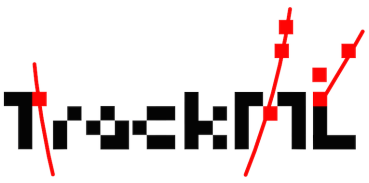
Algorithmes de reconstruction de trajectoire pour détecteurs de particules : Analyse des soumissions du challenge TrackML

Rapport de stage

Félix Faisant, L3 Physique fondamentale, Magistère 1
Encadré par David Rousseau

Laboratoire de l'Accélérateur Linéaire, CNRS / Université Paris-Sud, 91898 Orsay

Juin - Juillet 2018



Vives remerciements à David Rousseau et au LAL pour les deux jours de déplacement au CERN, riches en rencontres et en visites.

Remerciements à Andreas Salzburger, Moritz Kiehn, Dirk Zerwas et toute l'équipe TrackML pour les réponses à mes nombreuses questions.

Abstract

With the advent of the **HL-LHC** will come a five-fold luminosity increase in the detectors of the ATLAS, CMS and LHCb experiments. This will dramatically increase the complexity of track reconstruction, and with the current ATLAS algorithm, the reconstruction time would skyrocket. It is therefore necessary to develop new **reconstruction algorithms**. To bring new ideas to the table, the public **TrackML competition** is organized with the aim of bringing **machine learning**, data science and HEP communities together.

During my internship at LAL, my role was to develop tools to analyse in an automated fashion the proposed algorithms of the TrackML competition. These **analysis tools** are aimed at both organizers and competitors.

The simulated **dataset** provided to competitors and some of its characteristics are described. We then explain the attribution of a score to each algorithm, and the various statistical analyses made (physical efficiencies, track statistics...). In the second part, we present some analyses of a batch of submissions, especially similarities and differences, and some characteristics which allows us to guess some of the methods used by the competitors. Interestingly, a great number of submissions seems to be based on the DBSCAN clustering algorithm, which was initially considered not very promising. Finally, we analyse some algorithms which use completely different techniques.

Looking at the proposed ideas, the outcome of the competition is already very satisfying at the end of the first phase.

Résumé

Avec l'arrivée du **HL-LHC**, la luminosité quintuplera dans les détecteurs des expériences ATLAS, CMS et LHCb, et avec elle la complexité de la reconstruction des traces des particules. Il est donc nécessaire de développer un nouvel **algorithme de reconstruction**. Pour apporter une solution à ce problème, la compétition publique **TrackML** encourage l'apparition de nouvelles idées grâce aux communautés des sciences des données et de l'**apprentissage artificiel**.

Mon stage au LAL consiste à développer des **outils d'analyse** automatiques des algorithmes proposés (les soumissions) par les compétiteurs de TrackML. Ces outils sont destinés aussi bien aux organisateurs qu'aux compétiteurs.

On présente les données simulées fournies aux compétiteurs et certaines de leurs caractéristiques, puis on décrit l'attribution d'un score à chaque soumission, score servant à départager les compétiteurs. Les différentes analyses effectuées sont ensuite décrites. Dans un second temps, on présente quelques analyses d'un lot de soumissions, en particulier les similitudes et différences, et certaines caractéristiques permettant de deviner les techniques utilisées. Les soumissions basées sur l'algorithme de *clustering* DBSCAN, initialement considéré comme peu prometteur, sont nombreuses. On analyse enfin quelques algorithmes fondés sur des techniques différentes.

Du point de vue des idées proposées, le bilan est positif au terme de la première phase de la compétition.

Table des matières

1 Contexte

Le présent travail s'inscrit dans le cadre de la reconstruction de traces dans les détecteurs d'ATLAS, une des expériences du LHC. Le LHC est un collisionneur de hadrons, principalement proton-proton, et ATLAS détecte les particules résultantes de ces collisions. C'est en comparant les données récoltées et les données provenant de simulations du Modèle Standard que les mesures et découvertes physiques sont faites. Dans le LHC, ce sont des paquets de protons qui rentrent en collision. On appelle *évènement* la collision de deux paquets et les interactions qui en résultent.

1.1 Reconstruction des trajectoires dans les détecteurs de particules

Pour chaque évènement, les particules produites par les collisions proton-proton sont observées par différents détecteurs. On s'intéresse ici aux *trackers*, première couche de détecteurs autour du point d'interaction, qui ont pour but d'observer la trajectoire des particules. Lorsque qu'une particule traverse un module, elle y interagit [1], et le résultat, que l'on appelle un *hit*, est enregistré. Pour simplifier, on ne prend en compte ici que les coordonnées (x, y, z) du *hit* [2].

Pour connaître les paramètres de la particule (impulsion, charge), on plonge le détecteur dans un champ magnétique, qui courbe la trajectoire des particules.

À partir de ces *hits*, les trajectoires sont reconstruites grâce à un *algorithme de tracking*. C'est le sujet de ce stage.

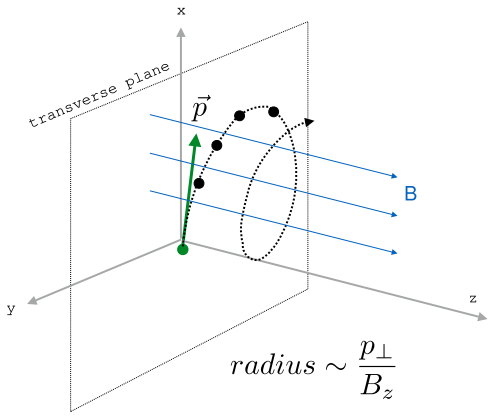


FIGURE 1.1 – Trajectoire hélicoïdale d'une particule dans le détecteur, plongée dans un champ magnétique. Les points noirs représentent les hits. Crédits : [1]

1.2 Défis posés par le HL-LHC

Dès 2018, et pour une mise en service en 2026, le LHC et les expériences associées, dont ATLAS, subiront des modifications (nommées HL-LHC, pour *high luminosity*) afin d'augmenter la luminosité, c'est à dire le nombre de collisions. Cela se traduira par un *empilement* (nombre de collisions simultanées) plus élevé (de $\mu \sim 40$ en 2017 à $\mu \sim 200$ pour le HL-LHC), ou en d'autres termes par un nombre de particules produites bien plus important, environ 10 000 par évènement.

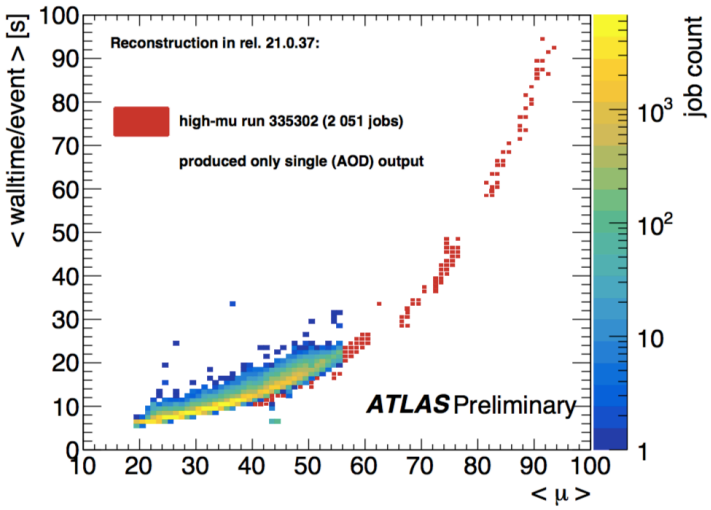


FIGURE 1.2 – Performance de l'algorithme de reconstruction actuel d'ATLAS, en secondes par évènement, en fonction de l'empilement (*pile-up*) de l'évènement. Crédits : ATLAS.

En ce qui concerne le *tracking*, en extrapolant les performances de l'algorithme actuel (fig. 1.2, $\sim \mathcal{O}(n^3)$) le temps de reconstruction exploserait, que l'on peut estimer à 500 s. C'est inacceptable au vu des ressources informatiques disponibles. De plus, l'algorithme actuel

1. Par exemple en créant des paires trous-électrons dans un détecteur au silicium
2. En réalité, la particule passe à travers *plusieurs* pixels de chaque module, et avec un degré (que l'on enregistre) proportionnel à la distance parcourue dans le pixel. À partir de cela, la position (x, y, z) du *hit* est reconstruite. Toutefois, on peut obtenir une plus grande précision en effectuant le tracking sur les données des pixels brutes. Ces données sont donc aussi fournies aux compétiteurs. Voir [1] p.10.

a déjà été hautement optimisé. Il y a donc peu d'espoir qu'il puisse être encore optimisé et conservé. Une méthode de reconstruction alternative est donc nécessaire.

1.3 Challenge TrackML

Afin d'apporter une solution technique à ce problème de performance, une compétition publique est organisée en 2018 : le challenge TrackML. Elle se décline en deux phases : la phase 1 (organisée sur la plateforme Kaggle[3], en mai-août 2018) invite les participants à développer des algorithmes de nouveaux types (par exemple basés sur le *machine learning*), avec la qualité/efficacité de la reconstruction des traces comme seul critère. Les participants génèrent des *soumissions* (associations *hits* du *dataset* → trace) grâce à leur algorithme. Ces soumissions sont ensuite envoyées sur Kaggle (sans l'algorithme, qui reste privé), puis un score est attribué (voir §2.5). La phase 2 (sur la plateforme CodaLab[4], septembre-octobre 2018) prend aussi en compte la rapidité des algorithmes proposés.

1.4 Analyse des soumissions

Le but de ce stage est de développer des outils d'analyse des soumissions du challenge TrackML, autant pour la phase 1 que pour la phase 2 (toutefois sans prendre en compte la rapidité des algorithmes), ainsi que d'assister à l'analyse tout le long de la phase 1. Le besoin de caractériser les soumissions est triple : surveiller le déroulement du challenge en déterminant les techniques utilisées par les compétiteurs, fournir des outils aux compétiteurs eux-mêmes, et enfin aider à la compréhension et au développement des algorithmes une fois le challenge terminé. Mon travail débute concrètement à la section 2.6, et je n'ai pas participé à l'élaboration du score ou du *dataset*.

Ces outils, qui seront disponibles publiquement avant ou après la phase 2, représentent environ 2400 lignes de code Python et 330 lignes de C++ (pour certains calculs, lourds en Python). Ils sont organisés sous forme d'une bibliothèque et de trois *notebooks*, permettant de générer 67 graphiques par soumission, et 64 graphiques permettant de comparer un lot de soumissions. Les graphiques sont générés en Python à l'aide de la bibliothèque `matplotlib`. Ce choix, plutôt que celui du *framework* `ROOT`, a été fait car c'est l'outil communément utilisé dans les sciences des données en dehors de la physique. Enfin, pour l'analyse basée sur les *jets* (dont on ne parlera pas ici), la bibliothèque FastJet a été utilisée.

2 *Dataset*, définitions et outils d'analyse

On présente ici les données fournies aux compétiteurs (le *dataset*), qui provient d'évènement simulés. Le *dataset* est simplifié par rapport à la réalité pour rendre accessible la compétition, tout en gardant les caractéristiques importantes. Ainsi, le problème reste proche de la reconstruction de trace à ATLAS. On présente ensuite les outils d'analyse des soumissions produits durant ce stage.

2.1 Quelques définitions

Les particules du *dataset* sont définies par leur origine (x_0, y_0, z_0) (le *vertex*, là où elles ont été produites), leur charge ($q = \pm 1$) et leur impulsion d'origine, $\vec{p}$. On définit l'*impulsion transverse* par $p_\perp^2 = p_x^2 + p_y^2$ et $\phi = \arctan(p_x/p_y)$, ainsi que la *pseudo-rapidité* $\eta = \operatorname{arcsinh}(p_z/p_\perp)$. Ici, étant donnée la géométrie du détecteur, la pseudo-rapidité est limitée à $[-4, +4]$. Une grande pseudo-rapidité caractérise une trajectoire vers l'avant/l'arrière du détecteur, alors qu'une faible pseudo-rapidité indique que l'impulsion est perpendiculaire à l'axe Z du détecteur. Dans toute la suite, η et ϕ seront utilisés sans unités particulières, et on prendra toujours implicitement en compte la topologie de ϕ ($2\pi \equiv 0$) dans les calculs.

3. https://www.kaggle.com/c/trackml-particle-identification
4. Le lien sera publié sur la page https://sites.google.com/site/trackmlparticle/home

2.2 Particules primaires

Dans le *dataset*, pour chaque évènement (tous indépendants entre eux), la collision proton-proton est simulée par une interaction principale (dite *hard scatter*) $t\bar{t}$ ([5]), ainsi que d'autres interactions, de plus faibles énergies. Les particules provenant directement de ces interactions sont dites *primaires*. La densité de collisions présente une extension spatiale gaussienne, centré en $z_0 = r_0 = 0$ avec $\sigma_{z_0} = 5.5\,\text{mm}$ et $\sigma_{x_0} = \sigma_{y_0} = 15\,\mu\text{m}$. Mais certaines particules intermédiaires produites ont une durée de vie faible non nulle, ce qui explique que l'origine des particules (*vertex*) est distribuée de façon un peu plus large (fig. 2.1). L'ensemble des particules primaires produites est globalement neutre.

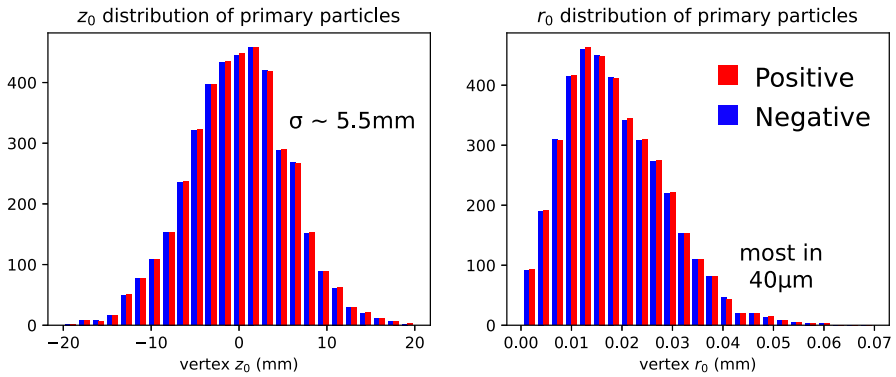


FIGURE 2.1 – À gauche, distribution de l'origine des particules primaires, sur 1 évènement (moyenne sur 30 évènements). En rouge, particules de charge électrique positive ($n = 4400$) ; en bleu, négative ($n = 4384$). La distribution *volumique* de l'origine en x_0 et y_0 est bien gaussienne, mais la distribution en r_0 ne l'est pas à cause du facteur géométrique : $\rho(r) \propto 2\pi r \cdot \exp\left(-r^2/\sigma^2\right)$. Environ 70% des primaires ont une origine $r_0 < 40\,\mu\text{m}$.

Ces particules produisent des *hits* ($\sim 114\,000$ par évènement) dans les détecteurs, dont la géométrie est représentée sur la figure 2.2, que l'on retrouve sur la densité de *hits*. Ce sont les coordonnées de ces *hits* qui sont fournis aux participants, et à partir desquelles les traces sont reconstruites.

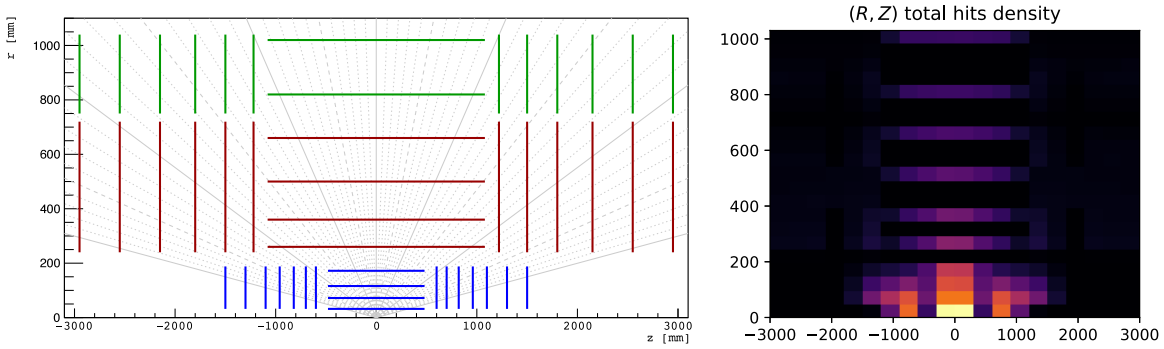


FIGURE 2.2 – À gauche, coupe des détecteurs simulés dans le plan longitudinal (R, Z). Les 3 couleurs représentent les 3 détecteurs, ayant des propriétés différentes (résolution, …). Les lignes horizontales définissent des détecteurs en cylindres d'axe Z (*barrel*), alors que les lignes verticale définissent des disques (*cap*). Les lignes grises sont les lignes à η constant. Crédits : [1]. À droite, densité du nombre de *hits* dans le plan (R, Z). La majorité des hits se situe dans le *barrel*.

2.3 Particules secondaires

En réalité, les particules interagissent avec la matière du détecteur et peuvent produire des particules secondaires (fig. 2.3.a). Ces effets ont été en partie reproduits dans la simulation, et les particules secondaires représentent en moyenne 9.2 % d'un évènement. De plus, du bruit est rajouté sous forme de *random hits* (fig. 2.3.b), de sorte que les algorithmes proposés soient robustes au bruit. Il y en a en moyenne 630 par évènement, soit 0.55 % des *hits*. Les secondaires de charge positive sont plus nombreux (fig. 2.4.a), car les secondaires sont principalement le résulat de l'interaction avec des noyaux, positifs.

5. quark top - anti-quark top, choisie pour sa richesse physique

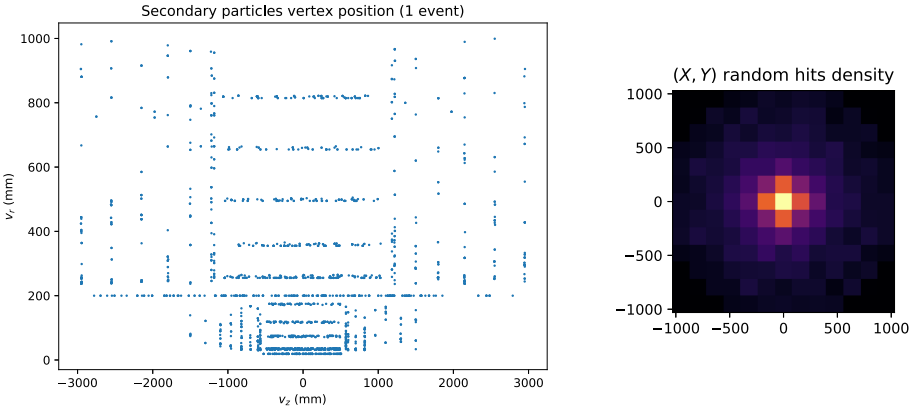
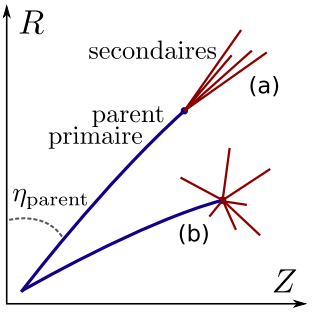


FIGURE 2.3 – À gauche, position de l'origine des particules secondaires dans le plan (R, Z), crées dans la matière (détecteurs) et non à l'origine. À droite, densité du nombre de *random hits* (bruit) dans le plan transverse (X, Y).



Lorsque l'on regarde la distribution (fig. 2.4.b) en η des particules-parent des secondaires (que l'on note η_{parent}), on observe deux pics symétriques en $\eta \simeq \pm 2$. C'est simplement dû à la géométrie du détecteur. Si l'on considère une production de gerbe de secondaires ayant une direction proche de celle de la particule-parent, comme sur le schéma ci-contre (a), on s'attendrait à ce que la distribution en η des secondaires eux-mêmes possède deux pics en $\eta \simeq \pm 2$, un peu plus larges. Or cette distribution (fig. 2.4.a) ne possède qu'un pic et ressemble à une gaussienne.

Cette absence de corrélation nette entre η et η_{parent} nous a initialement fait penser à un *bug* dans la simulation : les particules secondaires sont générées dans le référentiel de la particule-parent, et il aurait été oublié d'effectuer un *boost* relativiste pour se replacer dans le référentiel global. Si cela avait été vrai, la corrélation aurait été nulle aussi entre ϕ et ϕ_{parent}. Or, on observe (fig. 2.4.c,d) une faible mais non-nulle corrélation.

En réalité, les particules produisant des secondaires sont majoritairement des particules de faible impulsion (ou des particules déposant de faibles impulsions) et la majorité de l'énergie déposée est convertie en masse, et non en impulsion. Les secondaires partent donc dans toutes les directions, peu corrélées avec celle de la particule primaire. Elles forment plus un "feu d'artifice" (schéma (b)) qu'une gerbe. Ce fait rend d'autant plus difficile la reconstruction des secondaires, car les algorithmes supposent souvent que les traces sont des courbes passant par le centre.

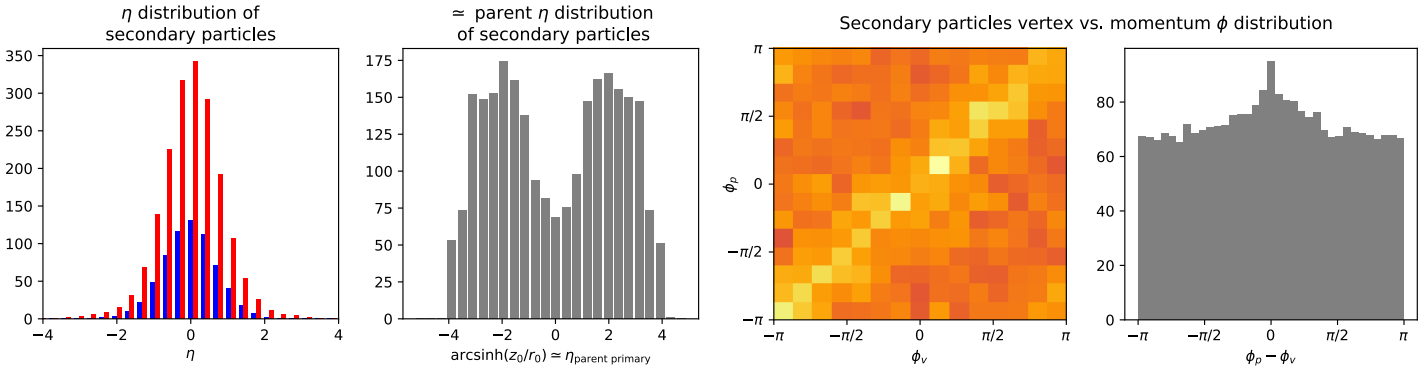


FIGURE 2.4 – De gauche à droite : distribution en η des secondaires ; distribution en $\arcsin(z_0/r_0)$, c'est à dire approximativement la distribution en η de la particule-parent des secondaire ; distribution 2D en $\phi_{\text{parent du secondaire}}$ (vertical) et en $\phi_{\text{secondaire}}$ (horizontal) des secondaires ; et enfin distribution en $\phi_{\text{parent}} - \phi_{\text{secondaire}}$ permettant de mieux visualiser la distribution 2D.

2.4 Classification des traces

Pour pouvoir calculer le score des soumissions, il est d'abord nécessaire de classer les traces reconstruites. Il est en effet nécessaire d'éliminer du calcul du score les traces "poubelle" (regroupant tous les hits non associés) ou les traces correspondant à plusieurs particules, par exemple.

La première étape est de déterminer, pour chaque trace reconstruite, la *particule majoritaire* (*majority particle*) de la trace, c'est à dire la (vraie) particule dont les *hits* sont les plus nombreux dans la trace proposée. On définit, pour chaque trace reconstruite, n_{hits}, $n_{\text{maj.p. event hits}}$ et $n_{\text{maj.p. track hits}}$ ainsi :

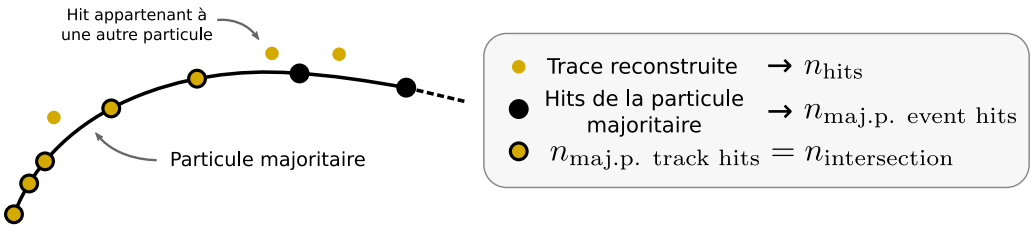


FIGURE 2.5 – Définition de n_{hits}, $n_{\text{maj.p. event hits}}$ et $n_{\text{maj.p. track hits}}$.

On calcule ensuite, pour chaque trace, les deux nombres :

$$\text{purity\_maj} = \frac{n_{\text{maj.p. event hits}}}{n_{\text{hits}}} \qquad \text{purity\_rec} = \frac{n_{\text{intersection}}}{n_{\text{hits}}}$$

Les traces sont alors classées en quatre groupes :

purity_maj	purity_rec	
$> 50\,\%$	$> 50\,\%$	*good track*
$\leq 50\,\%$	$> 50\,\%$	*split track*
$> 50\,\%$	$\leq 50\,\%$	*multiple track*
$\leq 50\,\%$	$\leq 50\,\%$	*bad track*

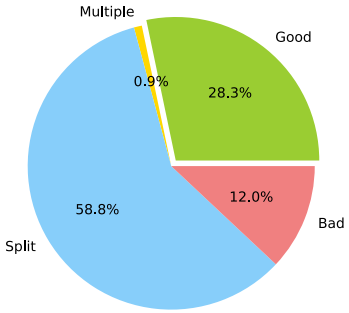


FIGURE 2.6 – Répartition typique des types de traces.

2.5 Poids et score

Toutes les particules n'ont pas la même importance dans l'analyse physique des évènements, et donc dans la reconstruction : les particules de grande impulsion (traces peu courbes) et ayant un grand nombre de *hits* (permettant de déterminer ses paramètres avec précision) doivent être favorisées par les algorithmes de reconstruction. De plus, dans chaque trace, les *hits* proches de l'origine (là où les détecteurs sont le plus précis) et éloignés (aidant à déterminer les paramètres de la particule) doivent être assosciés à la bonne particule en priorité. Les particules laissant moins de 4 *hits* sont en général inexploitables.

Ainsi, pour prendre en compte cet aspect dans le score des soumissions, on fixe un poids w_i à chaque *hit* d'un évènement, d'autant plus élevé qu'on le considère important, comme décrit ci-dessus. Le poids est déterminé à l'avance et gardé secret. Lors du calcul du score, on associe à chaque bonne trace (les traces dont purity_maj ou purity_rec $\leq 50\%$ ne rapportent rien au score) un score partiel W, valant la somme des poids des hits de la particule majoritaire (notés ●, comme en fig. 2.5). Pour obtenir le score de chaque évènement, on somme tous les scores W des *bonnes traces* reconstruites. Enfin, on moyenne sur tous les évènements du *dataset*. En d'autres termes :

$$W = \sum_{i \in \{\bullet\}_{\text{maj.p.}}} w_i \qquad \text{score} = \left\langle \sum_{\{\text{good tracks}\}} W \right\rangle_{\text{events}}$$

2.6 Efficacités

Si le score permet de classer les algorithmes, il est intéressant de connaître l'*efficacité* ϵ de l'algorithme pour la reconstruction des traces selon un paramètre physique α (impulsion, origine...). Pour construire l'histogramme d'efficacité, on compte dans chaque intervalle sur α, le nombre de bonne traces dont la particule majoritaire est dans l'intervalle, puis on normalise par le nombre de particules de l'évènement dans le même intervalle. En d'autres termes, si $\{n_{\text{rec}}(\alpha)\}$ représente l'histogramme

des particules correctement reconstruites (*good tracks*), et $\{n_{\text{true}}(\alpha)\}$ l'histogramme des particules véritables (*dataset*), alors l'efficacité est le rapport $\epsilon(\alpha) = n_{\text{rec}}(\alpha)/n_{\text{true}}(\alpha)$. Les barres d'erreur sont calculées avec la formule $\Delta\epsilon = \sqrt{\epsilon \cdot (1-\epsilon)/n_{\text{true}}}$ (loi binomiale, qu'on ne justifiera pas ici).

La figure 2.7 représente diverses efficacités. On calcule aussi les efficacités en se restreignant aux particules primaires ou aux secondaires, qui ont plus d'intérêt que l'efficacité brute. Les algorithmes se comportent en effet souvent très différemment sur les particules primaires et secondaires. On fait de même en se restreignant aux particules pour lesquelles $n_{\text{hits}} \geq 4$ (poids non nul dans le score).

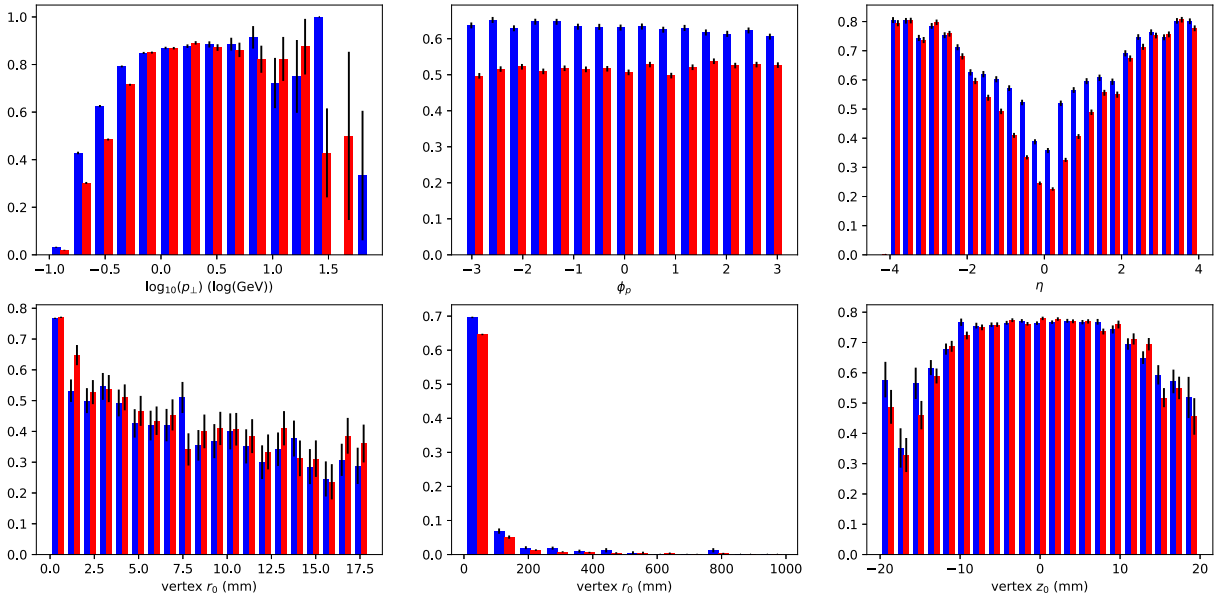


FIGURE 2.7 – Quelques efficacités totales (primaires + secondaires) d'un algorithme typique. La composante rouge représente les particules de charge positive, et la bleue les particules négatives.

En général, l'efficacité est indépendante de ϕ, augmente avec l'impulsion transverse $p_\perp$, et lorsque l'on se rapproche de l'origine ($z_0 = r_0 = 0$). Sauf pour les algorithmes à très bon score, l'efficacité est inférieure sur le *barrel* (petit η) et plus grande pour les particules orientées vers l'avant/arrière.

2.7 $\Delta_{\text{min}}R$

Un paramètre pour lequel il est intéressant de connaître l'efficacité des algorithmes est $\Delta_{\text{min}}R$. Soit une particule primaire dont l'impulsion initiale pointe dans la direction (ϕ, η). Alors $\Delta_{\text{min}}R$ est la distance euclidienne, dans l'espace (η, ϕ), à la plus proche particule primaire :

$$\Delta_{\text{min}}R = \min_{p \in \text{prim. particles}} \sqrt{(\phi - \phi_p)^2 + (\eta - \eta_p)^2}$$

On définit de même $\Delta_{\text{min, same sign}}R$ et $\Delta_{\text{min, opp. sign}}R$ la distance minimale dans l'espace (η, ϕ) en restreignant la recherche, respectivement, aux particules de même signe et de signe opposé.

La figure 2.9 représente l'efficacité $\epsilon(\Delta_{\text{min}}R)$ d'un algorithme. Comme attendu, l'efficacité augmente avec $\Delta_{\text{min}}R$. En effet, des particules bien séparées dans l'espace de direction d'impulsion initiale forment des traces spatialement plus séparées, donc plus faciles à distinguer à la reconstruction.

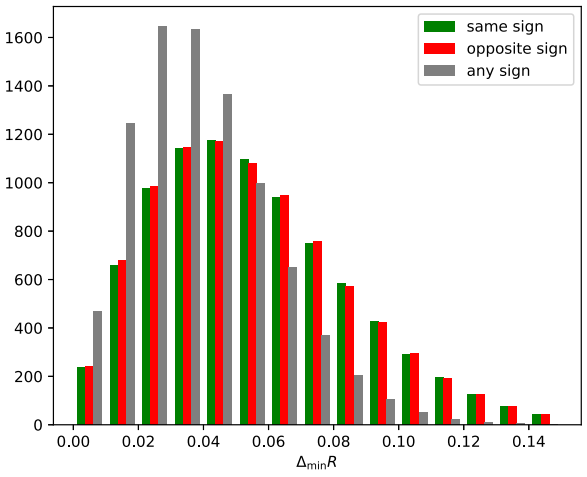


FIGURE 2.8 – Distribution moyenne des $\Delta_{\text{min}}R$ par évènement du *dataset*. On remarque que les distributions tenant compte des signes sont plus plus étalées vers les grands $\Delta_{\text{min}}R$, car il faut en moyenne chercher plus loin pour trouver une particule voisine de même charge ou de charge opposée.

De plus, on remarque dans cet exemple que, à faible ΔR, l'efficacité pour $\Delta_{\text{min,same}}R$ est plus faible que l'efficacité pour $\Delta_{\text{min,opp.}}R$, et que l'efficacité pour $\Delta_{\text{min,any}}R$ est intermédiaire. En effet, à même séparation d'impulsion initiale, des particules de charge opposée sont défléchies dans des sens opposés, ce qui augmente la séparation des traces et donc la facilité de reconstruction.

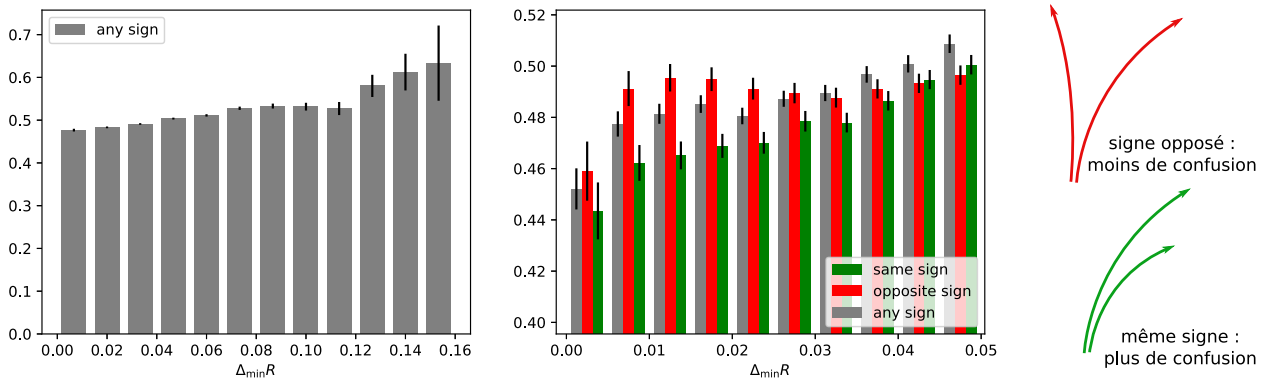
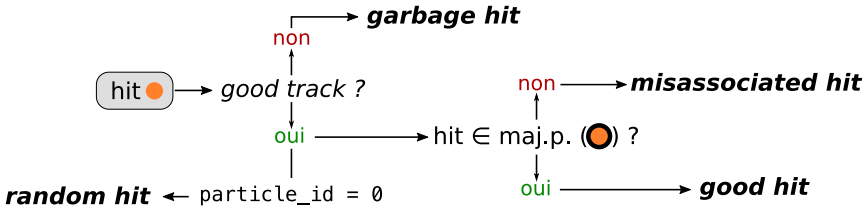


FIGURE 2.9 – Efficacité selon $\Delta_{\text{min}}R$ d'un algorithme typique. Au centre, zoom sur les efficacités selon $\Delta_{\text{min,same}}R$ et $\Delta_{\text{min,opp.}}R$, dont les différences se comprennent sur le schéma de droite.

2.8 Classification et distributions des *hits*

En plus des statistiques concernant les particules, il est intéressant d'étudier les *hits* eux-mêmes. Pour cela, on les classifie par rapport aux traces proposées de la soumission, selon le diagramme ci-dessous :



On remarque que trois catégories *garbage hits*, *misassociated hits* et *good hits* forment une partition l'ensemble des hits. Une fois tous les hits d'un évènement classés, on peut calculer une distribution spatiale de ces 4 types de *hits*. C'est ce que représente la figure 2.10, où chaque pixel est normalisé par le nombre total de *hits* dans le pixel : si $n_{\text{good}}(r,z)$ est le nombre de *good hits* et $n_{\text{tot}}(r,z)$ est le nombre total de hits dans l'intervalle autour de (r,z), alors le pixel vaudra $n_{\text{good}}(r,z)/n_{\text{tot}}(r,z)$.

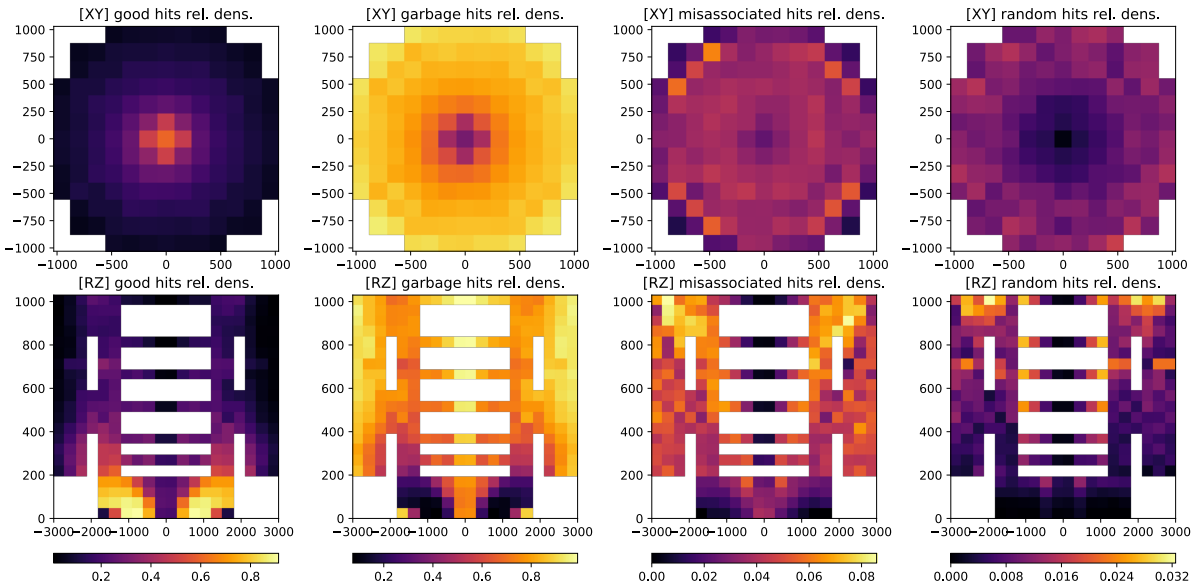


FIGURE 2.10 – Densités relatives des 4 types de hits pour un algorithme typique, dans le plan (X,Y) en haut et (R,Z) en bas (R verticalement, Z horizontalement). X, Y et R sont en mm, et chaque type de *hit* possède son échelle de couleur. L'addition des trois premières densités donnerait une distribution uniforme et valant 1.

La distribution des *good hits* dans le plan (R, Z) confirme l'observation de l'efficacité $\epsilon(\eta)$ (fig. 2.7) (que l'on observe aussi sur l'efficacité restreinte aux particules primaires) : les traces orientées vers l'avant/arrière du détecteur sont largement favorisées, alors que le taux de *good hits* sur le *barrel* est très faible et presque tous les hits du *barrel* sont attribuées à des mauvaises traces (excessivement scindées dans ce cas). On remarque aussi que les traces reconstruites sont ici relativement pures (on ne dépasse pas 10% de *misassociated hits*), et que cet algorithme est plutôt robuste au bruit.

2.9 Statistiques sur les traces

En plus des critères physiques, il est intéressant de regarder les statistiques sur toutes les traces proposées, notamment pour différencier les algorithmes entre eux et étudier les techniques utilisées par les compétiteurs :

- distributions du nombre de traces proposées / bonnes traces selon le nombre de *hits* de la trace, le nombre de *hits* de la particule majoritaire, le nombre de *hits*-bruit, le nombre de différentes particules dans la trace, le nombre de *hits* de la particule majoritaire manqués (voir fig. 2.12)
- distributions des traces par poids W, et efficacité selon le poids de la particule majoritaire (voir fig. 3.3 col. b)
- proportion de bonnes traces parmi les traces proposées, selon les paramètres déjà cités
- distributions de la pureté des traces (voir fig. 2.11 pour un exemple)
- proportions des types de traces (voir §2.4)

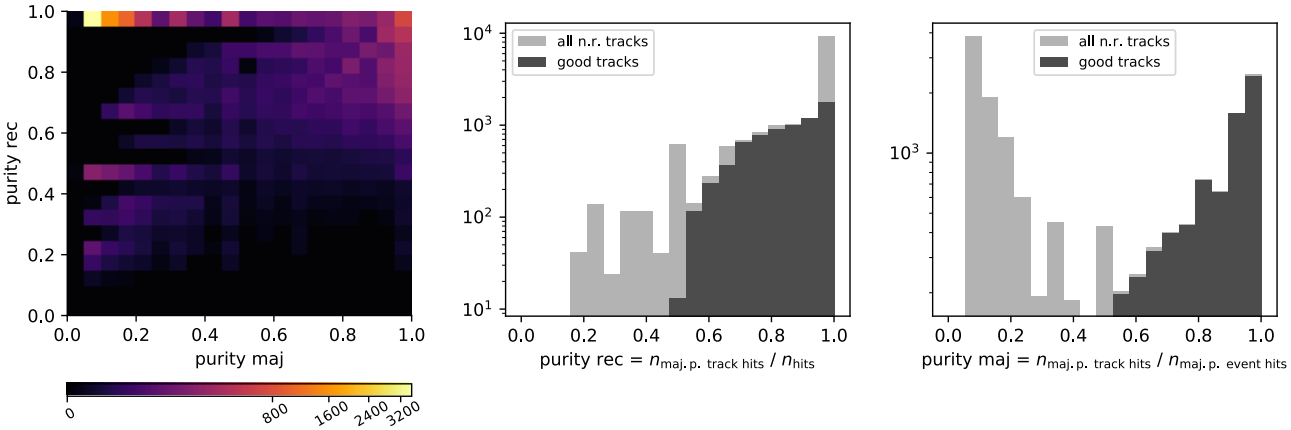


FIGURE 2.11 – Distributions de la pureté des traces reconstruites sur un exemple. À gauche, histogramme 2D du nombre de traces proposées par purity_rec et purity_maj. À droite, la projection de l'histogramme sur chacun des deux axes, avec de plus la fraction des bonnes traces (purity_rec & purity_maj > 0.5). Les effets de quadrillage/pics (par exemple à 0.5) sont dûs au fait que purity_rec et purity_maj sont le résultat de la division de deux entiers. On remarque ici que la majorité des mauvaises traces (66.4%) le sont à cause de purity_maj < 0.5, c'est à dire que la trace proposée contient moins de la moitié des *hits* de la particule majoritaire.

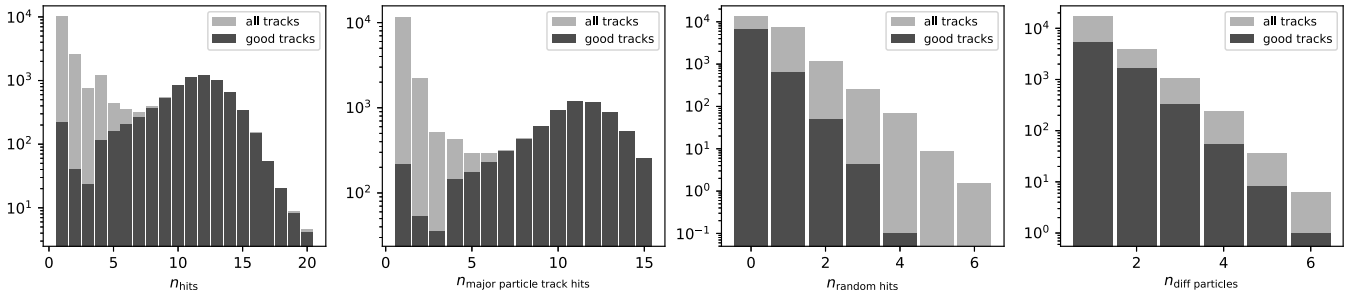


FIGURE 2.12 – Distributions typiques du nombre de traces reconstruites par évènement selon le nombre de *hits* de la trace n_{hits}, le nombre de *hits* de la particule majoritaire $n_{\text{maj.p. hits}}$, le nombre de *hits*-bruit $n_{\text{random hits}}$, et le nombre de différentes particules dans la trace proposée. En gris clair, toutes les traces ; en gris foncé, bonnes traces.

3 Analyse des soumissions du challenge

J'analyse ici certaines soumissions du challenge TrackML, motivé par le besoin d'analyse des organisateurs tout le long de la compétition. Par souci d'anonymisation [6] et de concision, les soumissions seront désignées par leur score. L'analyse de l'algorithme de reconstruction actuellement utilisé à ATLAS n'est pas faite ici car le code brut n'est pas accessible publiquement, et le projet ACTS [7], censé fournir une implémentation publique, n'est pas terminé.

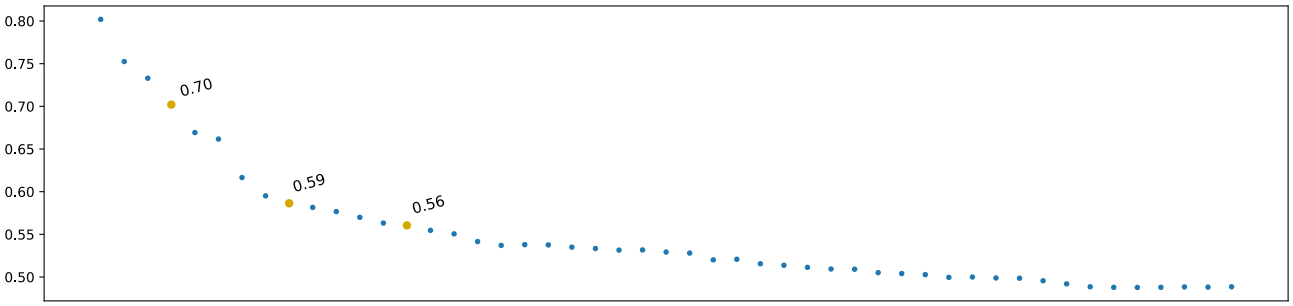


FIGURE 3.1 – Score de chaque soumission du lot analysé dans cette section, triées de gauche à droite par score. En orange, les trois premières soumissions analysées dans la section 3.1 (la quatrième est en dehors).

3.1 Soumissions de type DBSCAN

DBSCAN est un algorithme standard et simple de *clustering*, c'est à dire de regroupement de points "proches" dans un espace quelconque (fig. 3.2). C'est un algorithme générique, qui n'a *a priori* rien à voir avec la reconstruction de traces.

Il se trouve que durant la compétition — et à notre surprise [8] — les algorithmes basés sur DBSCAN se sont révélés être populaires (voir §3.3) et avoir un fort potentiel. En effet, certaines soumissions ressemblant fortement à des solutions que l'on sait être basées sur DBSCAN (annotées d'une étoile ($\star$) dans cette section) ont aisément dépassé 0.7 de score. On décrira de ces soumissions comme étant de *type DBSCAN*, sans toutefois pouvoir affirmer du tout qu'elles sont effectivement basées sur DBSCAN [9]. Voyons concrètement sur quelques-unes de ces soumissions ce que cela signifie, et les caractéristiques nous permettant de faire ce rapprochement. Au contraire, nous verrons en §3.2 des soumissions présentant de fortes différences avec celles de type DBSCAN.

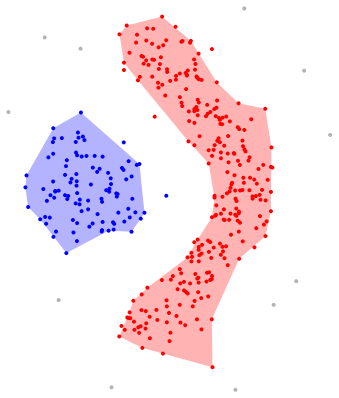


FIGURE 3.2 – Exemple de *clusters* de points, générés par DBSCAN. Crédits : Utilisateur *Chire*, Wikipedia

3.1.1 Similitudes

Lorsque l'on parcourt et compare visuellement les 15 figures des 60 soumissions en tête de classement, on remarque immédiatement une grande similitude entre une 50$^{\text{aine}}$ d'entre elles, par rapport à la 10$^{\text{aine}}$ restante. De plus, cette majorité est similaire à des solutions dont on sait être basées sur DBSCAN. On conjecture donc que ces soumissions sont elles aussi basées sur DBSCAN. On trouve sur la figure 3.3 quatre distributions caractéristiques (du moins visuellement) des algorithmes de type DBSCAN.

6. Les soumissions ont été analysées alors que la compétition n'était pas terminée, et certains résultats ont été repris pour des conférences publiques durant la compétition (CHEP 2018, ICHEP 2018, groupe ATLAS-LAL). Il était donc nécessaire que les compétiteurs ne puissent pas avoir d'indices sur les méthodes utilisées par un compétiteur en particulier.

7. `https://gitlab.cern.ch/acts/acts-core`. On estime qu'il attendrait un score de ∼0.95 en ∼500 s/évènement.

8. Le but premier du challenge est de voir se développer des solutions basées sur le *machine learning*, et DBSCAN étant un algorithme de *clustering* générique, les solutions basées sur DBSCAN nous semblaient avoir peu de potentiel.

9. Les informations apportées par certains compétiteurs à la fin de la compétition ne confirment ni infirment le rapprochement fait sur les quatre soumissions analysées ci-dessous.

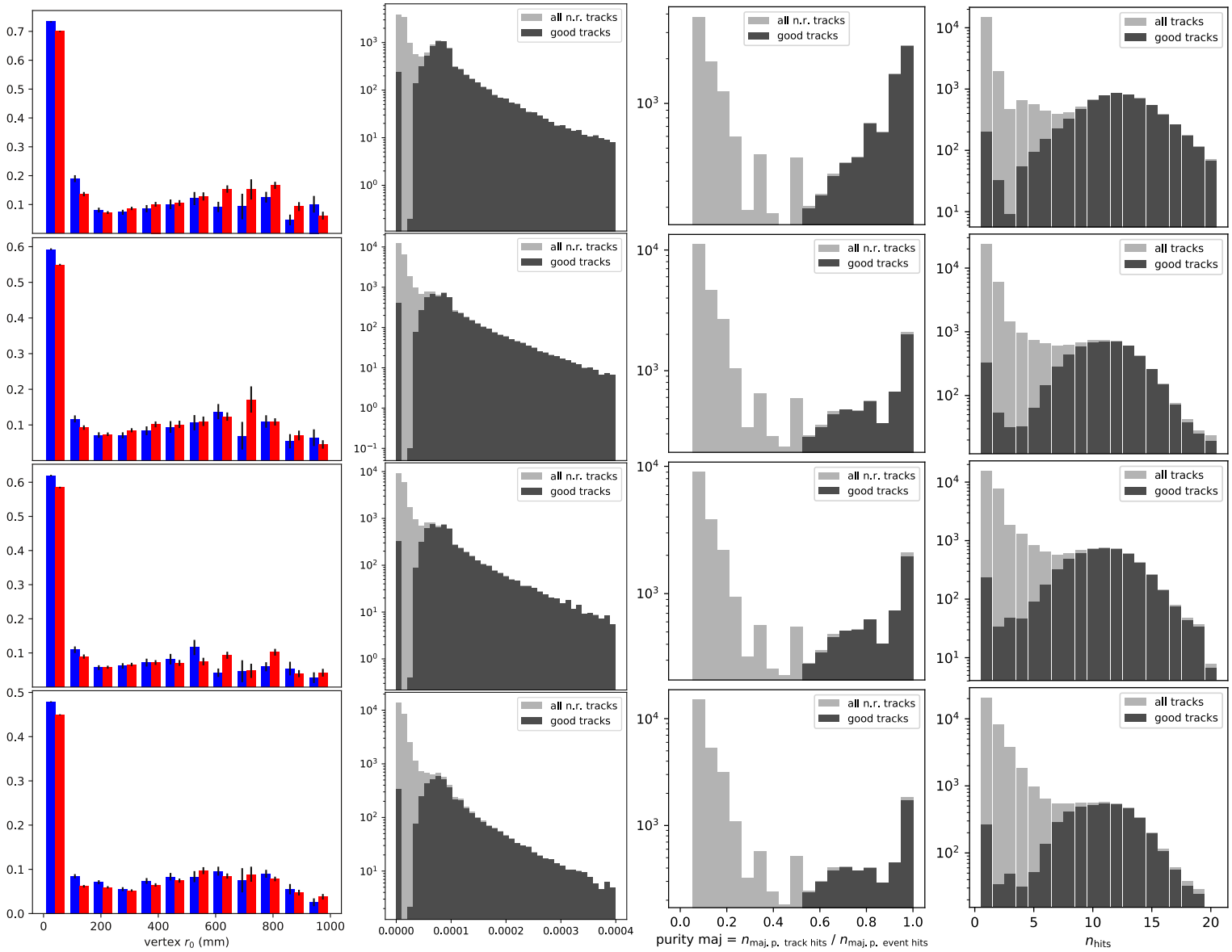


FIGURE 3.3 – Soumissions de haut en bas : 0.70, 0.56, 0.59 (⋆), 0.46 (⋆). De gauche à droite :
- Efficacité totale (particules primaires + secondaires) suivant r_0 (origine particule) sur tout le détecteur
- Nombre de traces proposées (en gris clair, toutes les traces ; celles valides en gris foncé) suivant :
  - le poids $W = \sum_{i \in \text{maj.p. track hits}} w_i$ des *hits* associés à la particule majoritaire
  - le paramètre *purity maj.* $= n_{\text{maj.p. track hits}} / n_{\text{maj.p. event hits}}$ de la trace
  - le nombre de *hits* de la trace

L'efficacité suivant r_0, en particulier pour $r_0 > 100\,\text{mm}$ (c'est à dire l'efficacité sur les secondaires) est non nulle (autour de 0.07), même si l'algorithme a un score médiocre, ce qui diffère notablement de certaines soumissions sortant de ce lot (voir fig. 2.7.e, fig. 3.4 ci-dessous, et fig. A.3). Et ce n'est pas une surprise : DBSCAN étant un algorithme de *clustering* générique, il parvient à suffisamment regrouper les *hits* des particules ayant une origine éloignée du centre (malgré les changements de coordonnées effectués). On remarque que l'efficacité présente un léger maximum secondaire autour de 700 mm, zone où la densité de *hits* est plus faible (confusion plus faible).

Les distributions du nombre de traces proposées suivant W, *purity maj.* et n_{hits} sont elles aussi très caractéristiques (comparer à fig. 2.11, 2.12, 3.8, 3.12) des soumissions de type DBSCAN. On remarque un grand nombre de traces proposées (∼80%) pour lesquelles $W < 0.00005$, de même qu'un grand nombre de traces dont *purity maj.* < 0.5 (*split tracks* et *bad tracks*), ce qui n'étonne pas venant d'algorithmes de *clustering* : il n'y a pas d'apprentissage favorisant les poids et/ou nombre de *hits* par trace élevés, résultant en des traces de faible poids et/ou coupées. Ce comportement diffère d'autres types de soumissions (fig. 3.5).

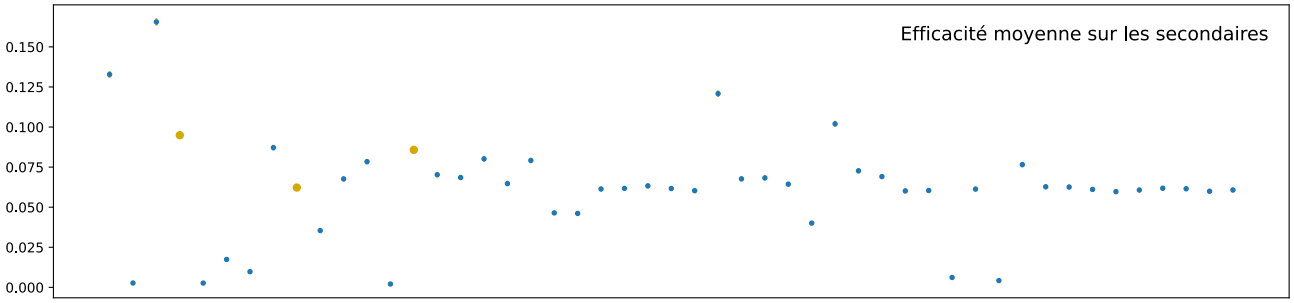


FIGURE 3.4 – Efficacité moyenne sur les particules secondaires de chaque soumission du lot. On constate que seule une dizaine de soumissions affiche une efficacité sortant de l'intervale $[0.05, 0.10]$.

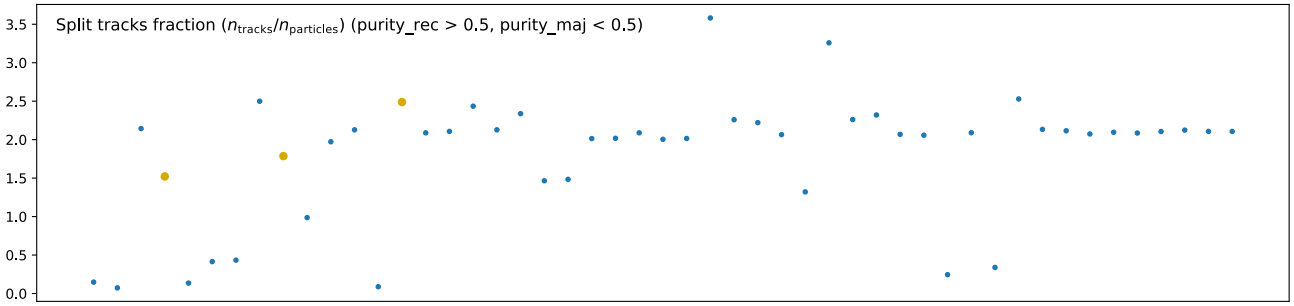


FIGURE 3.5 – Nombre de *split tracks* ramené au nombre de particules, pour chaque soumission du lot. On constate que de nombreuses soumissions affichent une fraction autour de 2. Évidemment, celles s'éloignant de cette valeur peuvent être basées aussi sur DBSCAN, et pourraient par exemple faire usage d'un post-traitement visant à regrouper des traces coupées/incomplètes [10].

On trouvera un indicateur supplémentaire en annexe (fig. B.2). Le dernier élément nous permettant d'affirmer qu'une grande part des soumissions de score inférieur à 0.60 se base probablement sur DBSCAN est l'activité des contributeurs sur la plateforme du challenge : les discussions sur le forum à propos de DBSCAN sont nombreuses, et plusieurs *kernels* utilisant DBSCAN ont été publiés. Mais on remarque déjà, comme on le verra au §3.3, que la plupart des soumissions de score supérieur à 0.60 sortent du lot.

3.1.2 Différences

Les soumissions de type DBSCAN présentent toutefois une certaine variabilité. En effet, même s'il est probable que DBSCAN (ou ses variantes) soit utilisé au cœur de ces algorithmes, toute une panoplie de pré-traitement et post-traitement est appliquée. Ce sont ces traitements qui font la différence parmi les soumissions. Citons-en quelques-uns [11] :

- distance utilisée entre les points, paramètre ϵ de DBSCAN
- changements de coordonnées (*Hough transform*, *helix unrolling*)
- dimension de l'espace de *clustering* utilisée (de 2 à 5 rapporté)
- stratégies d'élimination des *outliers* / points bruités, *cutoffs* sur n_{hits}, *helix fitting*…
- stratégies de fusion/combinaison de traces (*merging strategies*) simples ou itératives

La conséquence la plus visible est sur l'efficacité suivant η et sur les *hits maps*. On remarque que, suivant les soumissions, l'efficacité sur le *barrel* (faible η) prend des valeurs extrêmes (voir fig. 3.6), de maximale (pour les soumissions à haut score) à nulle (pour certaines soumissions de score plus faible). On remarque une efficacité par paliers sur la soumission 0.46 (*kernel* public de Grzegorz Sionkowski [12]), qui sont dûs au fait que cette solution exécute DBSCAN de façon segmentée, après changement de coordonnées.

10. Voir la discussion https://www.kaggle.com/c/trackml-particle-identification/discussion/60330.
11. Voir le forum pour plus de détails : https://www.kaggle.com/c/trackml-particle-identification/discussion
12. https://www.kaggle.com/sionek/mod-dbscan-x-100-parallel

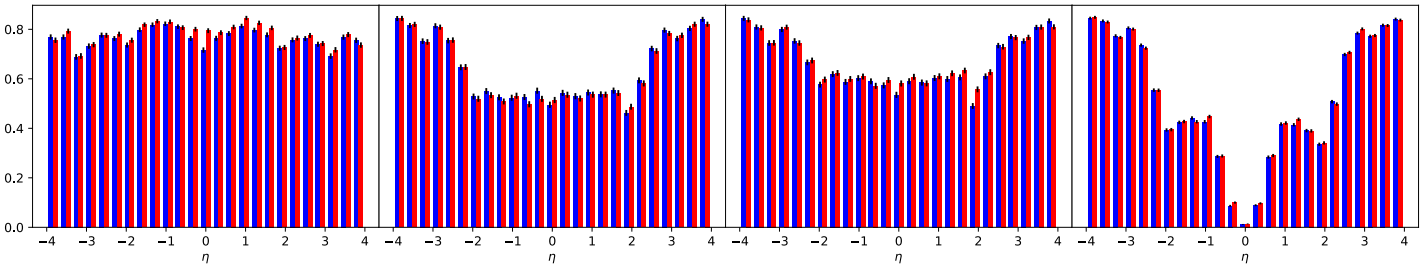


FIGURE 3.6 – Efficacité suivant η (pseudo-rapidité) sur les particules primaires. Soumissions de gauche en droite : 0.70, 0.56, 0.59 ($\star$), 0.46 ($\star$).

On retrouve l'efficacité moindre à petit η (impulsion p_z petite) sur les *hits maps* (fig. 3.7) : la densité de *good hits* est bien plus importante pour les traces vers l'avant ou vers l'arrière du détecteur (sur l'axe Z), et la densité de *garbage hits* est maximale dans le plan $Z = 0$. De plus, on remarque que même si la soumission 0.70 affiche une efficacité suivant η presque plate, la densité de *good hits* reste plus importante dans le cône $\theta \sim \pm 10°$ que sur le *barrel*. Les deux soumissions présentent une discontinuité en $R = 700\,\text{mm}$. Enfin, on remarque une plus forte sensibilité aux *random hits* de la soumission 0.70, ainsi qu'une forte disparité de la densité de *misassociated hits*, surtout pour $R < 300\,\text{mm}$.

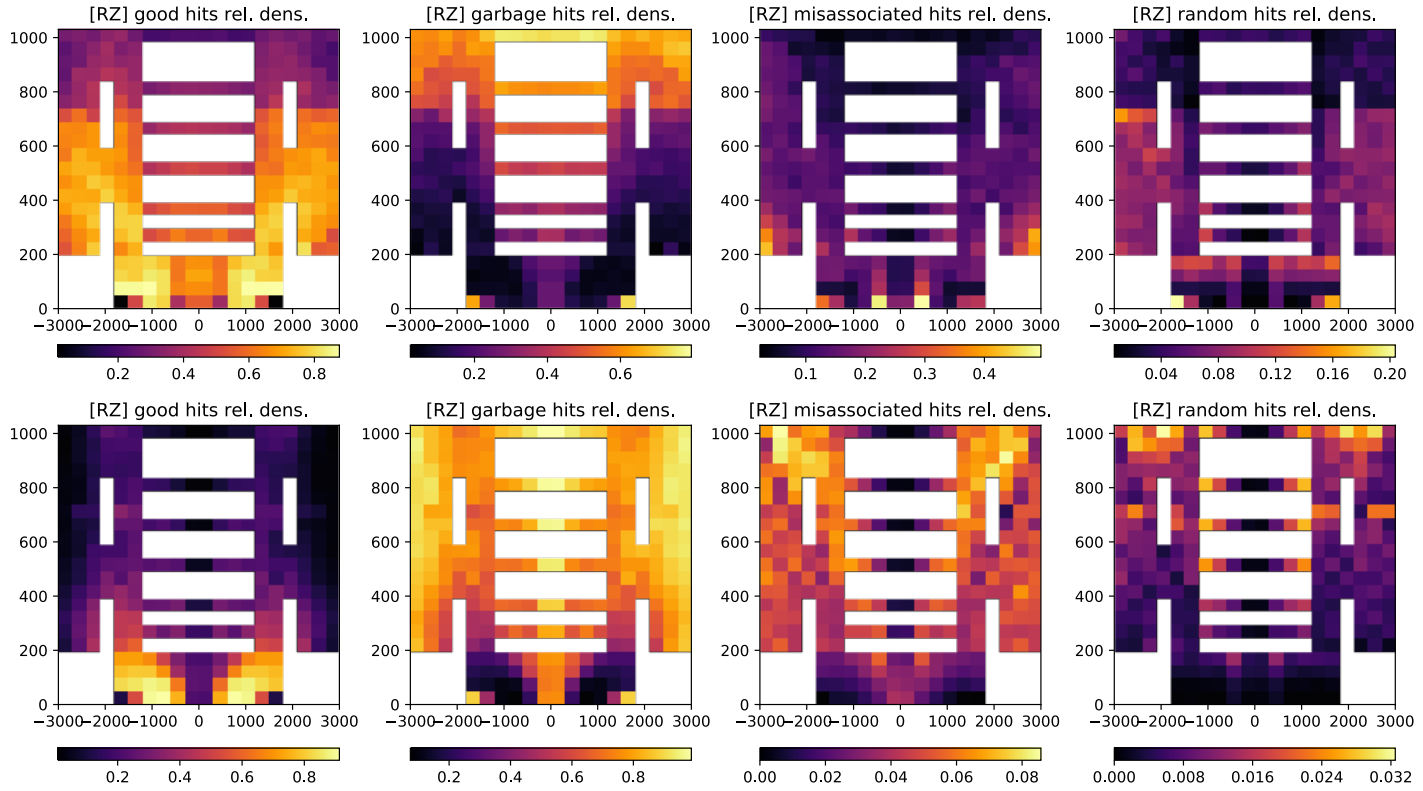


FIGURE 3.7 – Densité spatiale ($n_{\text{rec}}/n_{\text{true}}$ dans chaque case) de *good hits*, *garbage hits*, *misassociated hits*, *random hits* dans le plan R-Z du détecteur. En haut, soumission 0.70. En bas, soumission 0.46 ($\star$). L'unité des axes R et Z est le mm. Prêter attention aux échelles de couleur.

On trouvera en annexe (fig. A.1, A.2) des analyses supplémentaires de ces quatre soumissions.

3.2 Quelques soumissions sortant du lot

On analyse ici succinctement deux soumissions présentant de fortes différences avec celles de type DBSCAN, en l'occurence les soumissions des deux premiers gagnants de la compétition : *icecuber* (avec un score de 0.92) et *outrunner* (avec un score de 0.90), qui ont dévoilé les techniques utilisées.

3.2.1 *icecuber*

Les techniques utilisées sont d'ordre géométrique et statistique :

1. construction heuristique de paires de *hits* puis de triplets
2. extension en traces complètes par *helix fitting*
3. ajout de *hits* voisins puis assignation finale heuristique des *hits* aux traces, et résolution des conflits

Contrairement aux soumissions de type DBSCAN (voir fig. 3.3 col. b), beaucoup moins de traces proposées ont un poids faible, et pour $W > 0.00007$, plus de 99% des traces sont classées comme bonnes (fig. 3.10).

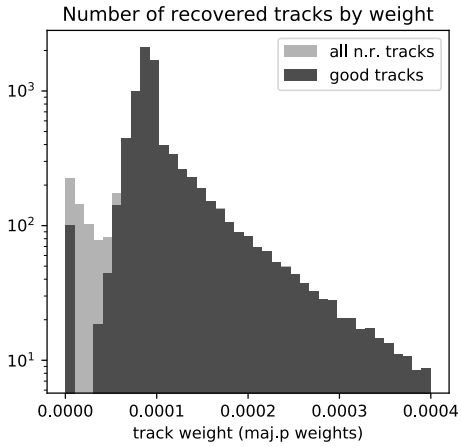


FIGURE 3.8 – Distribution du nombre de traces proposées / bonnes traces, selon le poids W de la trace.

L'efficacité sur les particules primaires approche 1 (et atteint 0.998 pour les particules primaires les plus faciles : $z_0 < 8\,\text{mm}$, $p_\perp > 1\,\text{GeV}$, $n_{\text{particule hits}} \geq 12$) et, pour les particules telles que $n_{\text{particule hits}} \geq 4$, est presque indépendante de tous les paramètres physiques (voir fig. 3.9), sauf pour les particules d'impulsion transverse entre 100 et 200 MeV, dont les trajectoires sont très courbées et courtes, donc difficiles à reconstruire (et dont le poids est faible).

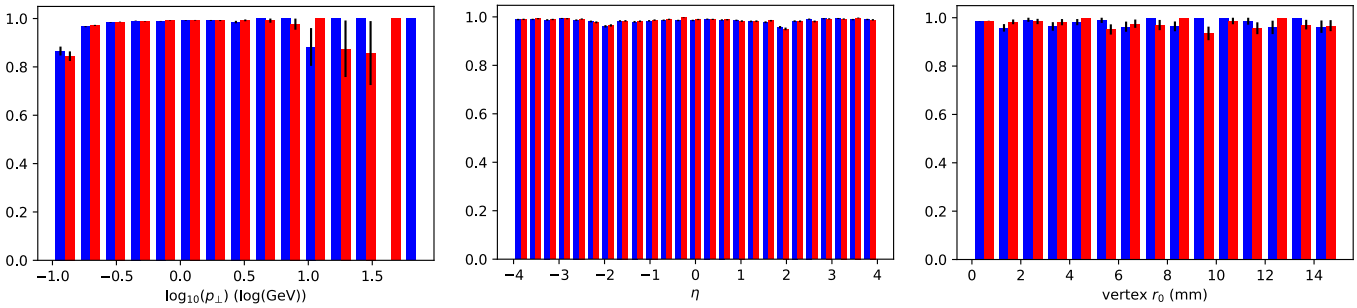


FIGURE 3.9 – Efficacité pour les particules primaires telles que $n_{\text{particule hits}} \geq 4$ selon $p_\perp$, η et r_0.

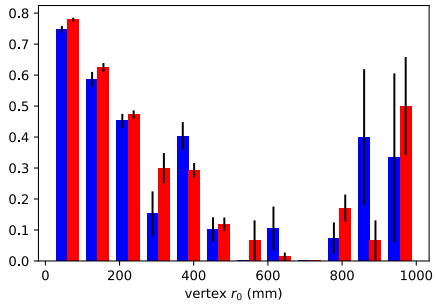


FIGURE 3.10 – Efficacité pour les particules secondaires telles que $n_{\text{particule hits}} \geq 4$ selon l'origine r_0.

De plus, l'efficacité pour les particules secondaires est très bonne (0.28 en moyenne pour tous les secondaires, et 0.63 pour les particules secondaires telles que $n_{\text{particule hits}} \geq 4$) par rapport aux autres soumissions du top 10 (aux alentours de 0.1 et 0.3 respectivement) et en connaissant la difficulté de la reconstruction des secondaires (traces courtes, décentrées...). Elle décroit avec la distance au centre.

Enfin, si les distributions spatiales des *hits* sont assez typiques, elles montrent un point améliorable : réduction de mauvaises associations dans les parties externes des détecteurs ($R > 700\,\text{mm}$, là où les détecteurs sont moins précis). Toutefois, la sensibilité au bruit est faible par rapport aux autres soumissions, et est similaire au #3 qui utilise des techniques proches.

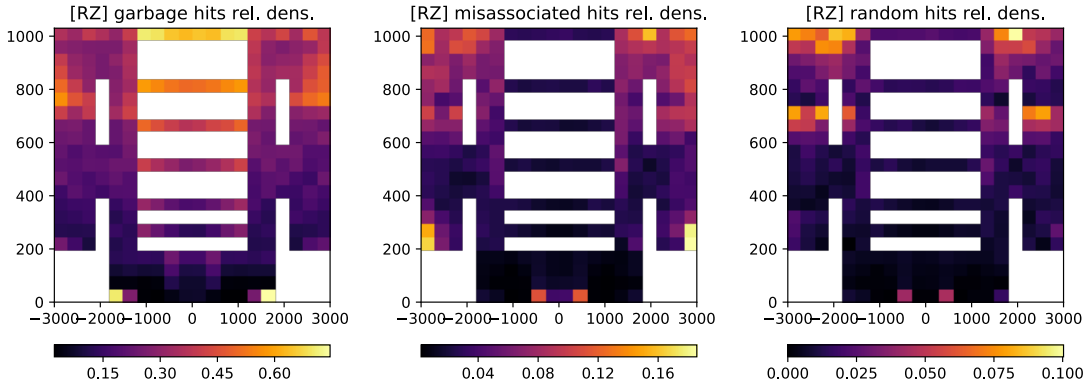


FIGURE 3.11 – Densité spatiale relative des *garbage hits*, *misassociated hits* et *random hits*.

3.2.2 *outrunner*

outrunner utilise un classifieur binaire en *deep learning* (une forme d'apprentissage supervisé) : pour chaque paire de *hits*, le classifieur entrainé indique si les deux hits appartiennent à la même trace ou non. À partir ce cela, les traces sont reconstruites. L'utilisation de *paires* de *hits* entraine un artefact bien visible : les traces de nombre de *hits* impaires sont largement favorisées (fig. 3.12, comparer à 2.12), sans que cela ne semble impacter la fraction de bonnes traces.

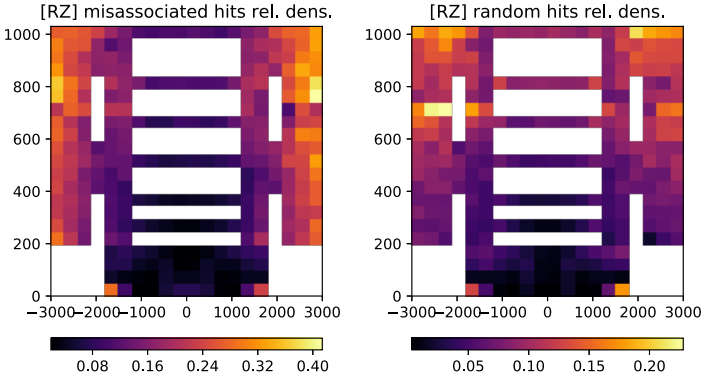


FIGURE 3.13 – Densité spatiale des *misassociated hits* et des *random hits*.

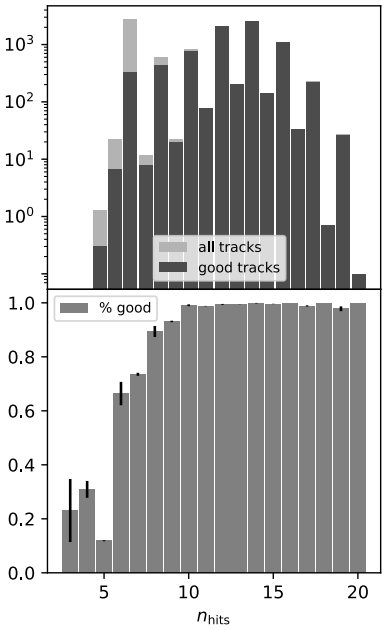


FIGURE 3.12 – En haut, distribution du nombre de traces proposées et de bonnes traces, selon le nombre n_{hits} de *hits* de la trace. En bas, fraction de bonnes traces selon n_{hits} (■ = ■/■).

On remarque que la densité de *misassociated hits* est forte par rapport aux autres soumissions (jusqu'à 0.40, contre 0.20 pour *icecuber*, voir fig. 3.13) et que leur distribution est différente (densité plus grande vers les grands Z), et que la sensibilité au bruit est assez grande.

Les efficacités sur les particules primaires sont similaires aux autres soumissions de score élevé, et l'efficacité sur les secondaire est proche de celle de *icecuber*, avec toutefois une efficacité plus faible (de ∼30%) sur les secondaires de charge négative (comme pour les soumissions de score plus faible).

3.3 Analyse globale des soumissions

Pour faciliter la comparaison des soumissions d'un lot, on choisit judicieusement et on calcule, pour chaque soumission, une soixantaine d'indicateurs (que l'on ne présentera pas ici) à partir des diverses distributions déjà présentées : des moyennes sur un certain intervalle, des maximums/minimums... À partir de ces indicateurs, on génère des figures du type fig. 3.5, 3.4 et B.2. Mais pour chercher les algorithmes "sortant du lot", il existe une technique statistique permettant de générer des inducateurs représentatifs : l'*analyse en composantes principales* [2]. C'est une transformation linéaire des indicateurs cités ci-dessus, en un ensemble d'indicateurs *décorrélés* entre eux.

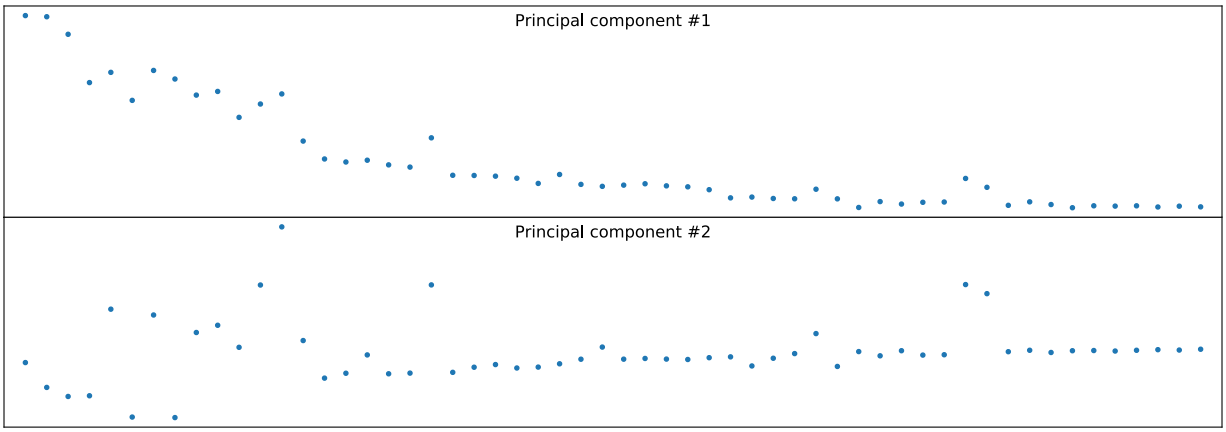


FIGURE 3.14 – 1ère et 2nde composantes principales du lot de soumissions de la section 3.1. Unités arbitraires.

La figure 3.14 représente les deux premières composantes principales. La première composante, qui est l'axe sur lequel la variance est la plus grande, ressemble fortement au score des soumissions. Outre le fait qu'elle montre que le score est un bon indicateur pour différencier les algorithmes, elle n'a pas d'intérêt particulier ici. La deuxième composante est, par définition, orthogonale à la première (décorrélée). On voit ainsi les soumissions qui "sortent du lot" et celles qui sont "dans la moyenne".

3.4 Retours de fin de compétition

J'aurais pu pousser plus loin les interprétations des très nombreuses statistiques produites, mais ce n'est ni l'objet de ce stage (qui était de développer les outils), ni pertinent. En effet, on ne peut rien affirmer de certain à partir de celles-ci sur les techniques utilisées, et la plupart des compétiteurs en tête de classement (et bien d'autres) ont dévoilés leurs techniques[13]. Elles sont bien plus diversifiées et intéressantes que je ne le pensais ; aussi décrivons-les succinctement :

- algorithme combinatoire/statistique pour les #1 et #3 : détermination heuristique rapide de bouts de traces, et assemblage combinatoire sur les bouts de traces (bien plus rapide que sur l'ensemble des *hits* lui-même) (voir §3.2.1, [3] et [4])
- #2 : utilisation du *deep learning* en classification binaire de paire de *hits* (voir §3.2.2 et [5])
- #9 : ensemble d'exécutions de DBSCAN (*clustering*) dans des espaces de coordonnées différentes, puis fusion simple des traces [6]
- #7 : ensemble d'exécutions d'un algorithme rapide de *clustering* basé sur un regroupement discret des *hits* (*sparse binning*), dans des espaces de coordonnées différentes, puis fusion et extension des traces par apprentissage supervisé [7]
- basé sur DBSCAN pour #12, #16, #19, #17, #54, avec pré et post-traitement
- probablement basé sur un algorithme de clustering (DBSCAN ou non) pour de nombreux autres
- pour certains, apprentissage supervisé (*deep learning* ou non) avec tous les *hits* en entrée et toutes les traces en sortie (c'est à dire exécution unique)

Notons enfin que la technique actuellement utilisée à ATLAS (filtres de Kalman [8]), complexe à mettre en place mais déjà connue, n'a pas été utilisée dans le top 3, et probablement pas dans le top 10.

4 Conclusion

Les outils développés durant ce stage se sont révélés utiles durant la première phase de la compétition ; j'espère qu'ils le seront aussi durant la seconde phase, ainsi qu'aux participants. La compétition elle-même montre qu'une variété d'autres algorithmes de reconstruction peuvent être développés et rivaliser avec l'algorithme actuel. En particulier, *icecuber* semble déjà très bien placé pour la seconde phase. Même si les solutions basées sur le *machine learning* se sont montrées difficiles à mettre en place dans cette compétition, les techniques d'apprentissage artificiel seront, et sont déjà, d'une grande importance pour traiter les quantités de données toujours grandissantes produites par les expériences de physique, particulièrement en physique des hautes énergies [9].

Ce travail, sortant un peu de l'ordinaire en physique, a été enrichissant. Il m'a permis une plus grande familiarisation avec Python, les bibliothèques `matplotlib` et `pandas`, et l'interface Python-C++.

J'ai découvert la physique des hautes énergies et ses méthodes de recherche et de travail, autant au LAL que durant la courte et passionnante visite au CERN. La grande variété de métiers œuvrant pour un but commun rend très intéressant ce domaine. De plus, les interactions avec les membres de l'équipe TrackML et du groupe ATLAS-LAL m'ont beaucoup apportées. Enfin, la compétition, ainsi que ma participation à un *bootcamp* TrackML à Paris, m'ont fait rencontrer la communauté du *machine learning*, sujet qui m'intéressait déjà auparavant.

13. Voir le forum Kaggle : https://www.kaggle.com/c/trackml-particle-identification/discussion

Références

[1] The TrackML Team. *Participant document : Particle Tracking and the TrackML Challenge*. 2018. URL : https://kaggle2.blob.core.windows.net/forum-message-attachments/321278/9331/trackml-participant-document-particle-v1.0.pdf.

[2] Wikipedia. *Principal component analysis*. URL : https://en.wikipedia.org/wiki/Principal_component_analysis.

[3] Johan Sokrates Wind. *Fast and scalable tracking by outlier density estimation*. URL : https://github.com/top-quarks/top-quarks/blob/master/top-quarks_documentation.pdf.

[4] Sergey Gorbunov. *Fast combinatorial search algorithm for the TrackML Particle Tracking Challenge*. URL : https://github.com/sgorbuno/TrackML_CombinatorialTracker/blob/master/doc/TrackML_AlgorithmDescription.pdf.

[5] URL : https://www.kaggle.com/c/trackml-particle-identification/discussion/63256.

[6] Jean-François Puget. *Pushing DBSCAN To Its Limit*. URL : https://github.com/jfpuget/Kaggle_TrackML/blob/master/doc/trackML.pdf.

[7] Yuval Reina et Trian Xylouris. *Ultrafast sparse binning clustering enhanced by ML track scoring*. URL : https://github.com/tx1985/kaggle-trackML/blob/master/trackML_solution.pdf.

[8] Saša Fratina. *Using Kalman Filter to Track Particles*. 2004. URL : http://www-f9.ijs.si/~cindro/seminars/kalman.pdf.

[9] « Machine Learning in High Energy Physics Community White Paper ». In : *ArXiv* (juil. 2018). arXiv : 1807.02876 [physics.comp-ph].

Annexe

A Soumissions : figures supplémentaires

A.1 Soumissions de type DBSCAN

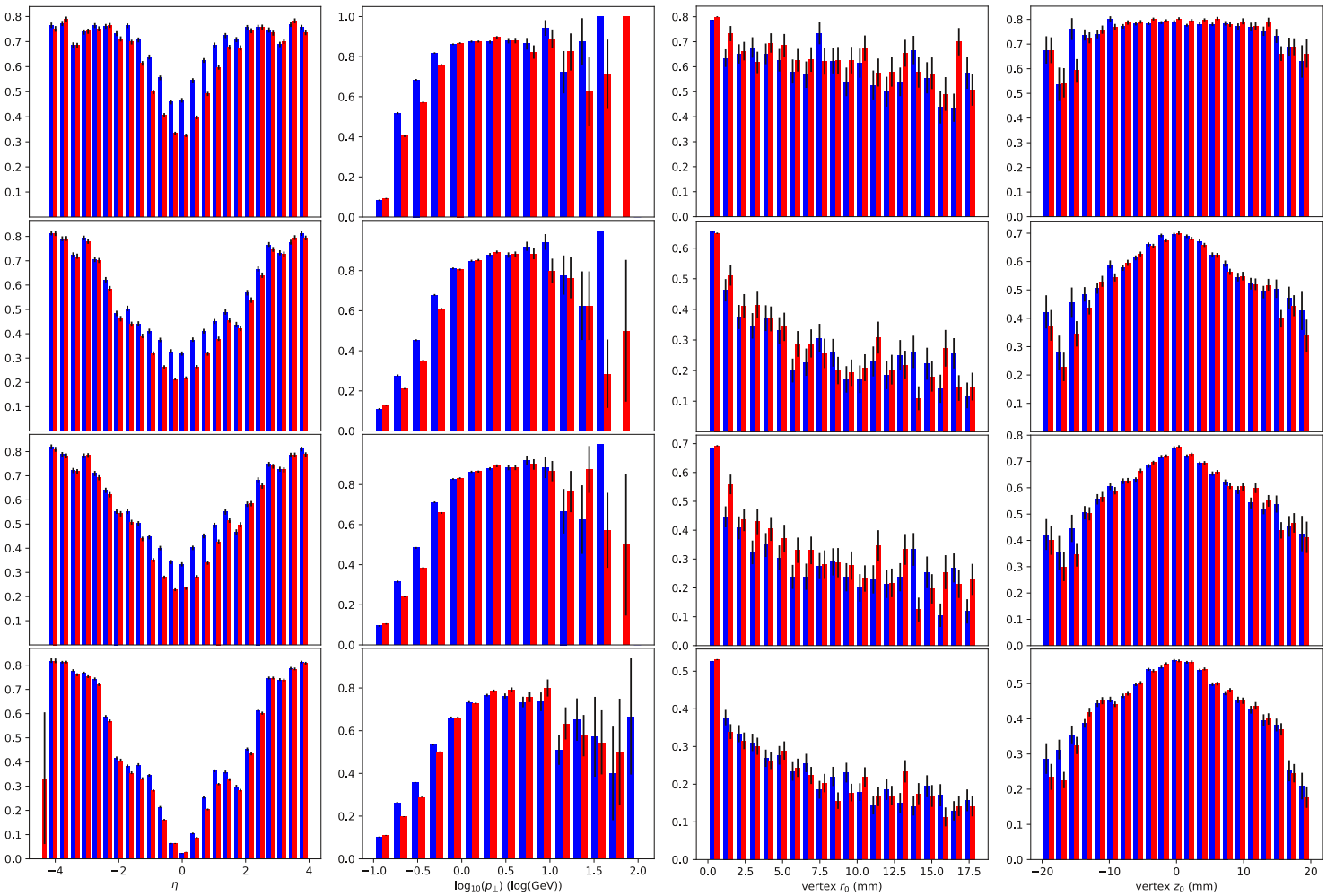


FIGURE A.1 – Histogrammes d'efficacité totale (primaires + secondaires) des soumissions de type DBSCAN analysées au §3.1, suivant η, $p_\perp$, r_0 (proche) et z_0. Soumissions de haut en bas : 0.70, 0.56, 0.59 ($\star$), 0.46 ($\star$).

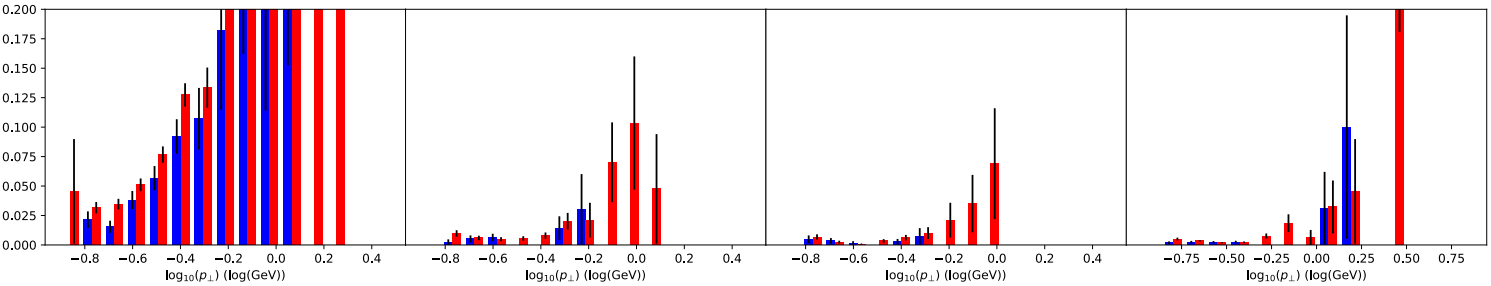


FIGURE A.2 – Histogrammes d'efficacité des soumissions de type DBSCAN analysées au §3.1, suivant $p_\perp$, pour les particules secondaires ayant $n_{\text{particle hits}} \geq 4$. Soumissions de gauche à droite : 0.70, 0.59 ($\star$), 0.56, 0.46 ($\star$).

A.2 Autres soumissions intéressantes

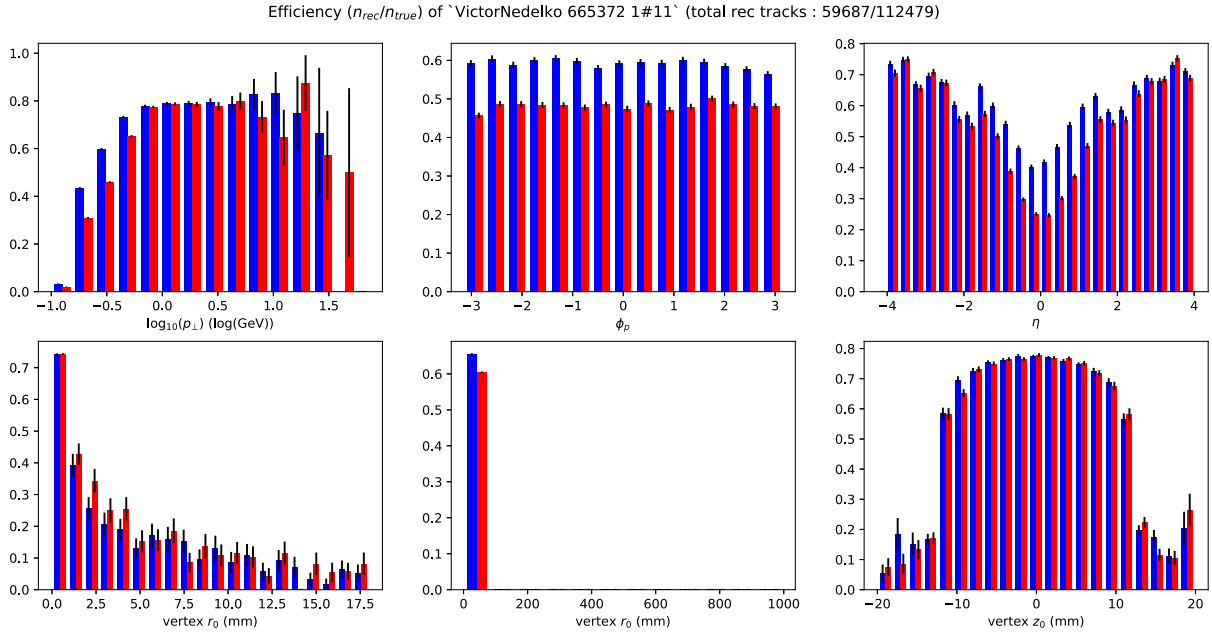


FIGURE A.3 – Efficacité sur toutes les particules selon $p_\perp$, ϕ, η, r_0 et z_0 de l'algorithme de Victor Nedelko.

A.3 Soumissions du top 3

On présente ici quelques figures des trois premières soumissions de la compétition : *icecuber*, *outrunner* et Sergey Gorbunov. Les figures ont ici toutes les mêmes échelles verticales.

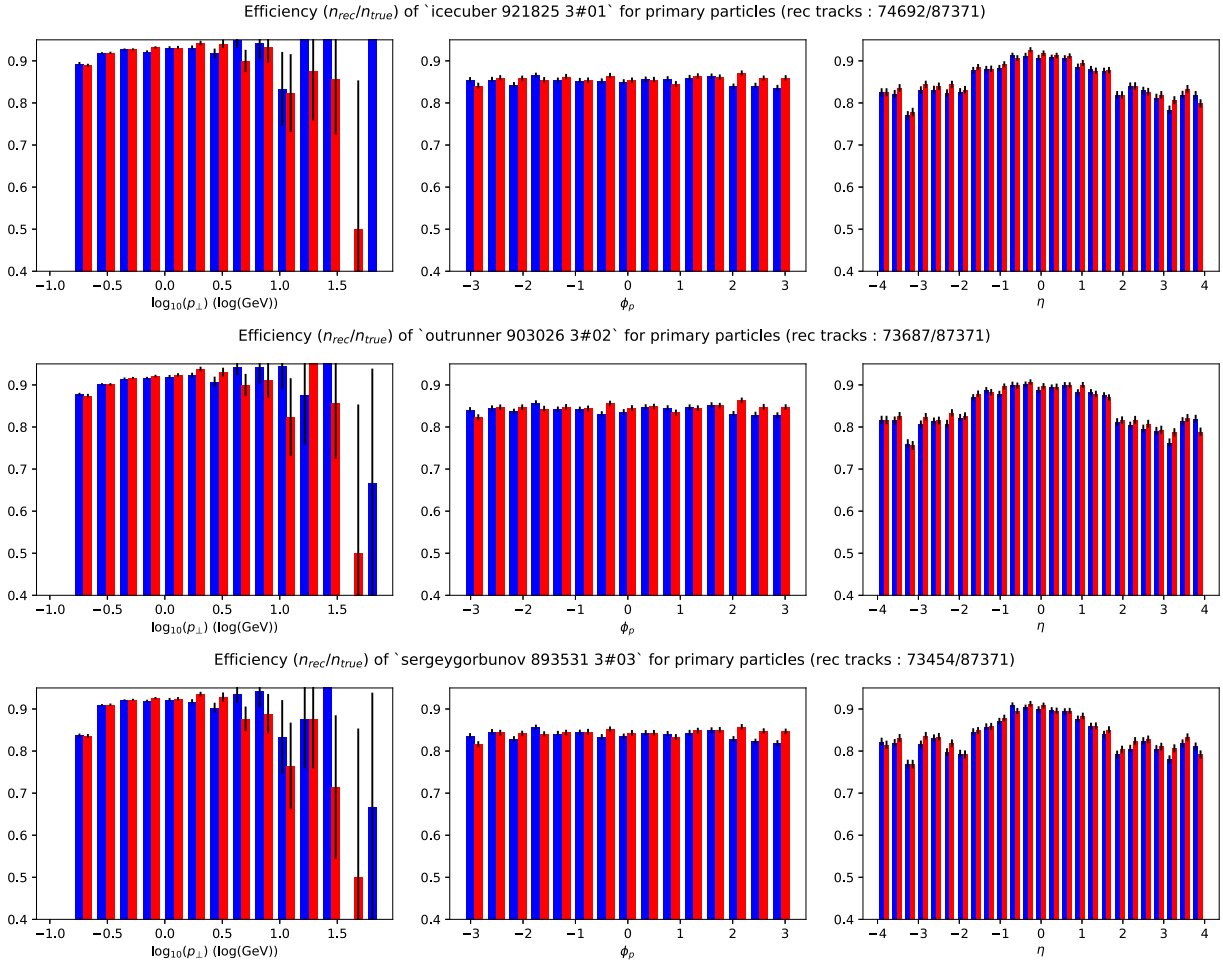


FIGURE A.4 – Efficacité sur les particules primaires selon $p_\perp$, ϕ et η.

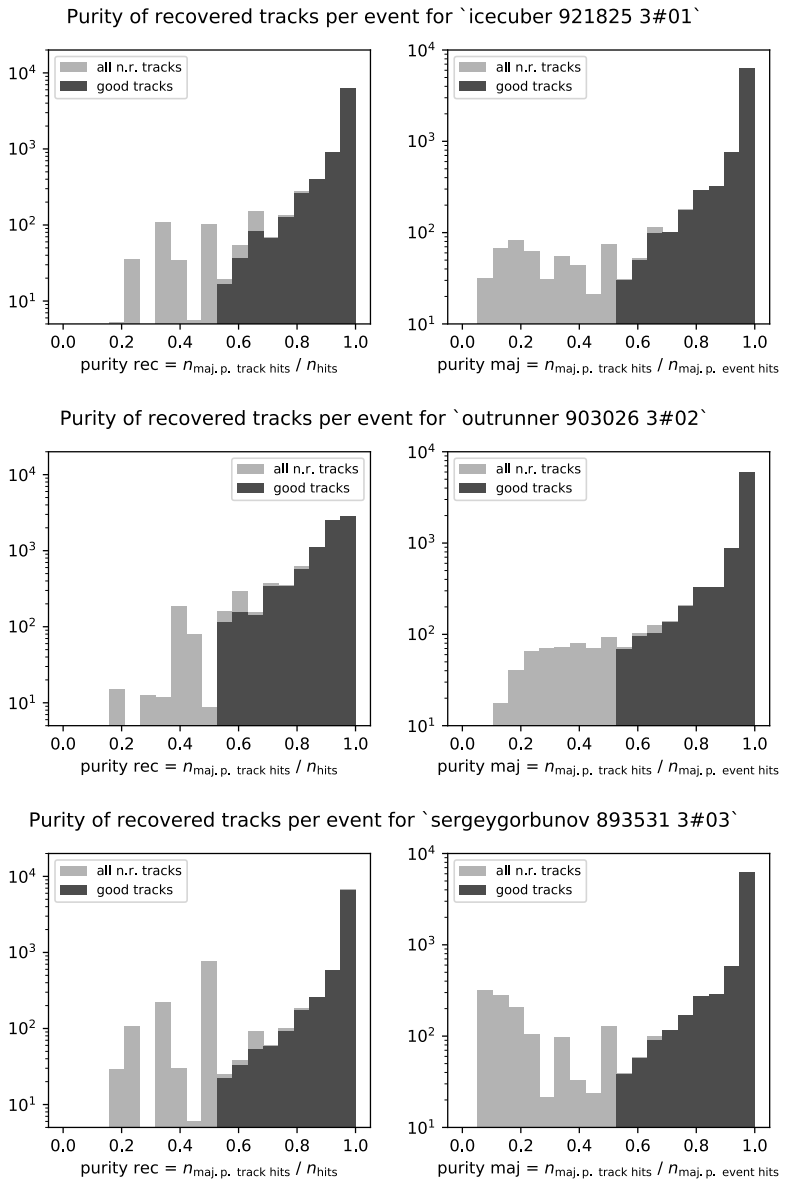


FIGURE A.5 – Nombre de traces reconstruires / bonnes traces en fonction de purity_rec et purity_maj.

B Analyses globales : figures supplémentaires

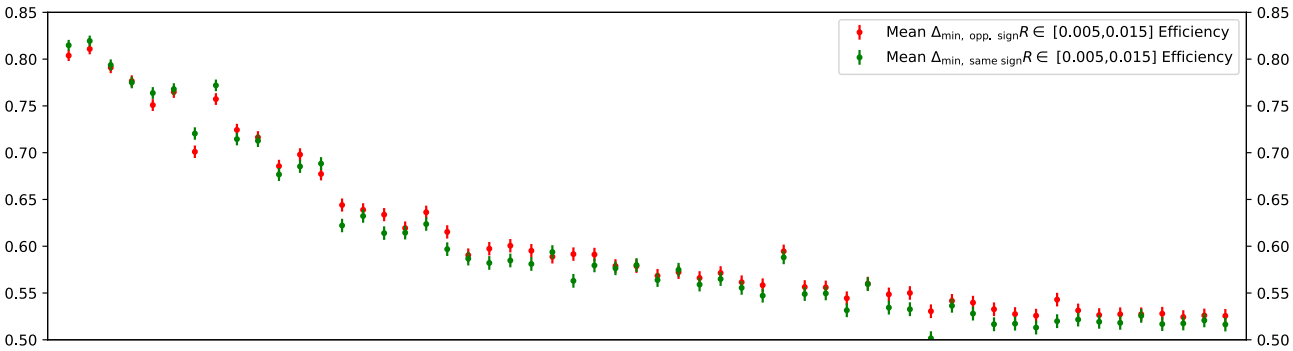


FIGURE B.1 – Efficacité sur les particules primaires en $\Delta_{\text{min,same}}R \simeq 0.01$ (en vert) et en $\Delta_{\text{min,opp.}}R \simeq 0.01$ (en rouge), sur le lot de la section 3.1. On remarque, sur les soumissions de meilleur score, un efficacité plus grande à même signe qu'à signe opposé. C'est l'inverse de ce qui est décrit au §2.7, sans que l'on ait d'explications pour le moment.

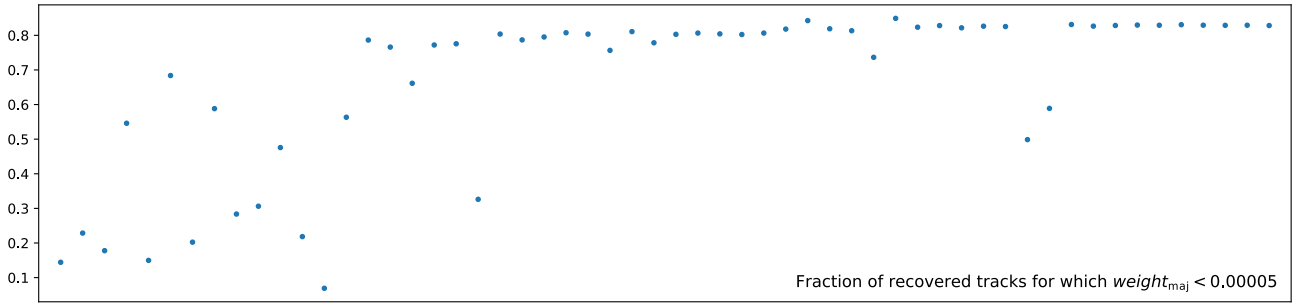


FIGURE B.2 – Fraction des traces proposées telles que le poids W des *hits* de la particule majoritaire soit inférieur à 0.00005 (voir fig. 3.3 col. b). On remarque que le lot se compose majoritairement de soumissions ayant $\sim$80% de traces pour lesquelles $W < 0.00005$ (surtout des soumissions de type DBSCAN).