



# Applying BLAS-Based Graph Algorithms to GPGPU-based Track Reconstruction

Miloš Barjaktarović

**Supervisors:** Stephen Nicholas Swatman, Alexander  
J. Pflieger

Summer, 2025

### **Abstract**

Reconstructing charged particle trajectories is an important step in identifying and characterizing particles produced in collisions. The process starts by detector-identified space points and includes filtering out more likely out of a big number of possibilities. In order to improve the efficiency, this project explores the ways to tackle the problem, so it is appropriate for execution on GPGPUs. We designed an algorithm based on GraphBLAS, that operates over two different semi-rings. It is tested on one dataset and the result aligns with the expectations.

**Keywords:** GraphBLAS, semi-ring, track, sparse adjacency matrix

# Contents

|          |                                       |           |
|----------|---------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                   | <b>3</b>  |
| <b>2</b> | <b>GraphBLAS and Semi-Rings</b>       | <b>3</b>  |
| 2.1      | Semi-ring $\min . +$ . . . . .        | 4         |
| 2.2      | Semi-ring $\max . +$ . . . . .        | 5         |
| 2.3      | Semi-ring $\max . \times$ . . . . .   | 6         |
| 2.4      | Semi-ring $\max . \min$ . . . . .     | 7         |
| 2.5      | Semi-ring $\vee . \wedge$ . . . . .   | 8         |
| 2.6      | Discussion . . . . .                  | 11        |
| <b>3</b> | <b>Data Overview</b>                  | <b>11</b> |
| <b>4</b> | <b>Algorithm</b>                      | <b>14</b> |
| 4.1      | Weakly Connected Components . . . . . | 14        |
| 4.2      | Heaviest path . . . . .               | 15        |
| 4.3      | Implementation . . . . .              | 15        |
| 4.4      | Results . . . . .                     | 16        |
| <b>5</b> | <b>Conclusion</b>                     | <b>16</b> |
| <b>A</b> | <b>Code Availability</b>              | <b>16</b> |

# 1 Introduction

ATLAS is a general-purpose detector at the Large Hadron Collider (LHC) which investigates various physics phenomena. Beams of particles from the LHC collide at the center of the ATLAS detector making collision debris in the form of new particles, which fly out from the collision point in all directions. Six different detecting subsystems arranged in layers around the collision point record the paths, momentum, and energy of the particles, allowing them to be individually identified.

It is the main objective of this project to recover trajectories from the sets of SPs and seeds, while using the language of graphs and utilizing the GraphBLAS concepts. Taking this approach, working with graphs represented as sparse matrices, rather than conventional data structures, allows for a high degree of parallelism and efficient execution of large-scale matrix operations on GPGPUs.

## 2 GraphBLAS and Semi-Rings

The GraphBLAS defines a set of mathematical operations that are useful for implementing a wide range of graph operations. At the heart of the GraphBLAS are matrices as mathematical objects, which are usually sparse. This means that most of the matrix elements are zero. These zeros are not stored in order to make the implementation more efficient. [3] The sparse matrix format used for this project is Compressed Sparse Row (CSR). [4] Additional information saved in my CSR data structure is the “shape” of the matrix, i.e. its dimensions  $m \times n$ . While the number of rows is readily accessible from the standard CSR format, the number of columns can be ambiguous. Although for this project we only work with square adjacency matrices that represent graphs, this addition was implemented for its generality.

The core idea of GraphBLAS is the following – operations with matrices over different semi-rings imply different transformations to the underlying graphs. Let’s observe a set  $S$  and two operations  $\oplus$  (addition) and  $\otimes$  (multiplication) over the set  $S$ . Then, the ordered triplet  $(S, \oplus, \otimes)$  is a suitable GraphBLAS semi-ring if it satisfies the following properties. [3] For each  $a, b, c \in S$ :

- i)  $a \oplus b = b \oplus a$
- ii)  $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
- iii)  $\exists \hat{0} \in S : a \oplus \hat{0} = a$  (existence of additive identity)
- iv)  $a \otimes b = b \otimes a$
- v)  $a \otimes (b \otimes c) = (a \otimes b) \otimes c$
- vi)  $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
- vii)  $a \otimes \hat{0} = \hat{0}$  (additive identity is multiplicative annihilator)

Let now  $A, B \in S^{m \times n}$  and we consider the induced addition of matrices, defined as:

$$C = A \oplus B, \quad C = (c_{ij})_{i=\overline{1,m}, j=\overline{1,n}}, \quad c_{ij} = a_{ij} \oplus b_{ij}.$$

We could also define element-wise (Hadamard) multiplication  $\otimes$ , so that the ordered triplet  $(S^{m \times n}, \oplus, \otimes)$  is a semi-ring with inherited all scalar properties i)-vii).

On the other hand, let  $A \in S^{m \times n}$  and  $B \in S^{n \times k}$ , and we define matrix multiplication as:

$$C = A \otimes B, \quad C = (c_{ij})_{i=\overline{1,m}, j=\overline{1,k}}, \quad c_{ij} = \bigoplus_{l=1}^n a_{il} \otimes b_{lj}.$$

In this project, we used only this multiplication definition. While matrix multiplication commutativity is lost, this definition is more susceptible to nicer transformations of the underlying graph. Furthermore, the main matrix operation used in the project is the power of a matrix. Since this matrix multiplication is still associative, the power of a matrix  $A^k = \underbrace{A \otimes A \otimes \dots \otimes A}_k$ ,  $k \in \mathbb{N}$ , is

unambiguously defined. For the first time, let's delve into adjacency matrix operations and their connection with the underlying graph. Let  $G = (V, E, w)$  be a directed acyclic weighted graph with  $|V| = n$  and  $A \in S^{n \times n}$  its adjacency matrix, then

$$A^k = \left( a_{ij}^{(k)} \right)_{i,j=\overline{1,n}}, \quad a_{ij}^{(k)} = \bigoplus_{\substack{i \rightarrow j \\ k\text{-paths}}} \bigotimes_{e \in i \rightarrow j} w(e), \quad i, j = \overline{1,n}, \quad (1)$$

where  $i \rightarrow j$  is a path from node  $i$  to node  $j$ ,  $k$ -paths stands for paths in the graph  $G$  of length  $k$  and  $e$  stands for edge of the graph.

**Note 1** *In general, if  $G$  is not an acyclic graph, the  $k$ -paths in (1) should be changed to  $k$ -walks. However, the nature of the problem we deal with in this project imposes that the graph is acyclic. Namely, as will be demonstrated later in this paper, our graph and its directed edges represent the trajectories of charged particles, which don't go "backwards" and therefore don't form a cycle.*

Without further ado, let's explore some concrete realizations of semi-rings and concepts tackled above.

## 2.1 Semi-ring $\min. +$

Let's observe the ordered triplet  $(\mathbb{R} \cup \{+\infty\}, \min, +)$ . It is an easy exercise to convince oneself that this triplet is a semi-ring that satisfies all properties i)-vii) given in the beginning of this section, with  $+\infty$  being the additive identity. With the same assumptions,  $A \in (\mathbb{R} \cup \{+\infty\})^{n \times n}$  being an adjacency matrix

over  $\min.+$  semi-ring and  $A^k = \left(a_{ij}^{(k)}\right)_{i,j=\overline{1,n}}$ ,  $k \in \mathbb{N}$ , the expression (1) then becomes:

$$a_{ij}^{(k)} = \min_{\substack{i \rightarrow j \\ k\text{-paths}}} \sum_{e \in i \rightarrow j} w(e), \quad i, j = \overline{1, n}, \quad k \in \mathbb{N}. \quad (2)$$

Since for  $k \in \mathbb{N}$ ,  $A^k \in (\mathbb{R} \cup \{+\infty\})^{n \times n}$  (in general, for  $A \in S^{m \times n}$ ,  $A^k \in S^{n \times n}$ , for  $k \in \mathbb{N}$ ),  $A^k$  is an adjacency matrix of some graph. Let's demonstrate this by example.

**Example 1** Let  $G$  be the graph as in Figure 1 and  $A$  its adjacency matrix.  $G$  is obviously an acyclic graph. We want to calculate  $A^3$ . Due to the expression (2), we look for the paths of length 3. Notice that there are two disjoint  $A_1 - A_5$  paths in graph  $G$ , with accumulative weights of 0.9 and 0.4. Since the addition in this semi-ring is  $\min$ , that means that in the graph corresponding to  $A^3$  there exists the edge  $(A_1, A_5)$  with weight of 0.4. The similar situation is with nodes  $A_2$  and  $A_6$ .

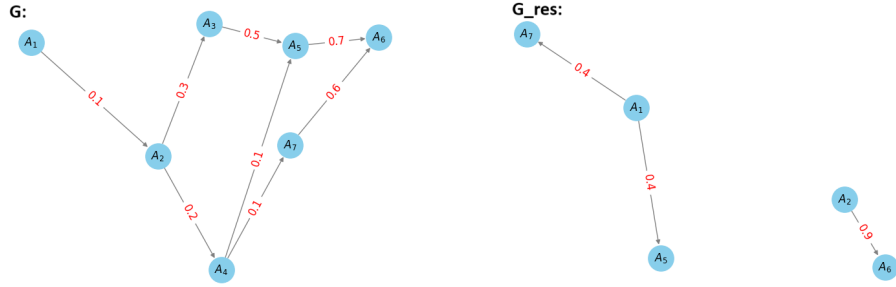


Figure 1:  $G_{\text{res}}$  is a graph corresponding to the adjacency matrix  $A^3$ , whereas  $A$  is the adjacency matrix of the graph  $G$  over  $\min.+$  semi-ring.

## 2.2 Semi-ring $\max.+$

We refer to  $(\mathbb{R} \cup \{-\infty\}, \max, +)$  as the  $\max.+$  (or tropical-max) semi-ring. The set of real numbers is extended with  $-\infty$  because it functions as an additive identity in this semi-ring. The expression (1) for the power of adjacency matrix  $A$  over this semi-ring becomes the following:

$$a_{ij}^{(k)} = \max_{\substack{i \rightarrow j \\ k\text{-paths}}} \sum_{e \in i \rightarrow j} w(e), \quad i, j = \overline{1, n}, \quad k \in \mathbb{N}. \quad (3)$$

**Example 2** Let's observe the same graph as in Example 1. There are two  $A_1 - A_5$  paths in graph  $G$  with accumulative weights of 0.4 and 0.9. Since adjacency matrix  $A$  in this example is over  $\max.+$  semi-ring, this means that  $a_{15}^{(3)} = 0.9$  i.e. the graph corresponding to adjacency matrix  $A^3$  contains the edge  $(A_1, A_5)$  with weight 0.9.

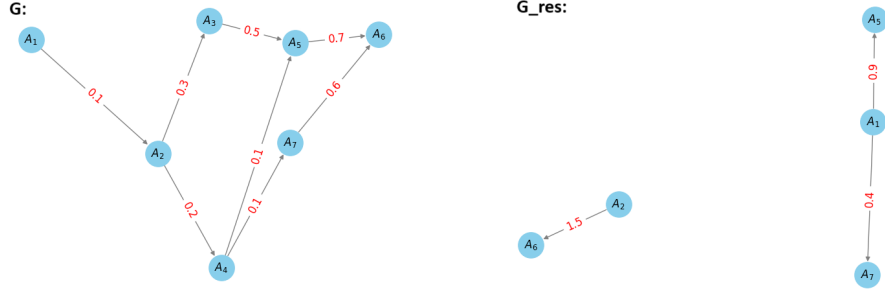


Figure 2:  $G_{res}$  is a graph corresponding to the adjacency matrix  $A^3$ , whereas  $A$  is the adjacency matrix of the graph  $G$  over  $\max. +$  semi-ring.

Let's say we want to find the heaviest path of arbitrary length  $k$ . Due to (3), it is enough to find the biggest element of matrix  $A^k$ . Furthermore, let's again consider the Example 2 with the aim of finding the heaviest path in the graph  $G$  (path that cumulatively has the biggest weight of its edges). It is easy to notice that the biggest path in graph  $G$  has length 4, implying – the adjacency matrix  $A$  of graph  $G$  satisfies that  $A^k$  corresponds to an empty graph<sup>1</sup> for  $k \geq 5$ . Considering everything explained so far, the problem of finding overall heaviest path reduces to finding maximum over the biggest edge in each of the graphs  $G^k$ ,  $k = \overline{1, 4}$  (which corresponds to the heaviest path of length  $k$ ), or

$$\max_k \{ \max w(E(G^k)) \}. \quad (4)$$

Looking at the Figure 3, we can conclude that the heaviest path in graph  $G$  is of the length 1.6 and it is  $A_1 - A_6$  4-path<sup>2</sup>.

### 2.3 Semi-ring $\max. \times$

Another variation of the previous semi-ring is the  $\max. \times$  semi-ring or the triplet  $(\mathbb{R}_{\geq 0}, \max, \times)$ , where  $\times$  stands for standard multiplication of real numbers. Over this semi-ring, the expression (1) becomes

$$a_{ij}^{(k)} = \max_{i \rightarrow j} \prod_{k\text{-paths } e \in i \rightarrow j} w(e), \quad i, j = \overline{1, n}, \quad k \in \mathbb{N}. \quad (5)$$

Figure 4 demonstrates how the resulting graph  $G_{res}$  differs over this semi-ring. However, this semi-ring shows no practical relevance to this project.

<sup>1</sup>A graph is said to be empty if the set of its edges is an empty set.

<sup>2</sup>4-path refers to the path in a graph that consists of 4 edges.

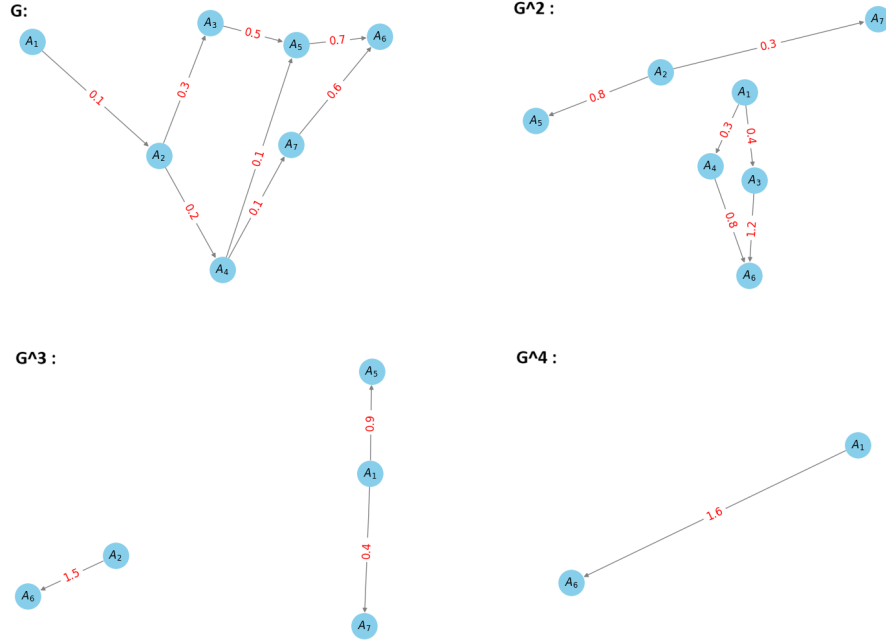


Figure 3: Graphs corresponding to iterative powers of adjacency matrix  $A$ .

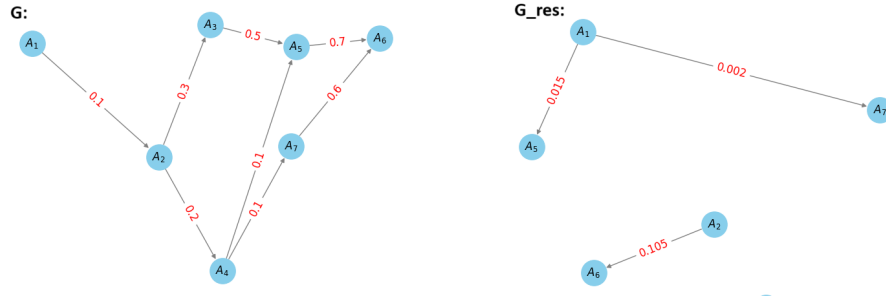


Figure 4:  $G_{\text{res}}$  is the graph corresponding to the adjacency matrix  $A^3$ , whereas  $A$  is the adjacency matrix of the graph  $G$  over  $\max.\times$  semi-ring.

## 2.4 Semi-ring $\max.\min$

The ordered triplet  $(\mathbb{R}_{\geq 0}, \max, \min)$  is called  $\max.\min$  semi-ring. Over this semi-ring, the general expression (1) becomes:

$$a_{ij}^{(k)} = \max_{i \rightarrow j} \min_{\substack{e \in i \rightarrow j \\ k\text{-paths}}} w(e), \quad i, j = \overline{1, n}, \quad k \in \mathbb{N}.$$

While it is interesting to observe some other behavior of this semi-ring, such as in (4), some practical usage within this project was not found.

## 2.5 Semi-ring $\vee, \wedge$

The last semi-ring we will discuss in this section is the Boolean ( $\vee, \wedge$  or, or.and) semi-ring  $(\{0, 1\}, \vee, \wedge)$ . This semi-ring obviously satisfies all properties given by i)-vii) in the beginning of this section, with 0 being the additive identity. The adjacency matrix over Boolean semi-ring is always a representation for an unweighted (directed) graph. The expression (1) over this semi-ring becomes:

$$a_{ij}^{(k)} = \begin{cases} 1, & \text{if there exists a } k\text{-path from } i \text{ to } j, \\ 0, & \text{elsewhere;} \end{cases} \quad i, j = \overline{1, n}, \quad k \in \mathbb{N}. \quad (6)$$

As this semi-ring completely emphasizes the existence of an edge, it is very helpful for tackling graph operations where this is the main interest, such as graph connectivity, neighborhoods, induced subgraphs, matchings, etc.

**Example 3** *Let us consider the graph  $G$  given in the chart below and its adjacency matrix  $A$  over the Boolean semi-ring. In this example, we aim to obtain the graph  $G[A_1]$  i.e. the subgraph of graph  $G$  induced by the node<sup>3</sup>  $A_1$ . Given in the table are graph  $G$ , matrix  $A$ , as well as graphs corresponding to  $A^2$ ,  $A^3$  and  $A^4$ , respectively. The so-called reachability matrix  $R$  is given by*

$$R = A \oplus A^2 \oplus \dots \oplus A^k \oplus \dots$$

where  $\oplus$  is element-wise matrix addition induced by scalar  $\vee$  addition. By observing that for  $k \geq 5$ ,  $A^k$  is a zero-matrix, the above formula reduces to

$$R = A \oplus A^2 \oplus A^3 \oplus A^4.$$

Calculating the matrix  $R$ , we obtain

$$R = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

The matrix  $R = (r_{ij})_{i,j=\overline{1,9}}$  contains the following information:

$$r_{ij} = 1 \Leftrightarrow \exists i \rightarrow j \text{ path in graph } G.$$

---

<sup>3</sup>Subgraph  $G[v]$  of a graph  $G$  consists of all nodes of graph  $G$  reachable from the node  $v$  and edges that produce this reachability.

*In other words,  $r_{ij} = 1$  iff node  $j$  is reachable from node  $i$  in graph  $G$ , implying that  $j \in V(G[i])$ . Reading from the first row of matrix  $R$ , we conclude that nodes  $A_2, A_3, A_5, A_6, A_7$  and  $A_8$  belong to the  $G[A_1]$ , and we can easily extract the induced subgraph we are looking for.*

| Graph                        | Adjacency matrix                                                                                                                                                                                                                                                  |
|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>G:</b></p>             | $A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & & 1 & & & & & & \\ 0 & & & 1 & & & & & \\ 0 & & 1 & & 1 & & & & \\ 0 & & & & & 1 & & & \\ 0 & & & & & & 1 & & \\ 0 & & & & & & & 1 & \\ 0 & & & & & & & & 1 \\ 1 & & 1 & 1 & & & & & \end{pmatrix}$ |
| <p><b>G<sup>2</sup>:</b></p> | $A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & & & 1 & & & 1 & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & 1 & 1 & & 1 & & 1 & & \end{pmatrix}$       |
| <p><b>G<sup>3</sup>:</b></p> | $A^3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & 1 & & 1 & & 1 & & \end{pmatrix}$                                  |
| <p><b>G<sup>4</sup>:</b></p> | $A^4 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & & & & \\ 0 & & & & & 1 & & 1 & \end{pmatrix}$               |

## 2.6 Discussion

In this report, we presented some semi-rings that could be appropriate for this project's setting. Namely, we work with specific types of graphs, with one of the important properties being the positivity of edge weights. For this reason, while some other semi-rings are rather known, such as  $(\mathbb{R}_{\leq 0}, \min, \max)$  or  $(\mathbb{R}_{\leq 0}, \min, \times)$ , they cannot be adapted for this problem.

Furthermore, while trying to unravel semi-rings with interesting properties, we would usually run into the problem of missing some of the properties given by i)-vii). Motivated by incorporating operation of concatenation in the semi-ring, the following semi-ring was considered. For  $a, b \in S$ ,  $S = \bigcup_{n=0}^{\infty} (\mathbb{R} \times \mathbb{N})^n$ , addition and multiplication are defined as

$$\begin{aligned}
 a &= \{(w_{n_1}^a, k_{n_1}^a), \dots, (w_{n_a}^a, k_{n_a}^a)\}, \quad b = \{(w_{n_1}^b, k_{n_1}^b), \dots, (w_{n_b}^b, k_{n_b}^b)\} : \\
 a \oplus b &= \{(w_{n_c}^c, k_{n_c}^c), \dots, (w_{n_c}^c, k_{n_c}^c)\}, \text{ where} \\
 (w_{n_s}^c, k_{n_s}^c) &= \begin{cases} (w_{n_{s_1}}^a, k_{n_{s_1}}^a + k_{n_{s_2}}^b), & \exists s_1 \in \{1, \dots, n_a\}, s_2 \in \{1, \dots, n_b\} : w_{n_{s_1}}^a = w_{n_{s_2}}^b \\ (w_{n_{s_1}}^a, k_{n_{s_1}}^a), & \nexists s_2 \in \{1, \dots, n_b\} : w_{n_{s_1}}^a = w_{n_{s_2}}^b \\ (w_{n_{s_2}}^b, k_{n_{s_2}}^b), & \nexists s_1 \in \{1, \dots, n_a\} : w_{n_{s_2}}^b = w_{n_{s_1}}^a \end{cases} \\
 \text{and } a \otimes b &= \{(w_{n_d}^d, k_{n_d}^d), \dots, (w_{n_d}^d, k_{n_d}^d)\}, \text{ where} \\
 (w_{n_d}^d, k_{n_d}^d) &= \begin{cases} (w_{n_{s_1}}^a, \min\{k_{n_{s_1}}^a, k_{n_{s_2}}^b\}), & \exists s_1 \in \{1, \dots, n_a\}, s_2 \in \{1, \dots, n_b\} : w_{n_{s_1}}^a = w_{n_{s_2}}^b \\ 0, & \text{elsewhere} \end{cases} .
 \end{aligned}$$

It can be checked that the triplet  $(S, \oplus, \otimes)$  satisfies all properties i)-vii), except for distributivity (vi). It has already been mentioned that the power of adjacency matrix is well defined, because (induced) matrix multiplication is associative. However, in order to guarantee matrix multiplication associativity, we need distributivity in our semi-ring. Consequently, exploration of this semi-ring ceased here with the conclusion that, in order to be correct, the later explained algorithm and approach really demands all listed semi-ring properties.

## 3 Data Overview

This project deals with track reconstruction of charged particles stemmed in beams collisions. After the collision, new particles scatter in different directions. It is known that the track of a charged particle follows a helical trajectory in  $x - y - z$  plane. As a consequence, the trajectory of a particle in transversal  $x - y$  plane follows a circular path, and a straight path in longitudinal  $r - z$  plane, with  $r = \sqrt{x^2 + y^2}$ . [2]

Specifically, the data is generated from the collision of beams of muons and represents the tracks of 4 muons. It is given in "sps", "dub" and "tri" file datasets. The space coordinates of SPs are given in the "sps" file. Their projections on

the transversal and longitudinal plane are given in Figure 5. Note that 4 different straight paths (in longitudinal plane) are easily distinguishable, nodding to the fact that the data has 4 underlying muon tracks.

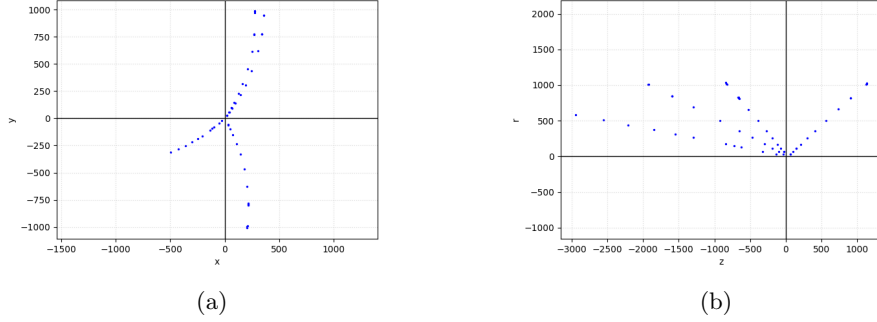


Figure 5: (a) projection of SPs in transversal plane; (b) projection of SPs in longitudinal plane.

Another dataset is “dub” and is given as doublets  $(i, j)$  corresponding to the indices of rows in “sps” dataset, and indicating the possibility that space points with the indices  $i$  and  $j$  belong to the same charged particle trajectory. If we position the SPs as in the longitudinal plane, and draw an arrow  $i \rightarrow j$ , for every  $(i, j)$  in the “dub” dataset, we obtain a plot of a directed unweighted graph, as shown in Figure 6. It is mathematically correct that the datasets “sps” and “dub” define and can be modeled by a directed unweighted graph. In particular, indices of SPs in “sps” represent nodes, doublets in “dub” directed edges, and concrete coordinates now serve to determine the positions of nodes in plots.

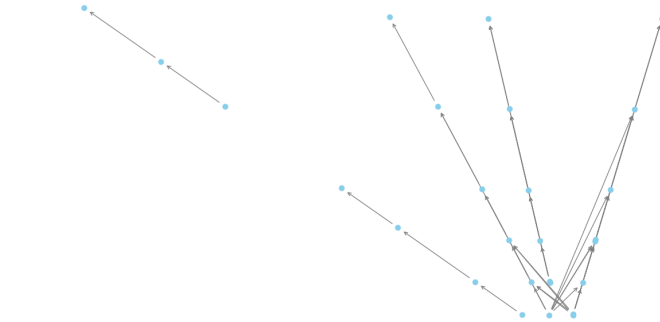


Figure 6: Doublets graph representing collision data

The problem of the project arises at this point – while positions of nodes

clearly form 4 separate trajectories, dataset produced edges between them. The goal becomes filtering out the edges and determining which tracks are real (which edges indicate the real track of a particle). In order to do that, we need to have some kind of measurement unit, since this process is not deterministic - many real-world caveats and technological constraints are encountered. That is where the “tri” dataset comes in.

Considering edges of a graph  $G$  as nodes, and two nodes are connected if the successor of one is the predecessor of other in  $G$ , we end up with line graph  $L(G)$ . Formally speaking, for directed graph  $G = (V, E)$ , line graph  $L(G)$  is a graph such that  $V(L(G)) = E$  and  $(u, v, w) \in E(L(G)) \Leftrightarrow (u, v) \in E \wedge (v, w) \in E$ . This is exactly what is happening in the “tri” dataset – it is the collection of triplets  $(i, j, k)$  that correspond to indices of SPs that form edges in the line graph obtained from the graph illustrated in Figure 6. These triplets coincide with prior mentioned seeds. An additional column in this dataset represent the weights assigned to the edges of the graphs. Weights are probabilities affected by the transversal and longitudinal impact parameters, and/or some other factors. [5]

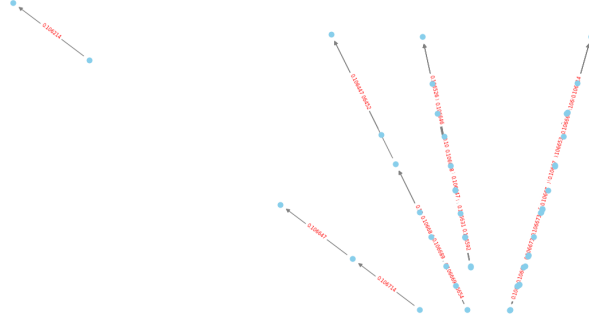


Figure 7: Triplets line graph

This line graph  $L(G)$ , that is, “tri” dataset, is shown in Figure 7. In this graph, the position of the node  $(u, v)$  in the line graph is the midpoint of the line determined by the edge  $(u, v)$  in the graph  $G$ . We can see that in this triplets graph, “straight lines” are still distinguishable and there are no edges between them, which is not surprising. The reason for this is the fact that all such “problematic connections” in the doublets graph stem from the same node, meaning that no such two edges are concatenated to form a 2-path in  $G$ . However, the triplet graph is very edge-dense, which can be demonstrated by applying curvature to the edges. The rightmost component of a triplet graph with curved edges is shown in Figure 8.

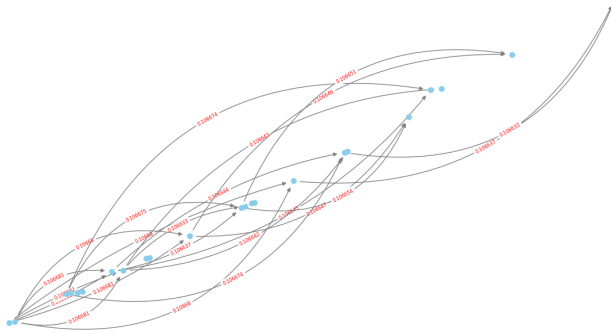


Figure 8: Rightmost component of the triplets graph

## 4 Algorithm

Considering the previously shown triplets graph, we can set a straightforward goal - for each component of this graph, try to extract the heaviest path in that component. The main ideas and implementation of this process are discussed in the following sections.

### 4.1 Weakly Connected Components

The first part of the process consist of obtaining components of graph, specifically - weakly connected components (WCCs). It has already been discussed in Section 2.5 that the Boolean semi-ring is a useful and elegant tool for tackling connectivity-based properties of a graph. Let  $A = (a_{ij})_{i,j=\overline{1,n}}$  be the adjacency matrix for our triplet graph  $G$  with  $n$  nodes. The first step is mapping the adjacency matrix  $A$  to an adjacency matrix over the Boolean semi-ring  $A_{bool}$ , which represents the graph  $G_{bool}$ . While this step can be purpose-adjusted, for the purpose of this project, we define mapping  $f : \mathbb{R}_{\geq 0} \rightarrow \{0, 1\}$  as follows:

$$f(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \end{cases}.$$

This mapping is appropriate, because the weights of edges in the triplets graph are nonnegative (more precisely, in range  $(0, 1)$ ), and we only really care about the existence of an edge. Consequently, the mapping  $\tilde{f} : \mathbb{R}_{\geq 0}^{n \times n} \rightarrow \{0, 1\}^{n \times n}$  transforms the adjacency matrix  $\tilde{f}(A) = A_{bool}$  into an adjacency matrix over the Boolean semi-ring, with  $\tilde{f}$  naturally being defined as:

$$\tilde{f}\left((a_{ij})_{i,j=\overline{1,n}}\right) = (f(a_{ij}))_{i,j=\overline{1,n}}.$$

Obtaining WCCs of a graph  $G$  is equivalent to finding connected components of undirected version of graph  $G$ . Therefore, we further consider the

symmetrized adjacency matrix  $A_{sym} = A_{bool} \oplus A_{bool}^T$  which corresponds to the undirected version of graph  $G$  over Boolean semi-ring. As in Section 2.5, we build a reachability matrix for matrix  $A_{sym}$ :

$$R = A_{sym} \oplus A_{sym}^2 \cdots \oplus A_{sym}^k \quad (7)$$

Now,  $R = (r_{ij})_{i,j=1,\overline{n}}$  satisfies:

$$r_{ij} = 1 \Leftrightarrow \text{nodes } i \text{ and } j \text{ belong to the same connected component.}$$

Going node by node and globally updating visited nodes allows us to readily and swiftly determine connected components of undirected version of the graph  $G_{bool}$ , which, as discussed, is sufficient for determining the WCCs of the graph  $G$ .

## 4.2 Heaviest path

Once we have the WCCs, the next step is to extract the heaviest path from each weakly connected component. Let  $H$  be one connected component of a graph  $G$ , and  $A_H$  its adjacency matrix.

**Note 2** *In this section we assume that  $H$ , with its adjacency matrix, is given as the connected component over the standard semi-ring.*

We continue by observing matrix  $A_H$  as a matrix over the  $\max. +$  semi-ring, which can be easily realized by an identity mapping. As discussed in Section 2.2, the biggest element of the matrix  $A_H^k$  is the accumulative weight of the heaviest  $k$ -path in graph  $H$ . In the iterative process of increasing the power of adjacency matrix  $A_H, A_H^2, \dots, A_H^k$ , we update global information about the biggest element found so far. In order to reconstruct the heaviest path, once we have the information about it, modified sparse matrix multiplication is implemented, which keeps track of predecessors.

**Note 3** *The highest power  $k$  in this section, as well as in (7), is  $n - 1$  at most, where  $n$  is the number of nodes in graph  $G$ . However, we can exit early as soon as all elements of some power of the matrix are equal to the additive identity in appropriate semi-ring.*

## 4.3 Implementation

First of all, the matrix over specific semi-ring is implemented by initializing the matrix element as *MyNumber(element)*. “MyNumber” is a custom class that overrides addition and multiplication, but also some other operations encountered in the code. For matrix initialization, we use the procedure *formSparseMatrices(tri, MyNumber=float)*. This procedure builds a sparse matrix in CSR format over custom semi-ring from triplets dataframe.

Starting from triplets “tri” dataset (represented as the weighted graph in Figure

7), we build sparse matrix over the Boolean semi-ring. Later, using the procedure that implements the ideas presented in Section 4.1, we obtain the subsets of “tri” dataset that correspond to different weakly connected components. For each of these subsets, we call `formSparseMatrices(tri_subset, MyNumber=MyNumber)` to build their appropriate sparse matrices over the `max.+` semi-ring. Finally, for each of these matrices, using ideas explored in Section 4.2, we extract the biggest path information for each WCC, and then trace it back.

While this was the general idea and flow of the algorithm, there are some tweaks implemented for efficiency. For example, while building the sparse matrix, the list `roots.idx` is also built to save and return index of all non-isolated roots<sup>4</sup> found in the graph. This is done with the purpose of reducing the number of “starting” nodes for the reachability matrix  $R$ , since every WCC of a directed acyclic graph contains a root.

**Note 4** *While isolated node is a connected component of a graph by itself, it is trivial and neglectable. Furthermore, by analyzing the data, we observed a rather big number of isolated nodes in the triplets line graph.*

## 4.4 Results

As a result of this algorithm, we obtain a list of lists of node indices (indices of triplets in “tri” dataset), each list of indices representing the heaviest path in one of the graph components. Plotting the extracted paths in the doublets-graph manner (Figure 6), together with the original starting graph, gives Figure 9. The result aligns with our expectation and aim, providing us with 4 paths that correspond to 4 muons trajectories produced in the collision.

## 5 Conclusion

This project explored the core of GraphBLAS algorithms and the possibility of adapting it for the problem of reconstructing the trajectories of charged particles stemmed in beams collision. As a result, the most demanding parts of the algorithm presented are implemented using GraphBLAS. However, this algorithm and method for tracks reconstruction deeply relies on weights of the graph. Therefore, crucial part in making this approach applicable is finding the way to assign appropriate weights to the triplet-type graph.

## A Code Availability

The full implementation and Python scripts used in this project are available at: <https://github.com/miloss-git/CERN-TrackReconstruction>.

<sup>4</sup>Root of a directed graph is a node that has no predecessors.

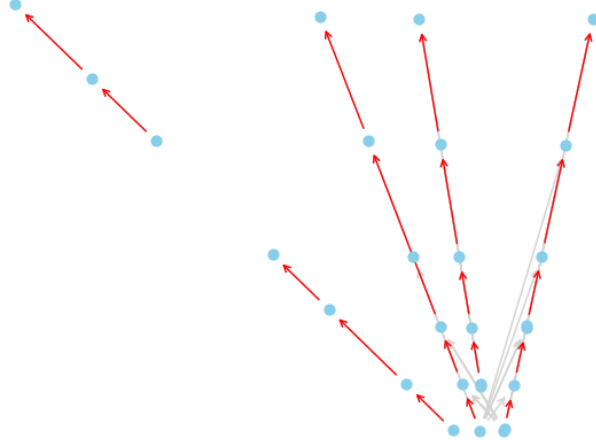


Figure 9: Heaviest paths found are colored in red.

## References

- [1] Paul Gessinger-Befurt, *Development and improvement of track reconstruction software and search for disappearing tracks with the ATLAS experiment*, 2021.
- [2] Xiaocong Ai, C. Allaire, et al, *A Common Tracking Software Project*, Computing and Software for Big Science, 2022.
- [3] Jeremy Kepner, *GraphBLAS Mathematics*, 2017.
- [4] H. Kabir, J. D. Booth and P. Raghavan, *A multilevel compressed sparse row format for efficient sparse computations on multicore processors*, 2014 21st International Conference on High Performance Computing (HiPC), Goa, India, 2014, pp. 1-10, doi: 10.1109/HiPC.2014.7116882. keywords: Sparse matrices;Program processors;Multicore processing;Bandwidth;Kernel;Symmetric matrices;Data structures,
- [5] M. Tanabashi et al. (Particle Data Group), *Passage of Particles Through Matter*, Phys. Rev. D 98 0300001 (2018) 2.