Filtering track candidates in TPC track reconstruction

Pavel Belecky (belecky@ispras.ru), Alexander Kamkin (kamkin@ispras.ru)
Veronika Burdelnaya, Jamilya Erkenova, Ilya Kozmin, Dmitriy Mikhalevich

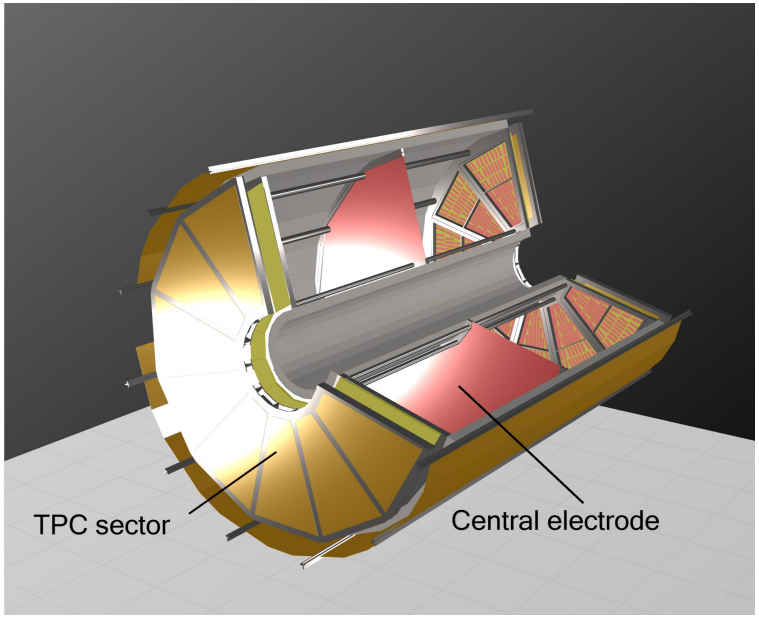
Plekhanov Russian University of Economics
Ivannikov Institute for System Programming of the Russian Academy of Sciences

JINR, Dubna | April 24, 2024

Tracking in MPD

Time Projection Chamber (TPC) is the main tracking detector of the central barrel. It provides precise momentum measurements and particle identification

ACTS Common Tracking Software* (version 21) was used



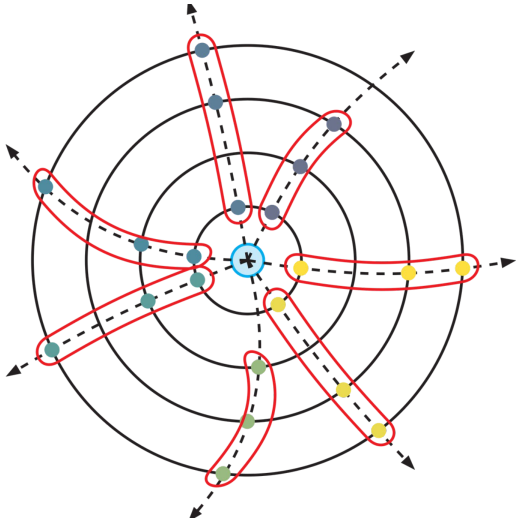
TPC schematic view

** https://acts.readthedocs.io/en*

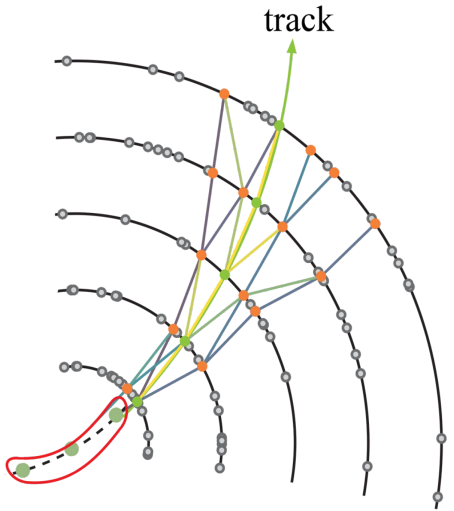
Track reconstruction

Reconstruction steps

1. Track seeding
2. Track-candidates finding
3. Ambiguity resolution



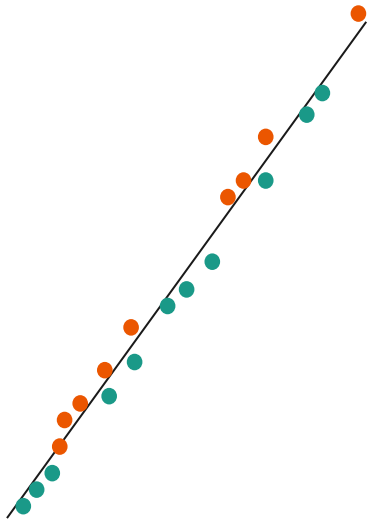
Schematic track seeding



Schematic track finding

Uncertainty resolution

Resolution of duplicate tracks (overlaps between track-candidates) and rejection of fake tracks (incorrect combinations of unrelated clusters)



green — 1st track-candidate
orange — 2d track-candidate

Aim and objectives

Aim

Decrease the number of fake and duplicate tracks among track-candidates

Objectives

1. Research existing methods for uncertainty resolution
2. Develop methods of uncertainty resolution
3. Implement methods of uncertainty resolution
4. Compare implementations

Comparison criterion

Tracking efficiency, duplicates and fakes are calculated only among the following particles:

* $|\eta| < 1.5$
* $p_t > 100$ MeV
* nHits ≥ 9

Metrics sorted by their importance:

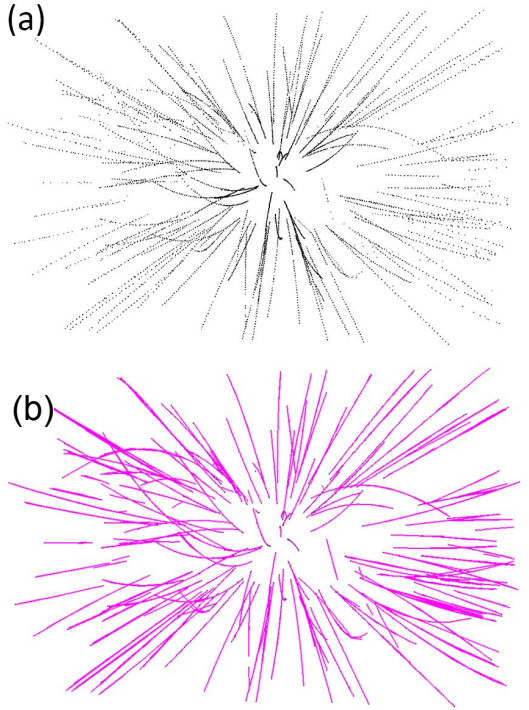
1. Fake rate should tend to zero
2. Efficiency should not decrease
3. Duplicate rate should decrease

Test data

Used package: UrQMD*

Parameters:

- particles — *197 Au+ Au*
- center of mass energy — *7 GeV*
- number of events — *10 000*
- Z vertex — *fixed to 0*

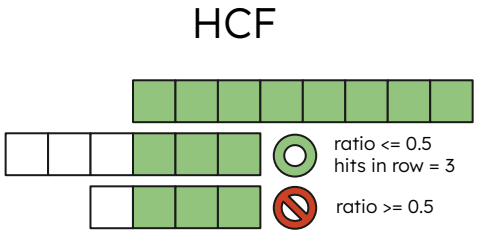


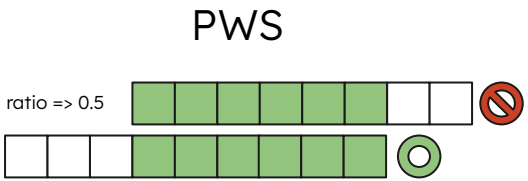
a) hits
b) recognized tracks

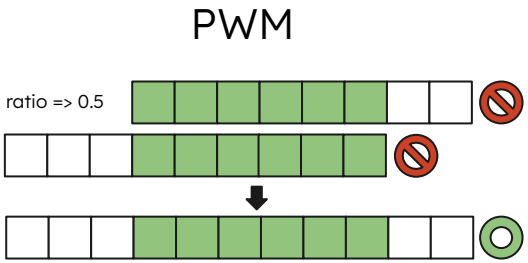
*https://urqmd.org

Methods for uncertainty resolution

1. Hit-coverage-based track filtering (HCF)
2. Pairwise track matching and selection (PWS)
3. Pairwise track matching and merging (PWM)
4. Track proximity graph selection (PGS)
5. Track proximity graph merging (PGM)
6. Neural network-based selection (NNS)*



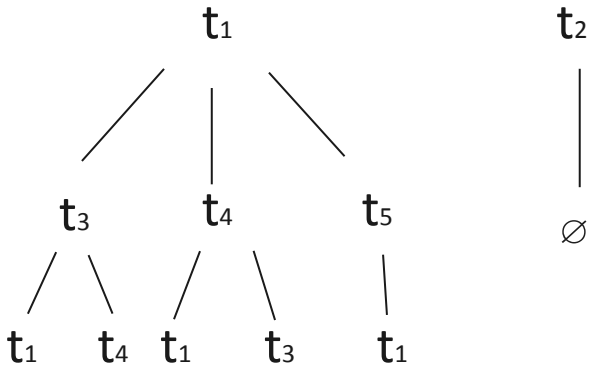


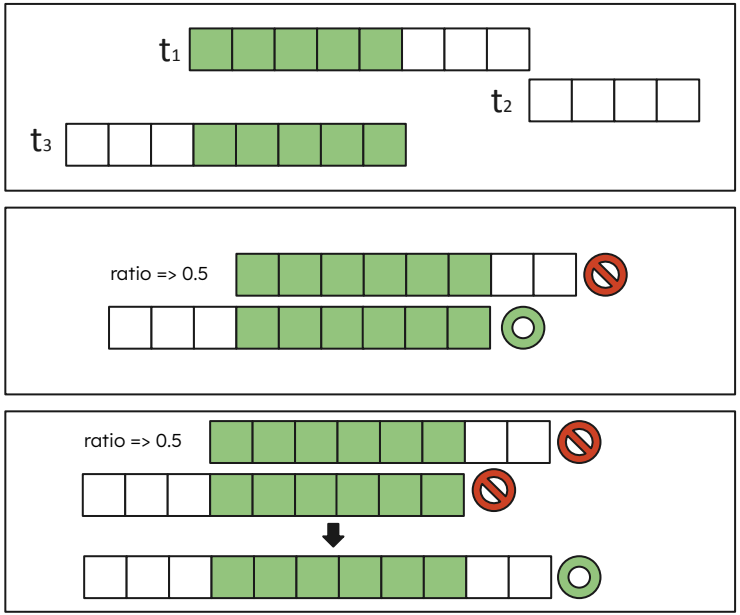


*https://acts.readthedocs.io/en/latest/plugins/MLAlgorithms.html#ambiguityresolutionmlalgorithm

PGS, PGM

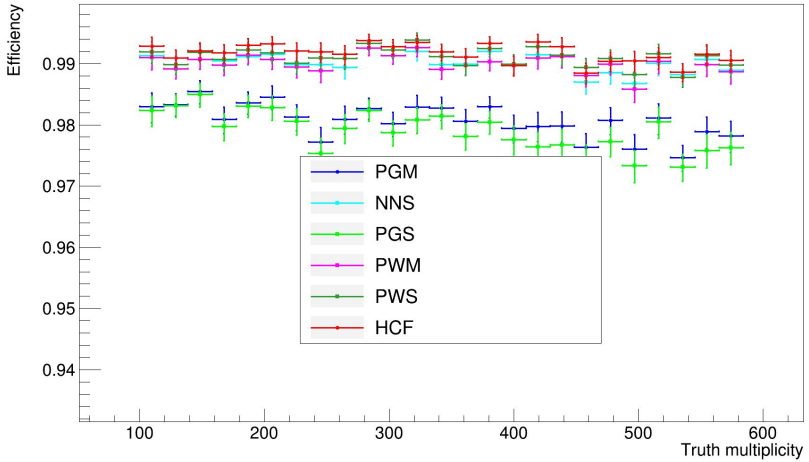
Tracks are clustered by graphs based on shared hits. Then the same algorithm as PWS and PWM is applied to clusters





Comparison: efficiency (M)

The graphs show average results on 10 000 events

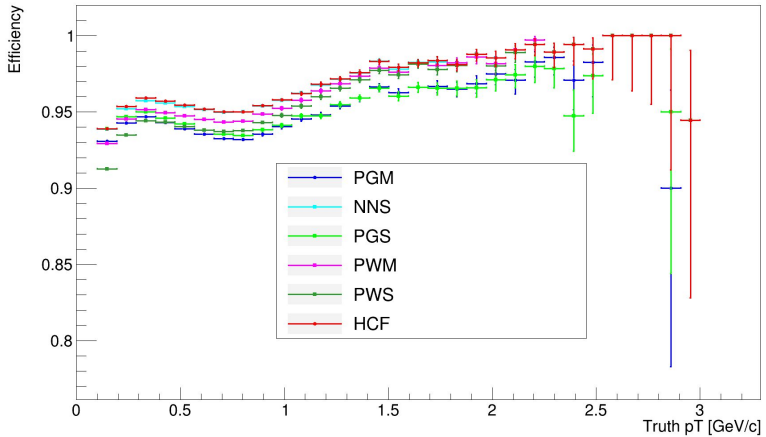


Efficiency VS multiplicity

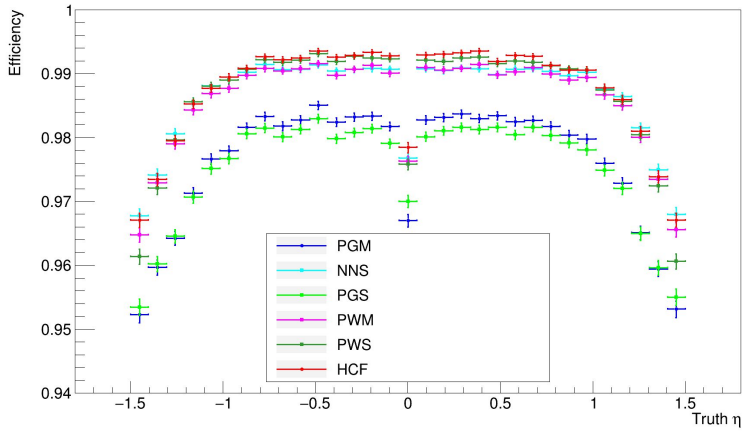
1. Hit-coverage-based track filtering (HCF)
2. Pairwise track matching and selection (PWS)
3. Pairwise track matching and merging (PWM)
4. Track proximity graph selection (PGS)
5. Track proximity graph merging (PGM)
6. Neural network-based selection (NNS)*

Comparison: efficiency (pt, η)

The graphs show average results on 10 000 events



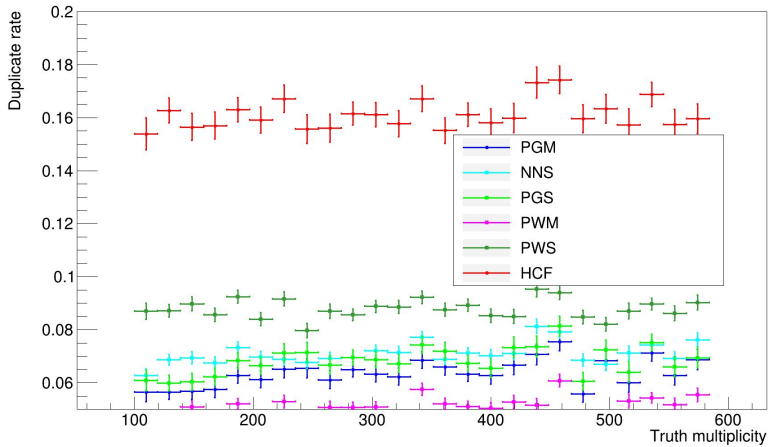
Efficiency VS transverse momentum



Efficiency VS pseudorapidity

Comparison: duplicate rate (M)

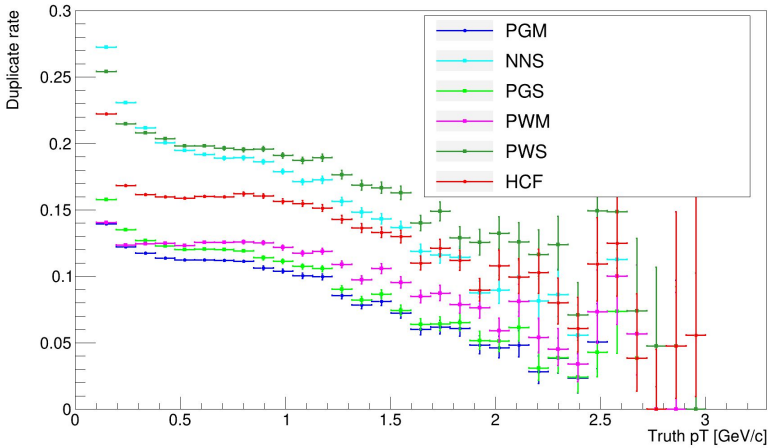
The graphs show average results on 10 000 events



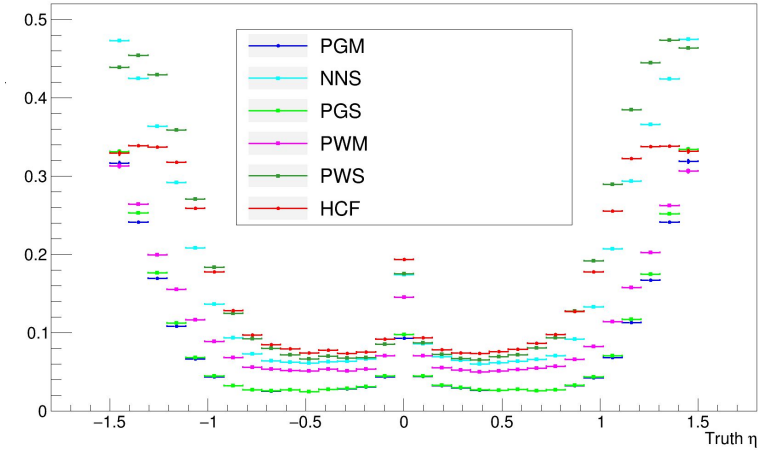
Duplicate rate VS multiplicity

Comparison: duplicate rate (pt, η)

The graphs show average results on 10 000 events



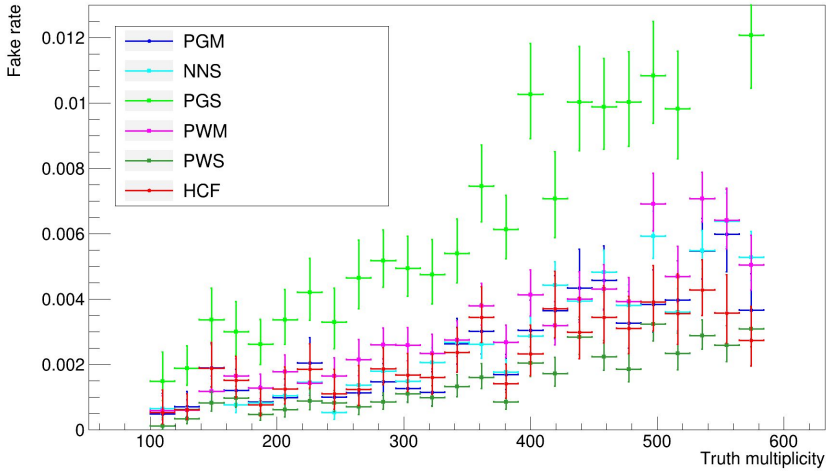
Duplicate rate VS transverse momentum



Duplicate rate VS pseudorapidity

Comparison: fake rate (M)

The graphs show average results on 10 000 events



Fake rate VS multiplicity

Results

	HCF	PWS	PWM	PGS	PGM	NNS
Fake rate, %	0.008	0.01	0.03	0.04	0.01	0.04
Efficiency, %	99.5	99.5	99.4	98.6	98.8	99.5
Duplicate rate, %	16.5	17.4	7.8	7.4	7	19

Future work

1. Tune seeding parameters using black-box optimization
2. Tune track finding parameters using black-box optimization
3. Adapt our new ACTS developments to a new ACTS version

Thank you!

Our repository:
github.com/PlekhanovRUE/mpd_tpc_tracker

Winning method algorithm

1. Track-candidates are sorted by length;
2. Starting with the first track, all hits are marked as “used”;
3. For every next track, ratio of the number of “unused” hits to the length of the track is calculated;
4. The number of segments of newHitsInRow hits in a track is calculated;
5. If the ratio in step 3 is less than newHitsRatio or there are no segments in step 4, then this tracks is discounted. Otherwise, all hits are marked as “used”;
6. Repeat 3-5.

newHitsRatio = 0.25

newHitsInRow = 6