Journal of Physics: Conference Series **664** (2015) 072041 doi:10.1088/1742-6596/664/7/072041

ATLAS strategy for primary vertex reconstruction during Run-2 of the LHC

G Borissov[1], D Casper[2], K Grimm[1], S Pagan Griso[3], L Egholm Pedersen[4], K Prokofiev[5], M Rudolph[6] and A Wharton[1] on behalf of the ATLAS Collaboration

[1] Lancaster University, UK
[2] University of California, Irvine, USA
[3] Lawrence Berkeley National Laboratory and University of California, Berkeley, USA
[4] Niels Bohr Institute (NBI), University of Copenhagen, Denmark
[5] Hong Kong University of Science and Technology
[6] University of Toronto, Canada

E-mail: matthew.scott.rudolph@cern.ch

Abstract. The reconstruction of vertices corresponding to proton–proton collisions in ATLAS is an essential element of event reconstruction used in many performance studies and physics analyses. During Run-1 of the LHC, ATLAS has employed an iterative approach to vertex finding. In order to improve the flexibility of the algorithm and ensure continued performance for very high numbers of simultaneous collisions in Run-2 of the LHC and beyond, a new approach to seeding vertex finding has been developed inspired by image reconstruction techniques. This note provides a brief outline of how reconstructed tracks are used to create an image of likely vertex collisions in an event, describes the implementation in the ATLAS software, and presents some preliminary results of the performance of the algorithm in simulation approximating early Run-2 conditions.

1. Introduction

The reconstruction of primary vertices from individual proton–proton collisions is essential for physics analysis with the ATLAS detector. An accurate reconstruction of the number and positions of interaction vertices is needed by algorithms that separate the effects of additional collisions ("pile-up") from the measurement of the properties of the particles generated by the hard-scattering collision of interest. During Run-1 of the LHC, ATLAS employed an iterative approach to vertex finding and fitting. This algorithm performed well for up to 40 inelastic collisions in one LHC bunch crossing. To quantify the amount of pile-up, the variable "μ" is used, equalling the average number of interactions per bunch crossing. During Run-2 and beyond, the amount of pile-up is expected to become even higher. A new approach inspired by imaging algorithms, as suggested elsewhere [1], is in development for future running. The goal is to perform better than the iterative algorithm at very high μ [2]. This new algorithm attempts to simultaneously identify all likely vertex locations in one LHC bunch crossing by using all tracks as input to a three-dimensional imaging algorithm (similar to those used in many medical imaging applications [3]). These locations are then used as seeds to the vertex finding and fitting process. This note presents a brief description of this algorithm both in terms of the

procedure and the software implementation. Some preliminary examples of its performance are also presented.

The results shown are obtained from simulations of the ATLAS detector [4], which is a large multi-purpose particle detector at the LHC. Monte Carlo generated collision events are processed through a GEANT4 [5] simulation of the ATLAS detector [6]. Minimum bias events for pile-up interactions are generated using the PYTHIA8 [7] Monte Carlo generator with the A2 tune [8]. To examine the vertex algorithm performance for collisions with a large number of tracks, top–antitop pair production events generated with POWHEG [9, 10, 11, 12] interfaced to PYTHIA6 [13] have been used. All interactions are simulated at an energy of 13 TeV under conditions expected in early LHC Run-2 during 2015. The inputs to the vertex reconstruction algorithms are reconstructed tracks in the Inner Detector, with a pseudorapidity coverage of $|\eta| < 2.5$.[1]

In general, the process of vertex finding and fitting in ATLAS is divided into three major steps – seeding, track assignment, and fitting. The procedure and performance of the iterative algorithm used in Run-1 are described in detail in Ref. [14]; a rough outline of the procedure is as follows:

(i) The impact parameters z_0 of all tracks with respect to the centre of the beam spot are used to produce a single seed at the location of the estimated mode in z [15], using an iterative method to find the most likely value.

(ii) Tracks compatible with the seed are grouped together for fitting.

(iii) The adaptive vertex fitting algorithm [16] is used to estimate the position and uncertainty of the vertex.

(iv) Incompatible tracks that are not used in a previous vertex are used to repeat the procedure starting from the creation of a new seed.

The iterative algorithm is primarily tuned to avoid splitting tracks from a single interaction vertex into multiple reconstructed vertices. However, as the amount of pile-up increases, merging two interactions by combining the reconstructed tracks from both becomes more common. Since seeds are produced one at a time, and the tracks from the two interactions may all be close enough to be compatible with a common vertex, the first seed created can result in a vertex merging the tracks from the two interactions together. All these tracks are then removed from consideration, so it is not possible to find a second seed in order to reconstruct a second vertex for these two interactions. As pile-up increases, this results in a negative quadratic dependence of the number of reconstructed vertices on μ. The primary motivation for the development of the imaging algorithm has been to implement a seeding strategy that will identify all probable vertex locations in a single pass. By producing two seeds for cases in which two close-by collisions took place before proceeding to track assignment and fitting, merging can be avoided by assigning tracks to both seeds, although the rate of split vertices may be higher.

2. Imaging algorithm description

In the imaging algorithm all seeds are output in a single step. The procedure is as follows:

(i) A three-dimensional binned histogram to be filled by the image is created containing the configurable volume in which vertex finding will be done. In these proceedings, the x and y dimensions of the box are 4 mm long and the the z dimension is 400 mm.

[1] The ATLAS experiment uses a right-handed coordinate system with its origin at the nominal interaction point (IP) in the centre of the detector and the z-axis along the beam direction. The x-axis points from the IP to the centre of the LHC ring, and the y-axis points upward. Cylindrical coordinates (r, ϕ) are used in the transverse (x, y) plane, ϕ being the azimuthal angle around the beam direction. The pseudorapidity is defined in terms of the polar angle θ as $\eta = -\ln\tan(\theta/2)$

(ii) Helical track trajectories are linearised and back-projected into the histogram using a voxel ray-tracing algorithm [17]; the histogram content in each bin crossed by a track is incremented by the path length of the linearised track in that bin. An example back-projection is shown in Fig. 1a.

(iii) The back-projection is Fourier transformed into frequency space using the FFTW3 [18] library.

(iv) A filter composed of two major parts is multiplied with the frequency space histogram. The image is reconstructed by applying a filter derived from the point spread function [3], adapted for the angular acceptance of the ATLAS tracking detector. It is essentially the inverse of the Fourier transform of the detector angular acceptance. In addition, a four-term Blackman-Harris window filter [19], which smoothly scales down higher frequencies up to a configurable cutoff frequency in each of the x, y, and z directions, is used to lessen the effect of high frequency variations. These variations can result, for example, in additional image peaks along outgoing track paths.

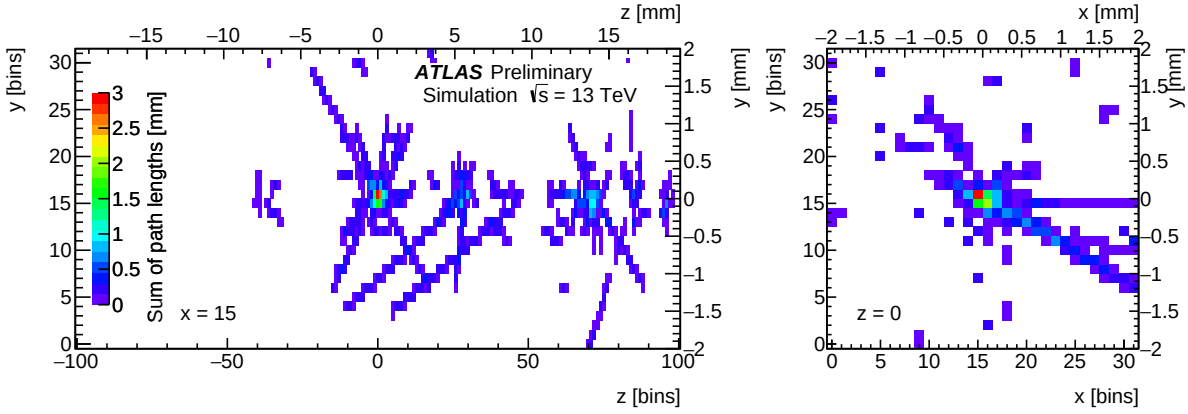
(v) The filtered frequency space image is then back transformed to position space, giving a final image as shown in Fig. 1b.

(vi) The resulting image is then passed to a separate clustering algorithm where all seeds are identified from peaks in the image.

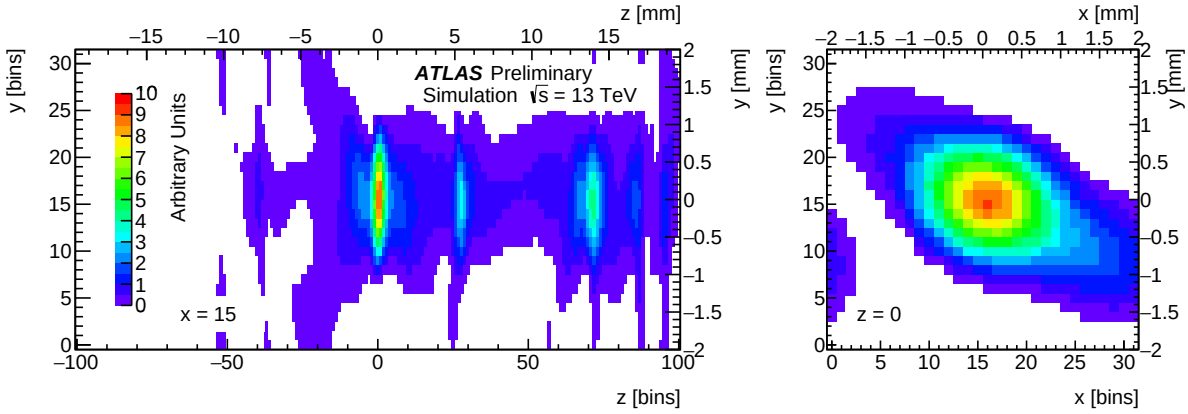
Two example configurations of the imaging seeding algorithm are used to compare performance with the iterative algorithm. The only difference between the two is in the choice of the number of bins to be used for the image histogram in the z-direction; for simplicity the other configurable elements were not re-tuned to give similar performance on a particular metric. In one, 1024 bins are used to cover a 400 mm range, while the other uses 2048 bins for the same range. Each uses 16 bins in the x and y directions, which was found to be an acceptable trade-off in terms of processing speed and performance. These configurations are chosen to provide example points where different minimum z separations between image peaks are possible. Figure 1 uses a different binning for purposes of illustration. But many more configurations can be made with different numbers of bins, ranges, and window filters. The interplay with the algorithm used to identify seeds is key here.

Development of a final clustering algorithm to produce vertex seeds is still ongoing; the results given in this note come from a simple, fast one-dimensional projection algorithm. Because of the modular nature of the seeding procedure, it is straightforward to substitute a more sophisticated image processing algorithm in the future to improve performance. The projection algorithm was chosen for first results because the magnitude of the track position uncertainties is large compared to the beam spot size and it is difficult to separate vertices in the x and y directions in the image. To avoid fluctuations at large values of x and y away from the expected beam-spot location, only bins within a 1σ box in xy are used in the projection. First, the standard deviation in x and y is calculated using the full image – its width is driven by the smearing of the method rather than the width of the true vertex distribution. Then this region is used to project the three-dimensional vertex image onto the z-axis; a view of the same event as in Fig. 1 (with an expanded z-range) is pictured in Fig. 2. To identify seeds in the projection, all local maxima above a configurable threshold are found. The lower of two peaks is eliminated if the minimum value between it and the other peak is greater than 90% of its value. The resulting set of vertex seeds is then used to perform the vertex finding and fitting.

In the track assignment step, each track is assigned to the closest seed to its trajectory. This is possible because the imaging algorithm produces all seeds simultaneously. This is the main difference with respect to the vertex finding for the iterative algorithm, which produces them one-by-one. Each resulting group of tracks is then fit with the same adaptive fitting algorithm for both imaging and iterative seeding. So if, for example, two close-by interactions produce two



(a) Track back-projection



(b) Reconstructed image

Figure 1: Illustration of the image reconstruction of part of a single simulated $t\bar{t}$ event using POWHEG and PYTHIA6, including pile-up, centred on the bin with the largest content in the reconstructed 3D histogram image. Slices are made through this peak in the zy and xy planes (at $x = 15$ and $z = 0$), and the axes are labeled both with the actual bin numbers used in the algorithm and the corresponding spatial extents. The results of the track back-projection step are shown in Fig. 1a; the bin content represents the sum of track path lengths in each bin. In Fig. 1b the full reconstructed image after Fourier transformation into frequency space, filtering, and back transformation is shown; this histogram is used as input to image processing algorithms to identify likely vertex locations that appear as peaks in the image. In addition to the interaction resulting in the highest peak (centred at $z = 0$), several other interactions are visible in the reconstructed image. From Ref. [2].

seeds, both are likely to have some tracks assigned to them and two final reconstructed vertices can be found.

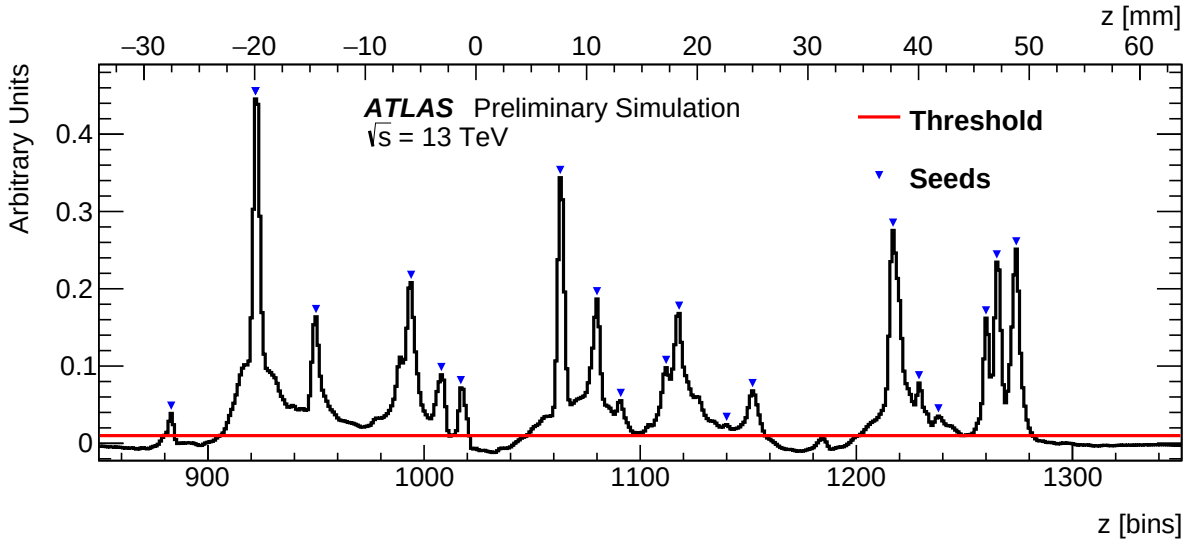


Figure 2: Image of a portion of a simulated event (the same $t\bar{t}$ event imaged in Fig. 1), after performing the full image reconstruction including transforms and filtering and then projecting onto the z-direction. The z-bin numbers have been changed from relative to absolute numbering; the peak at $z = 0$ in Fig. 1 corresponds to the one at $z \approx 920$. The seeds identified in the projection are indicated. From Ref. [2].

3. Algorithm structure and implementation

The software implementation of the imaging vertex finding algorithm is designed to be highly modular and configurable, relying on the features of the ATLAS Athena framework [20]. Each of the major steps in the vertexing algorithm (image creation, seed finding, track assignment, and vertex fitting) is performed by a separate tool implementing a defined interface specific to that step. Different versions of these tools can be developed, and the user can switch between them at run-time by configuring the tools used by the over-arching vertexing algorithm. For example, different strategies for identifying seeds in the reconstructed image can be developed in parallel and swapped in the configuration. The tools also define configurable properties, for example the number of bins in the image or cutoff frequencies in the filtering, allowing different settings to be configured at run time. In this section, an outline of these tools is provided.

The imaging seeding starts with the imaging itself – track back-projection, Fourier transforms, and filtering. A tool takes as input a collection of tracks from which to make the image, implements the algorithm as described above, and populates an array representing the 3D image histogram. Because of its large size, only one array is kept and the space is not re-allocated event by event. Accessors and helper functions are given to the user to facilitate image processing. In principle, this tool could be used in other algorithms besides the seeding one described here, such as beam-spot finding. At run-time the size of the volume to be used, the number of bins used, and the cutoff frequencies in the filter can all be configured.

For vertex seeding, the image is used as input to a cluster finding tool that will produce the actual vertex seeds which are output as a list of positions. The simple projection algorithm described above is one example of such a tool. In the future, new, more sophisticated algorithms

Table 1: Comparison of algorithm performance for simulated events with a single interaction. The efficiency to reconstruct any number of vertices for a single interaction, and the rate at which single interactions are split to form multiple vertices are shown for the iterative algorithm and for two example configurations of the imaging algorithm with 1024 or 2048 z-bins in the image. From Ref. [2]. .

	Minimum bias		$t\bar{t}$	
	Efficiency [%]	Split rate [%]	Efficiency [%]	Split rate [%]
Iterative	81.8	0.04	100.0	0.38
Imaging 1024 z-bins	81.5	0.2	100.0	0.96
Imaging 2048 z-bins	80.3	4.3	100.0	9.5

could be used to improve the performance of the seeding algorithm. This is straightforward to do because of the modular nature of the algorithm. New code can be created with the same interface as the current tool, and swapped in during run-time configuration. The track assignment is then performed using the full list of seeds by another tool. Currently this is done simply by assigning each track to be used to the closest seed among the full list. The vertex fit combining the tracks assigned to each seed is also performed by a separate configurable tool.

4. Performance studies

A comparison of the performance of the imaging algorithm with the iterative one for single interactions (no additional pile-up in the event) is found in Table 1. The performance has been tested with PYTHIA8 minimum bias and POWHEG+PYTHIA6 $t\bar{t}$. For single interactions, the most important metrics are the overall efficiency (was a vertex reconstructed at all for an event), and the split rate (in what fraction of events are two or more vertices reconstructed). Minimum bias events, with fewer tracks than $t\bar{t}$, are less efficient but less prone to splitting. Two example configurations of the imaging algorithm have been used with 1024 and 2048 z-bins. The imaging algorithm with 1024 z-bins produced similar performance to the iterative algorithm; with 2048 bins more split vertices are produced. The algorithm has not been adjusted to produce the same efficiency or split rate for the two image sizes. In the future, an additional algorithm may be used to recombine these extra split vertices with the full vertex fit information.

The advantage of the larger number of z bins is that vertices can be reconstructed at smaller z separations than in the iterative algorithm, as demonstrated in Fig. 3 showing the Δz separation between all pairs of vertices in minimum bias events with pile-up. The dip at low separation occurs when separate interactions merge into one reconstructed vertex; the small peak near $\Delta z = 0$ for the iterative algorithm includes contributions from splitting. The enhancement above 1 for imaging results from splits as well, but because of the binning used they are not produced at extremely small separations. The imaging algorithm with 2048 z-bins allows more closely spaced vertex pairs to be reconstructed since two seeds can be identified prior to track assignment.

The decrease in the number of merged vertices reduces the quadratic losses, as a function of μ, as shown in Fig. 4. The total number of reconstructed vertices in a minimum bias sample as a function of μ is described by a linear slope, ε, and additional quadratic losses due to merging, m:

$$N_{\text{vtx}}^{\text{reco}} = c_0 + \varepsilon\mu\,(1 - m\mu) \tag{1}$$

While the 1024 bin configuration had very similar single interaction performance to the iterative algorithm, it produces slightly more merging losses at higher μ. The 2048 bin configuration

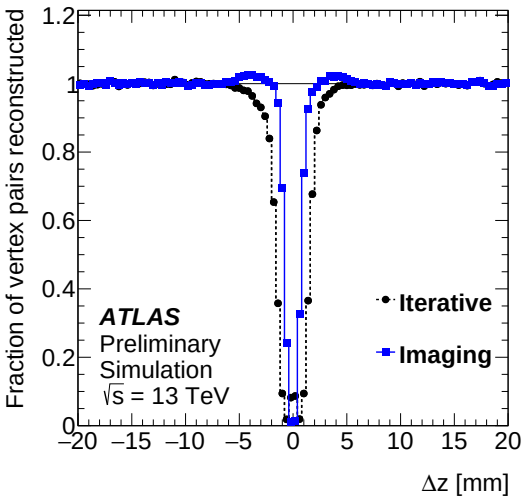


Figure 3: Distance in the z direction between all pairs of vertices in simulated events with pile-up, zoomed in on the region near 0 mm. The histogram is normalized by a Gaussian fit excluding the region $|\Delta z| < 5$ mm, giving the fraction of all pairs actually reconstructed. From Ref. [2].

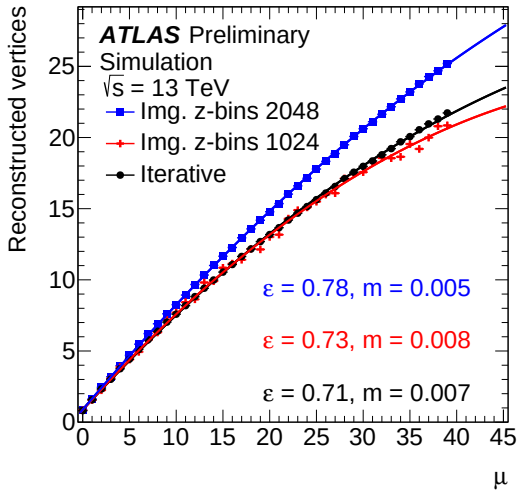


Figure 4: Comparison of the number of reconstructed vertices as a function of the number in-time pile-up collisions, μ, for PYTHIA8 minimum bias events. The iterative algorithm is compared to imaging using both 1024 and 2048 z-bin setups. The number of reconstructed vertices, N_{vtx} is parameterised with a linear term ε, multiplied by a quadratic loss term m: $N_{\text{vtx}}^{\text{reco}} = c_0 + \varepsilon\mu\,(1 - m\mu)$. From Ref. [2].

shows fewer losses, as expected from the behaviour seen in the Δz distribution; with $m = 0.005$ versus $m = 0.008$, at $\mu = 40$ on average the iterative algorithm loses 32% of vertices per event compared to a linear extrapolation of the efficiency, while the imaging algorithm loses 20%.

The other facet to performance involves the time taken to process each event. In Fig. 5, the total time per event to perform the seeding step is compared between the iterative algorithm and the imaging algorithm with 2048 z-bins. This was measured on a machine with a HEPSPEC scaling factor of about 13. It does not include the time spent assigning tracks to each seed or performing the final vertex fit; the total time spent reconstructing vertices, excluding seeding, ranges from < 1 ms at $\mu = 0$ to ≈ 50 ms at $\mu = 40$. Overall, the imaging seeding is slower up to very high values of μ. The imaging seeding time is dominated by the time spent performing Fourier transforms and applying the filter to reconstruct the image, which depend only on the number of bins used in the image histogram. It shows only a weak dependence on the value of μ from the back-projection and seeding steps, while for the iterative approach the seeding time increases quadratically.

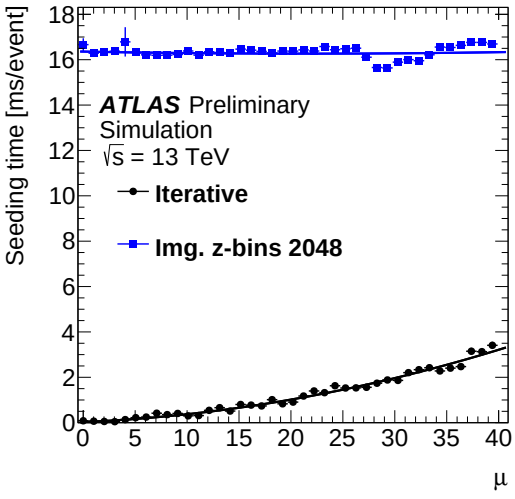


Figure 5: Comparison of the time spent creating vertex seeds in each PYTHIA8 minimum bias event as a function of μ, as measured on a machine with a HEPSPEC scaling factor of about 13. From Ref. [2].

5. Conclusion

An imaging algorithm has been developed for ATLAS in order to produce seed positions for vertex finding and fitting. It uses all tracks in the event to provide simultaneously a full set of seeds, instead of the previous approach of iteratively producing single new seeds and fitting them. This approach is highly modular and configurable in both the imaging itself and the identification of vertex seeds from the image. Preliminary performance studies in simulations corresponding to the conditions expected early in LHC Run-2 show that it is possible to identify more closely spaced interactions and thus to lessen the effects of merging in vertex reconstruction. While the algorithm has a larger overhead in the processing time per event at low amounts of pile-up, the scaling as a function of the number of pile-up collisions is better than the iterative approach. With future development and optimization still to come, this algorithm shows promise for vertex seeding as the number of simultaneous collisions increases in future LHC running.

References

[1] Hageböck S and von Toerne E 2012 *Journal of Physics: Conference Series* **396** 022021
[2] ATLAS Collaboration 2015 ATL-PHYS-PUB-2015-008
[3] Colsher J G 1980 *Physics in Medicine and Biology* **25** 103
[4] ATLAS Collaboration 2008 *JINST* **3** S08003
[5] GEANT4 Collaboration, S Agostinelli et al 2003 *Nucl. Instrum. Meth.* **A 506** 250
[6] ATLAS Collaboration 2010 *Eur. Phys. J.* **C 70** 823 (*Preprint* 1005.4568)
[7] Sjostrand T, Mrenna S and Skands P Z 2008 *Comput. Phys. Commun.* **178** 852 (*Preprint* 0710.3820)
[8] ATLAS Collaboration 2012 ATL-PHYS-PUB-2012-003
[9] Campbell J M, Ellis R K, Nason P and Re E 2014 (*Preprint* 1412.1828)
[10] Nason P 2004 *JHEP* **0411** 040 (*Preprint* hep-ph/0409146)
[11] Frixione S, Nason P and Oleari C 2007 *JHEP* **0711** 070 (*Preprint* 0709.2092)
[12] Alioli S, Nason P, Oleari C and Re E 2010 *JHEP* **1006** 043 (*Preprint* 1002.2581)
[13] Sjostrand T, Mrenna S and Skands P Z 2006 *JHEP* **0605** 026 (*Preprint* hep-ph/0603175)
[14] ATLAS Collaboration 2012 ATLAS-CONF-2012-042
[15] Robertson T and Cryer J D 1974 *J. Amer. Statist. Assoc.* 1012–1016
[16] Waltenberger W, Frühwirth R and Vanlaer P 2007 *J.Phys.* **G34** N343
[17] Amanatides J and Woo A 1987 *Eurographics '87* pp 3–10
[18] Frigo M and Johnson S G 2005 *Proceedings of the IEEE* **93** 216–231
[19] Harris F 1978 *Proceedings of the IEEE* **66** 51–83
[20] ATLAS Collaboration 2005 Tech. Rep. ATLAS-TDR-17 CERN