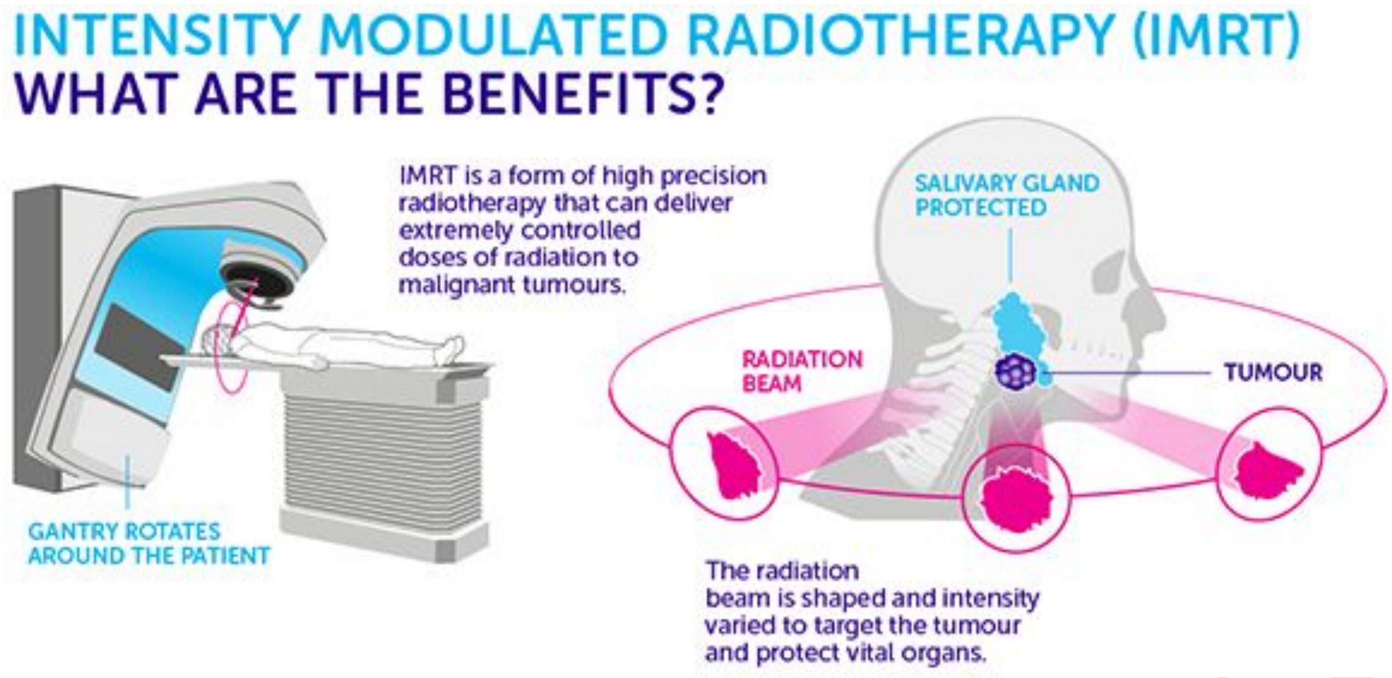
Exploiting Multicore Servers to Optimize IMRT Radiotherapy Planning

J.J. Moreno[1], **Savíns Puertas-Martín[1,2]**, J.L. Redondo[1], P.M. Ortigosa[1], E.M. Garzón[1]

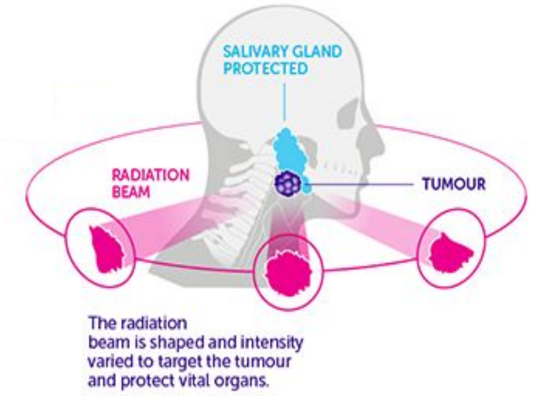
[1]*Supercomputación – Algoritmos (SAL), Universidad de Almería, (ceiA3), Almería, Spain*
[2]*Chemoinformatics Research Group, University of Sheffield, United Kingdom.*

1st workshop about High-Performance e-Science

- Intensity modulated radiation therapy (IMRT) is an effective cancer treatment that involves delivering doses of radiation to a tumour while sparing the surrounding tissues.



- Physicists in each planning must solve a **complex optimization problem**, in which the optimal adjustment of the intensity of all radiation beams is sought, in order to **maximize the dose in the tumor areas (PTV)** and **decrease it in the organs at risk (OAR).**

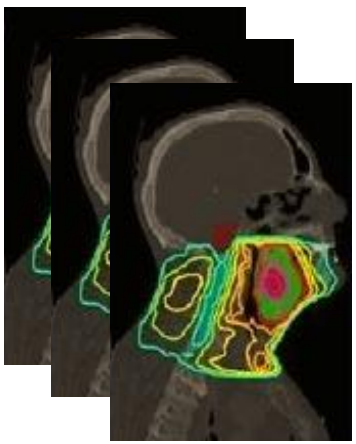


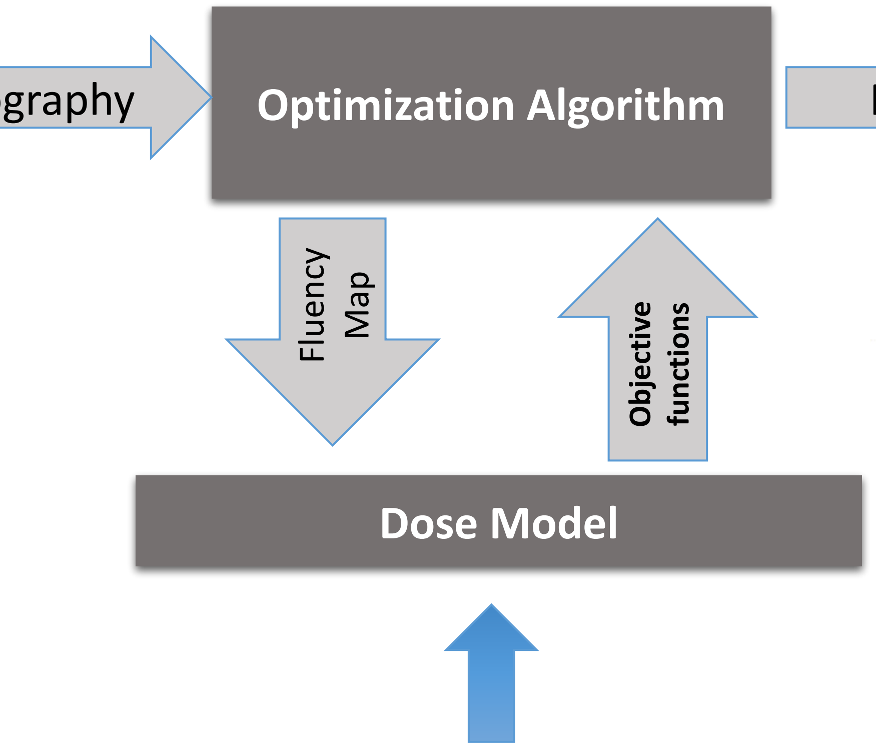
Computational tools that solve these schedules in a way that is:

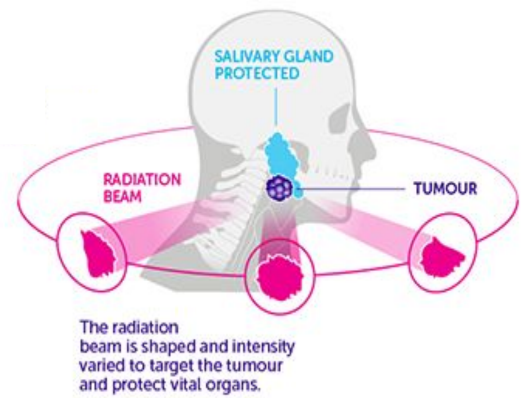
- ✔ **Automatic**
- ✔ **Accurate**
- ✔ **Fast**

Background. Present problem

- To solve this, they have to deal with the following workflow:







EUD Model (Equivalent Uniform Dose)

- Clinically meaningful RT plans can be obtained by computing the **maximum** of the following function:

$$F(x,\phi) = \prod_{t\in T} \frac{1}{1+\left(\frac{EUD_t^0}{EUD_t(x,a_t)}\right)^{n_t}} \cdot \prod_{r\in R} \frac{1}{1+\left(\frac{EUD_r(x,a_r)}{EUD_r^0}\right)^{n_r}}$$

Tumors | Organs at Risk

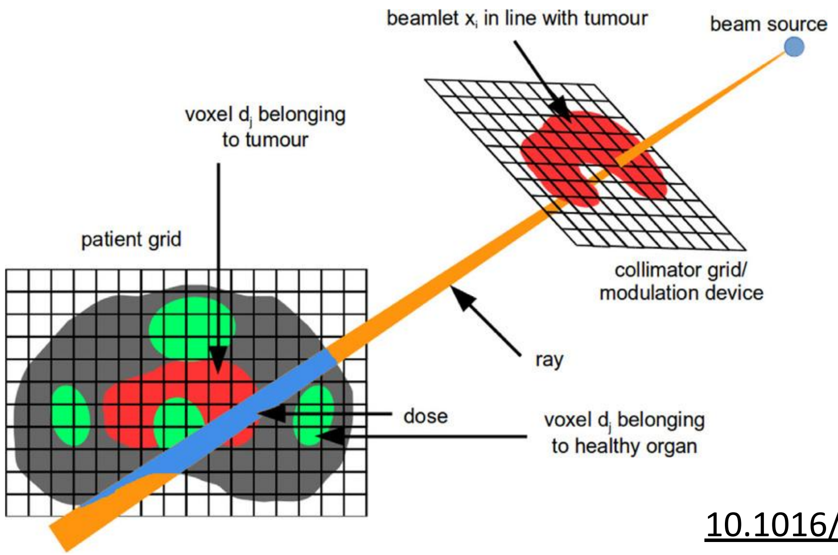
- EUD_t^0 is the prescribed dose for t-th PTV,
- EUD_r^0 is the maximum uniform dose at r-th OAR;
- n_r and n_t express the importance of the prescriptions for the corresponding structure;

ϕ represents the set of parameters involved in the F definition, i.e. ϕ is an instance of parameters n_t, n_r, a_t, a_r and EUD_r^0 with $t \in T, r \in R$.

EUD Model

- In EUD, radiation effects in a Planning Target Volumes (PTV) or an Organ At Risk (OAR), both referred as structure *s*, are evaluated by the following function that aggregates these effects over all voxels belonging to structure *s*:

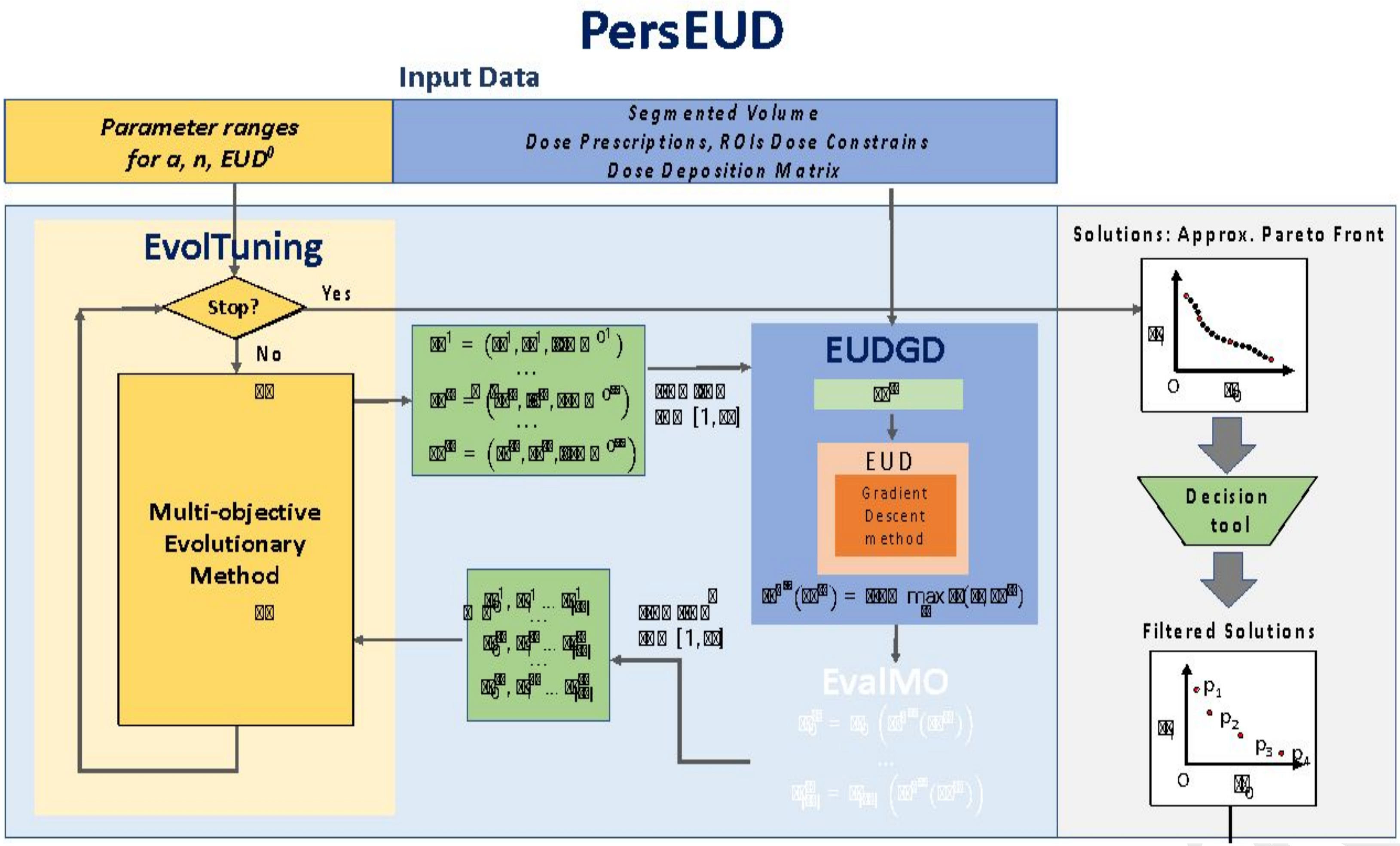
$$EUD_s(x, a_s) = \left(\frac{1}{|M_s|} \sum_{j \in M_s} d_j(x)^{a_s} \right)^{\frac{1}{a_s}}$$



10.1016/j.dib.2017.03.037

PersEUD

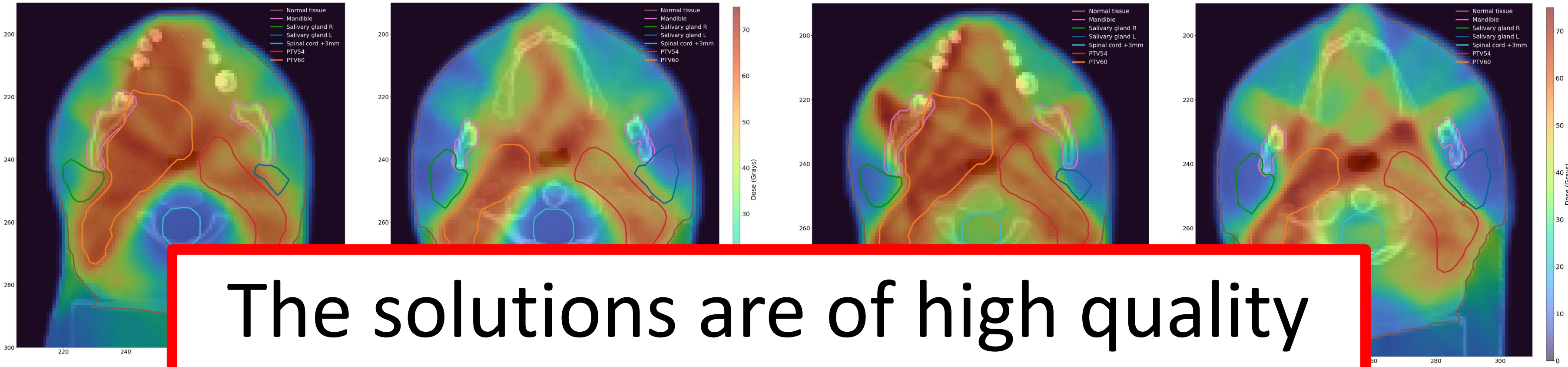
- We have an application of the EUD-based gradient descent technique capable of generating clinically acceptable plans.



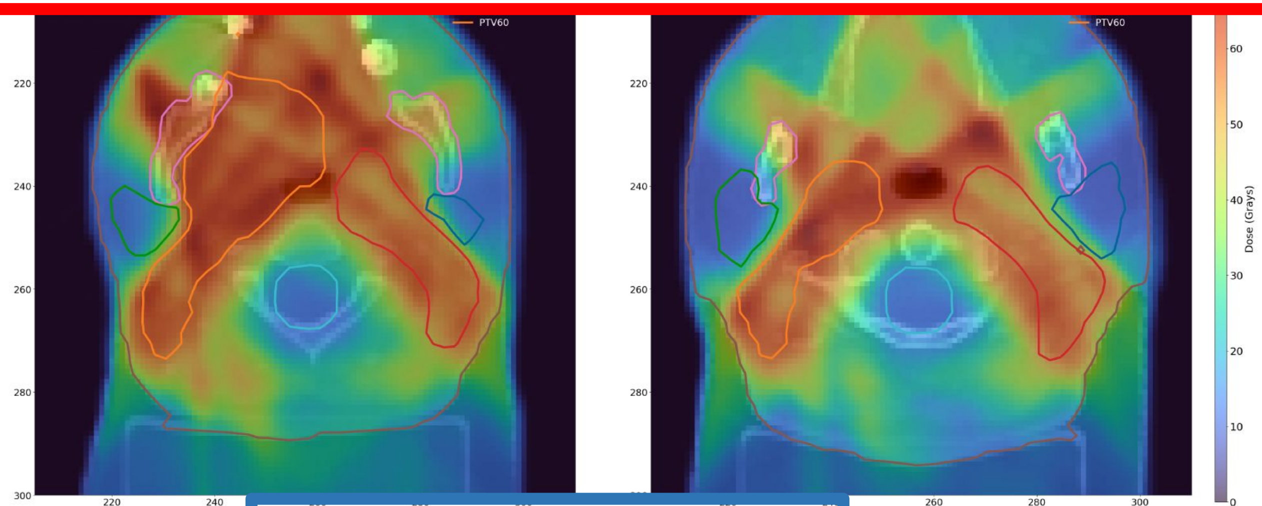
What do the solutions really represent?

Spinal cord

Salivary glands



The solutions are of high quality but we want to get them faster!!!



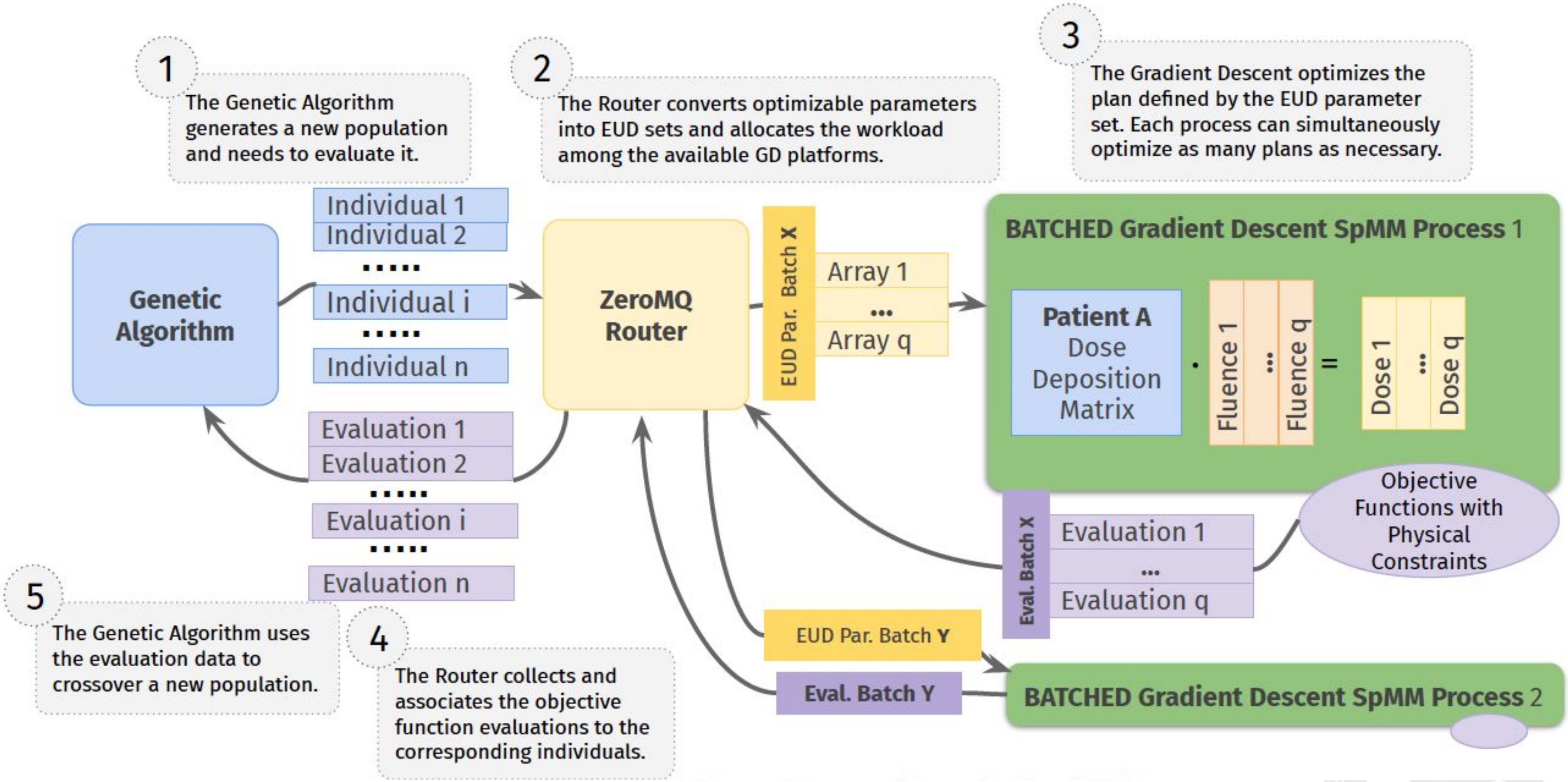
Spinal cord + Salivary glands

J.J. Moreno, **Savíns Puertas-Martín**, J.L. Redondo, P.M. Ortigosa, E.M. Garzón

Outline

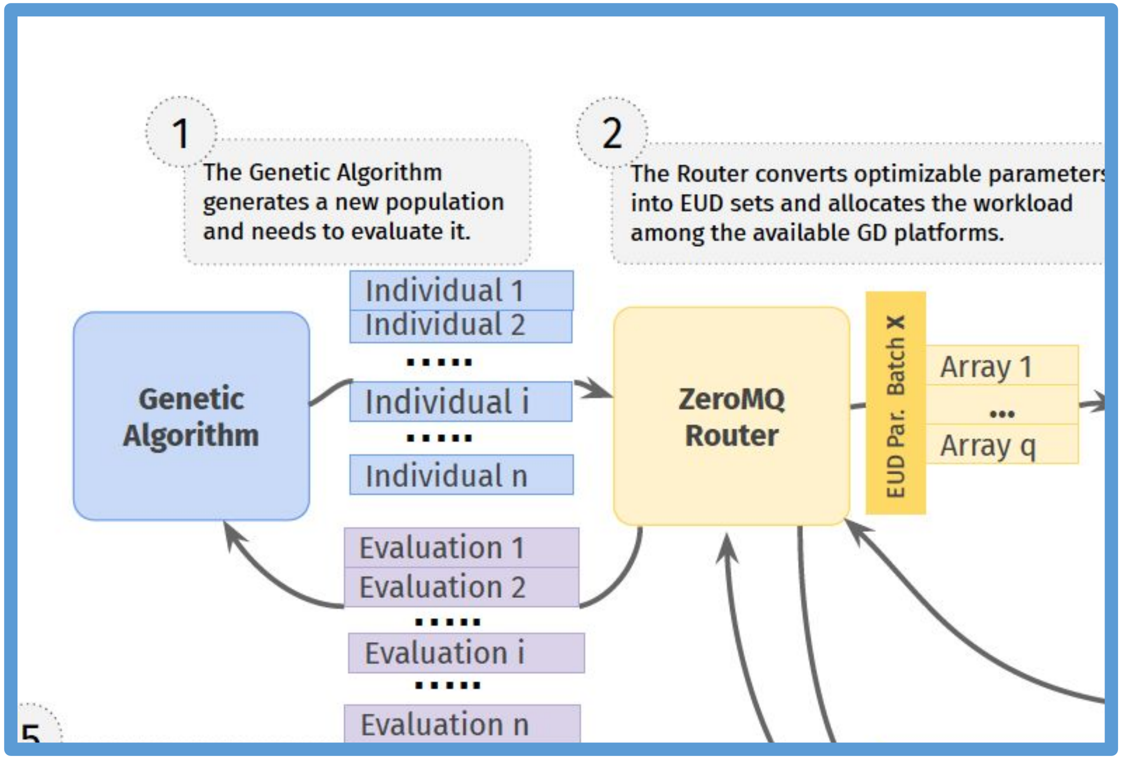
Process acceleration

- How can operations be accelerated?



Process acceleration

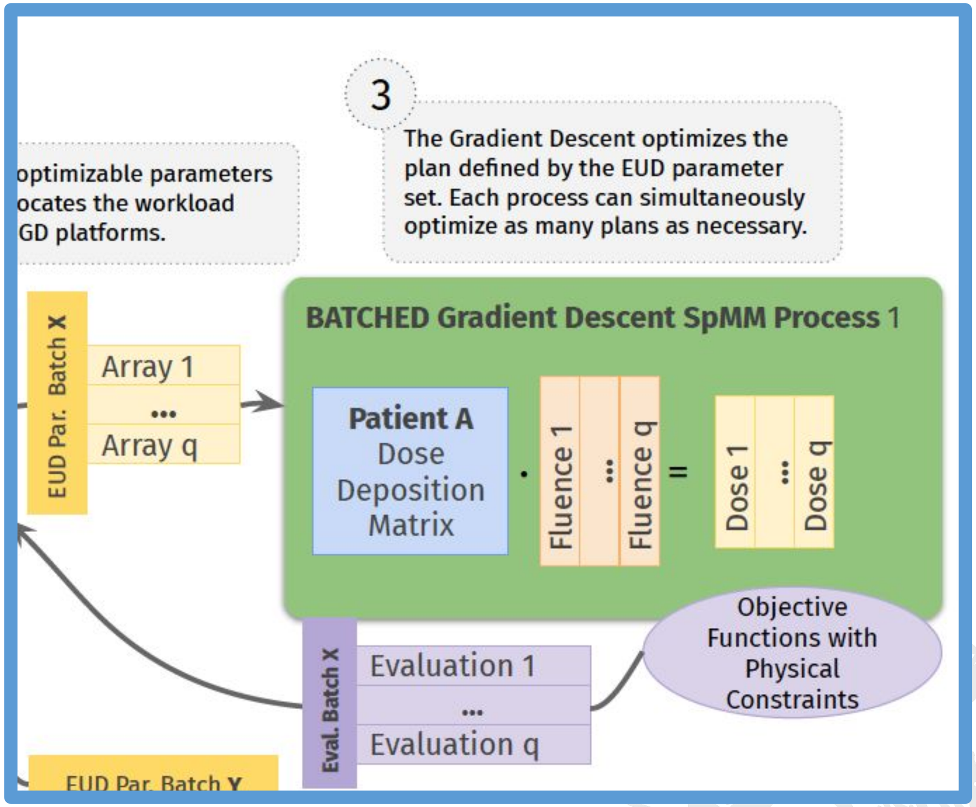
- How can operations be accelerated?



- The individuals generated by the genetic algorithm are sent to the Router.
- The router groups them in batches and stores the correspondence when it receives the results back.

Process acceleration

- How can operations be accelerated?
- The batch is received by the DG. A matrix-matrix product (BLAS level 3) is performed instead of array-matrix (level 2).
- All operations are parallelised with Intel oneAPI MKL 2023.0.0.
- The most time is consumed by the product of matrix D and D^t (large deposition matrix).



Outline

Data

Parameter	Patient A	Patient B	Patient C
Beam angles	9	9	9
Beamlets (N)	25,298	33,911	30,265
Voxels (M)	145,965	160,786	94,647
D nonzeros	67,544,881	106,792,251	64,991,188
Organs At Risk (OARs)	9	9	9
Planning Target Volumes (PTVs)	3	2	3
PTV_0 pr. dose (Gy)	54.0	59.4	54.0
PTV_1 pr. dose (Gy)	60.0	66.0	60.0
PTV_2 pr. dose (Gy)	67.5	-	66.0

Note the size of D

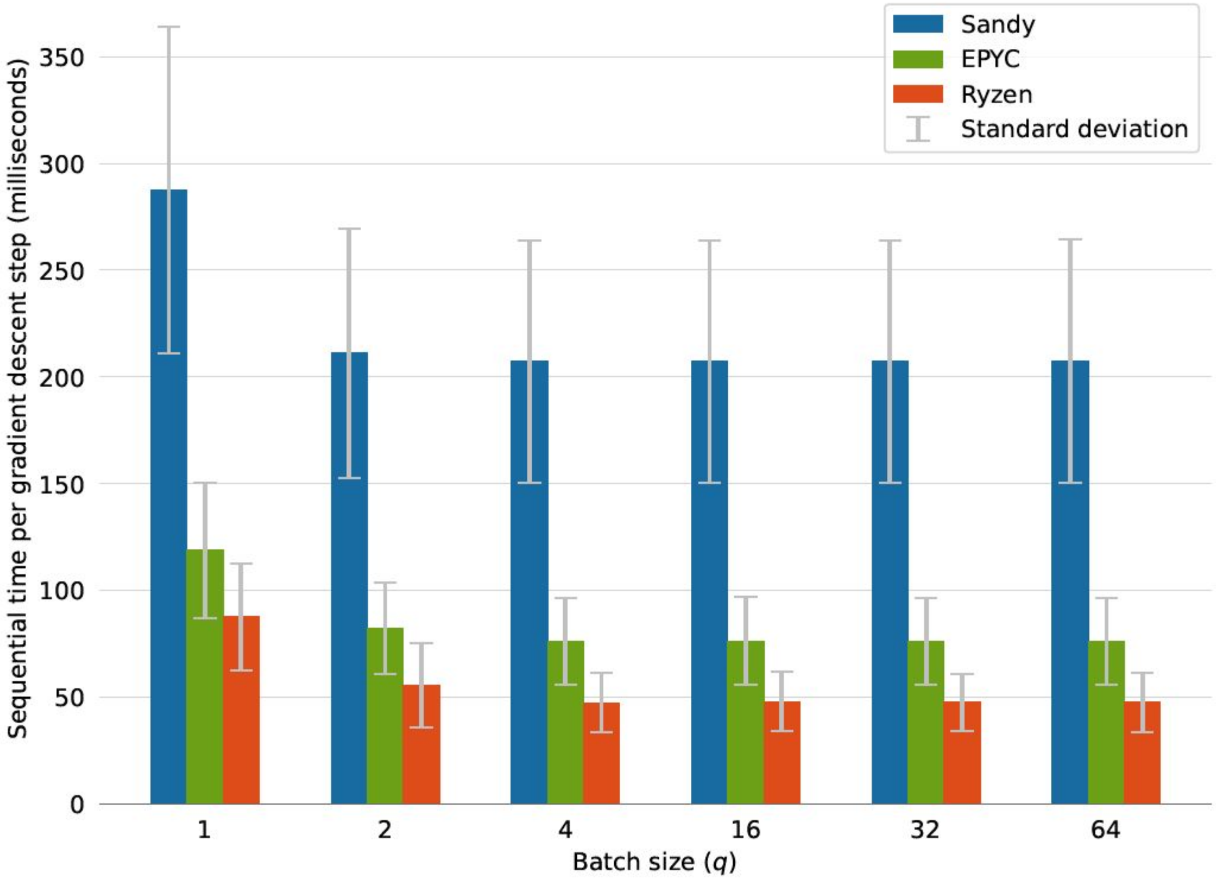
Two types of experiments have been carried out:

- Influence of batch size in sequential.
- Performance analysis of parallel versions on the following platforms:

Platform	CPU	Cores	RAM
Sandy	Intel Xeon E5-2650	16 (2 sockets)	64 GB DDR3
EPYC	AMD EPYC 7642	96 (2 sockets)	512 GB DDR4
Ryzen	AMD Ryzen 9 5950X	16 (1 socket)	32 GB DDR4

Outline

Influence of lot size on sequential

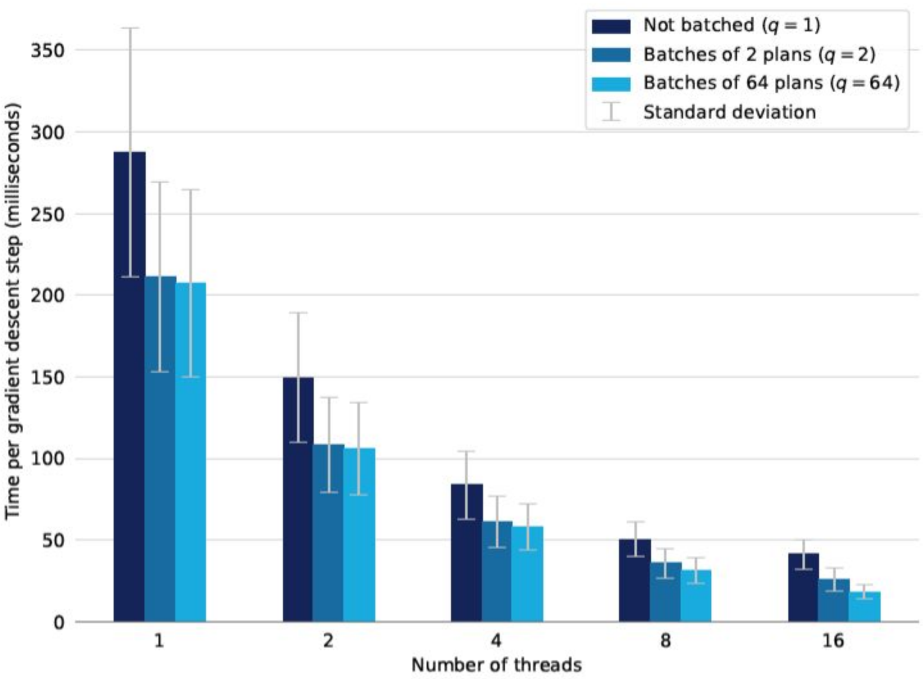


Average time of the 3 patients for each platform and different batch size. There is a considerable reduction in run time when applying a batch size 2, but thereafter the improvement is practically 0.
Total time = t*2000*pop*iter/1000/3600/24

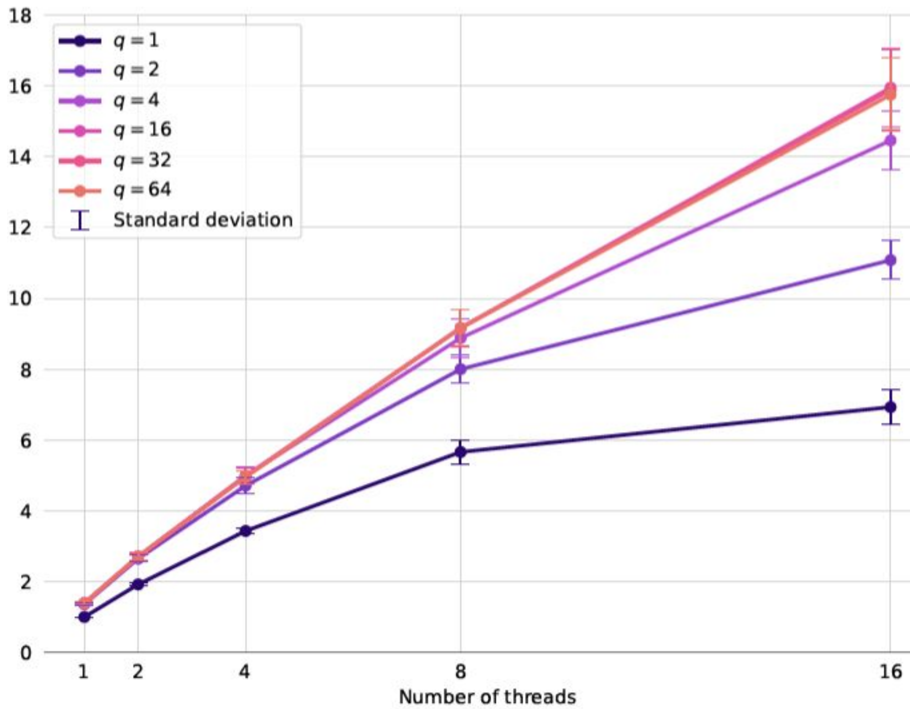
Results. Sandy Platform

Intel Xeon E5-2650
16 cores (2 sockets)
64GB DDR3

Execution time per batch size and threads



Speedup

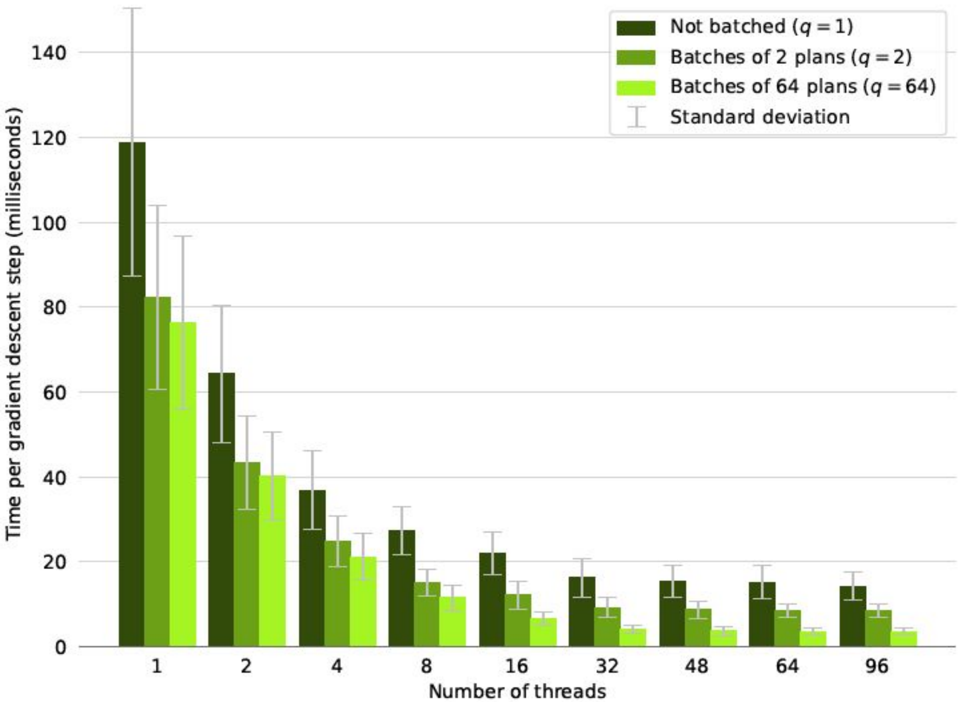


The time given is the time of one step of the gradient. This value has to be multiplied by 2000 iterations of the DG times the number of individuals and the number of iterations of the genetic.

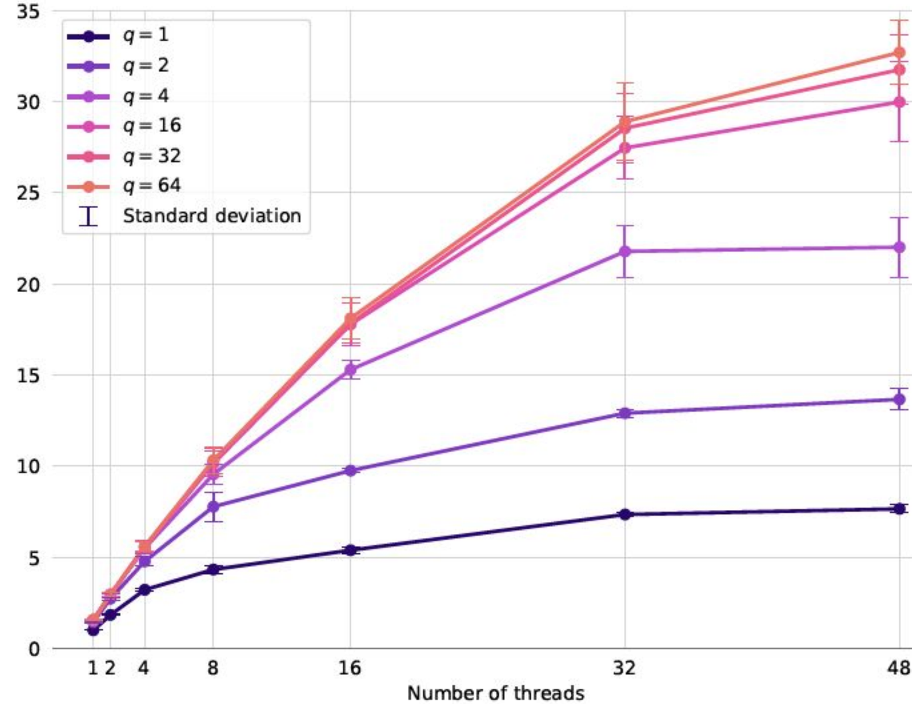
Results. EPYC Platform

AMD EPYC 7642
96 cores (2 sockets)
512GB DDR4

Execution time per batch size and threads



Speedup

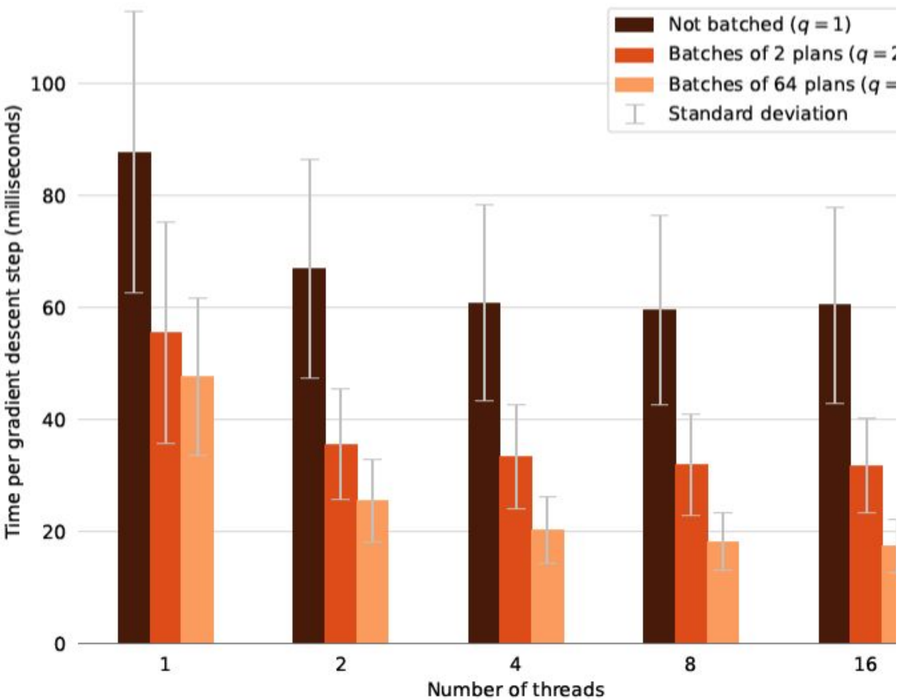


The time given is the time of one step of the gradient. This value has to be multiplied by 2000 iterations of the DG times the number of individuals and the number of iterations of the genetic.

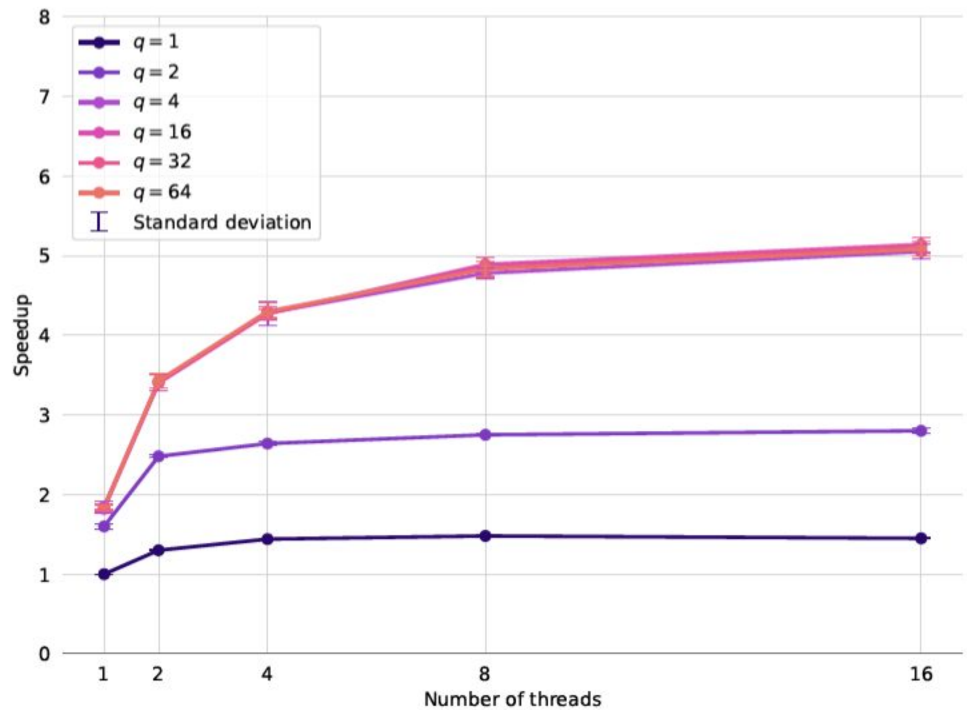
Results. Ryzen Platform

AMD Ryzen 9 5950X
16 cores (1 sockets)
32GB DDR4

Execution time per batch size and threads



Speedup



The time given is the time of one step of the gradient. This value has to be multiplied by 2000 iterations of the DG times the number of individuals and the number of iterations of the genetic.

Outline

Conclusions

- In this paper we have presented a new design that allows PersEUD to be accelerated using two approaches:
  - Parallelisation using batches.
  - Parallelisation using threads.

- We have tested our method on three different platforms with different architectures and we have evaluated its performance with different batch sizes and threads.
- The results show that the execution time is considerably reduced, making it feasible to use in real environments.
  - Ex: 128 individuals, 50 iterations and 2000 DG steps.
    - Total time = t*2000*pop*iter/1000/3600/24
    - Sequential without batches: 422.51 hours
    - Parallel version with 96 threads and 64 individuals per batch: 12.55 hours.

Supercomputación - Algoritmos

UNIVERSIDAD DE ALMERÍA

Exploiting Multicore Servers to Optimize IMRT Radiotherapy Planning

J.J. Moreno[1], **Savíns Puertas-Martín[1,2]**, J.L. Redondo[1], P.M. Ortigosa[1], E.M. Garzón[1]

[1]Supercomputación – Algoritmos (SAL), Universidad de Almería, (ceiA3), Almería, Spain
[2]Chemoinformatics Research Group, University of Sheffield, United Kingdom.

1st workshop about High-Performance e-Science