Performance of Primary Vertex Reconstruction on High Pile-up Events at LHC

Cong-Qiao Li

School of Phsyics, Peking University, Beijing 100871, China

Supervisor: Federico Meloni

ATLAS Group, Deutsches Elektronen-Synchrotron DESY, 22607 Hamburg, Germany

September 5, 2018

Abstract

An efficient algorithm of primary vertex reconstruction on high pile-up events is a prerequisite for the following analysis. This report focuses on the performance and optimization of the iterative vertex finder algorithm (IVF), the method adopted in LHC Run 1 data-taking. A set of evaluation indicators is introduced to assess and optimize the algorithm by matching the vertices to truth interactions. A revised version of IVF is further presented, containing two parameters as thresholds for different track containers. The parameters are optimized using a linear weight function with the designed indicators. The result implies no absolute criterion is available on the best performance and that the original IVF algorithm can be a good choice. Finally, three known algorithms are compared and applied to real data sets to see their actual performance.

Contents

1 Introduction

Primary vertex reconstruction, a process that reconstructs the proton-proton (pp) interaction points (known as primary vertices), is a main focus of ATLAS and CMS experiment in LHC. The process utilizes the reconstructed tracks as input to fit the position of each possible vertex and allocate each track to a fitted vertex. For high pile-up events in HL-LHC, the average number of inelastic pp interactions per bunch crossing (donated as μ) can be up to 200, which makes the procedure of vertex reconstruction a prerequisite to carry on the following analysis. A 3D visualization of tracks and reconstructed vertices are shown in Figure 1, indicating the complex pattern of tracks for higher μ.

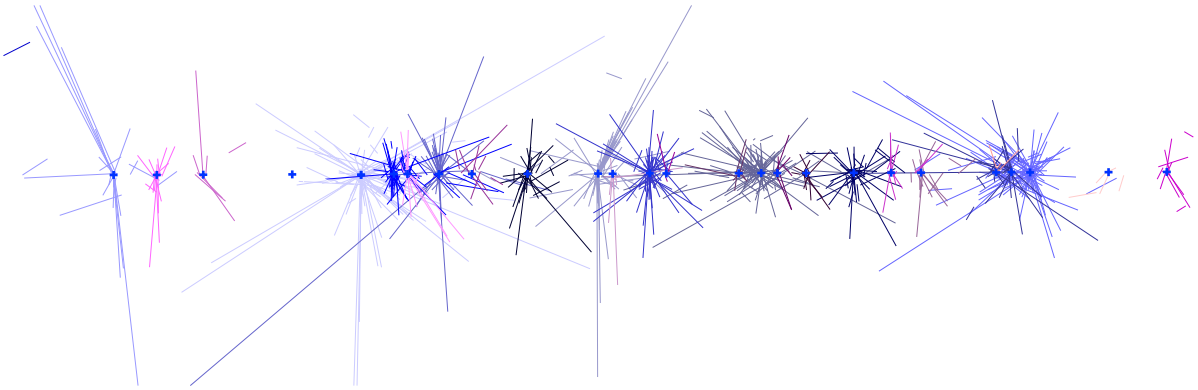

Figure 1: A 3D visualization of tracks and reconstructed vertices. The various colors of tracks indicate their allocations to different vertices.

In this document, we present the latest description of the most used algorithms in the ATLAS analysis software and apply a thorough check on their general performances. A set of indicators is introduced to evaluate the performance. A test on important variables which impact on the performance of IVF algorithm is implemented, and three kinds of algorithms are introduced and compared. The three algorithms are also applied to real data to see their actual performance.

2 Algorithm descriptions

All primary vertex reconstruction algorithms focus on two aspects: vertex finding and vertex fitting. Vertex finding refers to finding the position of a vertex candidate as a seed, using some selected tracks as input, while vertex fitting emphasizes the fitting techniques to obtain a more precise position of a primary vertex. Depending on different combinations of the finding and fitting algorithm, there are several methods ever designed as a vertex reconstruction tool in ATLAS. Below is a simple description of one algorithm that is mostly discussed in this paper, while the other two algorithms are described in Sec. 3.3.

2.1 Iterative vertex finder algorithm

The iterative vertex finder algorithm (IVF) was first adopted by ATLAS during the LHC Run 1 data-taking. The method is an iterative procedure to reconstruct the primary vertex and allocate some tracks to this vertex. The algorithm starts from a selection criterion on the track. Tracks passing the criteria form the seed pool and are used as input for the following iterative steps [1].

1. A vertex finding algorithm named `ZScan` is applied to tracks in the seed track pool to find a vertex seed (corresponding to the mode in z of the track distribution). Tracks near the seed vertex are roughly selected and enter the fitting procedure.

2. The tracks and the seed are used to estimate the best vertex position with a fit, which is an iterative annealing procedure. A weight factor is assigned to each track responsible for the compatibility with the vertex candidate. In each iteration, the less compatible tracks are down-weighted using

$$\omega(\hat{\chi}^2) = \frac{1}{1+\exp\left(\frac{\hat{\chi}^2-\hat{\chi}^2_{\text{cutoff}}}{2T}\right)}, \quad \text{where } \hat{\chi}_{\text{cutoff}} = 3, \tag{1}$$

   and the vertex position is recomputed. The temperature T decrease following a pre-defined sequence and converges at 1 in the last iteration.

3. After the vertex position is determined, tracks that are incompatible with the vertex by more than 7σ deviation ($\chi_{\text{cut}} = 7$) are removed and return to the seed track pool.

4. The procedure is repeated with the remaining tracks.

3 Performance

In this section, we will evaluate the performance of the vertex reconstruction algorithms in various ways. First, we show some general properties, then the parameter dependency of IVF algorithm is performed, and the variables are optimized. We also make a comparison between the performances of all three algorithms.

3.1 General performance

In real cases, it is impossible to perfectly reconstruct all interactions. Three main sources of inefficiencies are considered: (i) vertex merging, (ii) the splitting of a single interaction in multiple vertices, and (iii) fake vertices arising from the wrongly reconstructed tracks and combinatorics. It is interesting to see how they influence the vertex reconstruction as a function of μ. We use Monte Carlo (MC) simulation of $t\bar{t}$ event to test the performance of primary vertex reconstruction. The average μ over all 4000 events is 60. The distribution of the average number of reconstructed vertices as a function of

μ is shown in Figure 2. As can be seen, the increase in the number of reconstructed vertices is slower than that in the number of interactions, indicating the degradation of reconstruction algorithm as μ goes up.

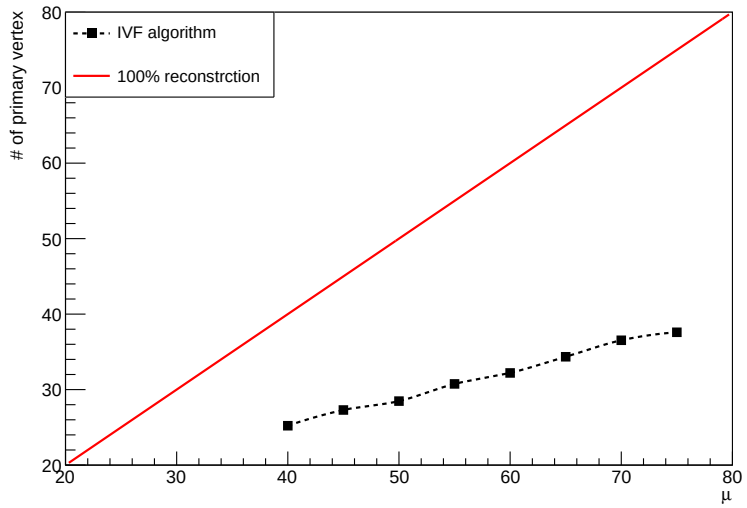

Figure 2: The average number of reconstructed vertices as a function of the number of interactions μ. The dashed black line and spots give the reconstruction result of IVF algorithm, while the solid red line shows the ideal case of 100% reconstruction efficiency.

To have a overall look at how vertices are influenced by μ, Figure 3 shows three kinds of normalized distributions on different ranges of μ for four variables characterize the vertices: the average number of associated tracks, sum of transverse momentum p_T of tracks, the fit χ^2, and the degrees of freedom used in the fit. As can be seen, the variables for larger μ spread a wider range. Figure 4 shows that all variables mentioned above tend to increase with μ. The degradation of vertex reconstruction for higher μ indicates that the fitted vertices are more likely to merge, thus more tracks are associated to the vertex. This also brings up the sum of p_T, χ^2 and degrees of freedom when μ goes up.

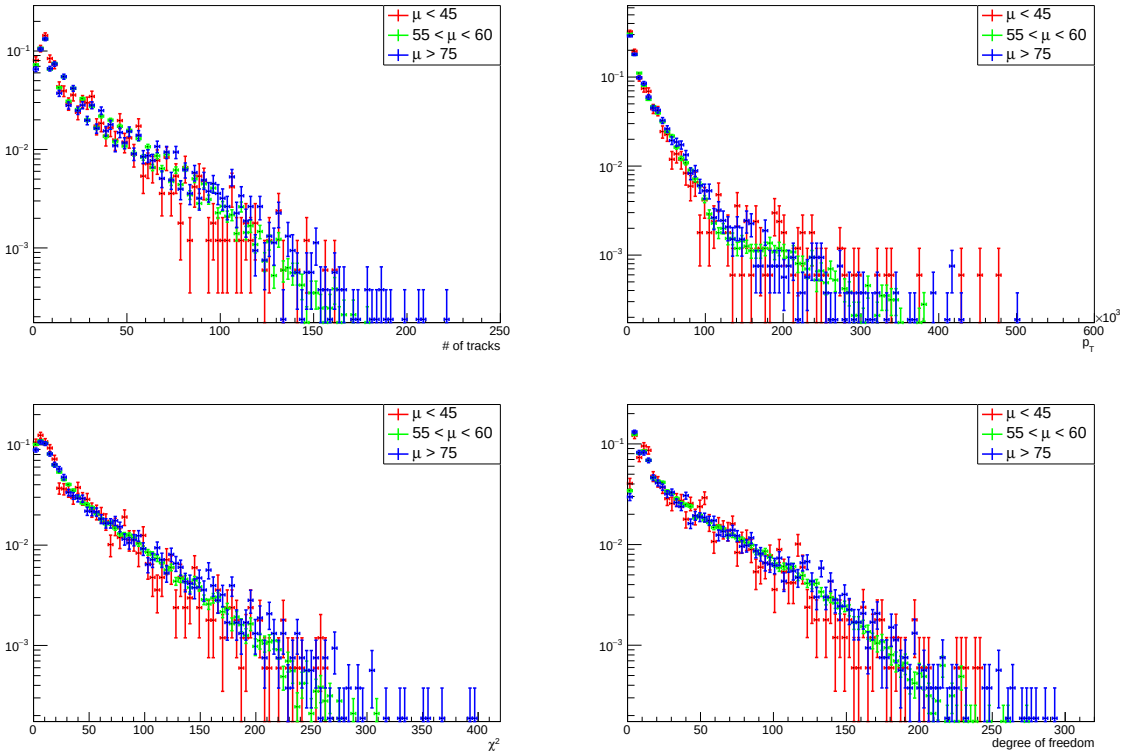

Figure 3: Normalized distribution of the average number of associated tracks (top left), sum of transverse momentum p_T of tracks (top right), χ^2 for the vertex (bottom left), and degree of freedom used in the vertex fit (bottom right) for three ranges of μ.

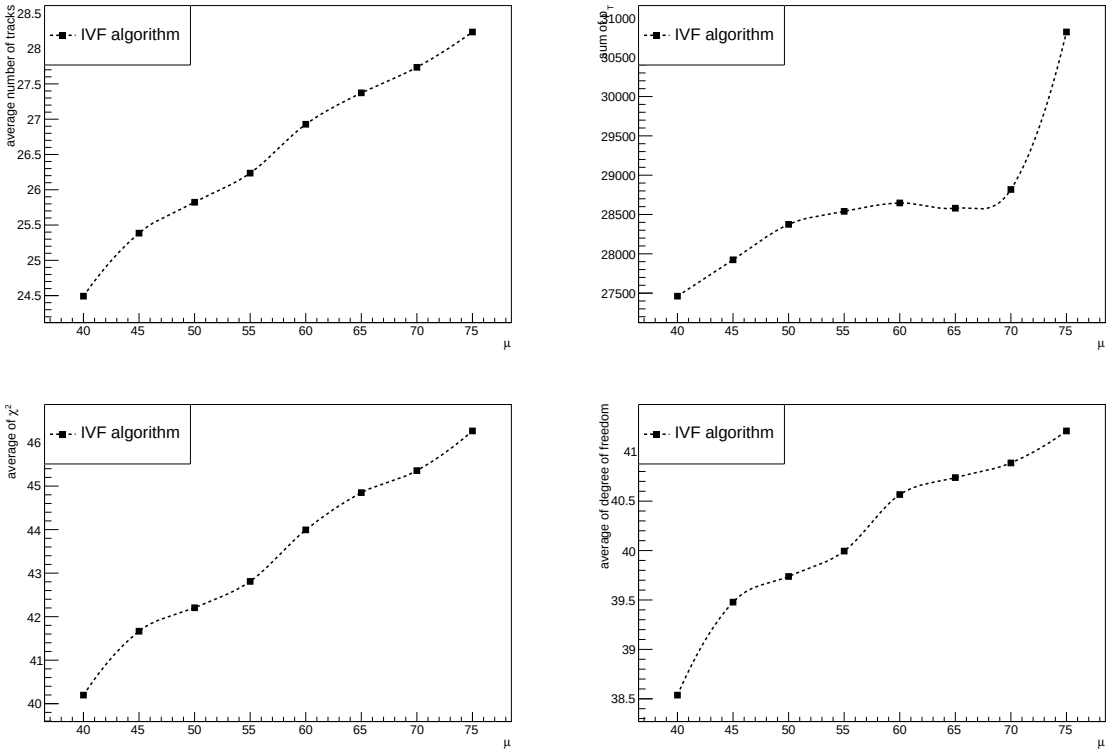

Figure 4: The average number of associated tracks (top left), sum of transverse momentum p_T of tracks (top right), χ^2 for the vertex (bottom left), and degree of freedom (bottom right) for each vertex as a function of the number of interaction μ.

An algorithm which matches the primary vertices and truth interactions is also introduced [1] to evaluate the quality of a reconstructed vertex. All vertices are classified into four types: matched, merged, split, or fake, with the method introduced as follows. The algorithm retrieves the weight for each tracks measuring its relevance with a truth particle. The sum of weights for a single vertex is normalized to 1. A fraction weight for a branch of tracks coming from a real interaction can be thus calculated, and the one with the highest weight is assigned as the "leading interaction". The track which has no corresponding interaction is classified as a fake track. The vertex is thus defined as:

(i) a matched vertex if tracks from the leading interaction contribute to 70% of the total weight, and no other vertex owns the same leading interaction;

(ii) a merged vertex if the leading interaction weights less than 70%, and no other vertex owns the same leading interaction;

(iii) a split vertex if it shares the same leading interaction with some other vertex.

(iv) a fake vertex if total weights of fake tracks take up the largest proportion.

The proportion for each type of vertices is shown in Figure 5, which shows that the main reason for the shrinking of matched vertices as μ goes up is the increasing number of merged vertices. For the intuitive purpose, visualization on the match of vertices is shown in Figure **??**. Figure 7 further plots the normalized distribution of the number of tracks, the sum of p_T, the fit χ^2, and the degrees of freedom for different types of vertices. As can be seen, merged vertices tend to have a larger value in those variables. The distribution of z position resolution of vertices (simply computed as the distance between a vertex and the closest truth interaction position) is also shown in Figure 8, revealing that large z resolution mostly corresponds to fake vertices, which makes sense because a fake vertex has no relevant truth interaction at all.

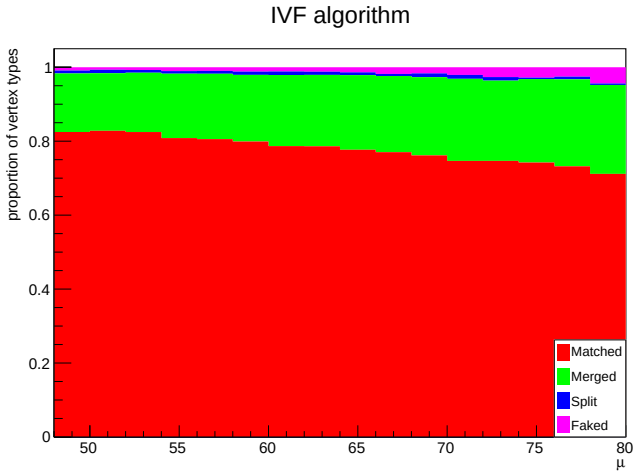

Figure 5: The proportions of each of the four types (i.e., matched, merged, split, and fake ones) as a function of the number of interactions μ.

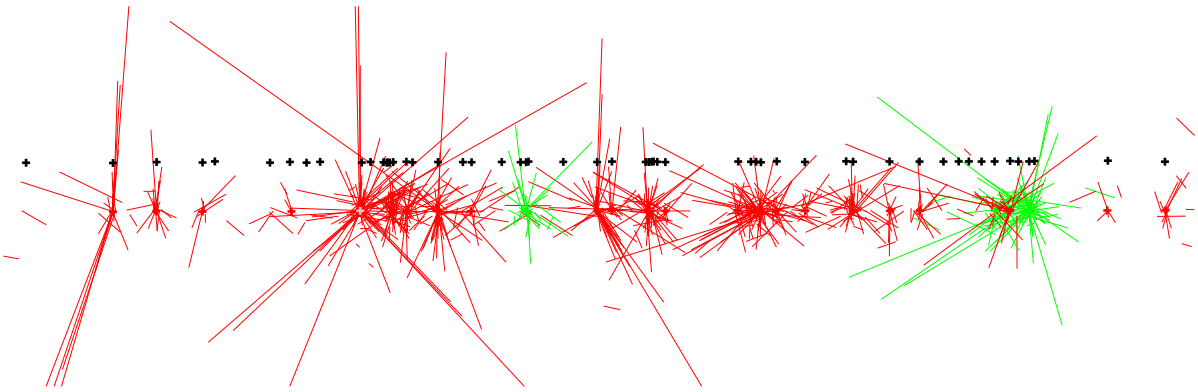

Figure 6: A 3D visualization showing the match result for vertices (colored crosses), tracks (colored lines), and truth interactions (black crosses). The color assignment to vertices and tracks follows the same rule as labeled in Figure 5.

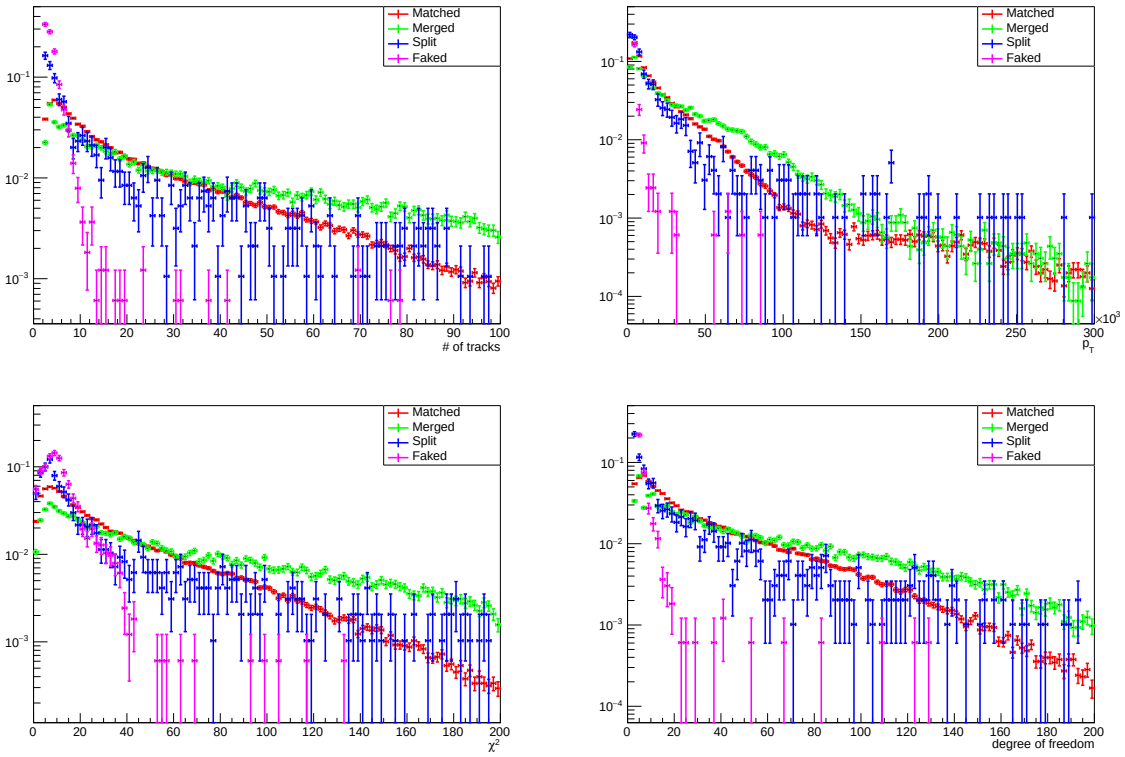

Figure 7: Normalized distribution of the average number of associated tracks (top left), sum of transverse momentum p_T of tracks (top right), χ^2 for the vertex (bottom left), and degree of freedom (bottom right) for four specific type of vertices: matched, merged, split, and fake ones.

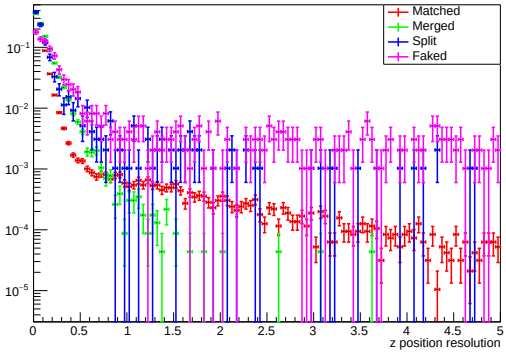

Figure 8: Normalized distribution of z position resolution for matched, merged, split, and fake vertices.

3.2 Parameter dependency

To present an overall test on parameter dependency, we introduce four evaluation indicators: matched efficiency, merged rate, split rate, and fake rate, which refer to the number of matched, merged, split, and fake vertices over the number of truth interactions . As an example, in Figure 9 we plot the four variable as a function of μ in the original IVF case. Please noted that for each data point, the medium line and the error bar represent the average and RMS of events confined in a specific μ range. The number of vertices is also used as a parameter, which is displayed in Figure 2.

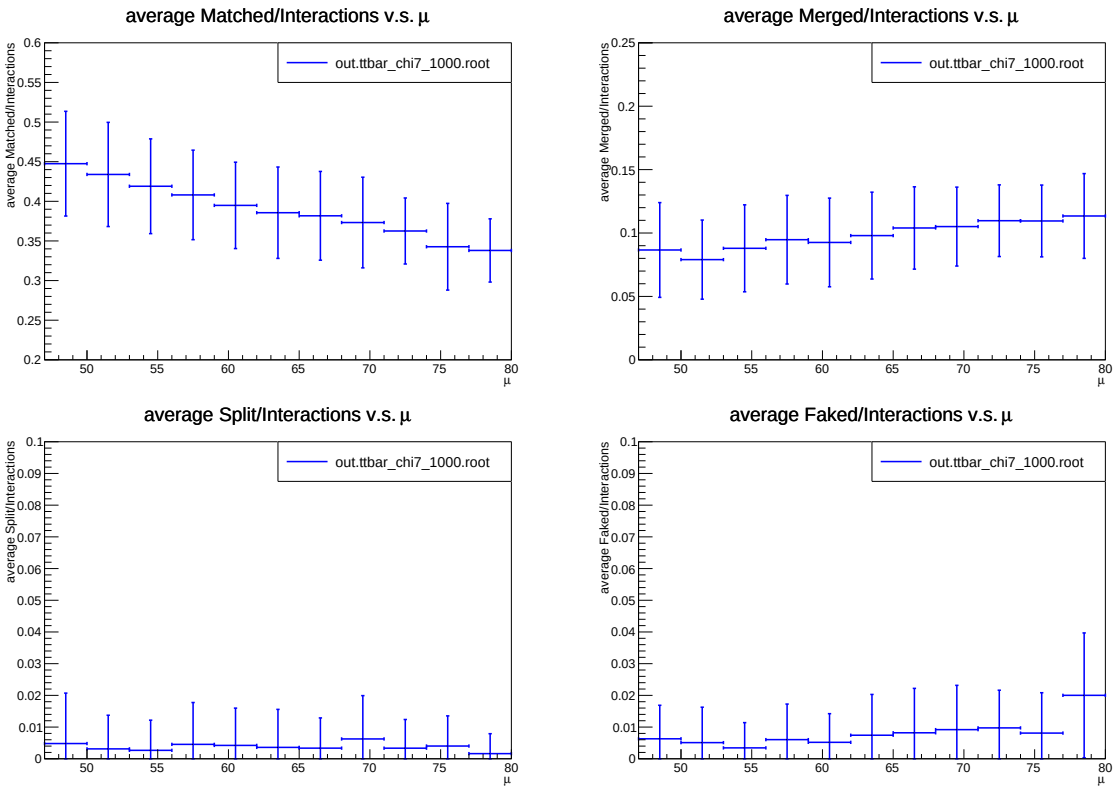

Figure 9: The matched vertex (top left), merged rate (top right), split rate (bottom left) and fake rate (bottom right) as a function of the number of interactions μ in the original IVF case. For each data point, the medium line and the error bar represent the average and RMS of events in the confined μ range.

3.2.1 Compatibility cut dependency

The compatibility cut χ_{cut} is introduced in step 3 of IVF algorithm (Sec. 2.1), which is referred to as a threshold on the compatibility between an input track and the fitted vertex that determines whether the track should be removed and returned back to the seed pool at the end of a loop for a vertex fit. Basically, higher the χ_{cut}, more associate tracks we have for a fitted vertex. Figure 10 shows the reconstruction efficiency versus μ as well as the distribution of our tested parameters for $\chi_{\text{cut}} = 3, 4, ..., 12$. As we can see, the vertex reconstruction efficiency drops significantly as the compatibility cut goes up, which means a looser criterion on the track selection. Since more tracks will enter a single fitted vertex as χ_{cut} goes up, fewer reconstructed vertices will be produced. Also, our indicator variables including matched efficiency, merged rate, split rate, and fake rate all decline accordingly, making it hard to decide the optimal value of χ_{cut}. The 2D plots for split/fake rate versus matched efficiency are thus shown in Figure 11, where the dashed red line represents the average of the related variable over all events. Results indicate that the split rate and fake rate are quite close to zero when $\chi_{\text{cut}} \gtrsim 7$, while the matched efficiency continues to drop as χ_{cut} goes up. Therefore, We can safely choose

$\chi_{\text{cut}} = 7$ as the optimum.

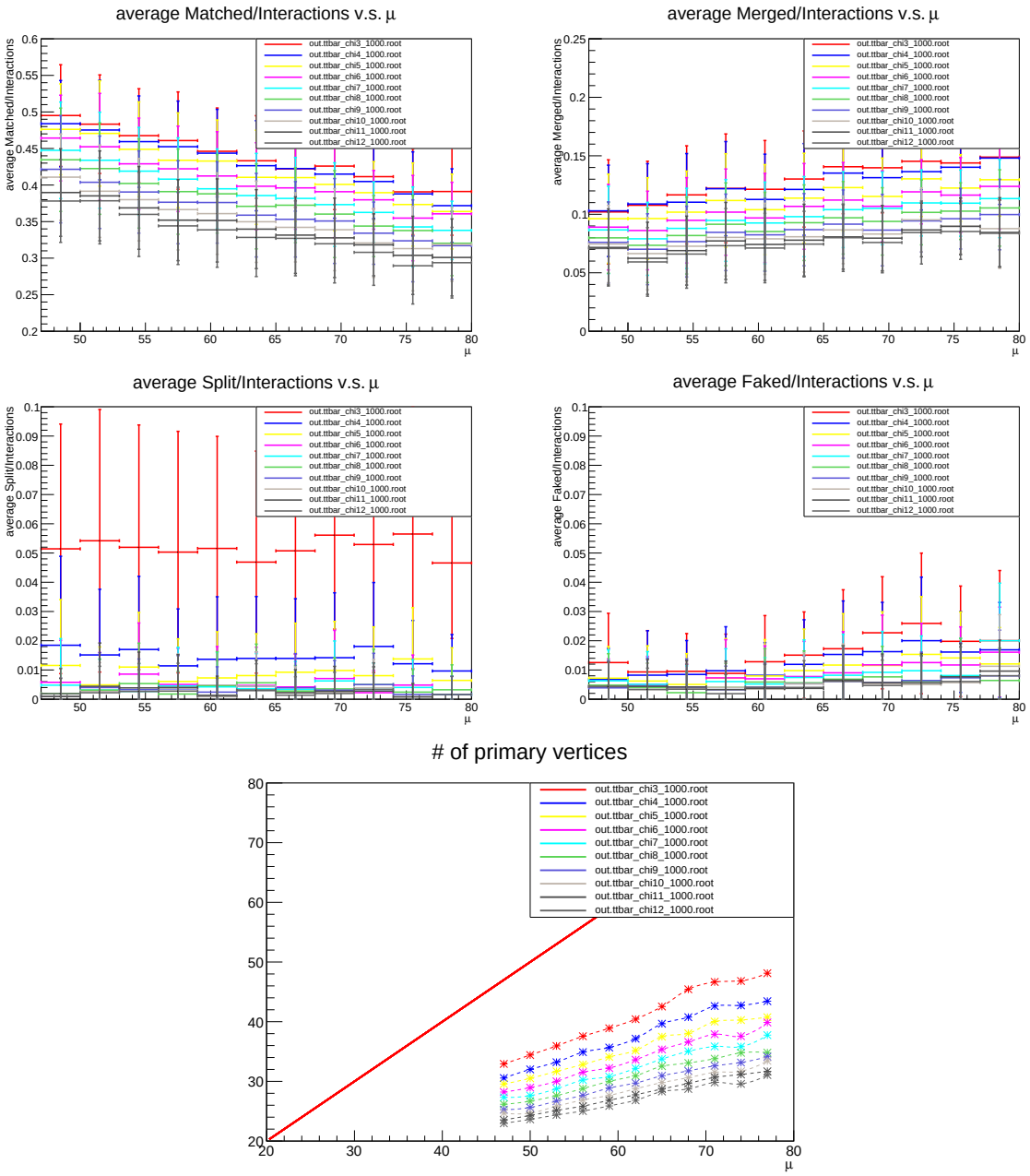

Figure 10: The matched vertex (top left), merged rate (top right), split rate (medium left), fake rate (medium right) and the number of reconstructed vertices (bottom) as a function of the number of interactions μ for $\chi_{\text{cut}} = 3, 4, ..., 12$ in IVF case.

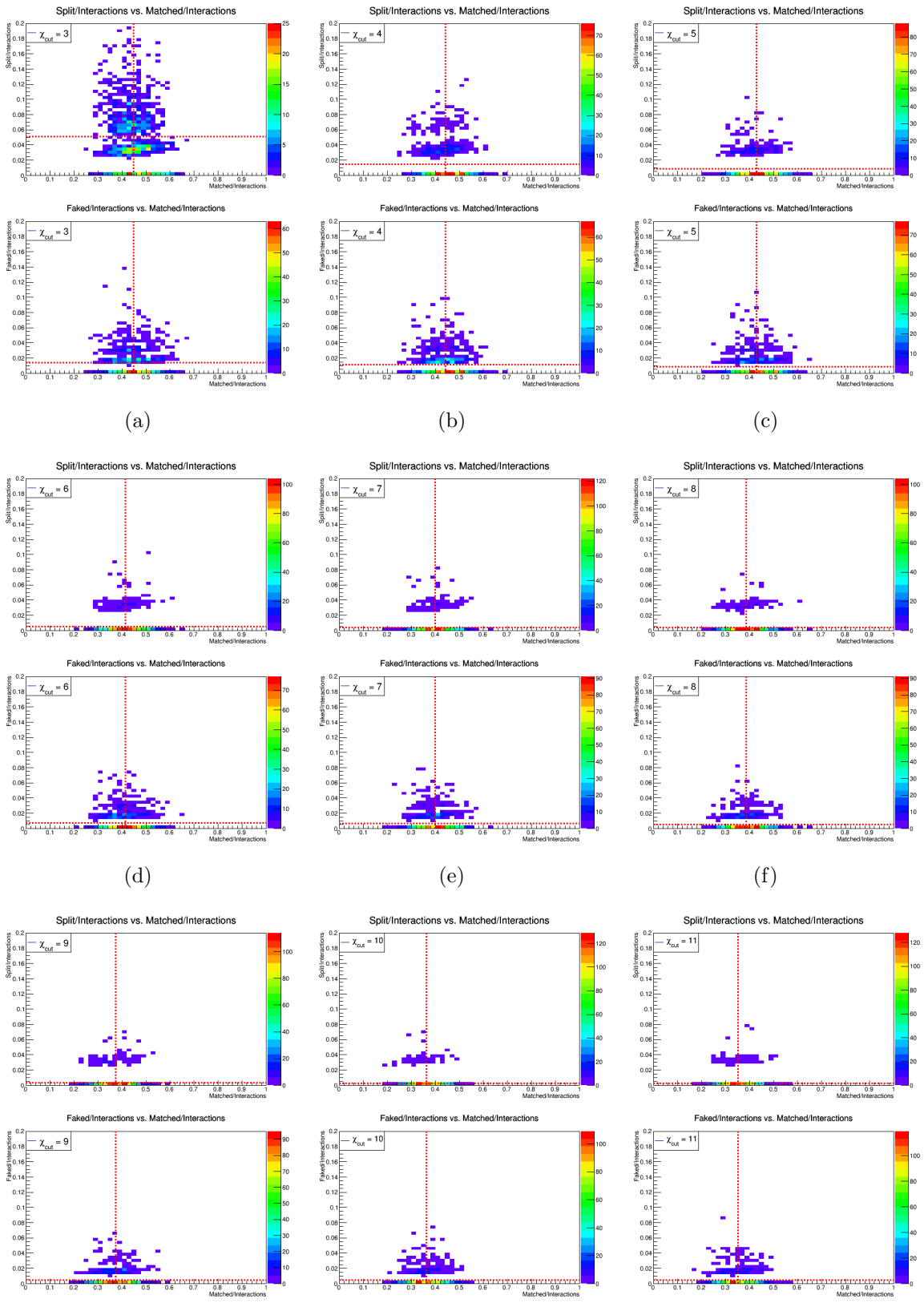

Figure 11: (a)–(i) each represents the 2D plot of split (above) or fake (below) rate versus matched efficiency for $\chi_{\text{cut}} = 3, 4, ..., 11$ in IVF case. Noted that the dashed red line indicates the average of the related variable over all events.

3.2.2 The modified IVF algorithm

Further attempts include a small revision of the original IVF algorithm. As described in step 1, both the candidate tracks for the seeding and fitting stem from the seed pool. However, since there is no extra requirement for a track which is currently in use for feeding may also be a candidate for the fitting procedure, there can be multiple choices to decide two track containers independently. Here we call the container used for seeding a seed pool and the one used for fitting as the perigee list. Thus the χ_{cut} can be applied respectively to decide the seed pool and perigee list in the next vertex finding and fitting procedure, which is referred to as $\chi_{\text{cut,seed}}$ and $\chi_{\text{cut,perigee}}$.
A pre-analysis is applied to evaluate the possible effect on such revision. On the one hand, if $\chi_{\text{cut,seed}}$ decreases with $\chi_{\text{cut,perigee}}$ fixed, more tracks—even already allocated tracks—may enter the seed pool in the next vertex finding iteration, contributing to the position of the next vertex. Since merged vertex occasionally appears, leaving the close-by interaction unreconstructed, we hope in that way the opportunity to reconstruct both nearby interactions can increase, reducing the high merge rate. On the other hand, if $\chi_{\text{cut,seed}}$ increases with $\chi_{\text{cut,perigee}}$ fixed, it means only part of unused tracks return to the seed pool in the following loop, avoiding the outliers (tracks that probably have large uncertainty that may do harm in vertex finding) from entering the seed pool.
According to the pre-analysis, we test the performance of revised IVF with $\chi_{\text{cut,seed}} = 3$ or 12, as well as the original case $\chi_{\text{cut,seed}} = 7$, with $\chi_{\text{cut,perigee}}$ unchanged. The plots on matched efficiency and merged/split/fake rate are shown in Figure 12. The result disappointingly shows that a decline in $\chi_{\text{cut,seed}}$ significantly brings up the split rate. Although it makes the close-by interactions easier to be identified, the uncertainty in tracks tends to be more likely to confuse the algorithm, raising the probability to mis-reconstruct two vertices which should have come from one interaction. Therefore, one needs to strike a balance between controlling the split rate as well as making it possible to identify nearby interactions. Besides, in the case we increase $\chi_{\text{cut,seed}}$, all evaluation indicators tend to decline, which attributes to the decrease in the total number of vertices being reconstructed. The above result refers that there are dozens of variables, with complicated relations, that may have an impact on the final performance, thus our initial intention to improve the algorithm by revising one parameter is untenable. Therefore, a global evaluation of the result is suggested with all variables free floating to optimize the parameter.
As a simple attempt, we introduce a weight function that compromise all evaluation indicators to optimize $\chi_{\text{cut,seed}}$ and $\chi_{\text{cut,seed}}$. A linear function is easily constructed as

$$\omega = \sum_{i=\text{all type}} w_i r_i, \tag{2}$$

where r_i represent the matched efficiency (i = matched) and the other three unmatched rate (i = merged/split/fake). We present two sets of weight vector here and show how they influence the optimal parameter set. For $\boldsymbol{w} = (0.5, -1, -1, -1)$ and $\boldsymbol{w} = (1, -1, -1, -1)$ the top 10 rank for the parameter sets are listed in Table 1 (a) and (b), respectively.

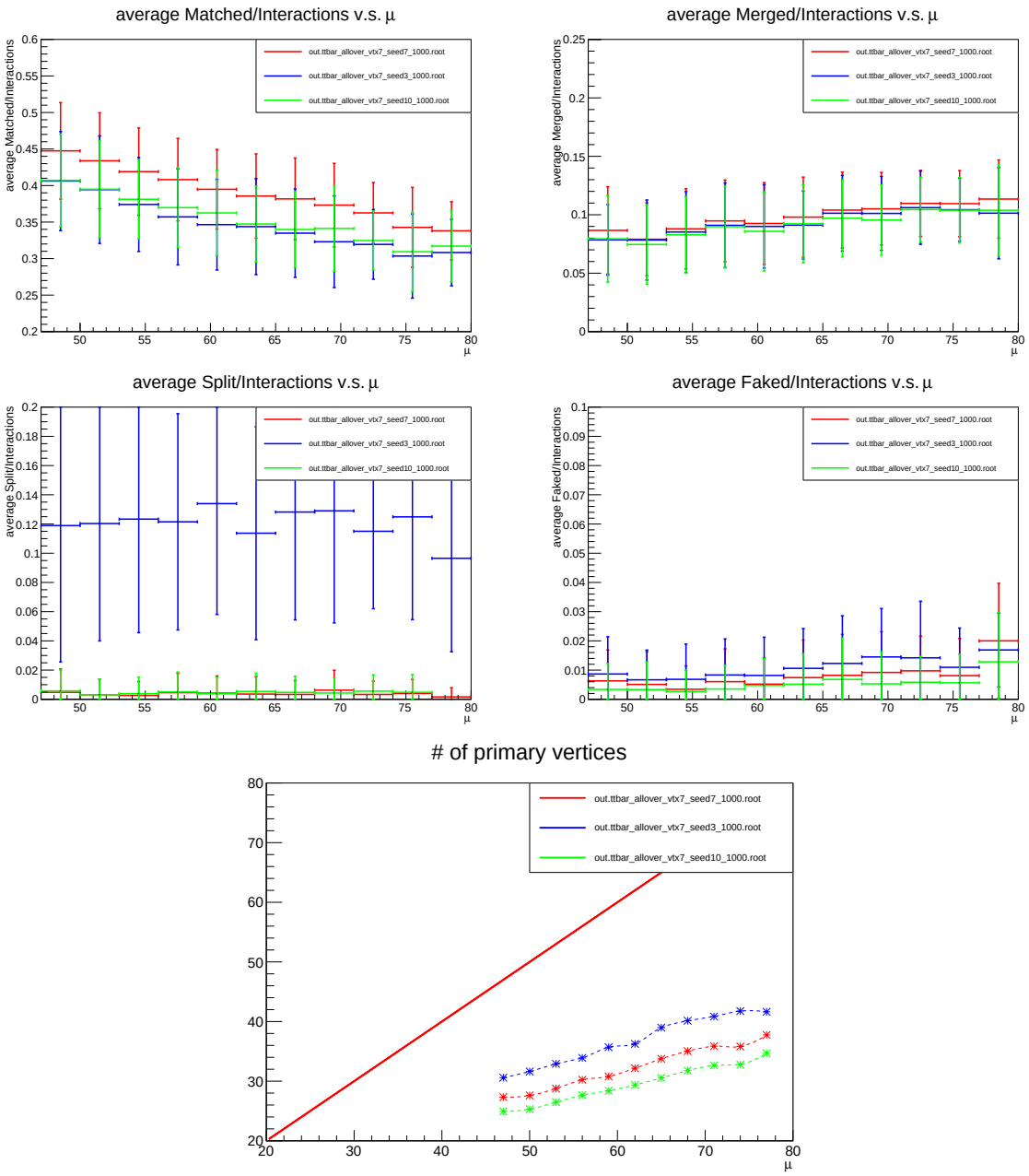

Figure 12: The matched vertex (top left), merged rate (top right), split rate (medium left), fake rate (medium right) and the number of reconstructed vertices (bottom) as a function of the number of interactions μ for $\chi_{\text{cut,seed}} = 7$ (red), 3 (blue) and 10 (green) with $\chi_{\text{cut,perigee}} = 7$ fixed in revised IVF case.

As expected, the optimal choice is highly dependent on the weight function. However, there is no absolute criterion on weight function which guarantees a "best" result, so it may simply depend on one's preference. Since the original choice of parameter both rank high in the lists, it suggests that $\chi_{\text{cut,seed}} = \chi_{\text{cut,perigee}} = 7$ can be a good choice.

Table 1: The top 10 optimal parameter set for two example weight vector $\boldsymbol{w}$.

(a) weight vectors $\boldsymbol{w} = (0.5, -1, -1, -1)$

ω	$\chi_{\text{cut,seed}}$	$\chi_{\text{cut,perigee}}$
0.0950	8	8
0.0946	9	9
0.0939	10	10
0.0937	7	7
0.0925	6	6
0.0919	9	10
0.0913	7	8
0.0906	10	11
0.0904	8	9
0.0896	10	9

(b) weight vectors $\boldsymbol{w} = (1, -1, -1, -1)$

ω	$\chi_{\text{cut,seed}}$	$\chi_{\text{cut,perigee}}$
0.3015	5	5
0.2994	6	6
0.2954	4	4
0.2931	7	7
0.2931	5	6
0.2897	6	7
0.2889	4	5
0.2879	8	8
0.2847	7	8
0.2813	5	7

3.3 Comparison between three algorithms

There are also other alternative algorithms that are designed but not used, two of which are the adaptive multi-vertex finder algorithm (AMVF) and Gaussian iterative vertex finder algorithm (GIVF). A comparison between the three algorithms is applied, indicating that GIVF is slightly better than IVF.

First, we briefly introduce the algorithms. GIVF is all the same with IVF algorithm except that in step 1 (Sec. 2.1), it adopts a different vertex finding algorithm named `TrackDensity`. The finding algorithm searches for a global maximum of the track density since a vertex is most likely to appear near branches of tracks. The first and second derivatives of track density at the current point is calculated to find the maximum. AMVF adopts an entirely different procedure in vertex fitting, while it still utilizes the `ZScan` vertex finding method which is the same as IVF. Unlike IVF where vertices are fitted one after another, AMVF attempt to fit all the vertices and optimize the track assignment to vertices "globally". In this way, all the vertices and allocation of tracks are finalized after the last vertex finding loop.

The performances of the three algorithms are shown in Figure 13, applied the same test. The AMVF algorithm gives a relatively bad performance among the three, with a significant increase in merged, split and fake rate. Besides, GIVF tends to win out IVF a little bit. The superiority is that it produces more vertices, resulting in an increase in matched efficiency while the unmatched rate keeps unchanged. The mechanism on why GIVF is capable of producing more matched vertex can call for subsequent research.

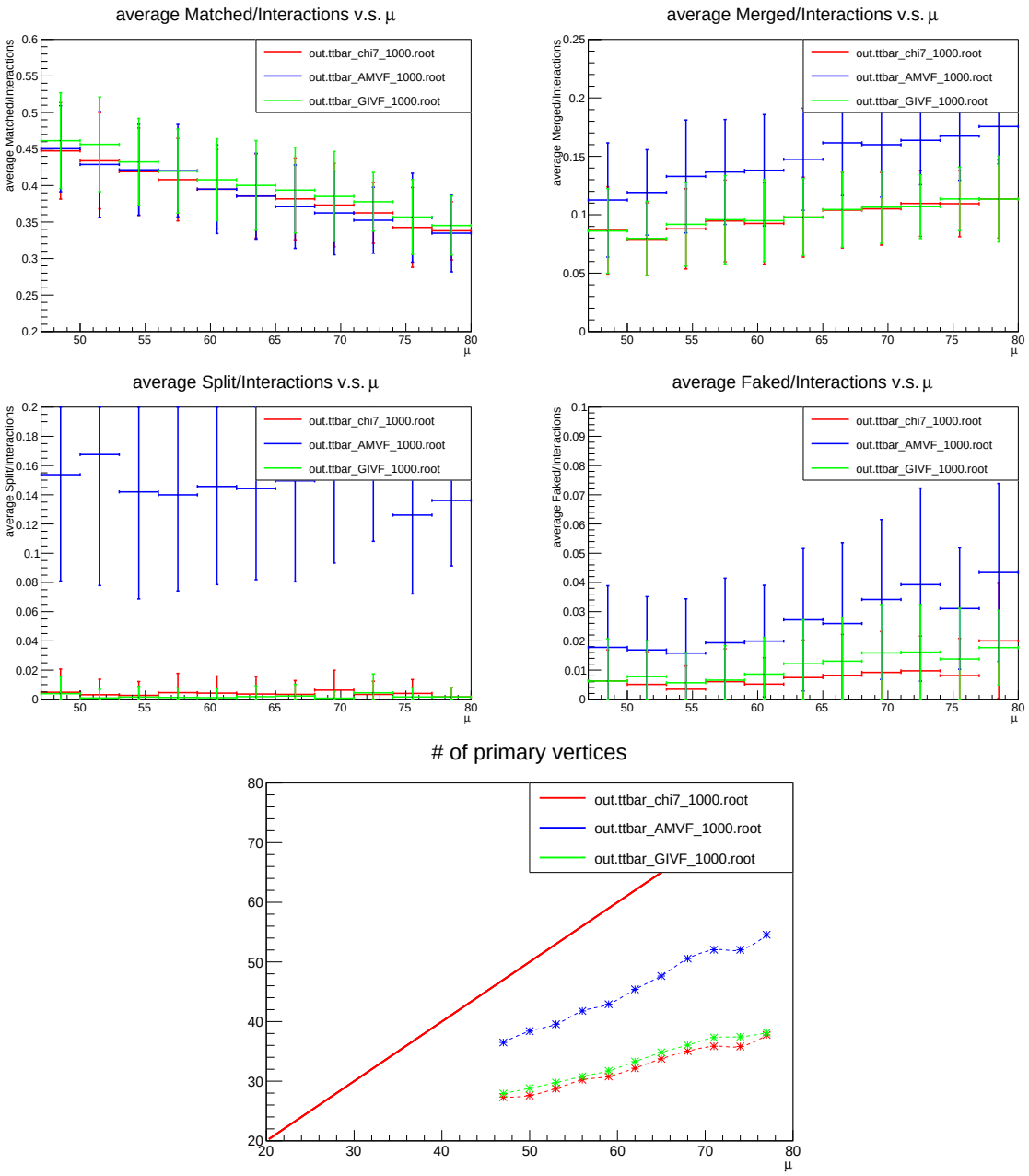

Figure 13: The matched vertex (above left), merged rate (above right), split rate (medium left), fake rate (medium right) and the number of reconstructed vertices (bottom) as a function of the number of interactions μ in the case of unrevised IVF algorithm (red), AMVF algorithm (blue) and GIVF algorithm (green).

4 Applications on real data

In this section, we apply the algorithms to real data in ATLAS experiment. The performances of the original IVF, AMVF and GIVF algorithm are investigated. Two sets of data for lower and higher region of μ are used respectively to carry out the test. In real experiment the average number of interactions μ is obtained from

$$\mu = \frac{L\sigma_{\text{inel}}}{n_{\text{b}} f_{\text{r}}}, \tag{3}$$

where σ_{inel} denotes the inelastic cross section, n_{b} marks the interaction bunches per beam and f_r is the revolution frequency [1]. The number of reconstructed vertices as a function of μ is thus plotted in Figure 14.

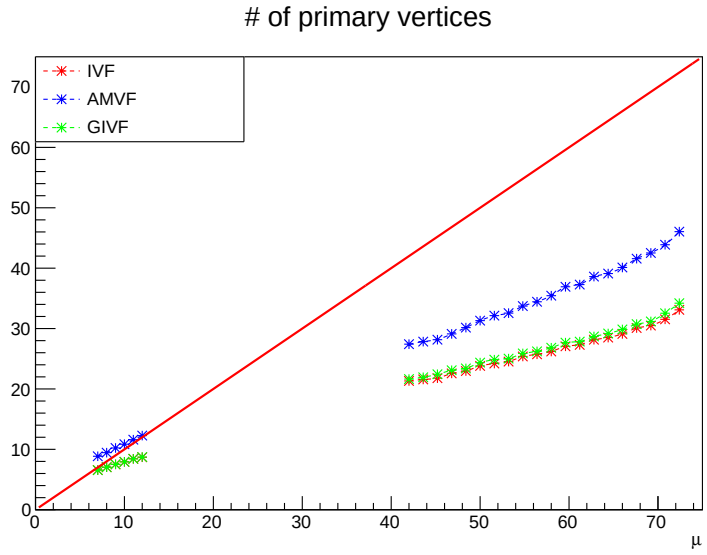

Figure 14: The average number of reconstructed vertices as a function of μ for two data sets with different range of μ. Noted that μ is calculated from the beam luminosity. The dashed line and spots give the result of IVF (red), AMVF (blue) and GIVF (green) algorithm, while the solid red line indicates 100% reconstruction.

As can be seen, for each algorithm, the number of vertices show a similar pattern as Figure 13. However, for lower μ, AMVF produces even more vertices than interactions. A single-event picture for tracks and vertices from AMVF in Figure 15 indicates that AMVF algorithm tends to split one interaction to many vertices, even for lower-μ data. While the increase of merged rate is mainly due to the increasing close-by interactions in higher-μ case, split vertices, brought by the complexity of a single interaction and the large uncertainty of related tracks, can still be high for lower-μ data. This brings up the reconstruction efficiency even above 100%. Therefore, reducing the split rate is an essential task for any algorithm, and that is why a loose criterion $\chi_{\text{cut}} = 7$ is chosen

in IVF case [1]. Furthermore, GIVF can produce slightly more vertices than IVF, and hopefully, it can raise the matched efficiency by a little.

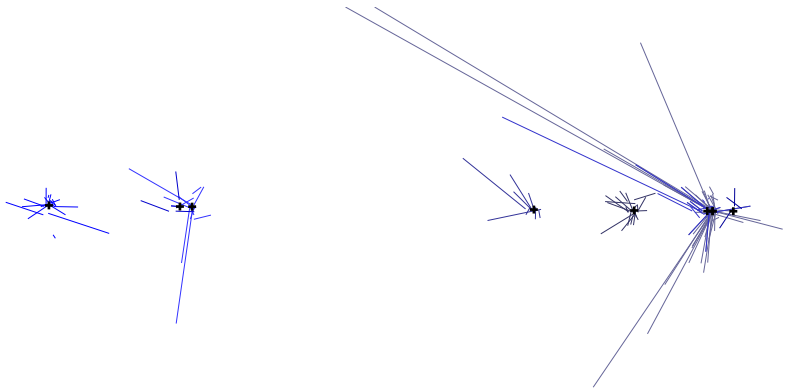

Figure 15: The visualization for AMVF performance in lower region of μ.

5 Conclusion

Primary vertex reconstruction is a main focus in high energy physics experiments. It utilizes reconstructed tracks as input to fit the position of vertices and allocated tracks to each of them. For high pile-up events in HL-LHC, the number of interactions can be up to 200, making vertex reconstruction an important task that ensures the following physical analysis to carry on.

Some basic properties of the iterative vertex finder algorithm (IVF)—adopted by ATLAS during the LHC Run 1 data-taking—is analyzed. The result shows that the reconstruction efficiency degrades as the average number of interactions (donated as μ) goes up, and more tracks will be associated to a vertex since the merging of vertices appears more often in the higher region of μ. A matching algorithm is further presented, classifying all vertices as matched, merged, split and fake ones according to their relations with truth interactions. Four indicators: the matched efficiency, the merged/split/fake rate are introduced in the hope of assessing and optimizing the algorithm. Based on the evaluation indicators, a particular analysis on the dependency of the compatibility cut χ_{cut}, a threshold measuring the compatibility of tracks linked to a vertex, shows that the original case with $\chi_{\text{cut}} = 7$ is a good choice, which gives the lowest split/fake rate as well as an acceptable matched efficiency. An attempt to revise IVF algorithm is also implemented, which defines independent parameters as thresholds on the seed pool and perigee list, i.e., two important track containers during the vertex finding and fitting process. A linear weight function is thus introduced to compromise all indicators to find the optimal choice for the two parameters. Results show that the best option is heavily dependent on the weight function, and no absolute criteria exist. Still, the original IVF can be a rather good choice. A brief comparison with another two algorithms—adaptive multi-vertex finder algorithm (AMVF) and Gaussian iterative vertex finder algorithm

(GIVF)—is implemented. We find that GIVF is slightly better than IVF, which may call for follow-up studies.
The three algorithms are finally applied on real data sets, the properties of reconstructed vertices are verified as compared to MC data sets.

6 Acknowledgment

The author gratefully acknowledges the guidance and invaluable assistance from the supervisor Dr. Federico Meloni, who often have inspiring ideas to implement analysis in various way. The author also gives special thanks to Fionn Bishop and Namgyong Jeong for their technical help, and Yebo Chen, Ruijia Yang for inspiring discussion. This work is supported by DESY ATLAS group and DESY Summer Student Programme 2018.

References

[1] The ATLAS Collaboration. Reconstruction of primary vertices at the ATLAS experiment in Run 1 proton-proton collisions at the LHC. *The European Physical Journal C*, 77(5), 332.