EPJ Web of Conferences **251**, 03071 (2021) https://doi.org/10.1051/epjconf/202125103071
CHEP 2021

Evolutionary Algorithms for Tracking Algorithm Parameter Optimization

Peter Chatain[1], *Rocky* Garg[1], and *Lauren* Tompkins[1,*]

[1]Stanford University

Abstract. The reconstruction of charged particle trajectories, known as tracking, is one of the most complex and CPU consuming parts of event processing in high energy particle physics experiments. The most widely used and best performing tracking algorithms require significant geometry-specific tuning of the algorithm parameters to achieve best results. In this paper, we demonstrate the usage of machine learning techniques, particularly evolutionary algorithms, to find high performing configurations for the first step of tracking, called track seeding. We use a track seeding algorithm from the software framework A Common Tracking Software (ACTS). ACTS aims to provide an experiment-independent and framework-independent tracking software designed for modern computing architectures. We show that our optimization algorithms find highly performing configurations in ACTS without hand-tuning. These techniques can be applied to other reconstruction tasks, improving performance and reducing the need for laborious hand-tuning of parameters.

1 Introduction

Charged particle trajectory finding, called tracking, is typically the most computationally expensive task for particle physics experiments at colliders. Precise and efficient measurements of charged particle trajectories are critical for the entire physics program: from reconstructing low momentum tracks in order to identify primary vertices, to using medium momentum tracks to identify heavy flavor decays and calculate jet energies, to accurately measuring high momentum messengers of electroweak particles or new physics. Great effort is put into optimizing tracking algorithms as this effort yields both resource savings and physics performance improvements. However, these algorithms have many tuneable parameters which can make finding an optimal parameter set difficult. Approaches which automate the process of finding high performing parameter sets have the potential to reduce the effort spent in finding the configurations, as well as these approaches scan a larger range of parameters, allowing for potentially more efficient solutions.

Tracking typically has two components: track finding, a pattern recognition step in which track candidates are assembled from the data, and track fitting, a parameter estimation step in which the track parameters are determined from the track candidates. This paper focuses on the first stage of track finding, called track seeding, as a proof of principle for our parameter optimization approach. In the track seeding step, shown schematically in Fig. 1, groups of three space points, or clustered detector hits, are tested for compatibility with a helical

*e-mail: laurenat@stanford.edu

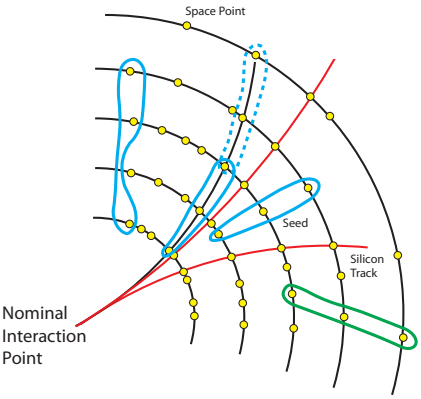


Figure 1. Schematic illustration of track seeding. Yellow filled circles represent space points. Blue and green loops indicate seeds, red lines indicates full tracks. Figure, modified with permission, from [1].

trajectory. Only seeds passing a set of selections are passed on for the track following step, which adds in space points from other detector layers. An ideal track seeding algorithm is very *efficient* at finding seeds for true particle trajectories, yet does not produce a proliferation of *fake* seeds which do not correspond to true trajectories, and does not produce *duplicate*, or multiple seeds per true charged particle. The later requirement on fake and duplicate seeds reduces the processing load on the downstream elements of the tracking algorithms. A more detailed description of the specific track seeding implementation is presented in section 2 .

In this work we consider two methods for parameter set tuning, evolutionary algorithms [2] and parameter sweeps [3], and compare these results to hand-tuning of the algorithms. Evolutionary algorithms, described in more detail in Section 4, are machine learning techniques inspired by natural evolutionary processes. Parameter sweeps are considered as a simple alternative, where a linear, iterative scan over parameter values is automated. As will be shown, evolutionary algorithms obtain the highest performing parameter sets for track seeding, as determined by the efficiency, fake rate, and duplicate rate. Processing time is not explicitly checked in this work, although it is implicitly tracked through duplicate and fake rates, and could be incorporated in a future iteration of this work.

2 Description of the Problem: Track Seeding

2.1 A Common Tracking Software (ACTS)

The work presented here is based on track seeding as implemented in the A Common Tracking Software (ACTS) suite [4]. ACTS is an experiment agnostic tracking software framework which supports parallel processing through strict thread safety. It is independent of both the detector technologies used and the magnetic field description of the experiment it is being used for. It is able to achieve this independence through the development of an event data model and geometry description that does not depend on any specific details of a particular detector and by implementing algorithms which are general mathematical formulations with no assumptions on detector geometry. Thus far it has been used with the ATLAS experiment geometry, the ATLAS HL-LHC upgraded detector geometry, the Open Detector [5], the Belle-II detector [6] and the LDMX detector [7].

2.2 Track Seeding in ACTS

As mentioned in the introduction, a seed is comprised of three space points, where each space point is the x, y, z coordinates of clustered detector hits. Because there is a known magnetic

field, these three space points define a unique helical particle trajectory. The seeding algorithm employed by ACTS [8] loops over many possible combinations of three hits, so the algorithm has $O(n^3)$ complexity. Utilizing configurable radius parameters, a subset of the detector space points can be selected to form seeds only from particular detector layers, reducing the number of combinations which must be tested. Then, the seeding algorithm relies on many other configurable parameters, described below, to reduce the number of possible good combinations and arrive at a reasonably quick algorithm. These parameters are heavily detector dependent, and many of them don't have an obvious value.

The tracking stage will use this seed as a starting point to find the full particle trajectory. If no seed exists for a particle, that particle will not be reconstructed. On the other hand, finding too many fake or duplicate seeds increases the time needed for tracking. We use the following definitions of the efficiency, fake rate and duplicate metrics to assess our track seeding performance:

- **Efficiency:** Fraction of true charged particles matched to a seed. True charged particles are required to have transverse momentum, p_{T}, greater than 500 MeV, pseudorapidity, $|\eta|$, of less than 2.5, and to have produced at least 3 hits in the layers used for seeding. This corresponds to a technical efficiency, which indicates an algorithm's ability to find a seed if a particle produced a sufficient number of hits.
- **Fake Rate:** Fraction of seeds that don't correspond to a true charged particle. A seed is considered not matched to a true charged particle if all of it's three space points do not correspond to a single truth particle.
- **Duplicate Rate:** Fraction of seeds that match to a true charged particle already found by another seed.

The goal of our work is to develop an algorithm that automatically finds the configuration on any detector geometry which achieves the highest efficiency while maintaining a low fake and duplicate rate. In addition, it should give insight on how the different parameters are affecting the performance to aid future work.

2.3 Configurable parameters

The track seeding algorithm has a total of 22 configurable parameters. In this work we have chosen to tune the following parameters as they have the largest effect on the overall performance:

- ***maxPtScattering:*** Charged particles will undergo multiple scattering due to material interactions in the detector. The seeding algorithm allows for seeds to have points on helical trajectories consistent with the expected multiple scattering based on the estimated p_{T} of the underlying particle. High p_{T} particles have very small scattering, however the seed estimate of the p_{T} is very poor in this regime. *maxPtScattering* is the maximum value of p_{T} considered for the multiple scattering consistency criterion. Any seed with an estimated p_{T} above this cut will not apply the multiple scattering consistency criterion. In later implementations of the track seeding code available after this work, this cut is applied such that use the multiple scattering estimated at this cut p_{T} value is used for the compatibility criterion.
- ***impactMax:*** The impact parameter is the closest distance between the interaction point (where the particles were created) and the helical path defined by the seed. The algorithm discards seeds with an impact parameter greater than *impactMax*.
- ***deltaRMin:*** The minimum radius between two space points in a seed.

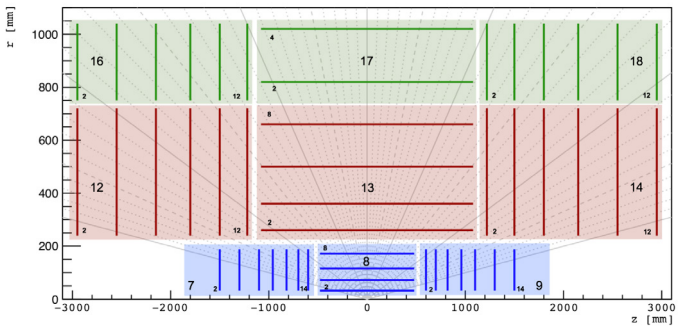


Figure 2: An $r - z$ view of the Generic Detector. The blue shaded area corresponds to the inner pixel layers used for seeding.

- ***deltaRMax:*** The maximum radius between two space points in a seed.
- ***sigmaScattering:*** This cut determines how many standard deviations of multiple scattering angle are allowed when calculating the compatibility of hits with a trajectory.
- ***radLengthPerSeed:*** The average radiation length a particle traverses. Similar to sigma scattering, this is used when calculating how much a particle will scatter and depends upon the material of the detector.
- ***maxSeedsPerSpM:*** Given a middle space point, if multiple seeds exist, they are sorted based on how likely they are to be a good seed. Then, only the first $n + 1$ seeds, where n is less than *maxSeedsPerSpM* are chosen.

The majority of these parameters are related to the detector geometry, either through the positioning of the physical layers (e.g., *deltaRMin*, *deltaRMax*) or through the material of the detector (e.g., *sigmaScattering*, *radLengthPerSeed*).

3 Detector and Datasets

We tested our algorithm on the **Generic Detector**, a detector design created for the Track ML challenge [5] and meant to function in a high pile-up, High Luminosity LHC-like environment. The Generic Detector is a hermetic collider detector design, with four layers of barrel short pixels and 7 short pixel disks. It is shown in Fig. 2. We used two input datasets for this detector: a sample containing 10000 muons per event and a sample of top-quark pairs ($t\bar{t}$) produced at $\sqrt{s} = 14$ TeV with 200 additional pile-up vertices per event generated with Pythia8 [9]. The $t\bar{t}$ sample had approximately 2000 findable particles per event. The single muons were produced with flat distributions in $1/p_{\mathrm{T}}$, $|\eta|$ and impact parameter over the acceptance of the detector.

4 Methods

4.1 Evolutionary Algorithms

We implemented an evolutionary algorithm using Distributed Evolutionary Algorithms in Python (DEAP) [10]. Evolutionary algorithms use a process analogous to natural evolution. The algorithm defines a *population* of *individuals*, where each individual represents

one parameter set of the seeding algorithm. We initialized the population as 50 copies one track seeding parameter configuration. Then, in each iteration of the evolutionary algorithm, called a *generation*, every individual is evaluated by running the seeding algorithm on the test dataset, and then calculating a score for that individual utilizing a custom scoring function. After evaluating the population, a selection function randomly chooses individuals such that better performing ones are more likely to stay, and worse performing ones are more likely to be removed.

For our selection function, we chose a tournament function with size 3, the default size in DEAP. A tournament of size n proceeds as follows: choose n random individuals, pick the one with the highest score, and repeat this 50 times so that the population stays a constant size. Next, each individual has a 30% chance of being selected for mutation. If selected, each parameter within an individual has a 20% chance of being mutated. The mutation for a parameter j in individual i is given by

$$x_j^{(i)} := x_j^{(i)} + \epsilon \mid \epsilon \sim \mathcal{N}(0, s_j^{(i)}) \tag{1}$$

where $s_j^{(i)}$ is the unique *strategy* for each parameter j within an individual i. This strategy is initialized randomly between 0.01 and 0.3, and also gets mutated itself along with $x_j^{(i)}$. To make sure each parameter is mutated by a consistent magnitude, we normalize x_j to be close to 1 before mutation, and then scale it back after mutation. Population sizes of 40–100 were tested and found to have no effect on the outcome. 16 generations were chosen to match the number of cores available for this study. [11] suggests that outcome of the evolutionary algorithm does not depend strongly on the parameters of the evolutionary algorithm itself (e.g., population size, number of generations, tournament size, mutation rate, etc.).

Considerable discretion is given to the choice of scoring function. Ideally it reflects the overall goals of the optimization problem. The simplest possible case would be to optimize a single metric, such as the efficiency of the track seeding. However, in most cases the optimization problem is more complicated and a variety of metrics must be combined, with a relative weight reflecting their relative importance in the overall optimization problem, into a single scoring function. We choose to optimize a high efficiency and low fake and duplicate rate, with relatively higher importance given to maintaining a high efficiency, by constructing the following scoring function:

$$\texttt{Score} = \texttt{Efficiency\%} - \frac{\texttt{Fake\%} \times \texttt{Duplicate\%}}{K} \tag{2}$$

We tested several K values for our study and found that $K = 1000$ offered a reasonable balance between efficiency and fake/duplicate rates. For example, with $K = 100$ maximum efficiencies of $\sim 60\%$ were obtained, whereas with $K = 1000$ efficiencies of greater than 98% were obtained for reasonable fake and duplicate rates.

During the training process, certain configurations break the seeding algorithm. For example, we can not have negative values for most parameters, so very loose bounds were placed on each parameter during training.

4.2 Parameter Sweep

Although the user will not know certain parameters, they can still define reasonable bounds on each parameter. For example, we can exclude checking *deltaRMin* of less than a value known from the detector geometry. This allows us to "sweep" over ranges of allowed parameters automatically.

For each parameter, we split up the range between the maximum and minimum for that parameter into 100 different values. Then, holding the other parameters fixed, we chose the best performing value as determined by the scoring function (Eqn. 2), and use this value when optimizing the next parameter. Then, we loop over the parameters until the score of the algorithm remained unchanged. This is a greedy algorithm as it optimizes one parameter at a time.

5 Results

5.1 Generic Detector

The generic detector track seeding performance was first "hand-tuned" by performing linear scan of the parameters above and picking the configuration which gave the best efficiency for a suitably low total number of produced seeds, which is a proxy for the fake and duplicate rate. The values obtained are listed in Table 1 and the corresponding parameters are in Table 2. Several iterations were done of this process, but because the exploration of the multi-dimensional parameter space was very inefficient, we do not claim that our working point was the best that could possibly be found by hand-tuning. We took a best effort for reasonable resources approach and then explored automated alternatives.

To test the evolutionary algorithm's ability to optimize the track seeding parameters, four of the parameters listed in Section 2.3 were changed from their hand-tuned value by a factor of two, and the sigmaScattering parameter, a key parameter for the efficiency, was changed from it's default value of 2 to 0.2 to prove that the algorithm works without starting from a hand-tuned working point. The top row of Figure 3 shows the evolution of the best performing individuals for two variables, maxSeedsPerSpM and sigmaScattering, over the course of 200 generations as a function of the performance metrics they most affect: the duplicate rate and efficiency, respectively. These figures show the typical progression of a variable in the evolutionary algorithm process, and show the impact on one component of the scoring function in terms of the best performing individual. Table 1 shows the achieved efficiency, fake rate and duplicate rate for both the muon and $t\bar{t}$ sample. A higher efficiency was obtained for both the muon and $t\bar{t}$ samples as compared to the hand-tuned values. The scoring function for the evolutionary algorithm allowed a higher duplicate rate than found with the hand-tuned method and the fake rates were $\sim$ 25% lower for the muon sample and $\sim$ 25 % higher in the $t\bar{t}$ sample. The difference in these two datasets are most likely attributed to the higher proportion of low momentum tracks and nuclear interactions in the $t\bar{t}$ sample. Results for the parameter sweep for the muon sample are also shown in Table 1, which finds performance similar to the hand-tuned working point. A plot of the best efficiency, fake and duplicate rate as a function of the sweep iteration is shown in Figure 4a. It can be seen that the fake and duplicate rate are relatively unchanged while the efficiency is improved. No results are available for the $t\bar{t}$ sample because the parameter sweep algorithm failed to converge too many times to be a useful comparison.

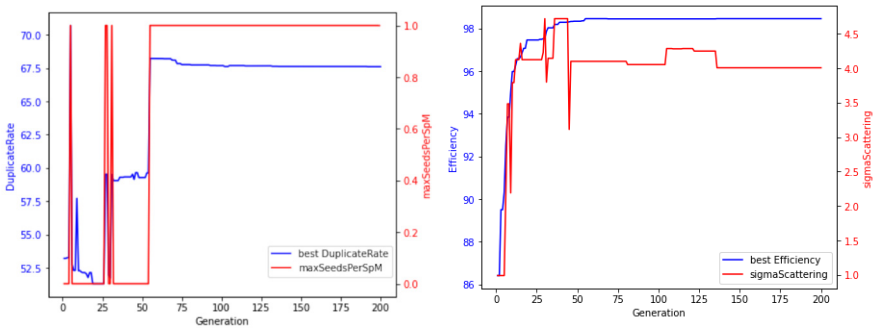
In the case of the evolutionary algorithms, overfitting was checked by dividing the samples into test and train samples, where the parameters were derived from the training sample and the efficiencies, fake and duplicate rates were checked in the test sample. The test and train sample metrics were compatible within statistical uncertainties, indicating no overfitting.

6 Conclusions

We have shown that evolutionary algorithms are a powerful and efficient method for determining high performing parameter sets for track seeding. We have also used our methods

Dataset	Notes	Efficiency %	Fake %	Duplicate %
Muon	Hand	98.9	8	54
Muon	EA	99.4	6	70
Muon	PS	98.28	4.44	63.95
$t\bar{t}$	Hand	96.6	38	34
$t\bar{t}$	EA	98.34	47	72.8

Table 1: Generic Detector results comparing best values found by hand-tuning (Hand), evolutionary algorithm (EA) and parameter sweep (PS) for the muon and $t\bar{t}$ samples. Note there are no parameter sweep values for the $t\bar{t}$ sample due to a high rate of failure of this method for this sample. Uncertainties on the efficiencies are O(0.25%).

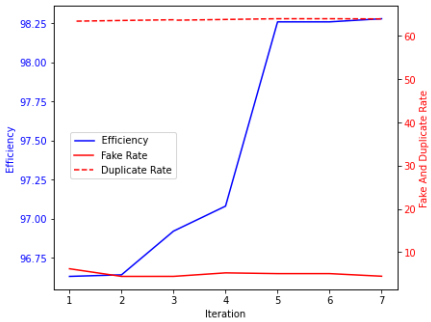


(a) (b)

Figure 3: Results for the best individual as a function of generation number for the muon sample with the Generic Detector (top). The blue curve and left-axis in each plot shows the duplicate rate (left) or efficiency (right) for the best performing individual of that generation and the red curve shows the value of the variable of interest, with it's name and value on the right $y-$axis, for that individual.

Variable	Generic Hand	Generic EA
maxPtScattering [GeV]	1200	30
impactMax [mm]	3	1.1
deltaRMin [mm]	1	0.25
sigmaScattering [σ]	4.0	4.0
deltaRMax [mm]	80.	60
maxSeedsPerSpM	2	1
radLengthPerSeem [λ]	0.005	0.0023

Table 2: Comparison of optimized parameter values for the generic detector, as determined from the muon sample.



(a)

Figure 4: Efficiency, fake and duplicate rate for the parameter sweep as a function of the iteration number in the first sweep for the Generic Detector.

for other detectors such as the LDMX and ATLAS detectors, and preliminary results indicate similar success. In future work we plan to embed this approach into the ACTS framework and to apply it to other stages of track reconstruction.

We note that the process described above is naturally suited to parallelization. For our studies we used a machine of 16 cores, which allowed us to evaluate 12-16 configurations per generation per event in parallel, leading to 30 minutes of total (wall) running time for an optimization run. This is certainly a lower bound on what could be used for this process. More generations and/or more events could be used in a system with higher parallelism.

7 Acknowledgments

This work was supported by the National Science Foundation under Cooperative Agreement OAC-1836650.

We thank Andreas Salzburger, Xiaocong Ai and Robert Langenberg for discussions about ACTS and track seeding.

References

[1] H.M. Gray, Ph.D. thesis (2010), presented on 09 Nov 2010, `https://cds.cern.ch/record/1309943`

[2] P.A. Vikhar, *Evolutionary algorithms: A critical review and its future prospects*, in *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)* (2016), pp. 261–265

[3] M.E. Samples, M.J. Byom, J.M. Daida, *Parameter Sweeps for Exploring Parameter Spaces of Genetic and Evolutionary Algorithms* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2007), pp. 161–184, ISBN 978-3-540-69432-8, `https://doi.org/10.1007/978-3-540-69432-8_8`

[4] P. Gessinger, H. Grasland, H. Gray, M. Kiehn, F. Klimpel, R. Langenberg, A. Salzburger, B. Schlag, J. Zhang, X. Ai, EPJ Web Conf. **245**, 10003 (2020)

[5] S. Amrouche, L. Basara, P. Calafiura, V. Estrade, S. Farrell, D.R. Ferreira, L. Finnie, N. Finnie, C. Germain, V.V. Gligorov et al., The Springer Series on Challenges in Machine Learning p. 231–264 (2019)

[6] T. Abe, I. Adachi, K. Adamczyk, S. Ahn, H. Aihara, K. Akai, M. Aloi, L. Andricek, K. Aoki, Y. Arai et al., *Belle II Technical Design Report* (2010), 1011.0352

[7] T. Åkesson, A. Berlin, N. Blinov, O. Colegrove, G. Collura, V. Dutta, B. Echenard, J. Hiltbrand, D.G. Hitlin, J. Incandela et al., *Light Dark Matter eXperiment (LDMX)* (2018), 1808.05219

[8] *Track Seeding, ACTS Core Library Docs*, https://acts.readthedocs.io/en/latest/core/seeding.html, accessed: 2020-02-22

[9] T. Sjöstrand, S. Ask, J.R. Christiansen, R. Corke, N. Desai, P. Ilten, S. Mrenna, S. Prestel, C.O. Rasmussen, P.Z. Skands, Comput. Phys. Commun. **191**, 159 (2015), 1410.3012

[10] F.A. Fortin, F.M.D. Rainville, M.A. Gardner, M. Parizeau, C. Gagné, Journal of Machine Learning Research **13**, 2171 (2012)

[11] M. Sipper, W. Fu, K. Ahuja, J.H. Moore, BioData Mining **11** (2018)