Metaheuristic algorithms in nuSTORM and MICE

Ao Liu
Fermilab

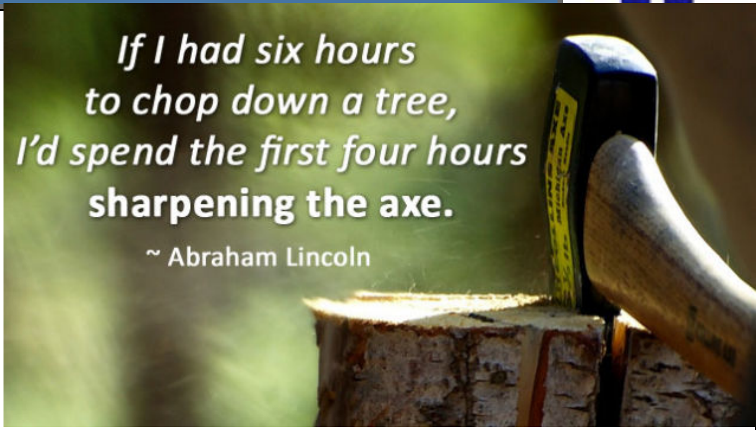
Outline

- Introduction to metaheuristic algorithms
  - – Genetic algorithm
  - – Simulated Annealing
  - – Implementing MPI in the above algorithms
- Optimization studies in nuSTORM and MICE
  - – Motivation
  - – Optimization objectives and setup
  - – Results
- Summary

What is a metaheuristic algorithm

- A **heuristic search** finds solutions for a problem by *trials and errors.*
  - Unlike deterministic algorithms such as golden section search, Gauss−Newton algorithm, greedy algorithm, etc;
  - It is nondeterministic, i.e. solutions are proposed by guesses;
  - Pros: efficient, free and thorough;
  - Cons: inaccurate, can not guarantee to find the absolute optimum
- A **metaheuristic algorithm** is a strategy that guides a heuristic search
  - It is more advanced: a good guide to new trials results in faster and more thorough search in the parameter space, so that the absolute global optimum can be visited.

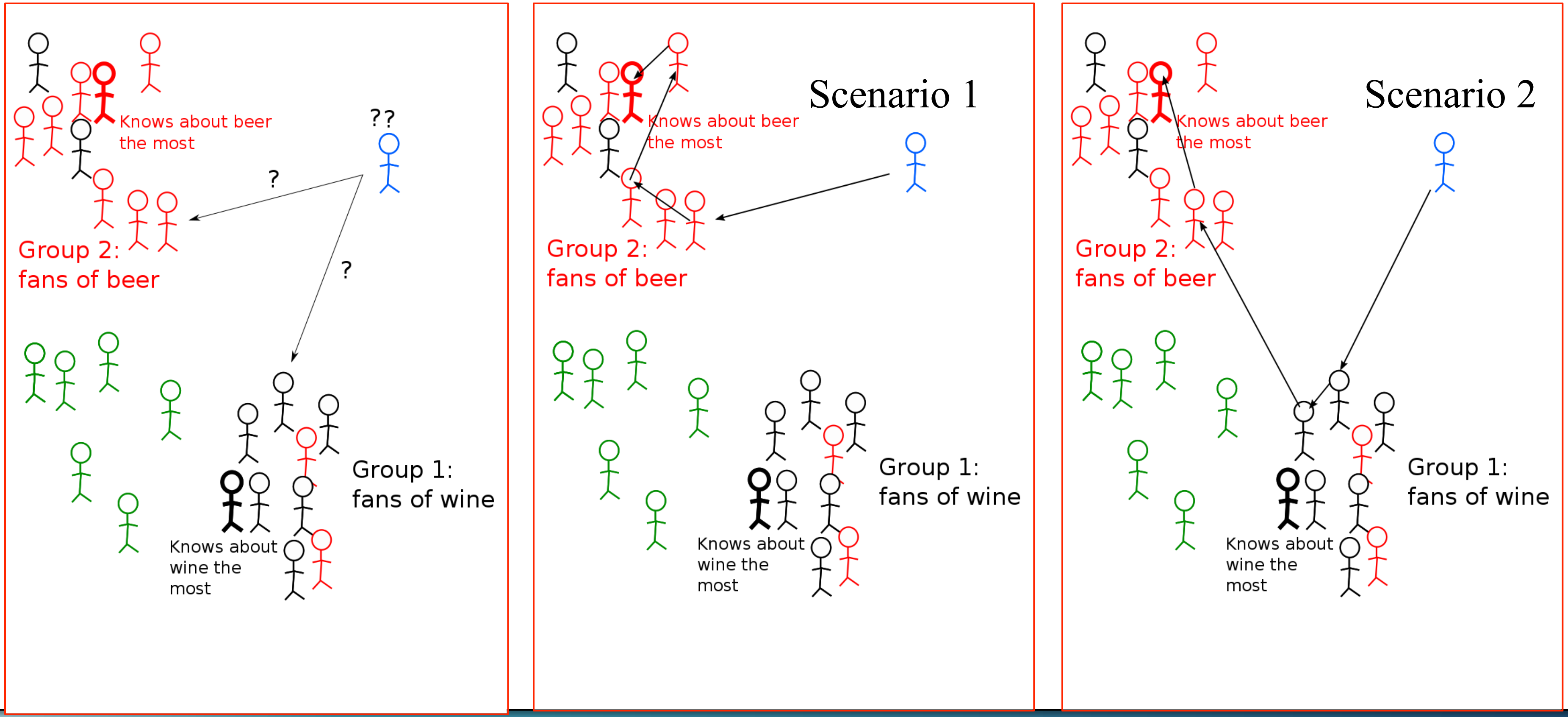
Properties of metaheuristic algorithms



- Not problem-specific
  - Generally can be easily adapted to any optimization problem
- Have mechanisms to avoid getting trapped in local optimum areas
- The more complicated a system is, the more advantageous it is to use metaheuristic algorithms
- There are many applications in accelerator physics
  - Cavity design, magnetic horn design, lattice design, target design optimizations
  - Commissioning and operation

A real life example of metaheuristic algorithms

- Suppose you are at a workshop reception where people taste different drinks
  - You want to find the person who knows about beers the most. Choose the next person you talk to by his/her neighbors



What is a genetic algorithm

- A **genetic algorithm** (or **GA**) is a metaheuristic algorithm;
- Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination).

Big but friendly

Aggressive but small

Survives well, but not always. What if their genes crossover?

Many generations

Tibetan mastiff

How GA works

- Starts with a population of randomly generated *individuals*;
  - A “known to be good” individual can be put in the population as a seed;
  - One individual is a set of variable values ($\boldsymbol{x_i}=(x_{0i}, x_{1i}, x_{2i}, \ldots,)$)
- In each *generation*, the *fitness* of every individual in the population is evaluated, a part of the individuals are selected, and modified by *crossover* to form a new population.
  - *Crossover* is a calculator that generates new values (usually two) based on two old values: can be binary or real value;
  - Mutation is often added: to randomly explore parameter space with a probability;
  - Generally, two parents produce two children, in some versions, elite parents have more children

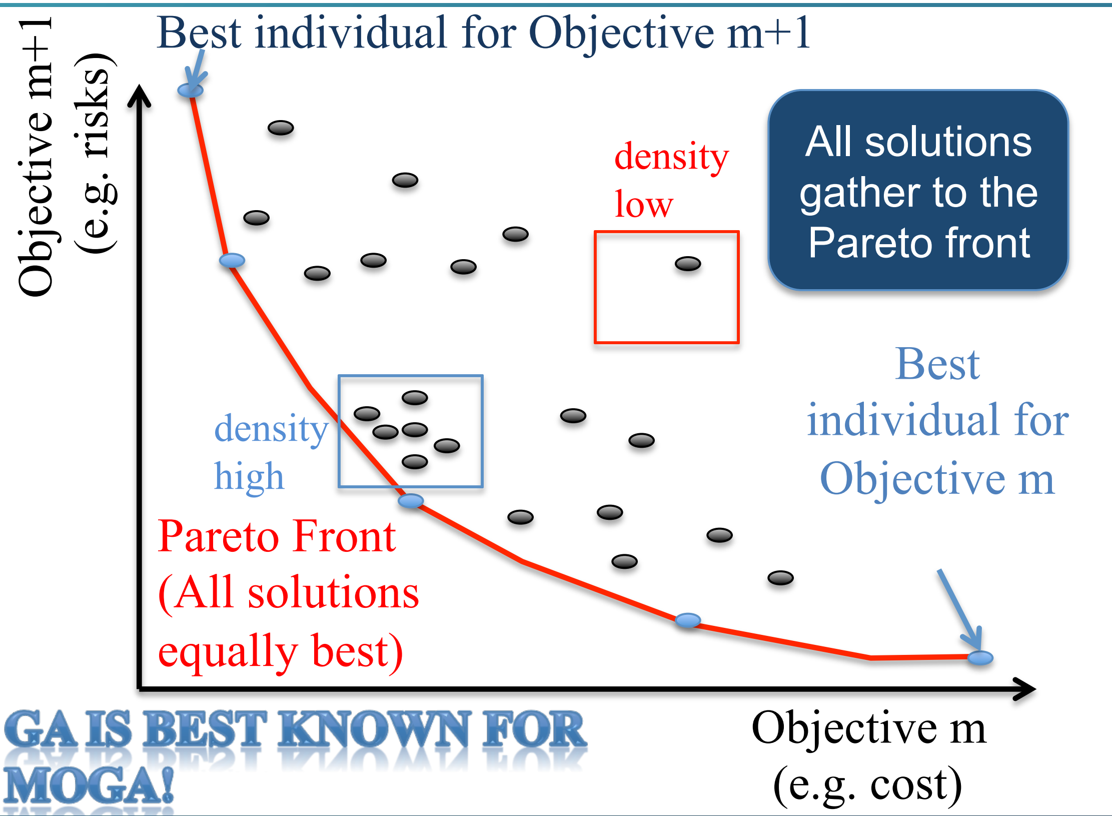
How GA works (cont'd)

- The new population is mixed with the old population
  - Individuals who do not generate children from the old population are abandoned;
  - The mix forms a new generation
- Usually, the algorithm terminates when either a maximum number of generations has been produced, or the fitness level has stopped increasing for the population.
  - There are ways to avoid premature termination, like a judgment day, or shuffle all parents
- There are lot of advanced improvements to the algorithm! It is still an area that is actively studied.

Multi-objective GA (MOGA)

- Introduces *pareto front* and *dominance*
  - Multiple objectives: McGrady is faster but shorter, Yao is taller but slower. They can not dominate each other so they are equally good

- In the *decision space*, any two of the objectives form a pareto front that can be visualized as in the right plot
- Introduces *crowding distance*, which measures the population density in the decision space
  - Can be used to choose parents



GA IS BEST KNOWN FOR MOGA!

My GA code

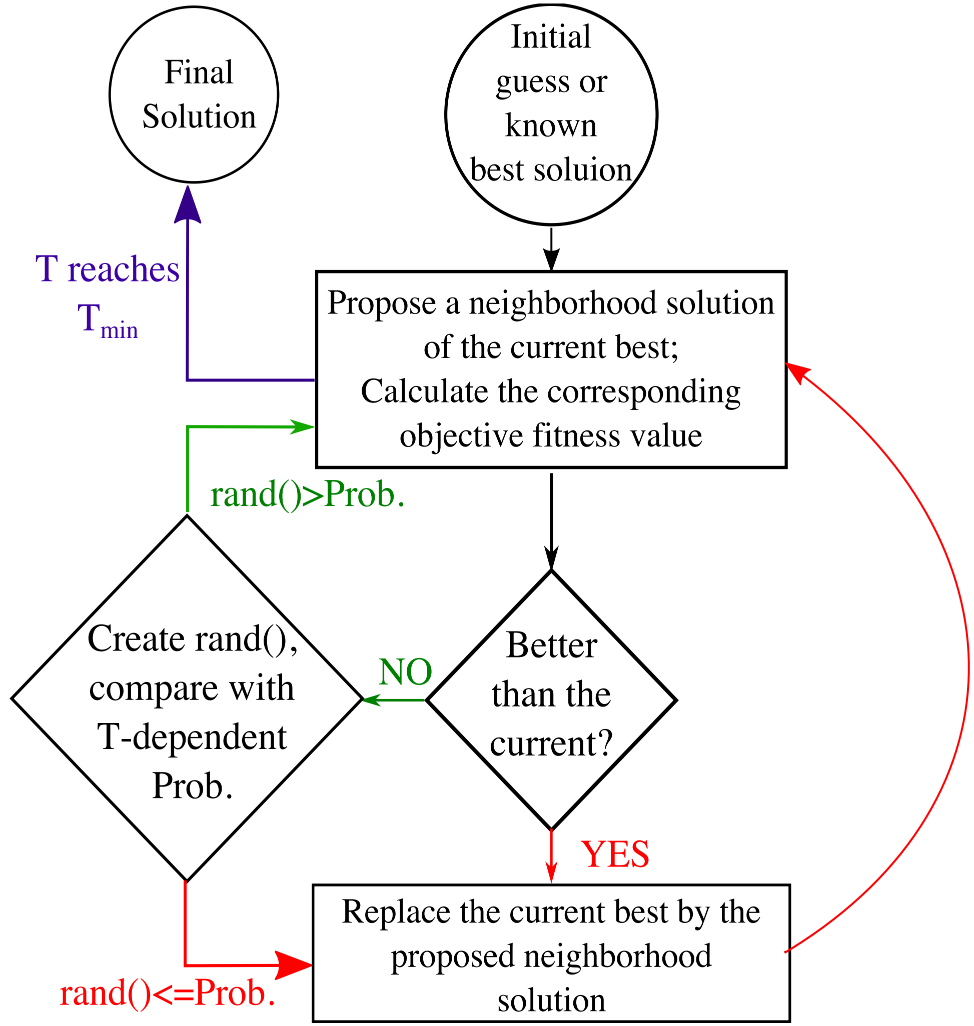
- Supports both MOGA and single objective GA (SOGA)
- Written in Python, with MPI implemented
  – Computes in parallel and gathers information together;
  – Implemented at NERSC, a national scientific computing center at LBNL;
  – Certainly any platform with scientific Python and MPI;
- Connects with other programs – GA provides solutions, other programs provide results
- Has the mechanisms mentioned above to deliver high performance optimization

What is a Simulated Annealing (SA)

- MOGA is powerful but SOGA is usually not very CPU efficient, especially at the converging stage;
- SA is a metaheuristic algorithm that can be based on both *trajectory* information and *population* information
  - Random walk + iterative improvement
  - New solution can replace the old solution when
    - New solution is better;
    - New solution is worse, with a probability of $p(T,s',s) = \exp\left(\frac{f(\vec{x}_{new}) - f(\vec{x}_{old})}{T}\right)$
  - T is the *temperature*, which reduces in each iteration;
    - When T is very low, the solution "freezes".

More about SA

- Pros:
  - Integrates local searches, and stochastic searches;
  - Fast, easy to converge
- Cons:
  - No memory: once new solution replaces the old one, the previous result (possibly better result) will be lost;
  - Relies on the temperature: the probability function works only if the fitness function and temperature are chosen wisely
    - e.g. fitness value ~= 1, but temperature ~= 1000, it never works;
    - Requires some knowledge of the system

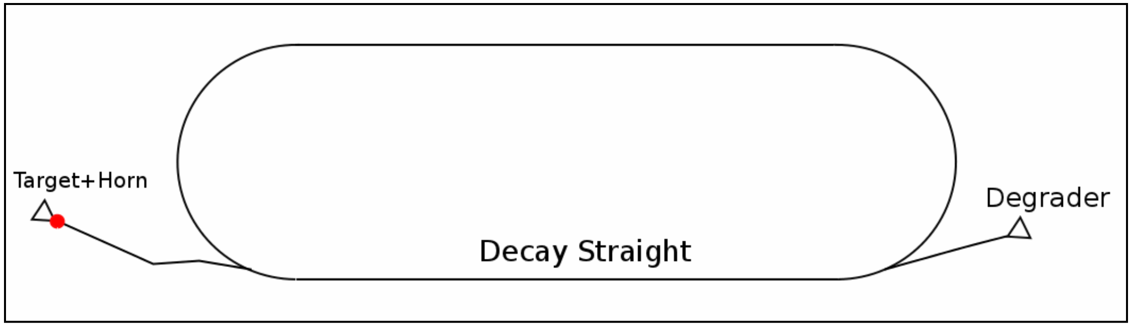


My SA code

- Use MPI controlled population to build memory about the global optimum
  - Use a rank (one MPI worker) to focus on performing local searches around the global optimum, never shifts to a worse solution;
  - Other ranks perform individual searches with their own path;
  - In each generation, the information from each rank is gathered and the global optimum is updated
- Python as the GA code, platform independent

Optimization cases in nuSTORM

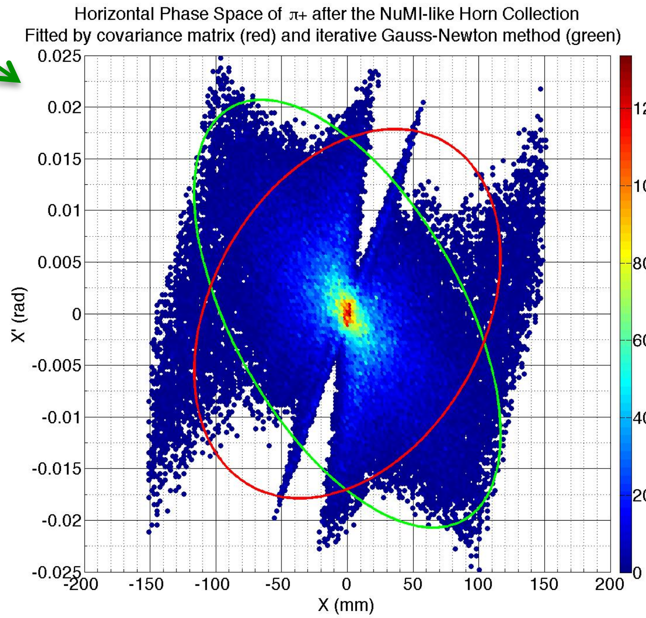
- neutrinos from STORed Muons (nuSTORM)
  - The simplest realization of a neutrino factory: provide a clean and precisely known ν_e source from muon decay in a storage ring



  - Need to re-optimize the magnetic horn to focus the pions into a beamline acceptance, rather than conventionally point-to-parallel;

Optimization of a nuSTORM horn

- Use MOGA to optimize the yield of muons in both:
  - Momentum acceptance of the beamline;
    - Includes the pion momentum acceptance and pion decay kinematics
  - Phase space acceptance of the beamline
    - Gauss-Newton phase space ellipse fitting used to guide the optics design
- Due to the nonlinear effects, the acceptance of the whole beamline can not be represented by a simple mathematical expression of the two.
- MOGA can optimize them simultaneously



Optimization work flow

GA starts, a number of random individual horns produced as the first generation

Model the B-field in the horns, based on the parameters of each horn

Track π+ in the horns, calculate fitness values for each individual

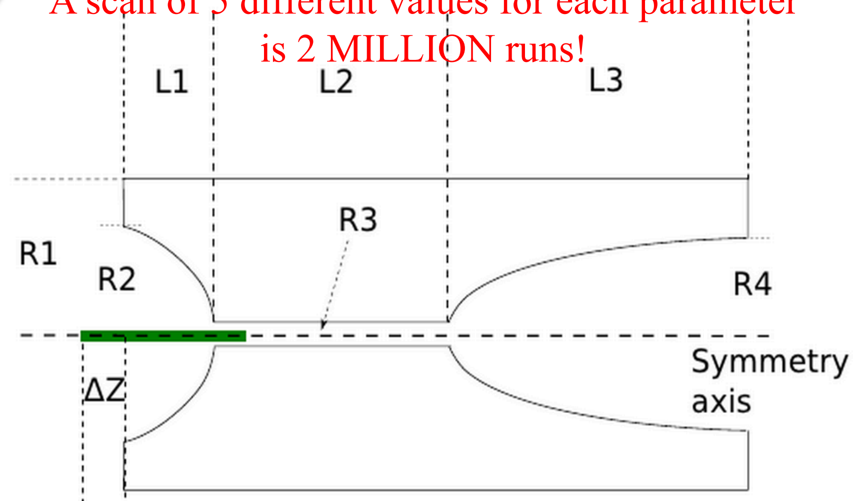
Make selection and the next population. A new generation is formed

When the maximum generation number is reached, or the fitness stops improving, stop the algorithm

9 parameters (8 listed below + horn current)

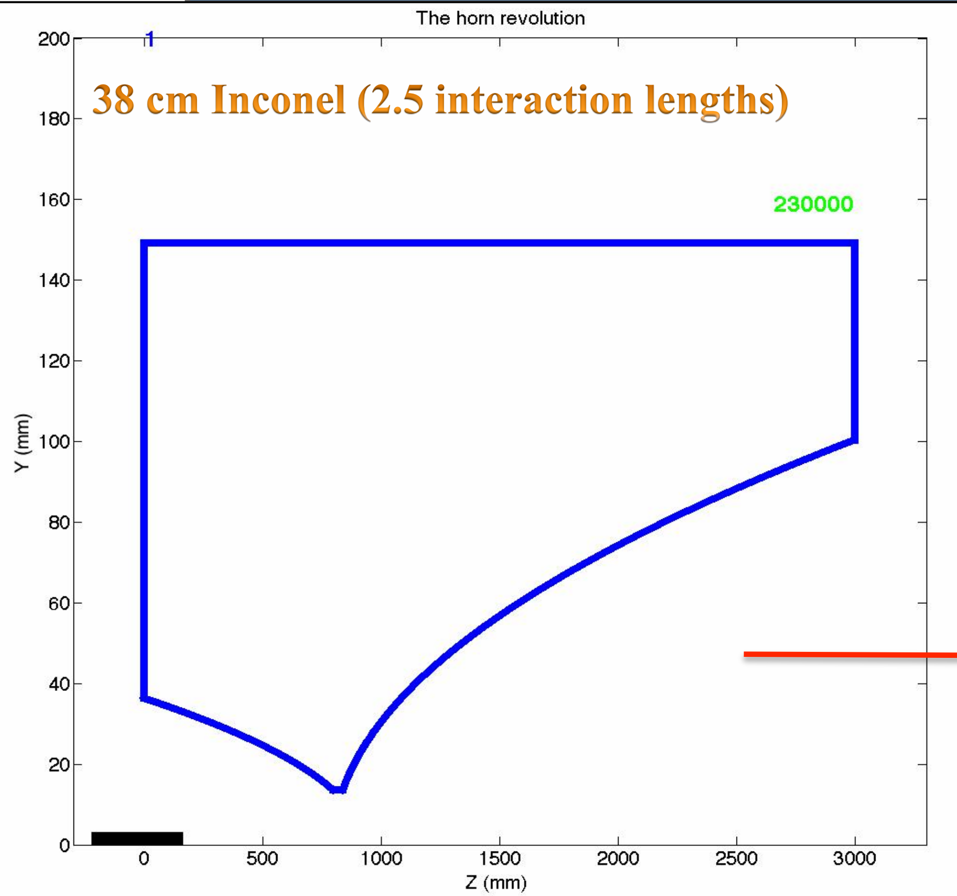
2 objectives

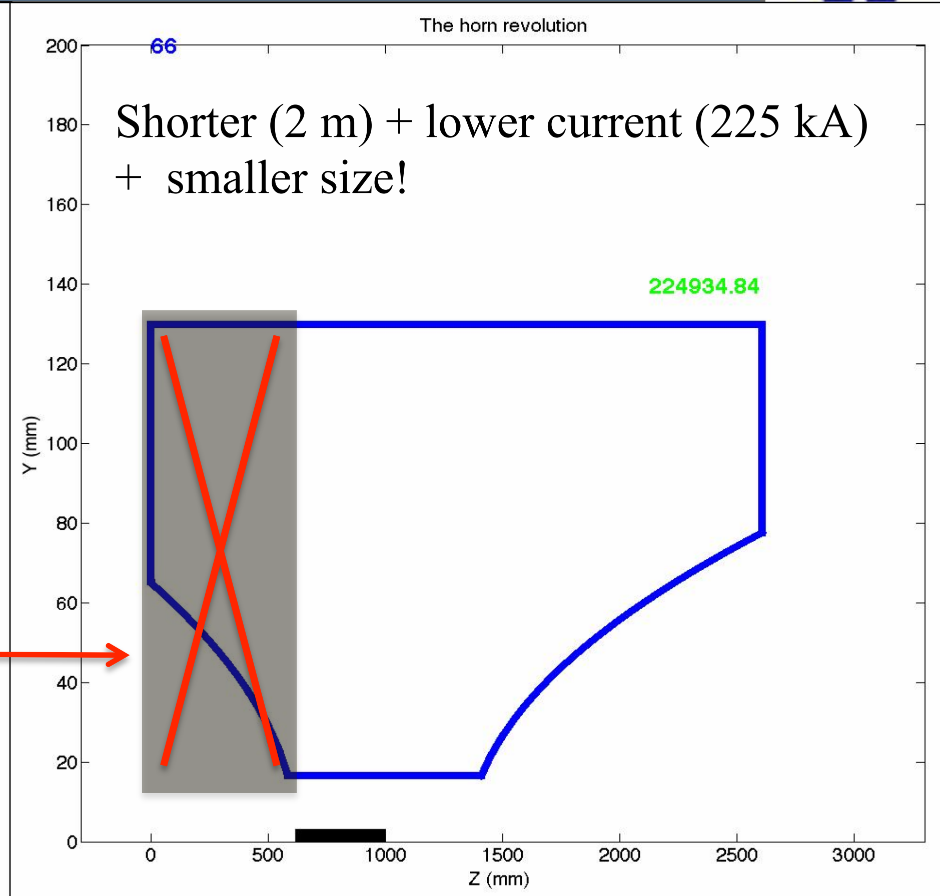
A scan of 5 different values for each parameter is 2 MILLION runs!



Constraints added: Maximum horn current, minimum neck radius, Feasible Twiss range

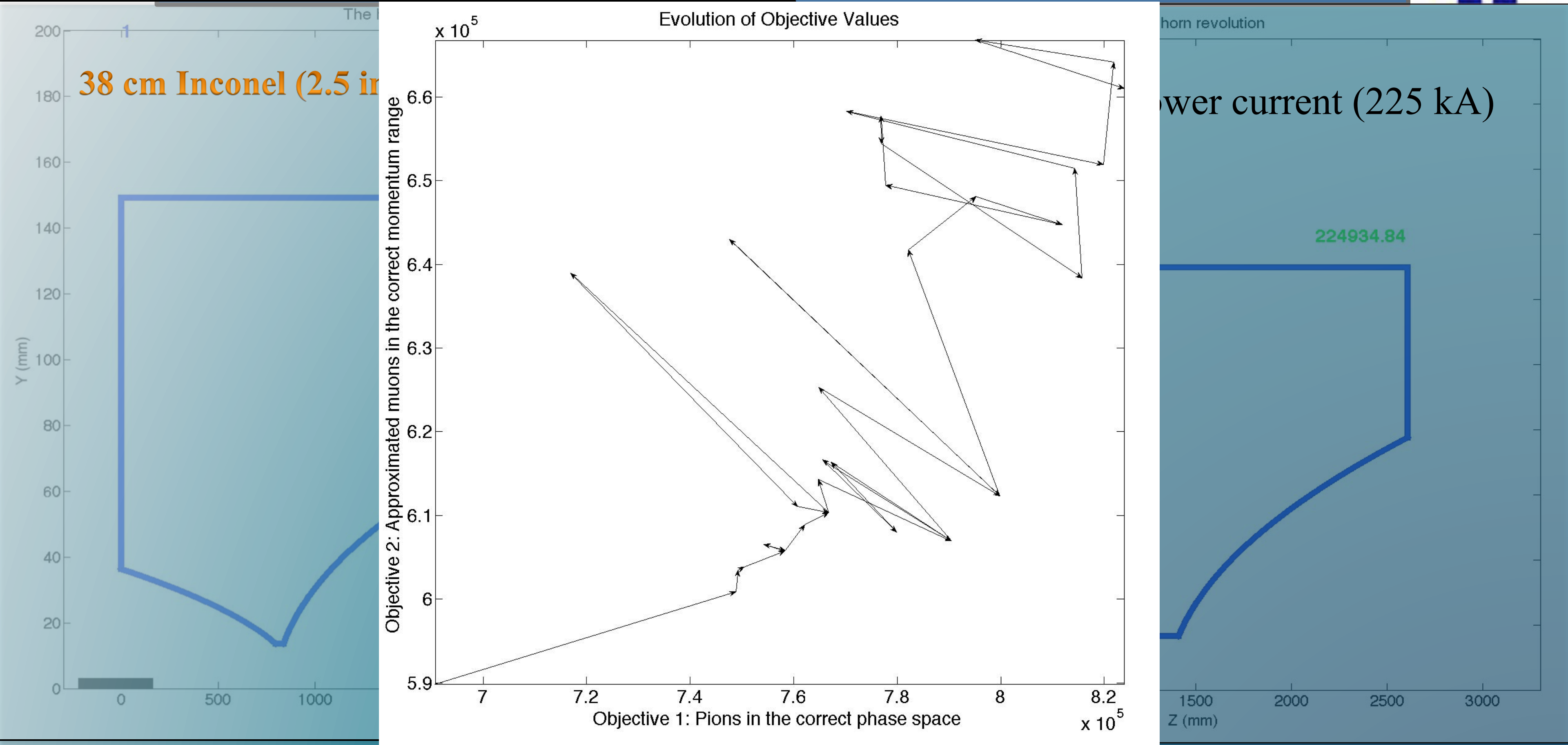
nuSTORM Horn Optimization – Results





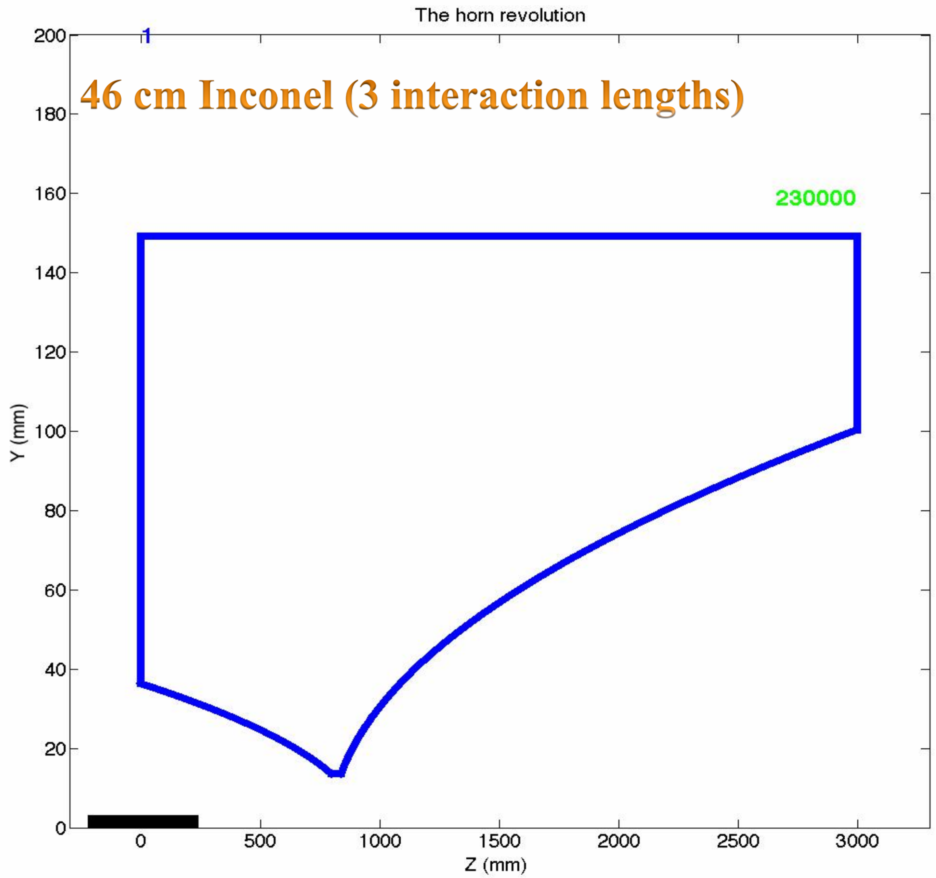
Changing the horn design without extending the target results in increasing μ+ in both 2000 μm and 3.8±10% GeV/c by 8.3%

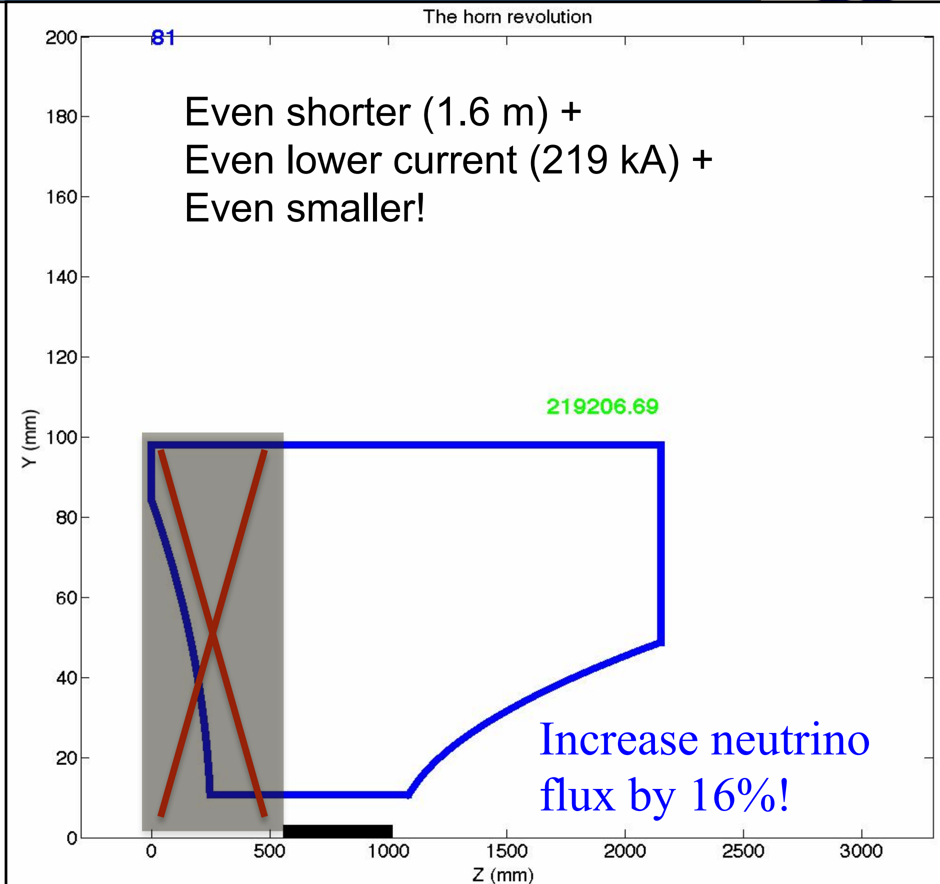
Horn Optimization – Results



Changing the horn design without extending the target results in increasing μ+ in both 2000 μm and 3.8±10% GeV/c by 8.3%

Horn Optimization – Results (Cont'd)





μ+ in both 2000 μm and 3.8±10% GeV/c increased by ~ 16% (Compared to the pre-optimization 38 cm Inconel + baseline horn)

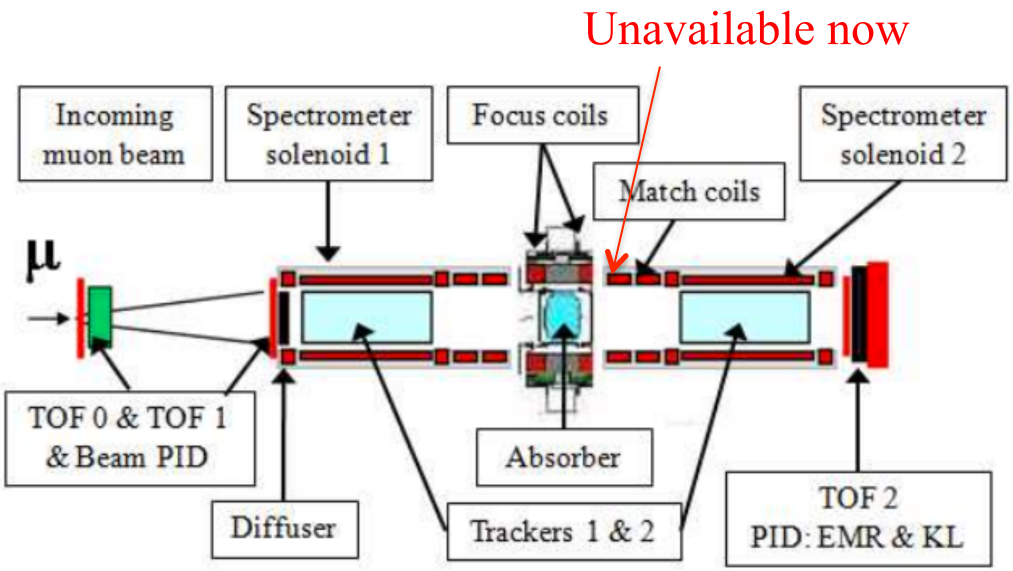
(If just changing the target length: ~ 5%)

Optimization cases in MICE

- Muon Ionization Cooling Experiment (MICE @ RAL)
  - Step IV:
    - Measurement of the cooling equation

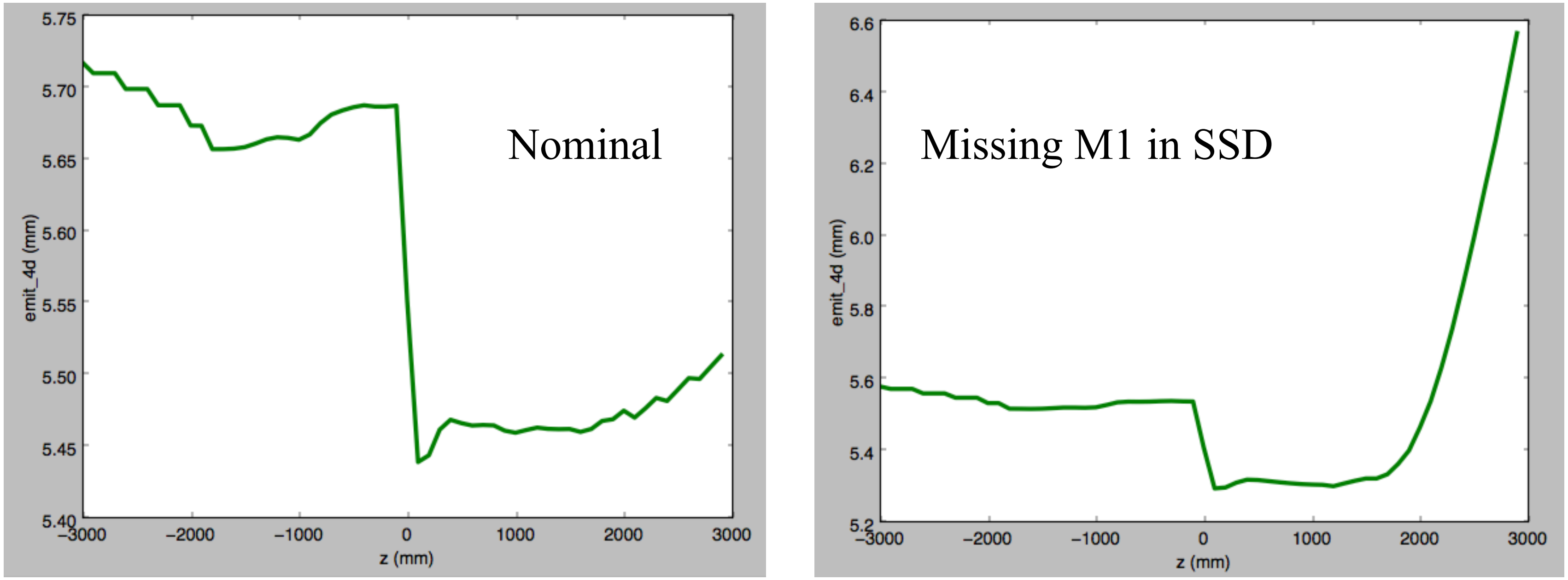
$$\frac{d\epsilon_n}{dz} = \frac{-\epsilon_n}{\beta^2 E}\left\langle\frac{dE}{dz}\right\rangle + \frac{\beta_\perp (14\ \mathrm{MeV})^2}{2\beta^3 E m_\mu X_0}$$

    - Measurement of the multiple scattering in the absorber materials
  - Incident:
    - Lost one of the matching coils in SSD
  - Action:
    - Re-optimize the lattice to obtain cooling and good transmission



Optimization cases in MICE (Cont'd)

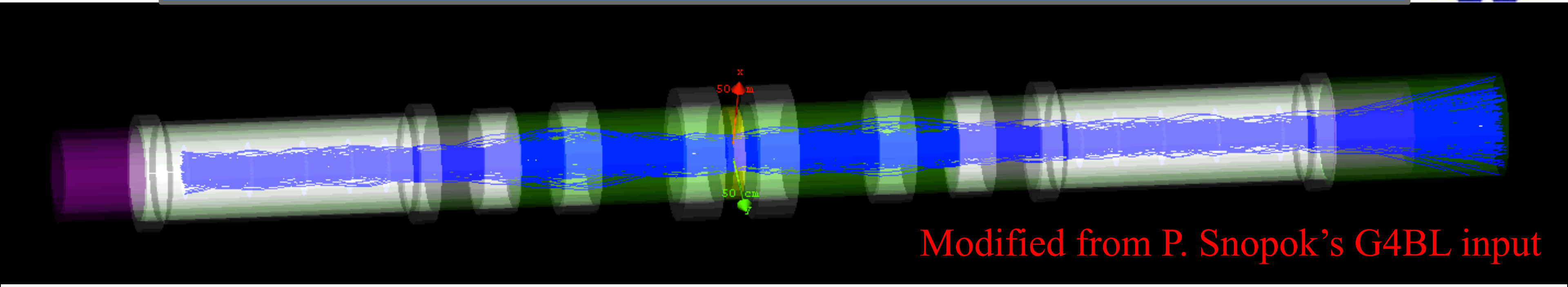
- Mismatch from missing the coil causes emittance growth and higher loss:



6 mm initial normalized 4D emittance, 200±10 MeV/c, survived muons only

- Use metaheuristic algorithms to ensure demonstration of cooling is possible without M1 in SSD

Optimization setup – G4Beamline

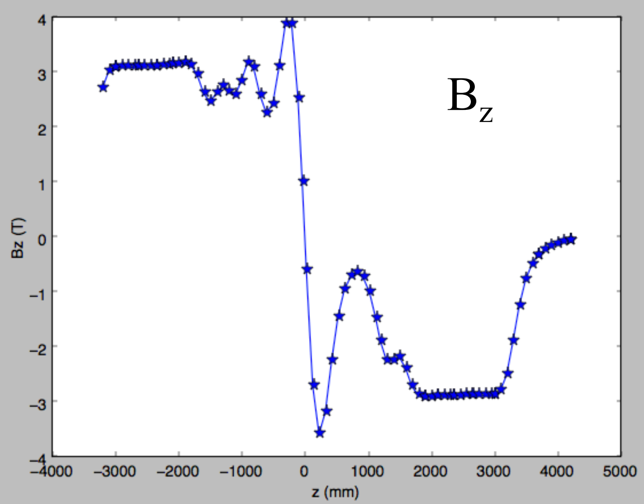


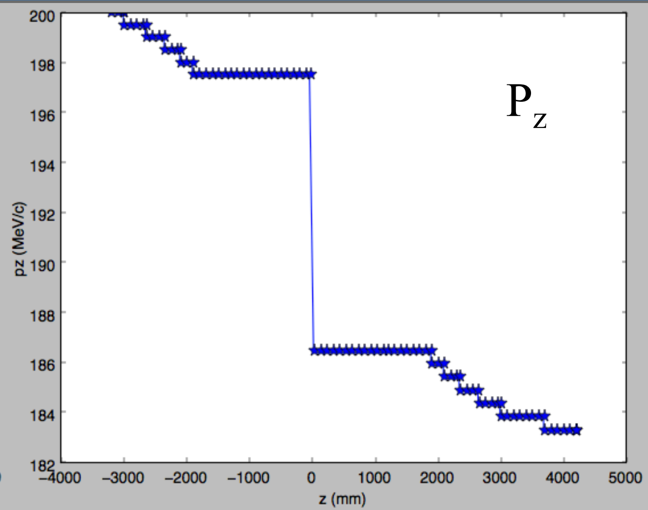
- Optimization is based on particle tracking
  - Performed in G4Beamline (author: T. Roberts, Muons Inc.)
  - SA and GA are both used.
- MICE coils and currents – both SS and FC
- Model materials in channel to match MAUS as accurately as possible:
- Use initial beam generated by constant solenoid mode Penn beam matrix, which matches the B_z at the starting point.
  - Beam starts at z=-3000 mm from the absorber (tracker0).

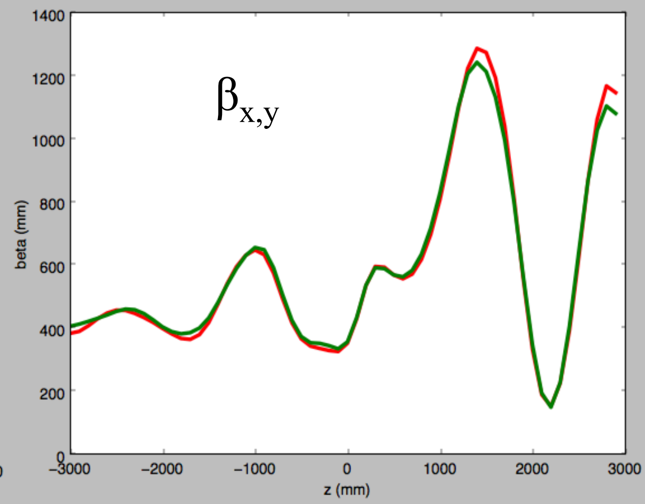
Optimization details

- Track reconstruction in the trackers requires uniform B_z on axis, which is used as a constraint in the optimization;
- Objective function:
  - $T^2 * (\varepsilon_{4D\_tracker1} - \varepsilon_{4D\_tracker0}) / \varepsilon_{4D\_tracker0}$
    - T: transmission to TOF2 (trigger): guarantees good transmission
  - $\varepsilon_{4D\_tracker0(1)}$ defined at -+1800 mm from the absorber (tracker ref. planes, i.e. where emittance is measured)
- Initial beam has 2.5% momentum spread to model a more realistic transmission

Optimization result

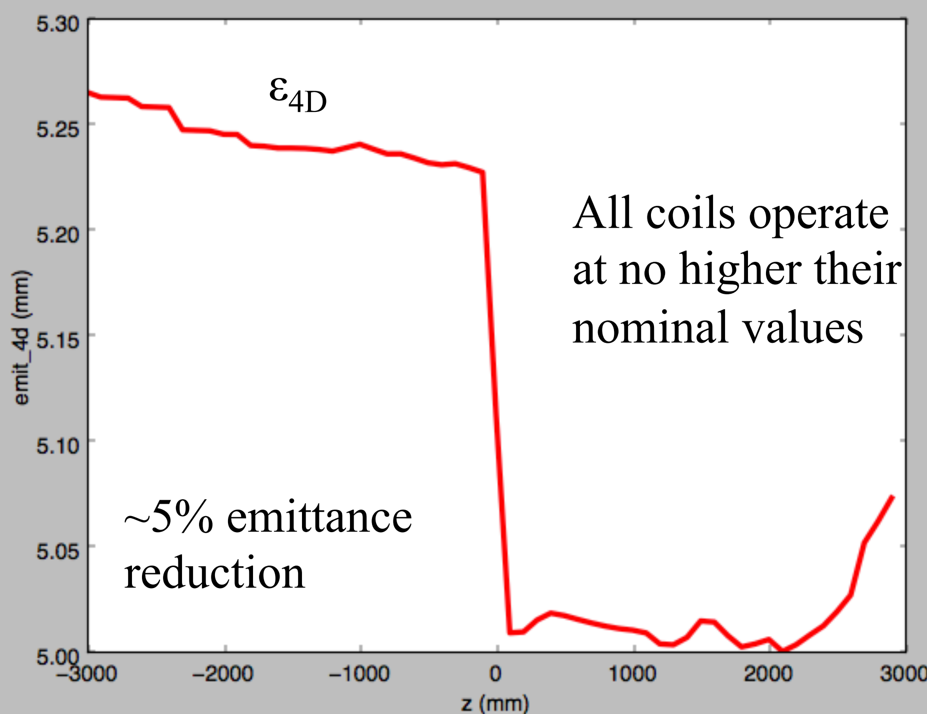






90% transmission to TOF2 with +-5 MeV/c momentum spread

FC are allowed to have different current, but in this configuration, FCs accidentally sit at the same current – good.



T2: 253.23*0.77
C: 277.98*0.77
T1: 246.2*0.77
M2: 257.85
M1: 246.48
FC: 229.86
FC_down: -222.01
M1_down: 0
M2_down: -256.12
T1_down: -246.2*0.71
C_down: -277.98*0.71
T2_down: -253.23*0.71

Conclusions

- Metaheuristic algorithms are extremely useful for design/operation optimizations with lots of variables or complicated performance mechanism;
- Using a MOGA, a nuSTORM target + horn combo can be optimized to deliver 16% more useful muons to the decay ring;
- Using a SOGA/SA, decent cooling with good transmission can be obtained even without the unavailable downstream matching coil in MICE;
- The algorithms can be easily applied to your project: whenever you need to make a decision!

Q & A

QUESTIONS? YES OR NO, THANKS!