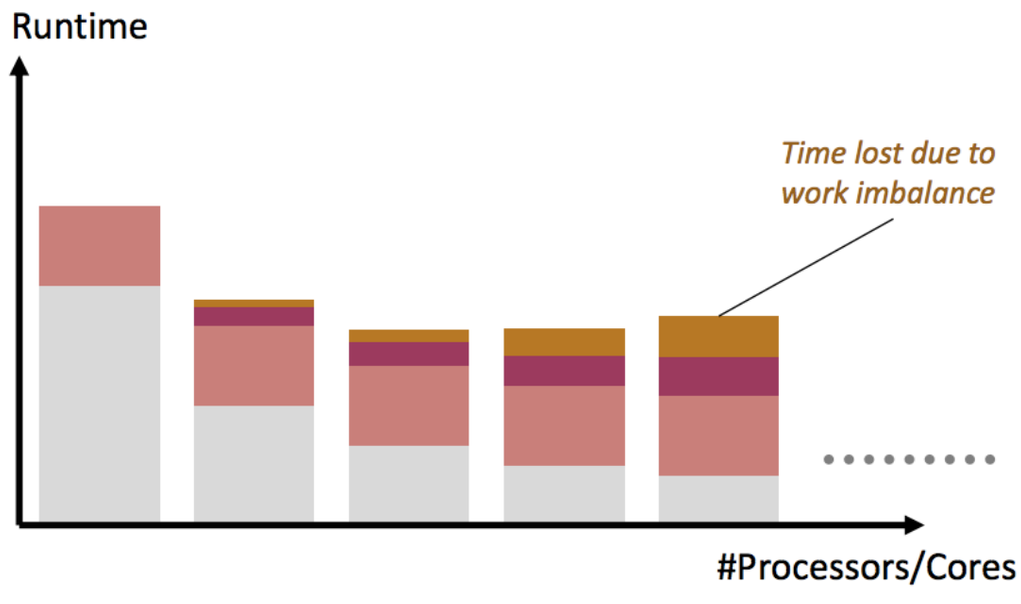
Why we need to tune?

Amdahl's Law: Runtime

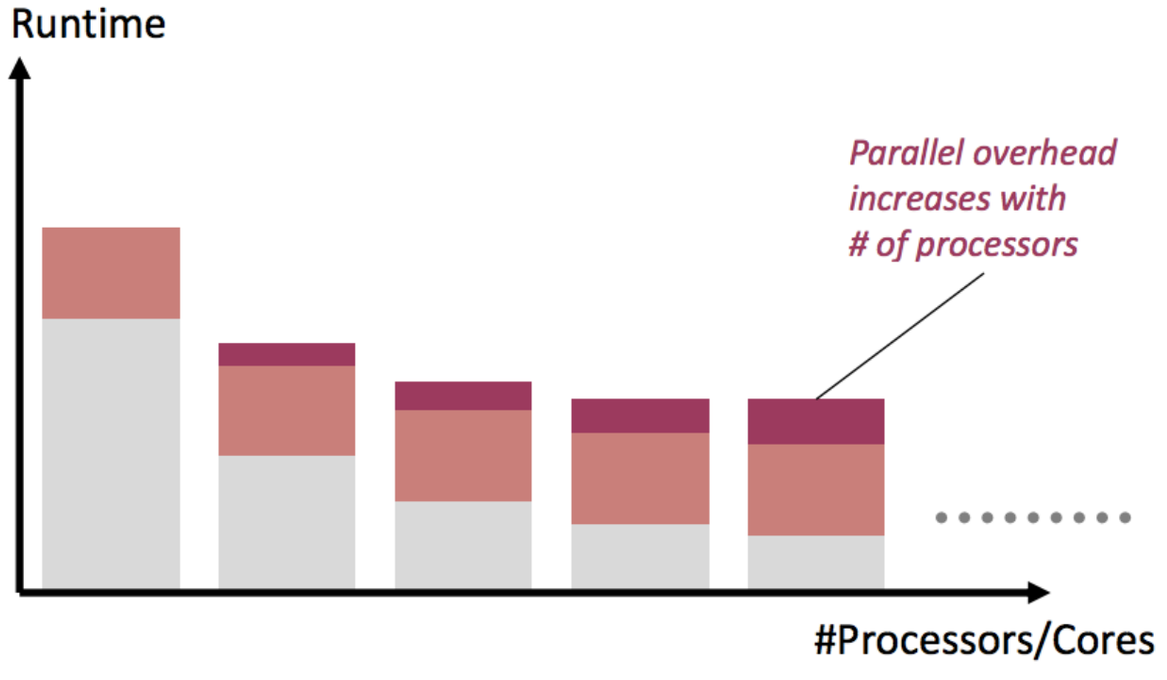




Runtime With Parallel Overhead And Work Imbalance



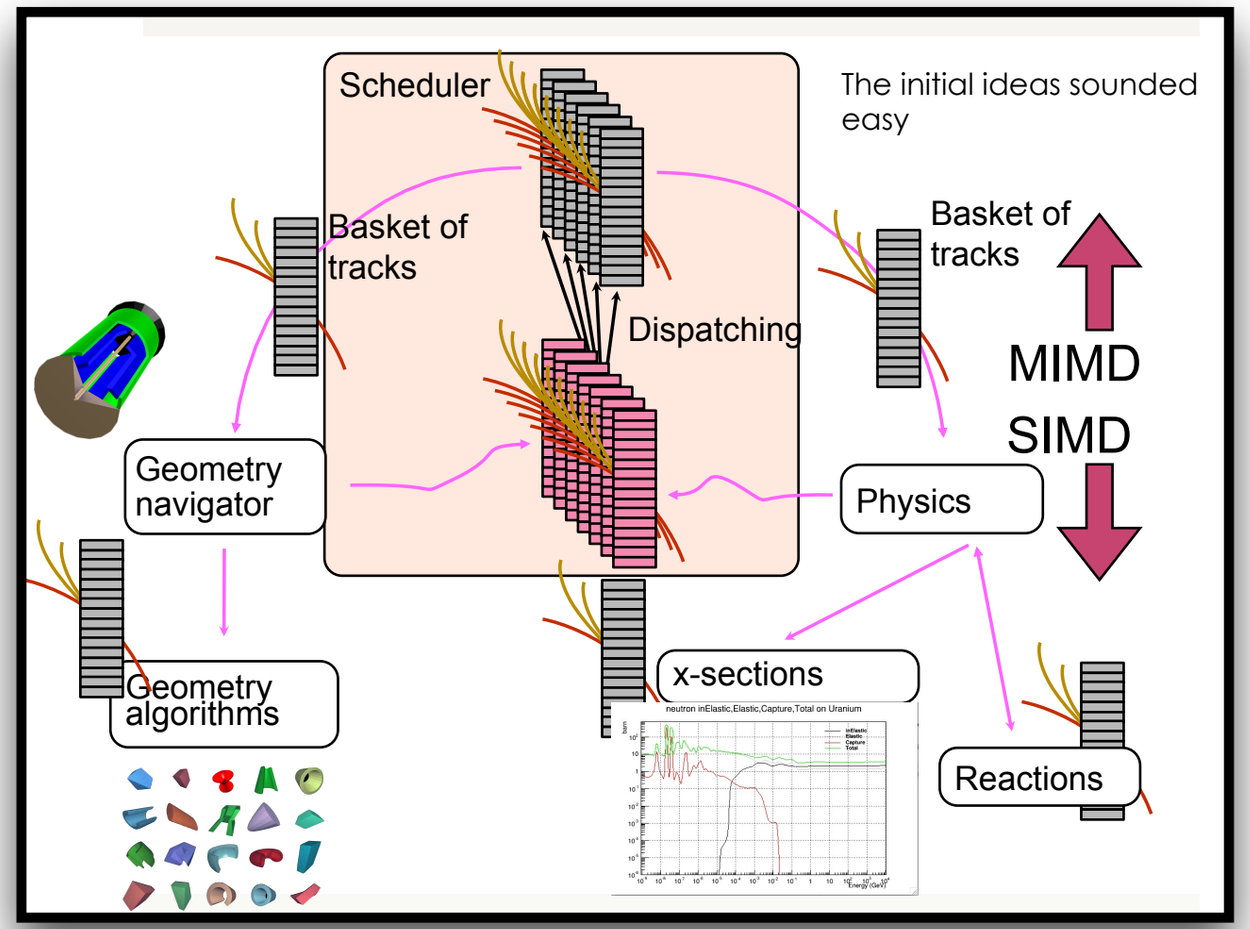
Runtime With Parallel Overhead



GeantV as black-box optimisation task

F

x



F(x)

Tuning parameters

Size of vector of events

N buffered events

N threads

Number of learning steps

Threshold for prioritizing events

Number of tracks to be reused

N propagators

Kill threshold

Maximum user memory limit

Cost functions

FPU Utilisation

Memory bandwidth

Cache

Instruction count

CPI Rate

CPU utilization

Run Time

I/O

Memory consumption

Concurrency metrics

Wait rates

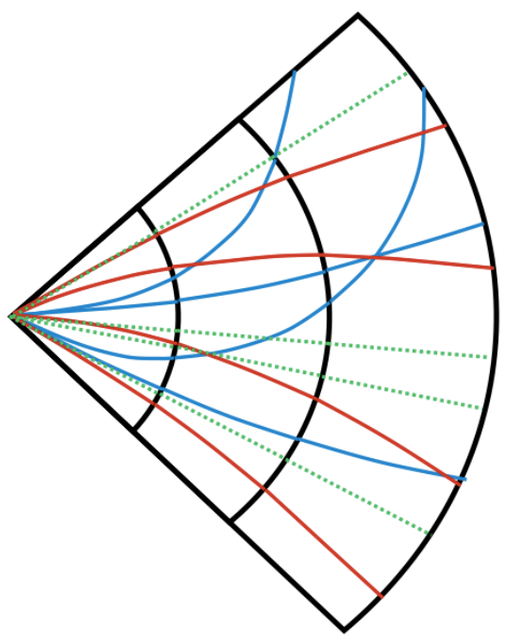
CPU Load

GeantV single-node concurrency approach

GeantV – Adapting simulation to modern hardware

Classical simulation

Flexible, but limited adaptability towards the full potential of current & future hardware



GeantV simulation

Engineered to profit from all processing pipelines

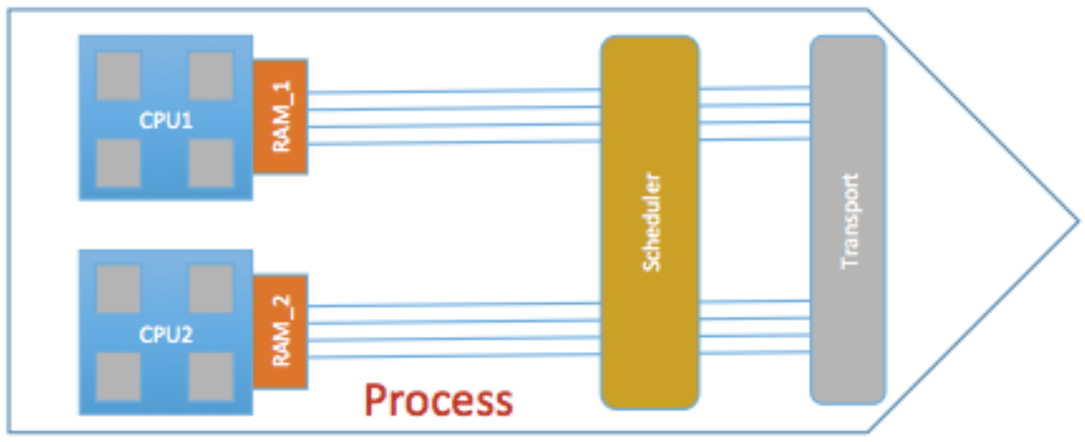
- One track at a time
- Three stacks (leptons, γ, other)
- Single event transport
- Event-level parallelism (threading)
- Cache coherency – low
- Vectorization – low (scalar auto-vectorization)

- Lots of tracks in flight
- Many baskets – 10s to 1000s
- Multi event transport
- Fine-grain parallelism + threads
- Cache coherency – high
- Vectorization – high (explicit multi-particle interfaces)

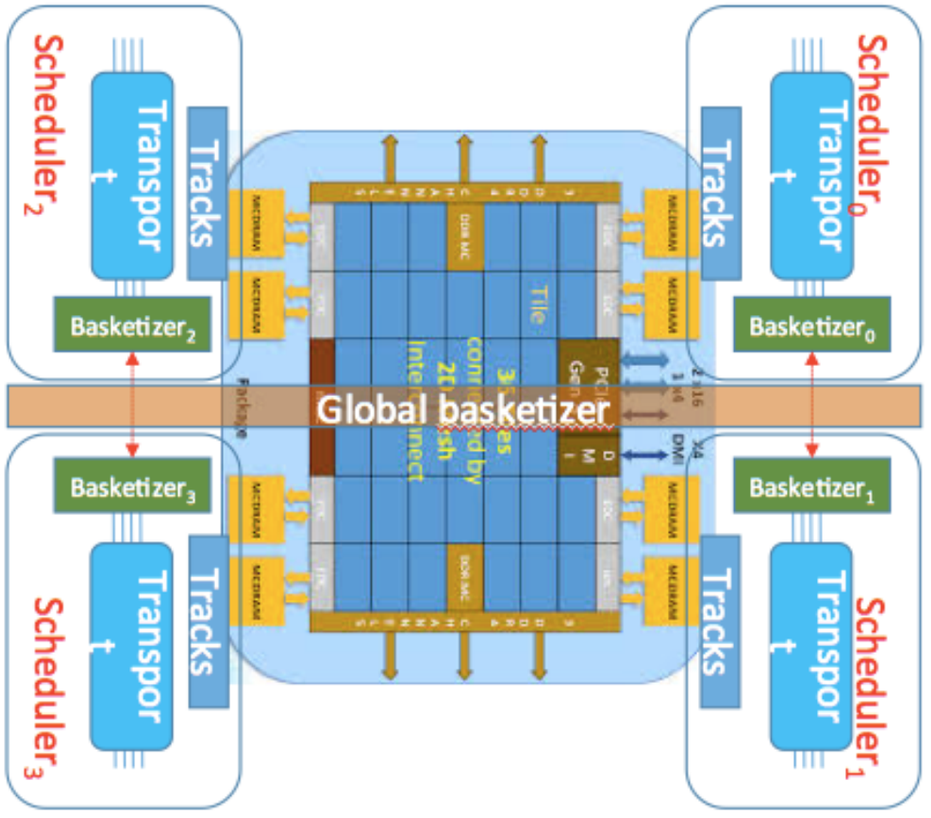
Credit: A.Gheata

Multi-task,multi-thread,multi-core approach

- **Multi-threaded approach;**
- **TBB task parallelism;**
- **Multiprocess approach** with common events queue as feeder;
- Job scheduling on **NUMA clusters.**



Credit: A.Gheata

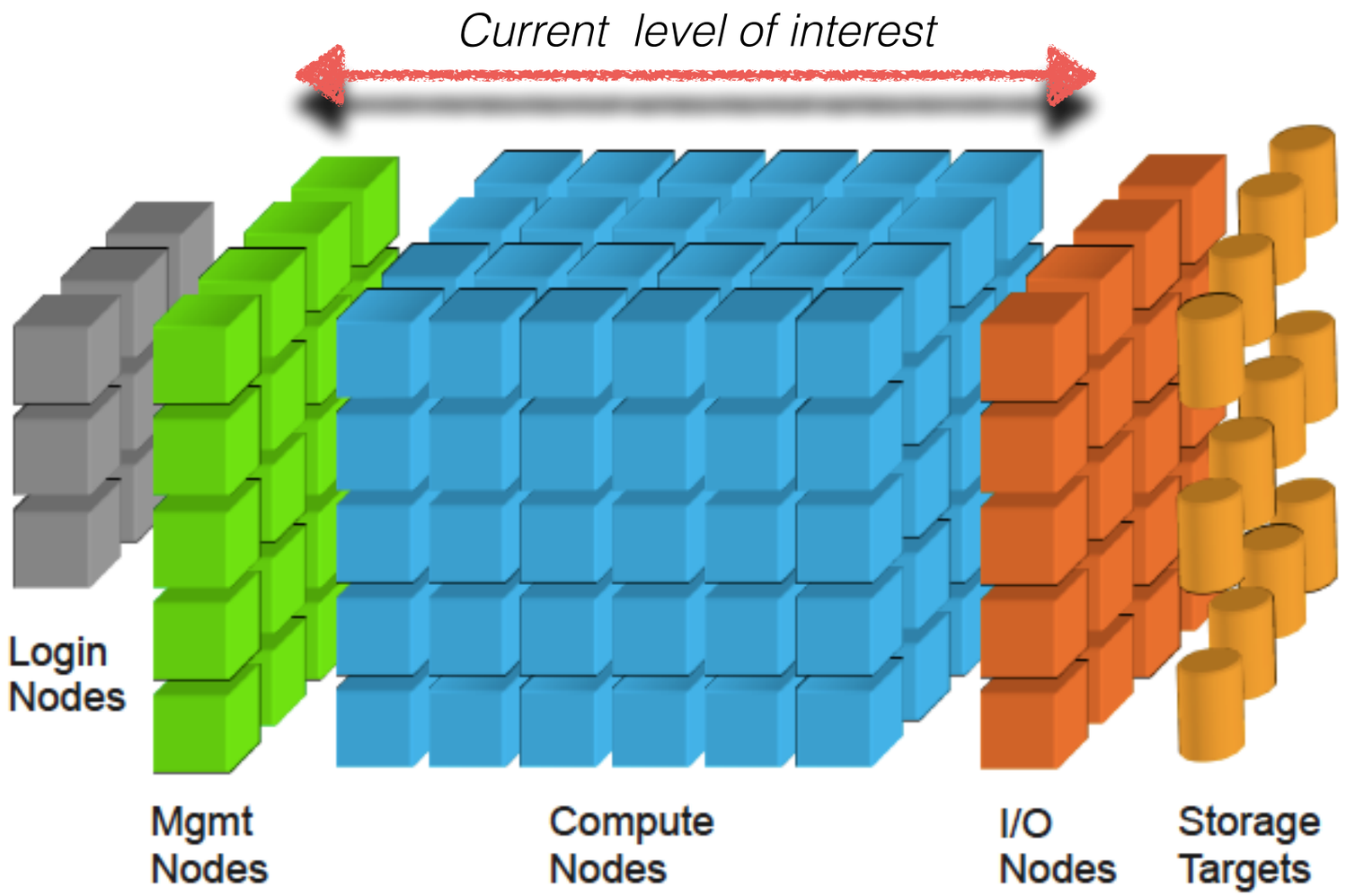


GeantV HPC approach

(multi-node approach)

Future excascale HPC

MPI will be used to communicate between address spaces in conjunction with some other “shared memory” programming model (such as OpenMP, UPC, CUDA, or OpenCL) for programming within an address space.



Thakur, R., Balaji, P., Buntinas, D., Goodell, D., Gropp, W., Hoefler, T., ... & Träff, J. L. (2010). MPI at Exascale. Procceedings of SciDAC, 2, 14-35.

Optimisation GeantV challenge: Tuning of HPC system

- Classification of set of computing nodes
- Load-balancing and event server introduction

- *Handling parallel I/O*
- *Large-scale merge of histograms*

→ *Future work*

Evolution of GeantV RunManager

RunManager class

Configuration object
**includes tuning parameters*

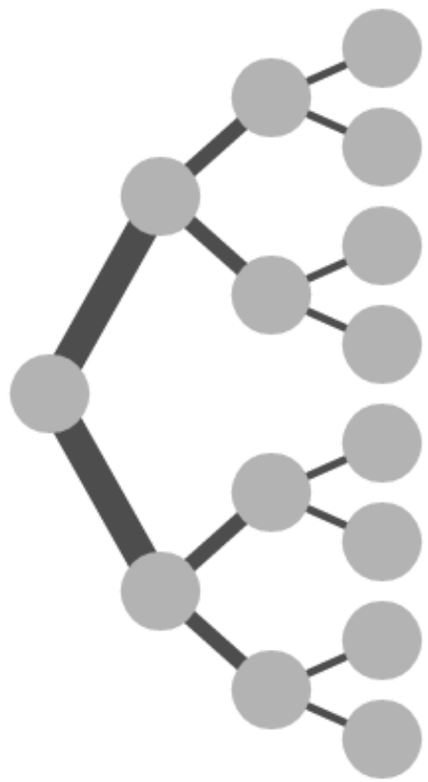
Tree-based detection of socket hierarchy

Machine learning library

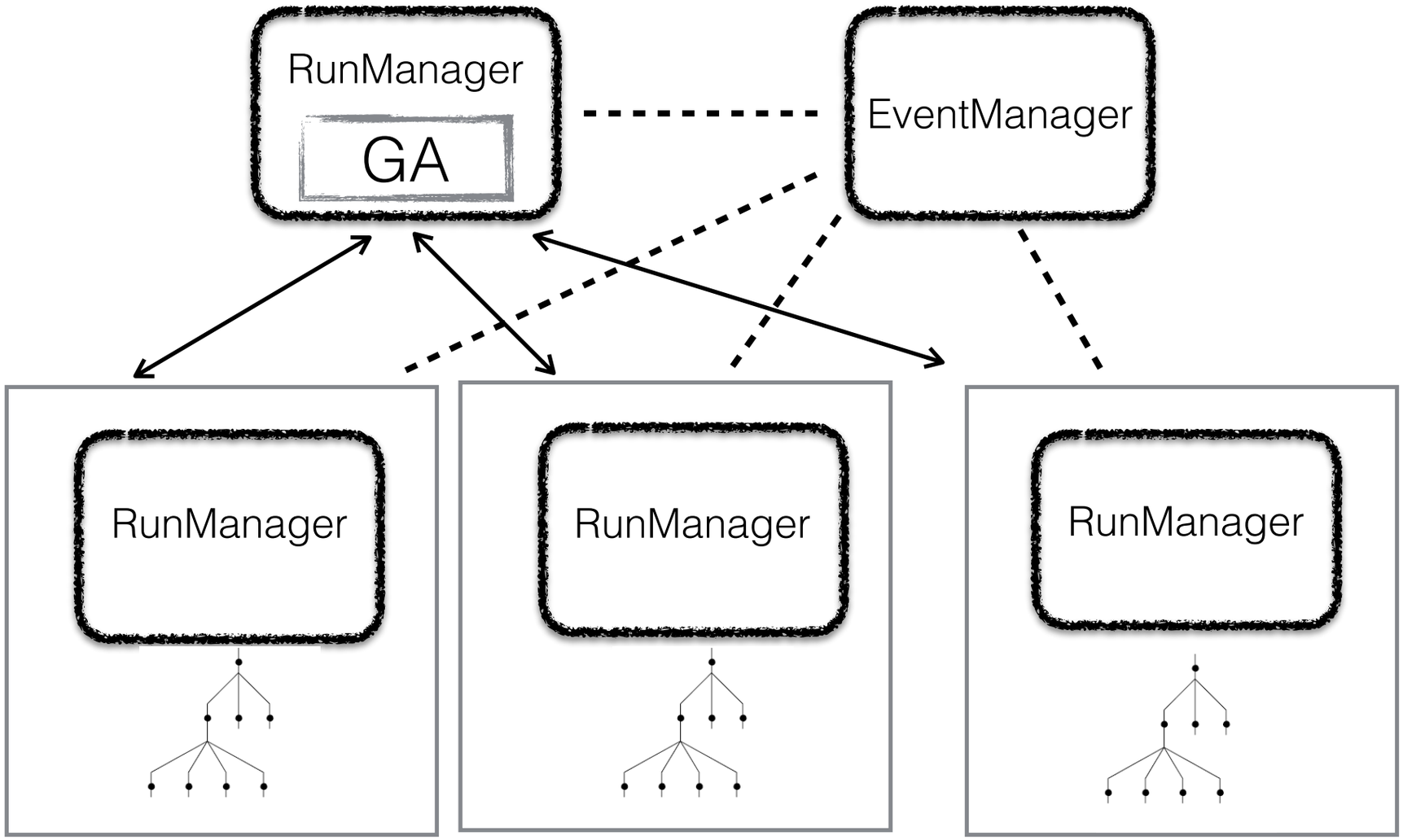
EA

PCAs

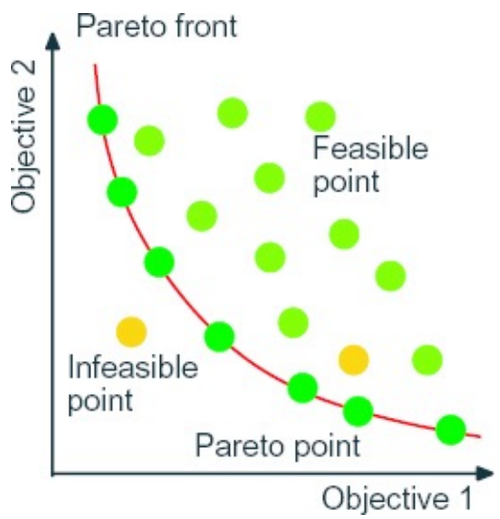
Clustering and etc.



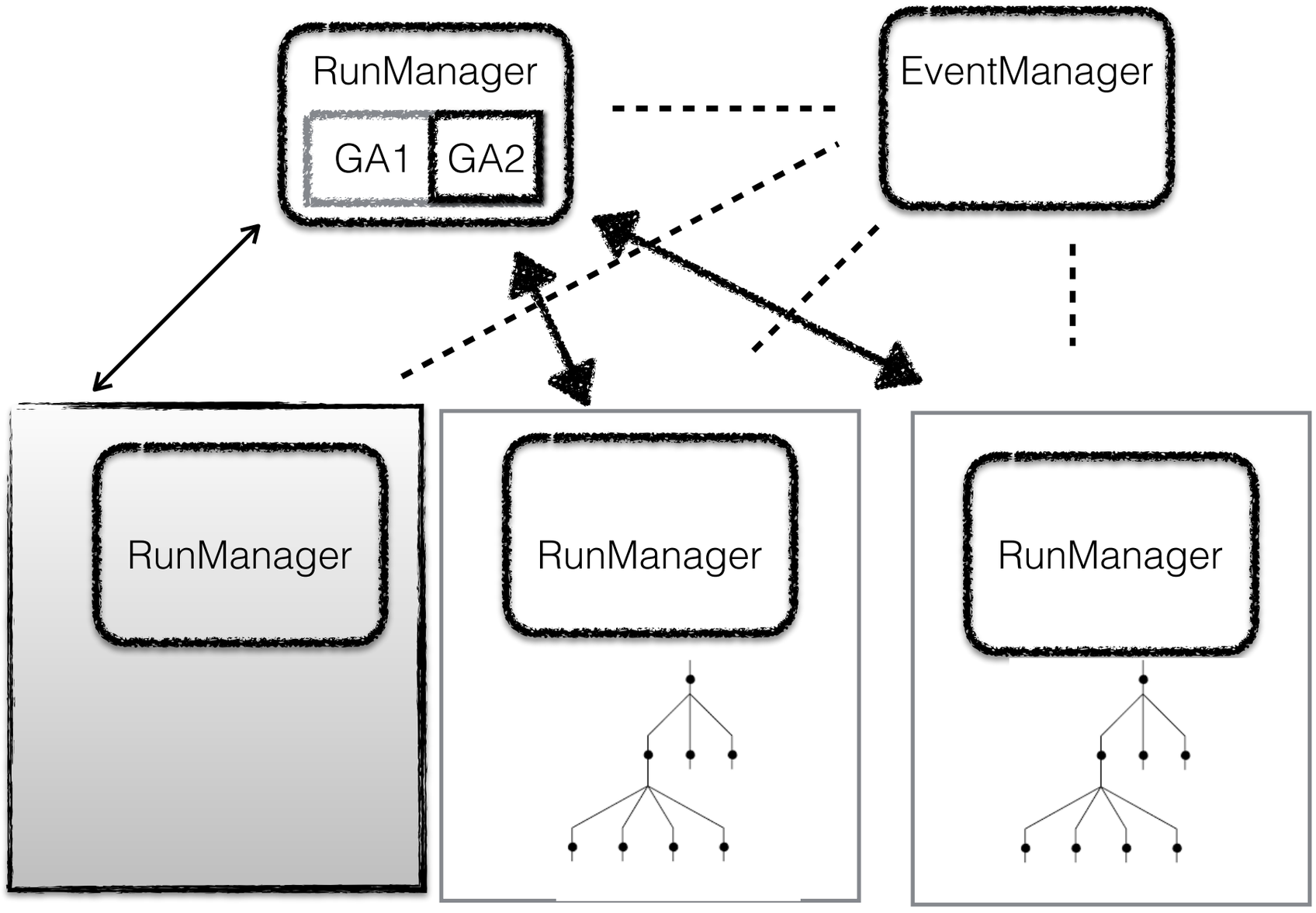
Tuning homogenous cluster case



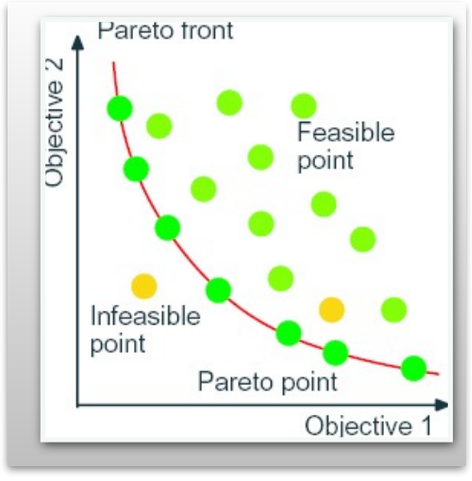
Expected output:

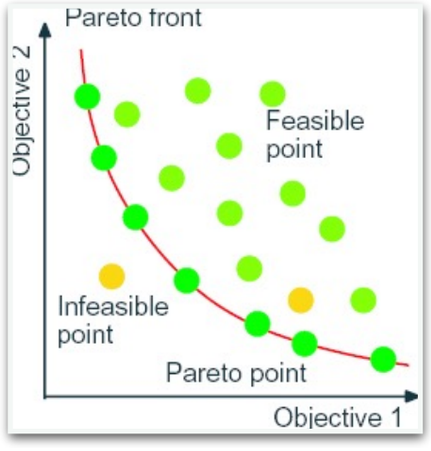


Tuning heterogeneous cluster (manually labeled)

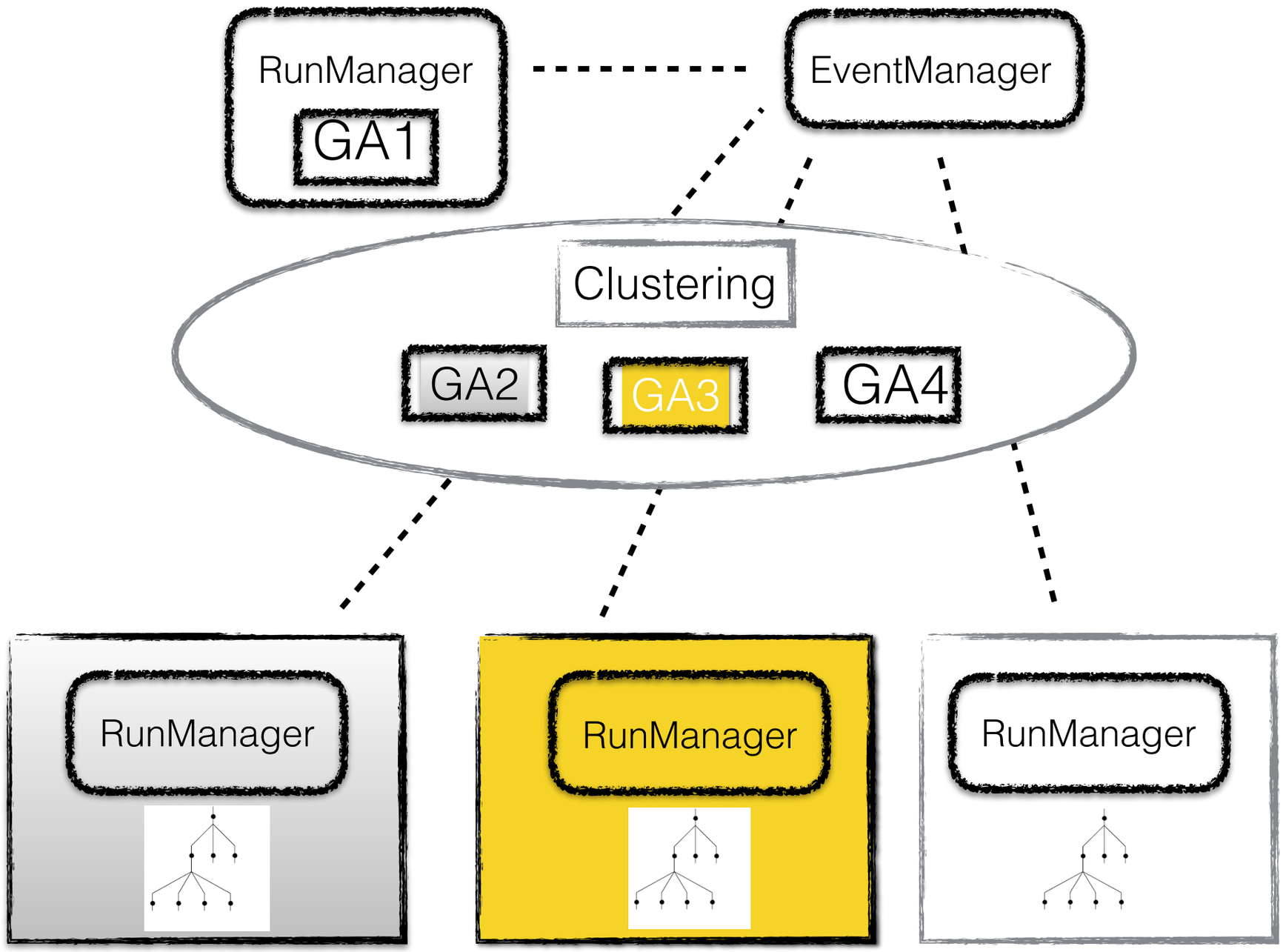


Expected output:



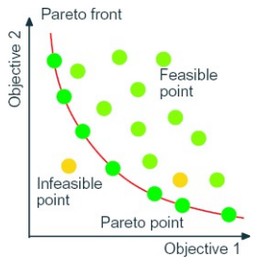


Tuning heterogeneous cluster (clustering)

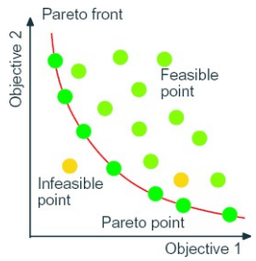


Expected output:

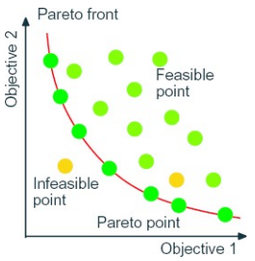
Label 1



Label 2



Label 3



. . . . .

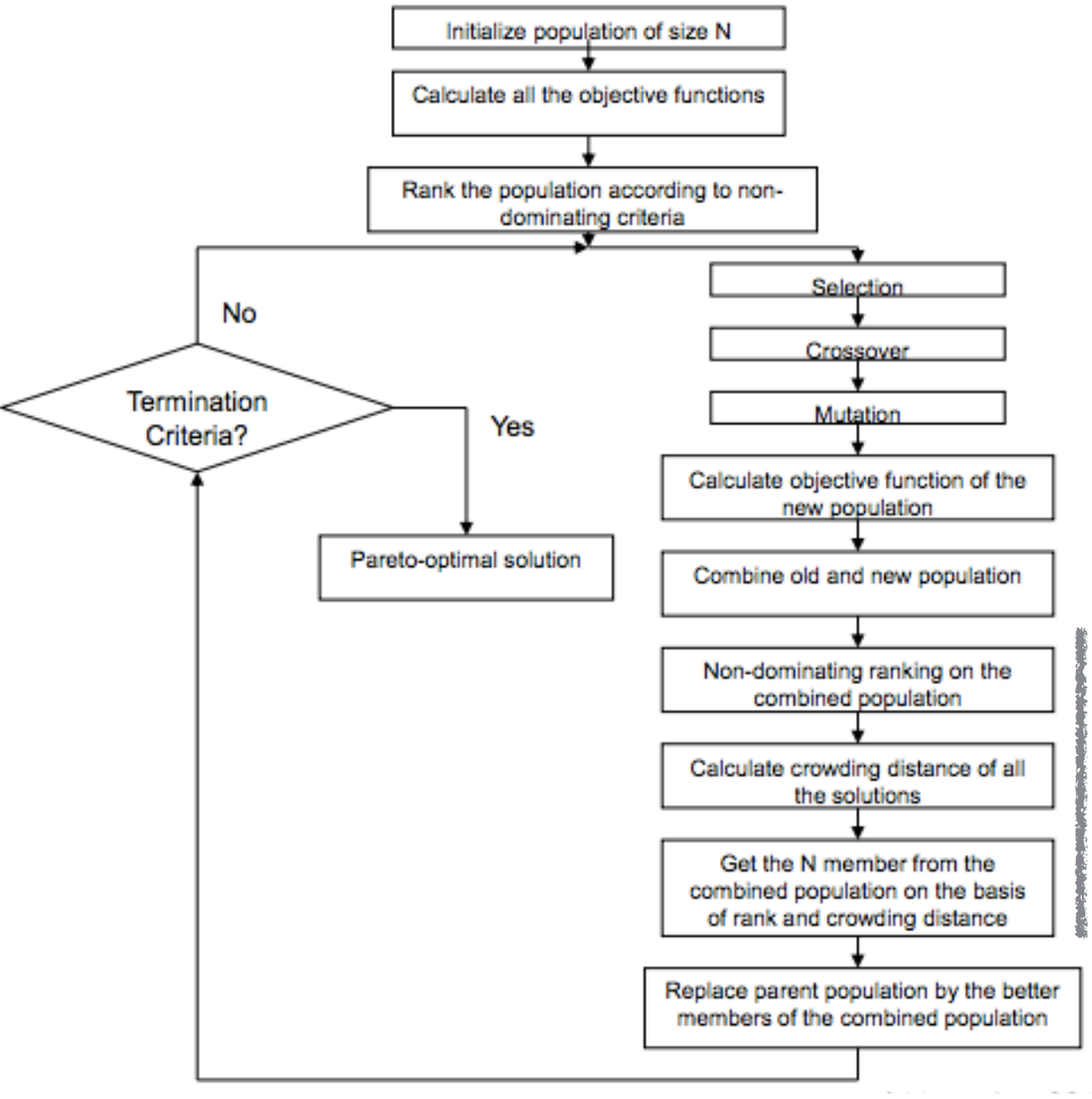
Conclusions

- Unsupervised learning methods and heuristic algorithms are powerful mechanism for optimisation and pattern detections for black-box functions;

- Performance optimisation of large exascale job processing could be simplified using machine learning methods.

Backup

NSGA-II vs NSGA-III



Input: H structured reference points Z^s or supplied aspiration points Z^a, parent population P_t
Output: P_{t+1}

$S_t = \emptyset, i = 1$
$Q_t =$ Recombination+Mutation(P_t)
$R_t = P_t \cup Q_t$
$(F_1, F_2, \ldots) =$ Non-dominated-sort(R_t)
repeat
  $S_t = S_t \cup F_i$ and $i = i + 1$
until $|S_t| \geq N$
Last front to be included: $F_l = F_i$
if $|S_t| = N$ **then**
  $P_{t+1} = S_t$, break
else
  $P_{t+1} = \cup_{j=1}^{l-1} F_j$
  Points to be chosen from F_l: $K = N - |P_{t+1}|$
  Normalize objectives and create reference set Z^r: `Normalize` $(\mathbf{f}^n, S_t, Z^r, Z^s, Z^a)$
  Associate each member s of S_t with a reference point: $[\pi(\mathbf{s}), d(\mathbf{s})] =$`Associate` (S_t, Z^r)
  % $\pi(\mathbf{s})$: closest reference point, d: distance between s and $\pi(\mathbf{s})$
  Compute niche count of reference point $j \in Z^r$: $\rho_j = \sum_{\mathbf{s} \in S_t/F_l} ((\pi(\mathbf{s}) = j)\ ?\ 1\ :\ 0)$
  Choose K members one at a time from F_l to construct P_{t+1}: `Niching` $(K, \rho_j, \pi, d, Z^r, F_l, P_{t+1})$
end if

CMAES

Step 1: Initialize the CMA-ES Parameters

D ← no. of dimensions

λ ← Offspring population size (4.0 + 3.0 Log(D))

μ ← Parent population for next generation (floor(λ/2))

σ_{Start} ← Initial standard deviation.

c_{cov} ← Covariance learning rate

Step 2: **While** stopping criterion is not met **do**

Step 3: Update the **Covariance Matrix** $C^{(g+1)}$ (see Covariance Matrix Adaption)

$$\boldsymbol{C^{(g+1)} \leftarrow (1 - c_{cov})C^{g} + \frac{c_{cov}}{\mu_{cov}} P_c^{(g+1)} P_c^{(g+1)^T} + c_{cov}\left(1 - \frac{1}{\mu_{cov}}\right)}$$

$$\times \sum_{i=1}^{\times} w_i \left(\frac{X_{1:\lambda}^{(g+1)} - M^{g}}{\sigma^{(g)}}\right)\left(\frac{X_{1:\lambda}^{(g+1)} - M^{g}}{\sigma^{(g)}}\right)^{T}$$

Step 4: Update the Step Size σ^{g} (see Step Size adaption)

$$\sigma_{g+1} \leftarrow \sigma_g \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{||P_\sigma||}{E\,N(0,I)} - 1\right)\right)$$

Step 5: Generate Sample Population for generation g+1

$$x_k^{(g+1)} \sim N\left(M^{(g)}, (\sigma^{(g)})^2\, C^{(g)}\right) \quad \text{for k = 1,…,}\lambda$$

Step 6: Update the mean for generation g+1

$$m^{(g+1)} \leftarrow \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g+1)}$$

Step 7: Update best ever solution

Step 8: End **While**

Fixed points

We consider the GA is as the discret dynamic system

$$\vec{p}_{k+1} = \hat{G}\,[\,\vec{p}_k\,]$$

or

$$\vec{p}_{k+1} = \hat{G}^k\,[\,\vec{p}_0\,], \quad \hat{G}^k = \hat{G} \circ \hat{G} \circ .... \circ \hat{G}\ (k{-}\text{times})$$

The fixed point is defined as

$$\vec{p}^{\star} = \hat{G}\,[\vec{p}^{\star}]$$

The fixed points for the GA will be investigated in monographs by Reeves and Rowe (2003). Gedeon (2010) had shown that the GA has the finite set of the hyperbolic fixed points.

The behavior of the map G in the neighborhood of a fixed point $\vec{p}^{\star}$ is determined by the eigenvalues of the linearization $LG\,[\vec{p}^{\star}]$

$$\Delta p'_i = p'_i - p^{\star}_i = \frac{\partial G_i}{\partial p_j}\,[\vec{p}^{\star}]\,(p_j - p^{\star}_j) = LG_{i,j}\,[\vec{p}^{\star}]\,\Delta p_j$$

If all eigenvalues of LG have absolute value less than one, then all iterates starting near $\vec{p}^{\star}$ converge to $\vec{p}^{\star}$. If there is at least one eigenvalue with absolute value greater than one, then almost all iterates will diverge from $\vec{p}^{\star}$. For the hyperbolic fixed points the LG has no eigenvalues lie on the unit circle in the complex plane.