Applications of Modern Computational Tools in Accelerator Physics

Balša Terzić
Center for Advanced Studies of Accelerators (CASA), Jefferson Lab
Center for Accelerator Science (CAS), Old Dominion University

Jefferson Lab Accelerator Seminar, September 12, 2013

Give Credit Where Credit is Due

- CASA scientists:
  - Alicia Hofler, Vasiliy Morozov, Yves Roblin, Fanglei Lin, Geoff Krafft
- Old Dominion University & Center for Accelerator Science:
  - Professors:
    Physics: Alexander Godunov
    Computer Science: Mohammad Zubair, Desh Ranjan
    Modeling, Simulation & Visualization Engineering: Bharat Madan
  - PhD students:
    Computer science: Kamesh Arumugam, Mohamed Aturban
- REU/SULI summer undergraduate students:
  - Matthew Kramer (UC Berkeley), Colin Jarvis (Macalester), Anton Zvezdin (Stony Brook → Northwestern), Alyssa Henderson (UVa)

Outline of the Talk

- New computational tools:
  - New methods
  - New computational hardware
- New methods: Multidimensional, non-linear optimization using a genetic algorithm (GA)
  - Brief motivation and background
  - Applications: What we *have done*, *will do* and *can do*
- New computational hardware:
  Parallel computation on Graphical Processing Units (GPUs)
  - Brief motivation and background
  - Applications
    - Multidimensional integration for use in CSR simulations
    - New code for simulation of long-term beam-beam dynamics
- Summary

GA Optimization: Motivation

- As the dimensionality of the problem *N* increases, *N*-dimensional non-linear optimization becomes more challenging/impossible:
  - Traditional, gradient-based methods (Newton, conjugate-gradient, steepest descent, etc…) are *not globally convergent*:
    - May get stuck in a local minimum and never come out
    - Final solution depends on the initial guess
    - Generally *not robust* in the non-linear regime
    - Direct multi-objective optimization not possible

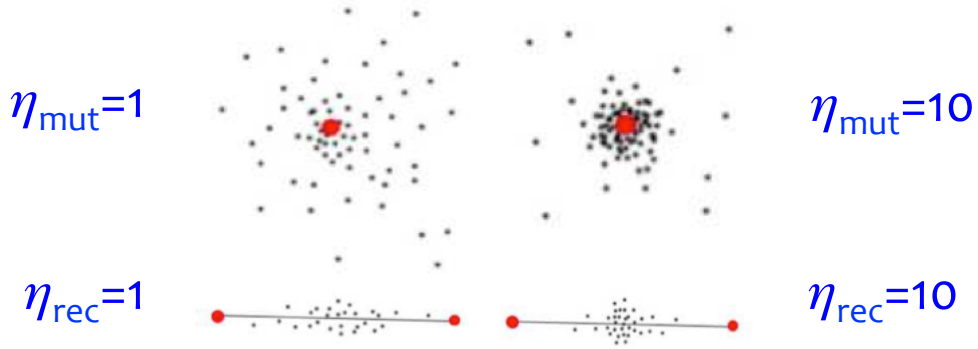
- This demonstrates a clear need for *globally-convergent, robust, multidimensional, multi-objective, non-linear optimization* methods
  - Genetic algorithm (GA):
    - Trade-off: not as efficient as traditional methods
    - Caution: *still not a silver bullet*

GA Optimization: Background

- GA uses principles of natural selection to solve an optimization problem

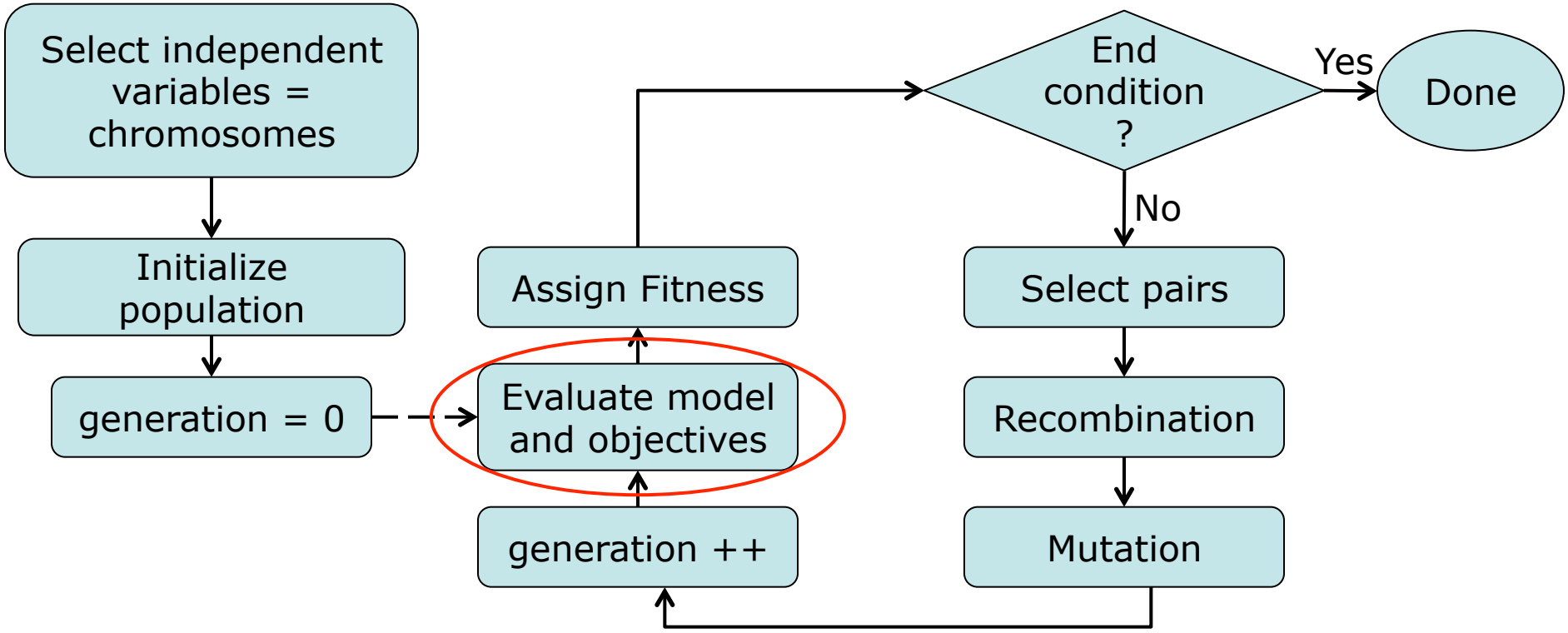
Evolution	Multidimensional optimization
Gene	Variable
Individual	Point in search space
Population	Set of points in search space
Mutation	Changing variables
Swap	Exchange of values of the same variable between two points in search space
Recombination (partial swap)	Change of values of the same variable between two points toward each other
Fitness	Value of the objective function

- Mutation
  - Asexual reproduction
- Recombination
  - Sexual reproduction



For details see: Hofler, Terzić, Kramer, Zvezdin, Morozov, Roblin, Lin & Jarvis 2013, *PR STAB* 16, 010101

GA Optimization: Background



$$\text{Minimize} \quad f_m(\mathbf{x}), \qquad m = 1,2,\ldots,M; \quad \leftarrow \text{objectives}$$

$$\text{Subject to} \quad g_j(\mathbf{x}) \geq 0, \qquad j = 1,2,\ldots,J; \quad \leftarrow \text{inequality constraints}$$

$$h_k(\mathbf{x}) = 0, \qquad k = 1,2,\ldots,K; \quad \leftarrow \text{equality constraints}$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)}, \qquad i = 1,2,\ldots,n; \quad \leftarrow \text{decision variable constraints}$$

Slide courtesy of Alicia Hofler

GA Optimization: Background

- GA simulation is only as accurate as its function evaluator
- We use two paradigms depending on objective function evaluation
- When objective function(s) are evaluated by a full simulation
  - Platform and Programming Language Independent Interface for Search Algorithms (*PISA*) from ETH Zürich and Alternate *PISA* (*APISA*) from Cornell:
    - Job control: spawning simulations, post-processing, communication
    - Redesigned to be more modular and script-based: easier to *plug-and-play*
    - Parallel execution of an entire generation: runs on JLab's cluster and farm
- When objective function(s) are analytically evaluated in a subroutine
  - Still can use PISA, but other options available (in various languages)
  - We choose *inspyred* Python package:
    - Contains various nature-inspired and other optimization methods: particle-swarm, differential evolution, simulated annealing, etc...
    - Contains efficient traditional methods: Newton, conjugate gradient, …
    - Parallel execution possible (using *pympi* package)

GA Optimization: Applications

- We applied GA optimization to many problems in accelerator physics:
  - When a separate simulation is needed to evaluate objective function(s)
    - Optimizing collider working point for luminosity (*BB3D*) — **Single objective**
    - Maximizing dynamic aperture in a collider ring (*elegant*) — **Single objective**
    - Decoupling of the beam optics in the injector (*elegant*) — **Single objective**
    - Optimizing dynamic aperture and chromaticity in a collider ring (*elegant*) — **Multiple objectives**
    - RF gun optimization for injector brightness (*Astra*, *Superfish*) — **Multiple objectives**

    [Hofler 2012, PhD and elsewhere]

    [Hofler, Terzić, Kramer, Zvezdin, Morozov, Roblin, Lin & Jarvis 2013, *PR STAB* 16, 010101]
    - Optimizing laser frequency modulation function in Thomson scattering — **Multiple objectives**

    [Terzić, Deitrick, Hofler & Krafft 2013, *PRL*, submitted]
  - When objective function(s) can be evaluated analytically
    - Chi-square fits to the CEBAF harp data — **Single objective**
- Future candidate for GA optimizations:
  - Optimizing heat load and trip rates in the CEBAF linacs — **Multiple objectives**
  - Injector optimization with the CEBAF as the function evaluator

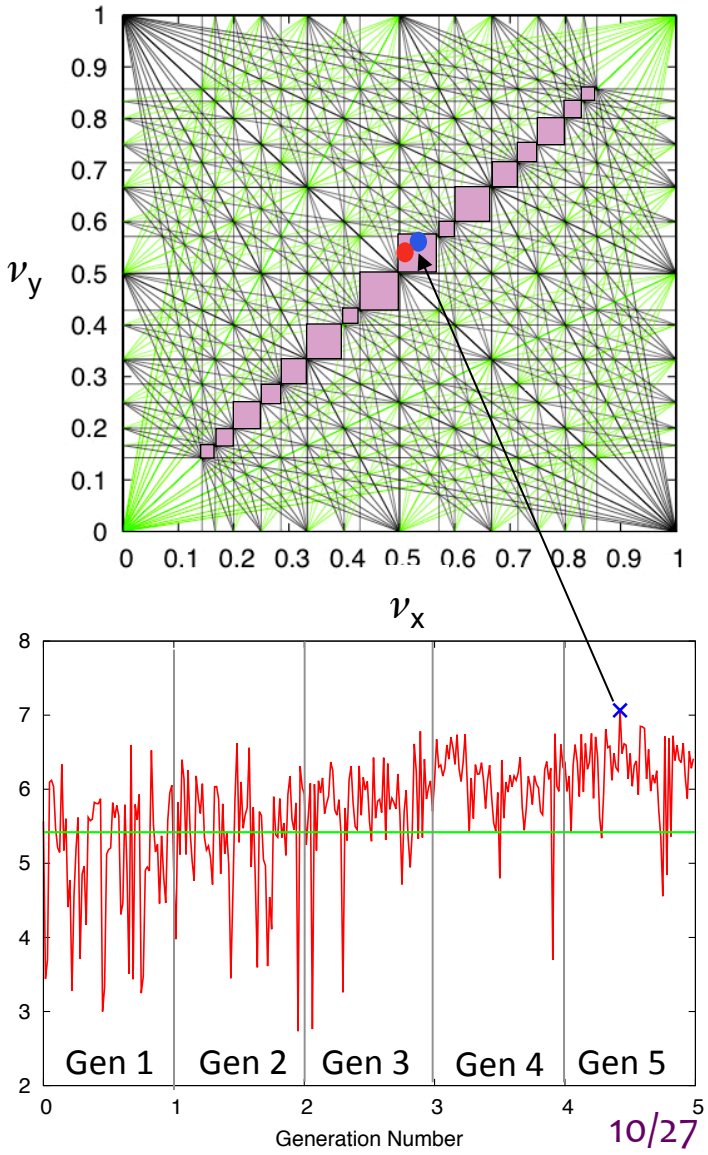
GA Application: Optimizing Collider Working Point

- Collider luminosity is sensitive to beam-beam effect and synchro-betatron resonances of the two colliding beams
- Careful selection of a tune *working point* is essential for stable operation of a collider as well as for achieving high luminosity
- We simulate the Medium-energy Electron-Ion Collider (MEIC)
- *Optimization problem*:
  - Independent variables: betatron tunes for the two beams $(\nu^1_x,\nu^1_y,\nu^2_x,\nu^2_y)$
    (Synchrotron tunes fixed for now; 4D problem)
  - Objective function: collider's luminosity $L(\nu^1_x,\nu^1_y,\nu^2_x,\nu^2_y)$
    (Evaluated via a simulation with *BeamBeam3D* parallel code on the JLab cluster)
  - Subject to constraints (e.g., confine tunes to particular regions)
- GA is the only non-linear optimization method that can work in a search space so violently fraught with resonances (*very* sharp peaks and valleys)

GA Application: Optimizing Collider Working Point

- Resonances occur when $m_x\nu_x+m_y\nu_y+m_s\nu_s= n$
  m_x, m_y, m_s and n are integers (m_s=0 for now)
- Green lines: *difference* resonances (stable)
- Black lines: *sum* resonances (unstable)
- Restrict search to a group of small regions along the diagonal devoid of black resonance lines. Restricts the search space by ~30 in 2D, ~1000 in 4D
- Found an excellent working point near half-integer resonance
  e-beam: ν_x = 0.530, ν_y = 0.548
  p-beam: ν_x = 0.501, ν_y = 0.527
- Luminosity about 33% above design value in only ~300 simulations (5 gen. of 64 individuals)
- Systematic scan with a modest 0.01 resolution: $100^4=10^8$ simulations!

  ➔ GA search orders of magnitude more efficient
- *This is just a proof of principle* – future realistic simulations will include other important effects: magnet errors, non-linear maps, IBS, cooling …

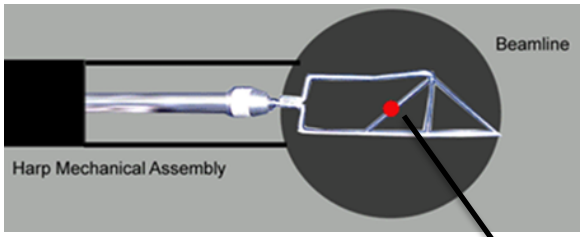


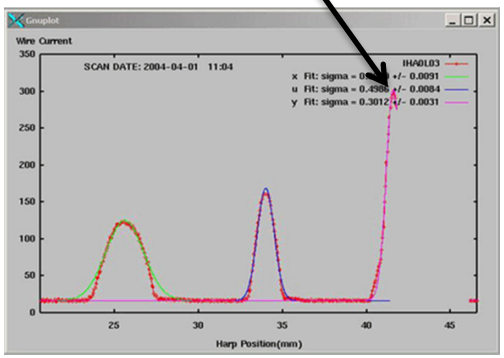
GA Application: Non-Linear Chi-Square Fitting

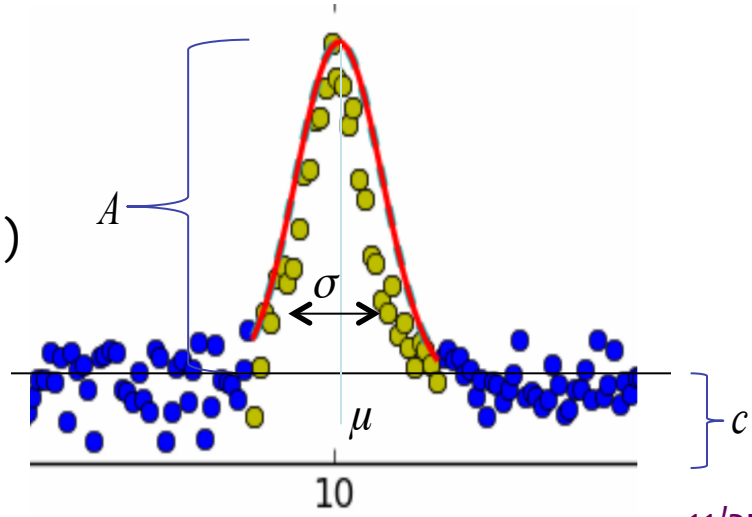
- Wire scanners (harps) are used in CEBAF to measure the beam's position and size
- Moved into the beamline at a constant velocity and angle (generally 45°)
- Data from these plots needs to be fit to Gaussians to obtain beam size and position
- Non-linear fitting cast into an optimization problem: *minimize chi-square* (χ^2)

$$\chi^2 = \sum_{i=1,N} [f(x_i) - y_i]^2$$

- Independent variables:
  Gaussian parameters: A, μ, σ, c (4D problem)
- Objective function:
  χ^2 of a Gaussian $f(x) = Ae^{-\frac{(x-\mu)^2}{2\sigma^2}} - c$
- Used Python package *inspyred*







Slide courtesy of Alyssa Henderson

GA Application: Non-Linear Chi-Square Fitting

- A *hybrid method* combines strengths of the two approaches
  - Traditional, gradient-based methods *converge quickly to a local optimum*
  - GA and other *nature-inspired methods* (PSO, DE) are *globally convergent* – they *eventually* zoom in on the global minimum (generally slow)
  - First reduce the search space with a globally convergent method, then use a fast gradient-based method for improved efficiency

Method	Time (s)
NCG	3.168
PSO	123.906
DE	41.416
GA	120.940

Average execution time for a harp scan fit with >50 points on a single CPU

- Harp fitting is still mostly solvable by a traditional, globally convergent method (Newton CG)
  - GA and other nature-inspired methods used for robustness & insurance
- Switching to other models is easy (just change the objective function)
- As the dimensionality of the problem grows, gradient-based methods becomes less reliable, and the need for GA more pronounced

GA Application: Dynamic Aperture & Chromaticity

- In a collider ring design, *dynamic aperture and momentum acceptance should both be maximized* – but they are *roughly* inversely proportional
- Reducing the higher-order chromaticity will lead to increased momentum acceptance
  - This, again, depends on the choice of the working point (betatron tunes)
- We simulate the MEIC ion ring
- *Optimization problem:*
  - Independent variables: betatron tunes for the beam (ν_x,ν_y) (2D problem)
  - Objective functions:
    (Both evaluated via a simulation with *elegant* code on the JLab farm)
    - Maximize mom. acceptance $\leftrightarrow$ Minimize 2nd-order chromatic function ξ^2
    - Maximize area of dynamic aperture A $\leftrightarrow$ Minimize $1/A$

GA Application: Dynamic Aperture & Chromaticity

- Physics judgment is needed to strike a balance between the two quantities
- Pareto-optimal front:
  Non-dominated solutions

Pareto-Optimal Front

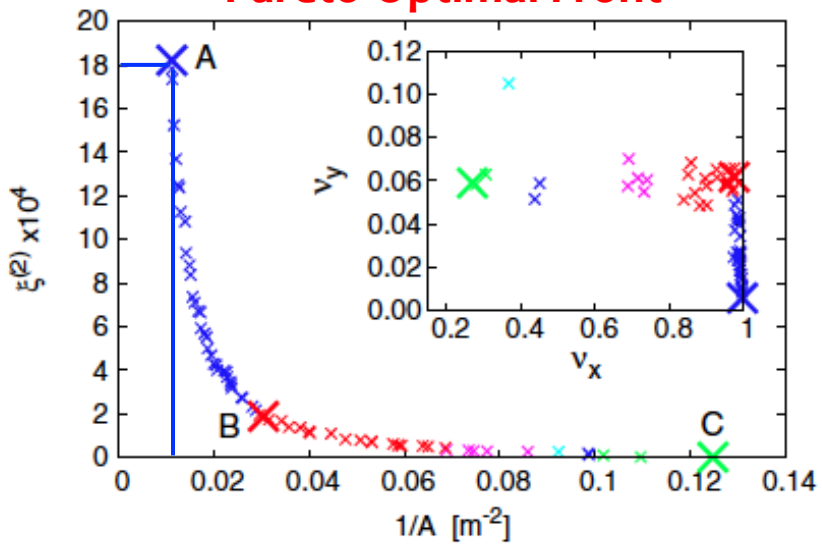
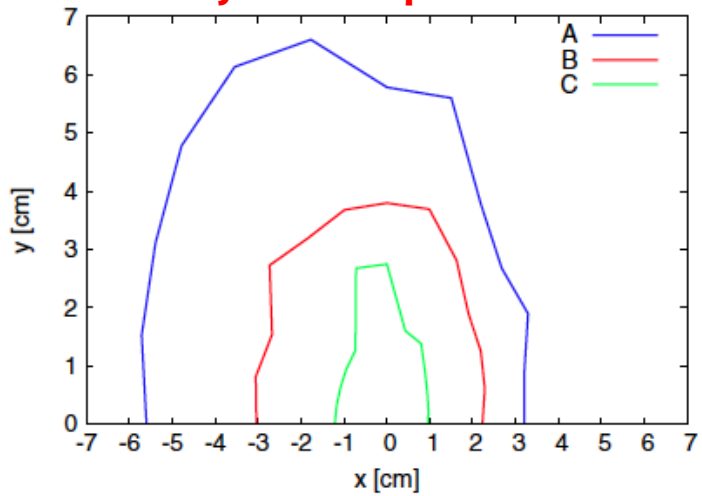
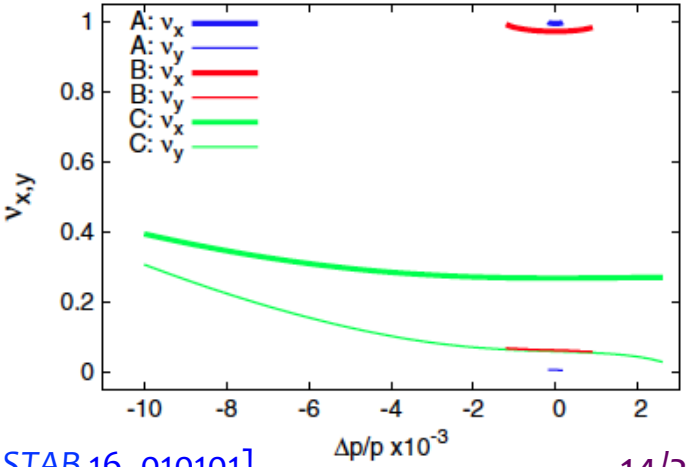


FIG. 17. Pareto front after 24 generations of 64 individuals. The large ×'s denote representative points A, B, and C.

Dynamic Aperture



Momentum Acceptance



[Hofler, Terzić, Kramer, Zvezdin, Morozov, Roblin, Lin & Jarvis 2013, *PR STAB* 16, 010101]

GA Optimization: Adapting to a New Problem

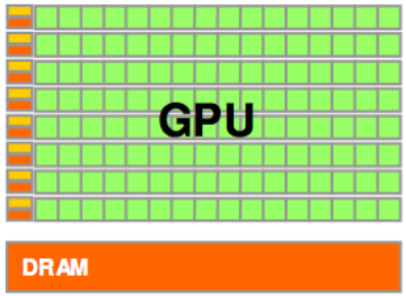
- Adapting the two paradigms to a new problem is relatively easy
  - Physics pre-screening
    - Restrict the search space as much as physically possible
      Convergence directly proportional to the volume of search space
  - Define the optimization problem:
    - *Objective function*: How is it computed? Choose paradigm
    - *Independent variables* (*parameters/"knobs to be turned"*): define ranges
    - *Constraints*: specify if present
    - Decide on simulation parameters: number of individuals/generations
  - If a separate simulation is needed for evaluating objective function(s)
    - Make sure program is installed on the platform to be used
      (Unlimited licenses are required for commercial software)
    - Modify scripts for spawning jobs and extracting results
    - Define parameter ranges for independent variables
  - If objective function(s) can be evaluated analytically
    - Write function(s)/subroutine(s)

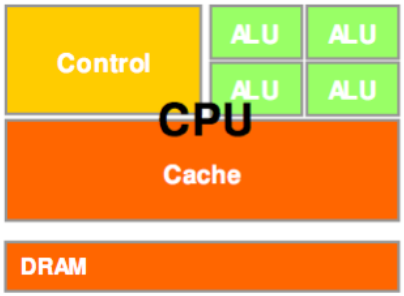
GPU Computation

- Second half of my talk is on parallel computation on GPUs
- *Why is it important?*
  - Making simulations much more efficient computationally *enables studying previously inaccessible physics*
- *What are we doing that is new and different?*
  - *Interdisciplinary approach – division of labor among experts in the field:*
    - Physicists: physics, algorithm development, prototyping
    - Computer scientists: algorithm development and implementation, parallel programming
- *What are our goals?*
  - Develop GPU-parallelized codes that will lead to *accelerator physics simulations beyond the present state of the art*
  - Design methods useful *beyond the scope of accelerator physics*
  - Develop expertise on the subject to use it on other problems

GPU Computation: Background

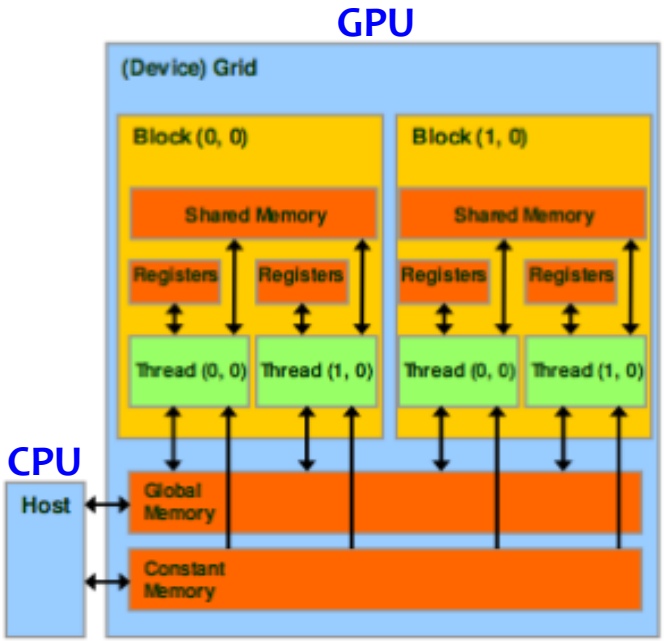
- Parallel computation on GPUs
  - Ideally suited for algorithms with *high arithmetic operation/memory access ratio*
  - Same Instruction Multiple Data (SIMD)
  - *Several types of memories* with varying access times (global, shared, registers)
  - Uses extension to existing programming languages to handle new architecture
  - GPUs have many smaller cores (~ 400-500) designed for parallel execution
  - *Avoid branching and communication* between computational threads





More space for ALU, less for cache and flow control

GPU: grid → blocks → threads



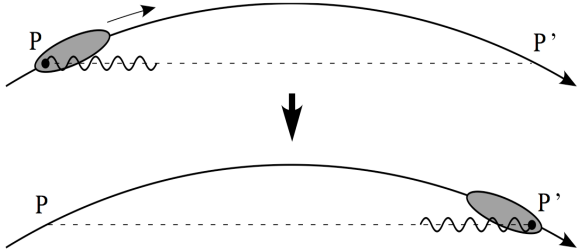
Example: Tesla M2090 GPU has 512 cores

GPU Computation: Motivation

- There are many problems in accelerator physics that can *greatly benefit* from a speedup from a GPU-based computation
  - Particle tracking codes (*elegant*, *VORPAL*, etc…)
  - Collision codes (*BeamBeam3D*)
  - Monte Carlo-based codes
- *Speedup*: ratio of execution times on a host CPU to that on a GPU
- Some have already been GPU-parallelized with impressive speedup
  - *elegant*: ~70 times
  - *VORPAL*: at least an order of magnitude
- In general, it has been shown that GPUs can deliver performance improvements of 1-3 orders of magnitude
- This kind of speedup means:
  - Simulation time: several months or a year → about a day
  - *Opening the doors to studying previously inaccessible physics!*

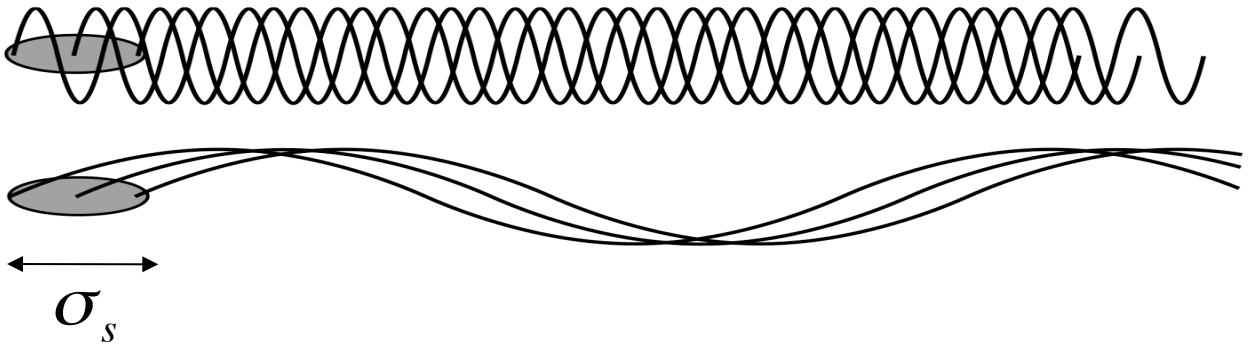
GPU Computation: CSR Simulations

- When a charged particle beam travels along a curved trajectory (bending magnet), beam emits synchrotron radiation



Incoherent (ISR) $\lambda < \sigma_s$

Coherent (CSR) $\lambda > \sigma_s$



largely cancels out

has systematic effects

- CSR adversely impact beam quality:
  - Increased energy spread and emittance, longitudinal instability (microbunching)
- CSR effects are important for FELs, light sources, ERLs, etc…
- It is of vital importance to have a trustworthy code to simulate and mitigate the CSR effects

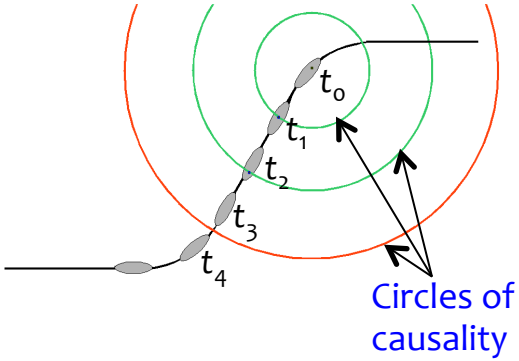
GPU Computation: CSR Simulations

- CSR simulations have proven to be extremely challenging
  - Computing *retarded potentials* requires integration over the retarded time *t'*:

$$t' = t - \frac{|\vec{r} - \vec{r}'|}{c}$$

Retarded time

$$\begin{bmatrix} \phi(\vec{r},t) \\ \vec{A}(\vec{r},t) \end{bmatrix} = \int \begin{bmatrix} \rho(\vec{r}',t') \\ \vec{J}(\vec{r}',t') \end{bmatrix} \frac{d\vec{r}'}{|\vec{r} - \vec{r}'|}$$



  - *Huge computational bottleneck!* Computations scale:
    - Particle-particle codes: ~ N^2, where N is the number of particles
    - Particle-in-cell codes: ~ N_{res}^2, where N_{res} is the grid resolution
      We have been developing a particle-in-cell (PIC) CSR code
- Solution: Develop an efficient, parallel multidimensional integrator on GPUs
  - Integration over grid is ideally suited for GPU parallelization (SIMD)
  - Used NVIDIA CUDA framework (extension to C++)
  - *Deterministic:* based on integration rules like Gauss or Newton – not Monte Carlo
  - Useful beyond this project: *outperforms Monte Carlo in medium dimensions*

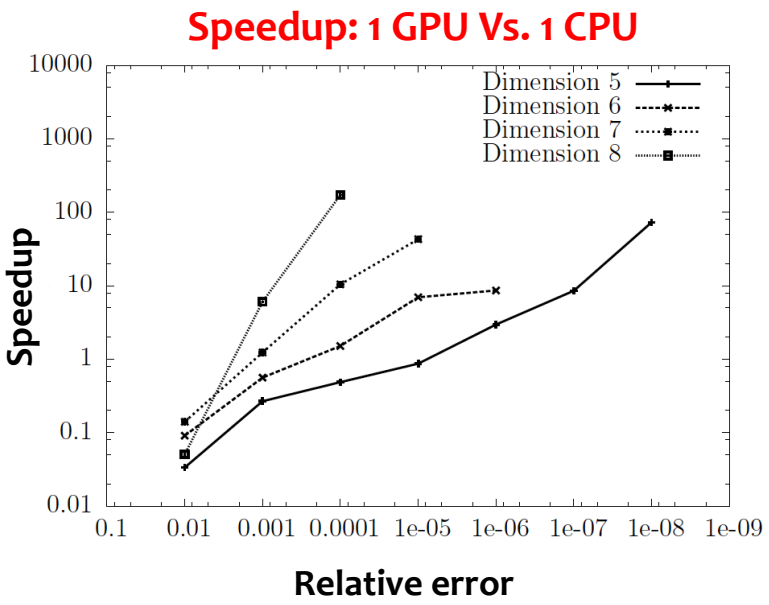
Adaptive Multidimensional Integration On a Single GPUs

- Direct parallelization of the serial methods *does not take advantage of GPU data parallelism* and *does not provide load balancing* → *inefficient code*
- *We developed a new two-phase parallel algorithm multidimensional integration on GPUs*
  - *Phase 1: Parallel identification of subintervals needing higher resolution*
  - *Phase 2: Parallel evaluation of identified subregions to prescribed accuracy*
- GPU-based implementation outperforms the best known sequential method (CUHRE) and achieve up to 10-100 times speedup on a single GPU

Benchmark functions in *n* dimensions

oscillatory, strongly peaked and of varying scales

1. $f_1(\mathbf{x}) = \left[\alpha + \cos^2\left(\sum_{i=1}^{n} x_i^2\right)\right]^{-2}$, where $\alpha = 0.1$
2. $f_2(\mathbf{x}) = \cos\left(\prod_{i=1}^{n} \cos\left(2^{2i} x_i\right)\right)$
3. $f_3(\mathbf{x}) = \sin\left(\prod_{i=1}^{n} i \arcsin(x_i^i)\right)$
4. $f_4(\mathbf{x}) = \sin\left(\prod_{i=1}^{n} \arcsin(x_i)\right)$
5. $f_5(\mathbf{x}) = \frac{1}{2\beta} \sum_{i=1}^{n} \cos(\alpha x_i)$, where $\alpha = 10.0$ and $\beta = -0.054402111088937$



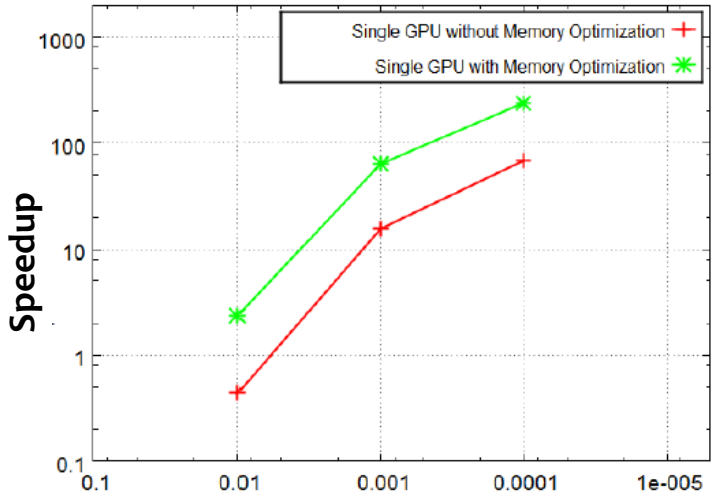
[Arumugam, Godunov, Ranjan, Terzić & Zubair 2013a]

Slide courtesy of Kamesh Arumugam

Adaptive Multidimensional Integration On Multiple GPUs

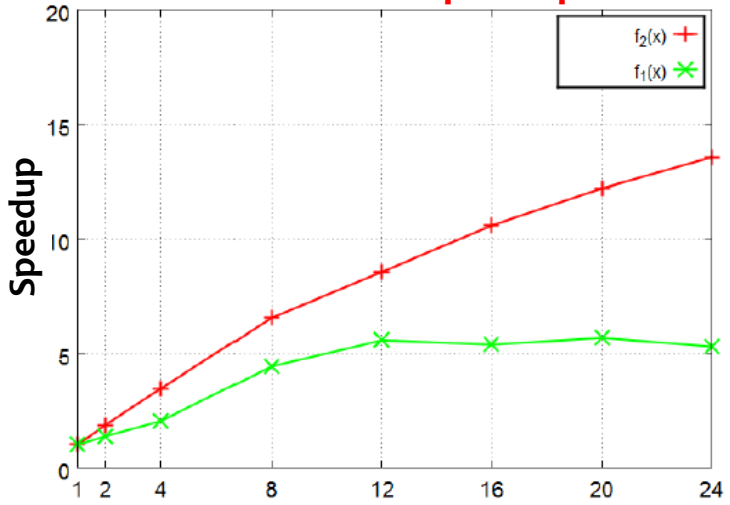
- Next, we *optimized* our new GPU-based algorithm *for memory efficiency* and *scaled to multiple GPU devices*
- The algorithm has been implemented on a cluster of Intel® Xeon® CPU X5650 computes nodes with 4 Tesla M2090 GPU devices per node (512 cores per device)
- On a single GPU device: Reached a speedup over a serial version of up to 240 as compared to a speedup of 70 when memory optimization was not used
- On a cluster of 6 nodes (24 GPU devices): Reached a speedup of up to 3250

1 CPU Speedup:
With Vs. Without Memory Optimization



[Arumugam, Godunov, Ranjan, Terzić & Zubair 2013a, 2013b]

Multi-CPU Speedup



Slide courtesy of Kamesh Arumugam

Monte Carlo Vs. Adaptive Multidimensional Integration With a Single GPU

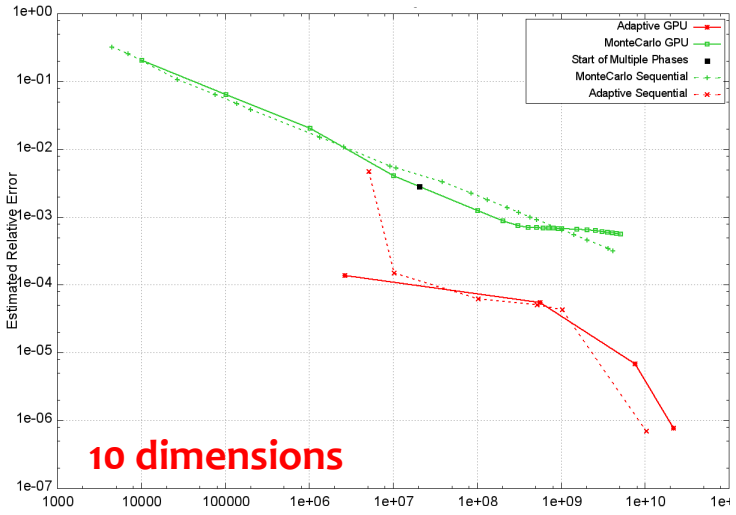
- Monte Carlo integration on GPU (VEGAS and BASES methods) has been published previously in *The European Physical Journal C* [Kanazaki 2011, 71:1559]
- We compare Monte Carlo Vs. our method on a set of *6 functions with exact solutions*
- *Preliminary results*: Even in higher dimensions our adaptive multidimensional integration method outperforms Monte Carlo method on a single GPU
- *Preliminary results*: Monte Carlo on GPU fails for large number of function evaluations
- *Possible ramifications*: Our new code can replace Monte Carlo in many physics application for improved performance

MC CPU
MC GPU
CUHRE CPU
CUHRE GPU

$$f_5(x) = \frac{1}{2\beta}\sum_{i=1}^{n}\cos(\alpha x_i)$$

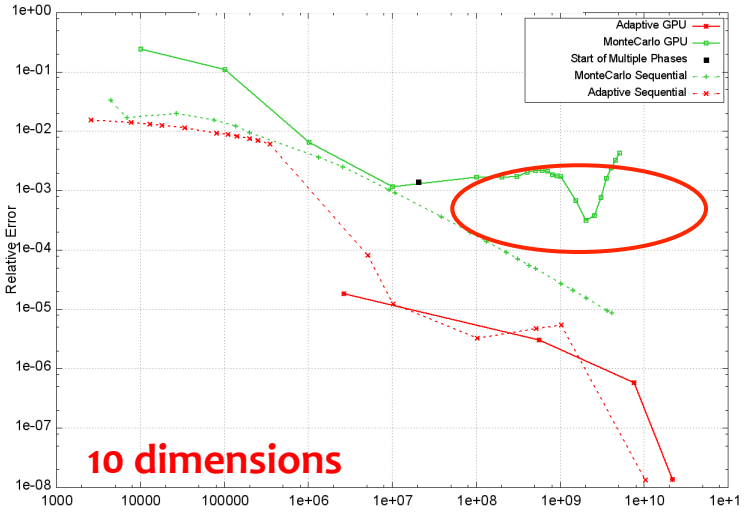
where α = 10.0 and
β = -0.0544

Internal Relative Error



# of function evaluations

True Relative Error



# of function evaluations

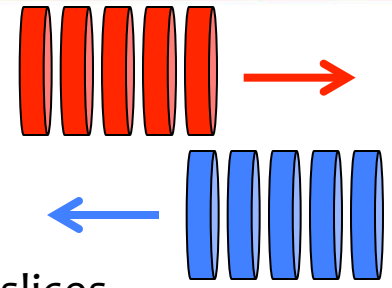
[Arumugam, Godunov, Ranjan, Terzić & Zubair 2013c]

Slide courtesy of Kamesh Arumugam

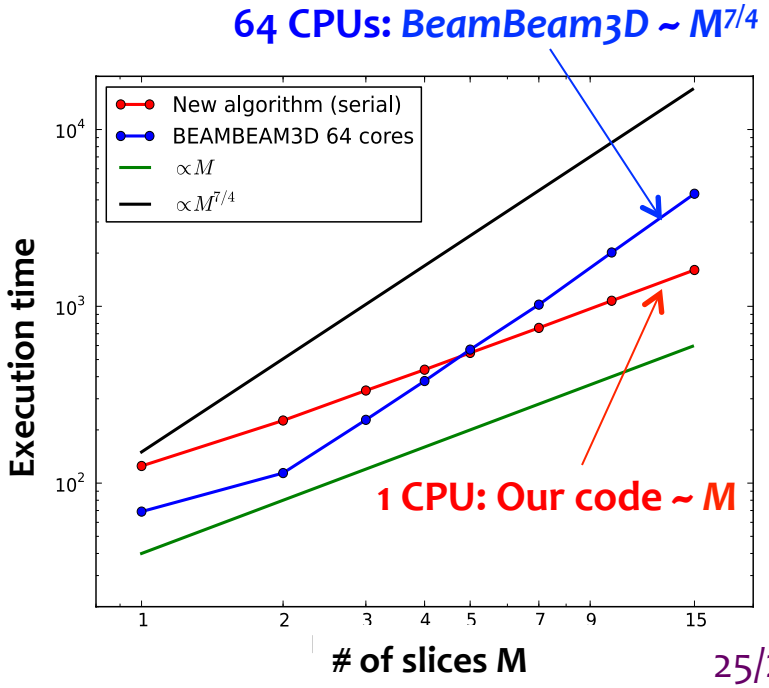
GPU Computation: Beam-Beam Code

- Studying beam dynamics in particle colliders requires specialized codes
- Simulation in beam-beam codes can be divided in two parts:
  - Particle tracking between consecutive beam collisions
    - Number of efficient tracking codes exist
  - Beam collisions at the interaction point
    - Solving the Poisson equation: major computational bottleneck
- Long-term dynamics is of particular interest
  - Simulate operational stability and luminosity lifetime
  - *Long-term*: on the order of beam lifetime (for MEIC ~ 10^9 collisions!)
  - This kind of long-term is *currently inaccessible* with existing codes
    - Too much time is spent on detailed solution of the Poisson equation
- Solution:
  - Simplify the beam-beam interaction (use Bassetti-Erskine approx.)
  - Parallelize both tracking and collision on GPUs

GPU Computation: Beam-Beam Code



- Bassetti-Erskine approximation:
  - When beams are *infinitesimally short* and *transversally Gaussian* the Poisson equation reduces to a *complex error function* (much faster than any other way of solving the Poisson equation)
    - Finite length of the beams simulated by stringing along thin slices
      - Collide each slice in one beam with each slice in the other beam
    - *Gaussian transverse distribution a good approximation for the MEIC*
- GPU parallelization: (*preliminary results*)
  - Particle tracking:
    - Symplectic 1-turn maps of an arbitrary order from *COSY-Infinity*
      - Single-CPU tracking is equivalent and as fast as *COSY-Infinity*
      - Speedup on 1 GPU ~ 170 times
  - Beam-beam collisions:
    - Scales better than *BeamBeam3D*
      - Speedup on 1 GPU ~ 70 times



Summary

- Presented two new powerful computational tools
- GA optimization for multi-dimensional, non-linear optimization:
  - In many physics cases, knobs have to be turned to improve performance
  - When the dimensionality of the problem is large, GA is the only hope
  - *GA simulation is only as good as its function evaluator*
  - Implemented two easily-adaptable computational paradigms
  - Applied to a number of problems in accelerator physics
  - *These serve as a template for other applications*

Summary

- Parallel computation on GPUs:
  - *Interdisciplinary approach*: physicists, computer scientists and engineers
  - Can lead to *orders-of-magnitude speedup* for some codes
  - GPU address the computational bottlenecks in two important accelerator physics problems:
    - CSR simulation
      - Numerical integration: *speedup of up to about 3 orders of magnitude*
    - Beam-beam interaction
      - Tracking and collision: *speedup of about 2 orders of magnitude*
  - We developed expertise on the subject to be used on future problems

- Overarching goal: Develop new tools to study *previously intractable* problems in accelerator physics and beyond

Details for the Work Presented

- Application of GA optimization (CASA collaboration):
  - Kramer, Jarvis & Terzić 2010, JLab Tech Note JLAB-TN-10-034
  - Terzić, Kramer & Jarvis 2011, PAC (*WEP167*)
  - Hofler 2012, PhD thesis
  - Hofler, Terzić, Kramer, Zvezdin, Morozov, Roblin, Lin & Jarvis 2013, *PR STAB* 16, 010101
    Mini-tutorial to be presented at NAPAC 2013 by Alicia Hofler
  - Terzić, Deitrick, Hofler & Krafft 2013, *PRL*, submitted
  - Henderson 2013, REU project
- GPU computation:

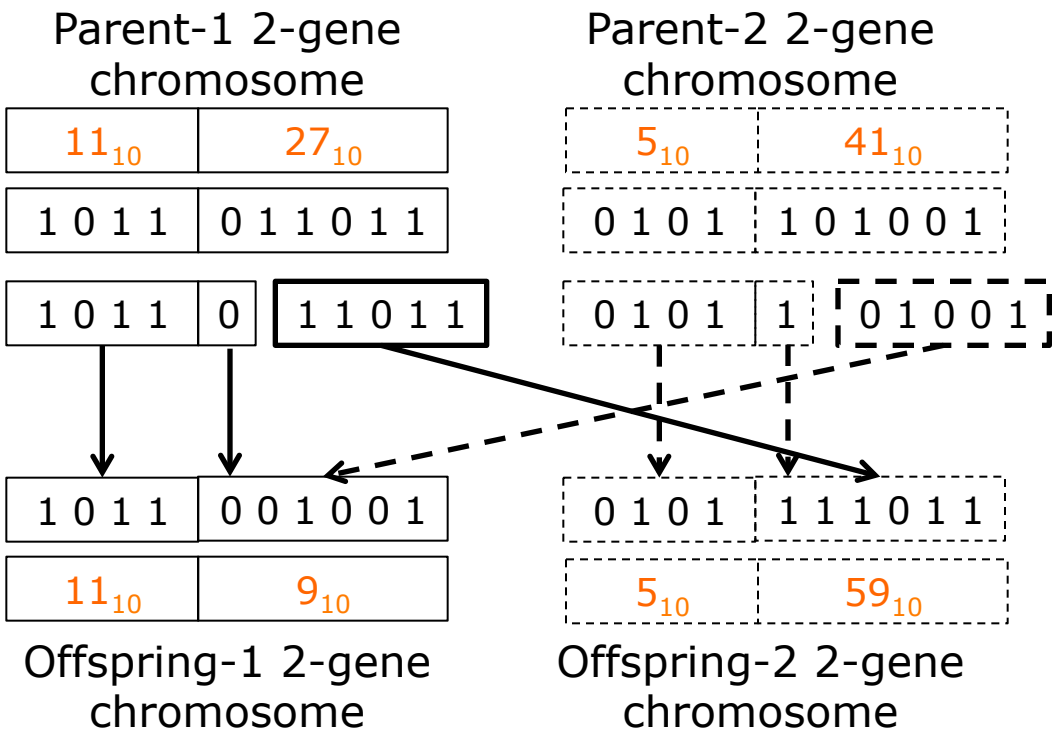
  Multidimensional integration (ODU / CAS collaboration)
  - Arumugam, Godunov, Ranjan, Terzić & Zubair 2013a,
    International Conference on Parallel Processing – 42nd Annual Conference (refereed)
  - Arumugam, Godunov, Ranjan, Terzić & Zubair 2013b,
    20th Annual International Conference on High-Performance Computing (refereed)
  - Arumugam, Godunov, Ranjan, Terzić & Zubair 2013c, in preparation

  Beam-beam code (ODU / CAS / CASA collaboration)
  - Roblin, Morozov, Terzić, Aturban, Ranjan & Zubair 2013, *IPAC* (MOPWO080)

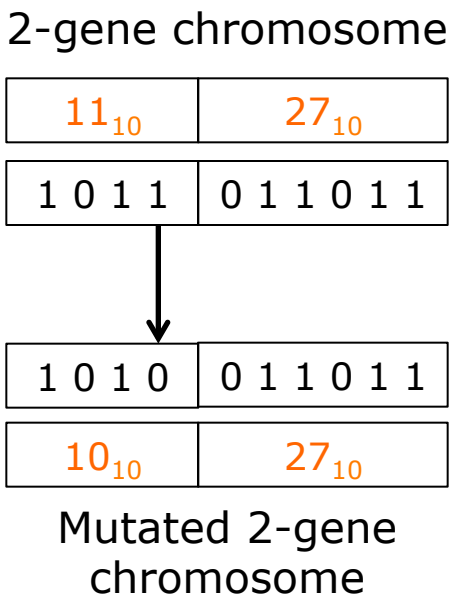
Backup Slides

Genetic Algorithm Basics

Crossover or Recombination



Mutation



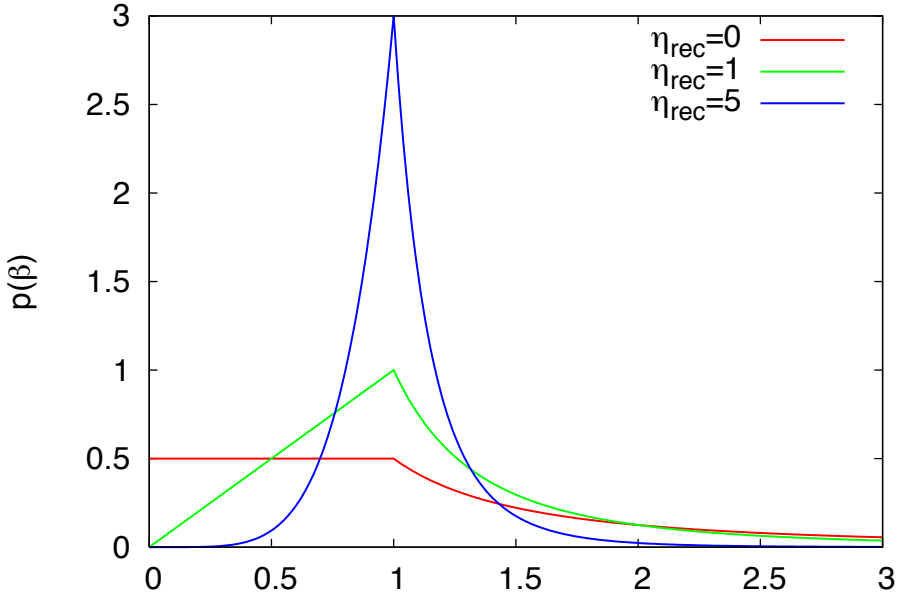
Slide courtesy of Alicia Hofler

Real Valued Recombination

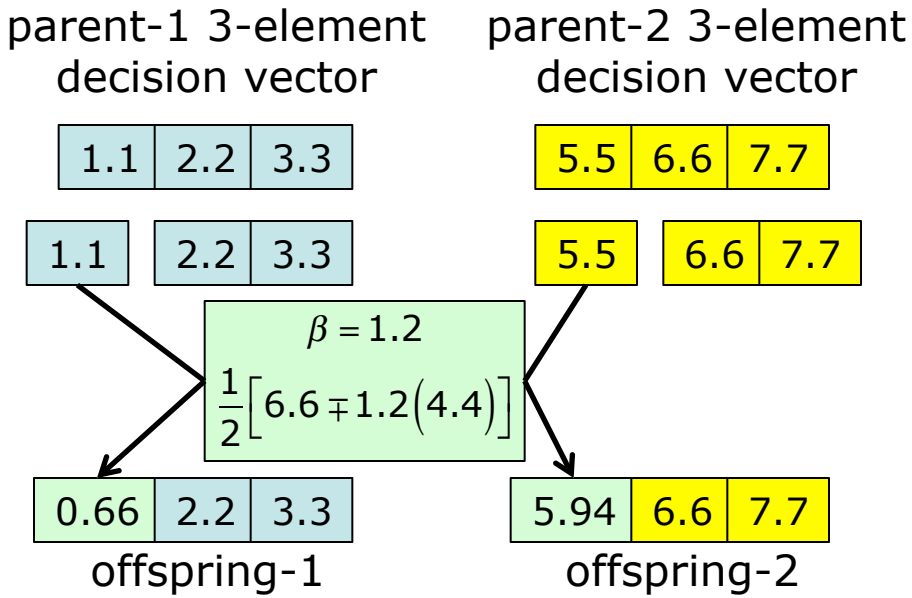
Simulated Binary Crossover

$$\beta = \left| \frac{x_{o_1} - x_{o_1}}{x_{p_1} - x_{p_1}} \right|$$

$$\left(x_{o_1}, x_{o_2}\right) = \frac{1}{2}\left[x_{p_1} + x_{p_2} \mp \beta \left| x_{p_1} - x_{p_2} \right| \right]$$



$$p\left(\beta\right) = \frac{1}{2}\left[1 + \eta_{\text{rec}}\right] \begin{cases} \beta^{\eta_{\text{rec}}} & 0 \le \beta \le 1 \\ \beta^{-\left(2+\eta_{\text{rec}}\right)} & 1 < \beta \end{cases}$$



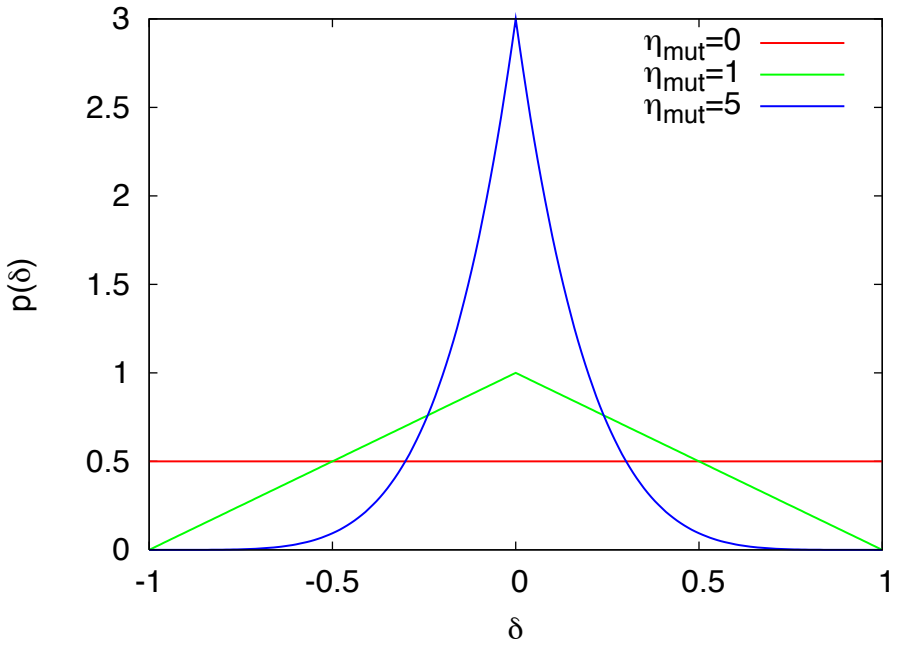
(Optionally, swap vector elements a priori)

Slide courtesy of Alicia Hofler

Real Valued Mutation

Polynomial Mutation

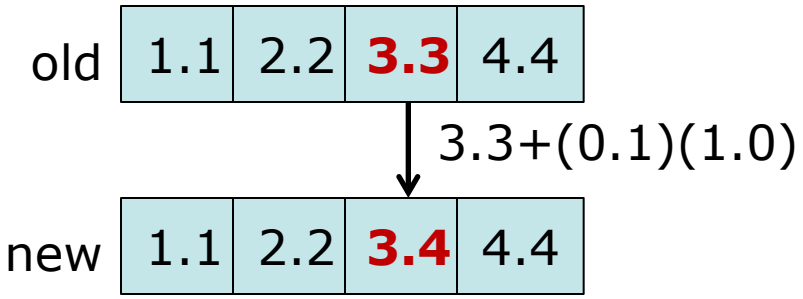
$$x_{mut} = x + \delta \Delta_{max}$$



$$\Delta_{max} = 1.0$$

$$\delta = 0.1$$

4-element decision vector



$$p(\delta) = \frac{1}{2}\left[1 + \eta_{mut}\right]\left[1 - |\delta|\right]^{\eta_{mut}}$$

Slide courtesy of Alicia Hofler

Optimizing Collider Working Point: GA Vs. Parameter Scan

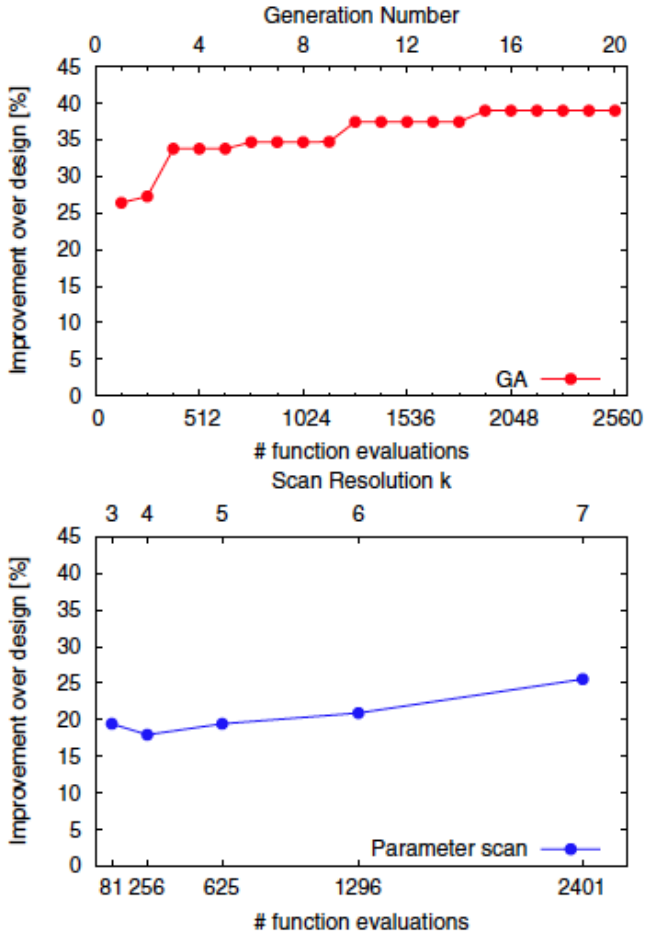


FIG. 6. Top panel: Improvement over the design luminosity after each generation for a GA-based optimization with 20 generations of 128 individuals in each. The improved working point is about 9% better than the one found in Fig. 4. Bottom panel: Improvement over design luminosity after a systematic parameter scan with resolution k in each parameter.

Hofler, Terzić, Kramer, Zvezdin, Morozov, Roblin, Lin & Jarvis 2013, *PR STAB* 16, 010101