

# TopoAna: A generic tool for the topology analysis of inclusive Monte-Carlo samples in high energy physics experiments

Xingyu Zhou<sup>a,\*</sup>, Shuxian Du<sup>b</sup>, Gang Li<sup>c</sup>, Chengping Shen<sup>d,\*</sup>

<sup>a</sup>*School of Physics, Beihang University, Beijing 100191, China*

<sup>b</sup>*School of Physics and Microelectronics, Zhengzhou University, Zhengzhou 450000, China*

<sup>c</sup>*Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China*

<sup>d</sup>*Key Laboratory of Nuclear Physics and Ion-beam Application (MOE) and Institute of Modern Physics, Fudan University, Shanghai 200443, China*

---

## Abstract

Inclusive Monte-Carlo samples are indispensable for signal selection and background suppression in many high energy physics experiments. A clear knowledge of the topology of the samples, including the categories of physics processes and the number of processes in each category, is a great help to investigating signals and backgrounds. To help analysts get the topology information from the raw data of the samples, we develop a topology analysis program, TopoAna, with C++, ROOT, and LaTeX. The program implements the functionalities of component analysis and signal identification by recognizing, categorizing, counting, and tagging events. Independent of specific software frameworks, the program is applicable to many experiments. At present, it has come into use in three  $e^+e^-$  colliding experiments: the BESIII, Belle, and Belle II experiments. The use of the program in other experiments is also prospective.

## Keywords:

topology analysis; component analysis; signal identification; inclusive Monte-Carlo samples; high energy physics experiments

---

## 1. Introduction

One of the most important tasks in the data analysis of high energy physics experiments is to select signals, or in other words, to suppress backgrounds. As for the task, inclusive Monte-Carlo (MC) samples<sup>1</sup> are extremely useful, in that they provide basic, though not perfect, descriptions of the signals and/or backgrounds involved. However, due to the similarities between signals and some backgrounds, it usually takes efforts to establish a set of selection criteria that retain a high signal efficiency and meanwhile keep a low background level. Further optimization of preliminary criteria is often needed in the process. Under the circumstances, a comprehensive

---

\*Corresponding author.

E-mail address: zhouxu@buaa.edu.cn, shencp@fudan.edu.cn

The program is now available at <https://github.com/buaazhouxingyu/topoana>.

<sup>1</sup>The samples are referred to as “generic MC samples” in the Belle and Belle II experiments.

understanding of the samples is required. In particular, a clear knowledge of the topology of the samples is quite helpful. To be specific, the topology information includes the categories of physics processes and the number of processes in each category, involved both in the entire samples and in the individual events. Here, the physics process could be a complete production and decay process involved in an event, or merely a part of it, such as the decay of an intermediate resonance. With the information, one can figure out the main backgrounds (especially the peaking ones), and optimize the selection criteria further by analyzing the differences between the main backgrounds and the signals.

The analysis of the topology information described above is a sort of component analysis. It is complex since it has to classify the physics processes. Another sort of topology analysis often required in practice is signal identification, which is relatively simple because it only aims to search for certain processes of interests. Mostly, signal and background events coexist in the inclusive MC samples. It is meaningful to differentiate them in such cases. The identified signal events can be used to make up a signal sample (removed to avoid repetition) in the absence (presence) of specialized signal MC sample. Occasionally, we have to pick out some decay branches in order to re-weight them according to new theoretical predictions or updated experimental measurements. Signal identification also plays a part in this occasion.

However, since the raw topology data of inclusive MC samples is counter-intuitive, diverse, and overwhelming, it is difficult for analysts to check the topology of the samples directly. To help them obtain the information quickly and easily, a topology analysis program called TopoAna is developed with C++, ROOT [1], and LaTeX. Here, C++ is the programming language, root is the C++ based data analysis software universally used in high energy physics experiments, and LaTeX is used for generating pdf documents containing the topology information. The program implements the functionalities of component analysis and signal identification by recognizing, categorizing, counting, and tagging events accurately, and exports the obtained topology information to plain text, tex source, pdf, and root files clearly.

The program is applicable to inclusive MC samples at any data analysis stage of high energy physics experiments. In the overwhelming majority of situations, it is run over the samples which have undergone some selections, in order to examine the signals and backgrounds in the selected samples as well as the effect of the imposed selections. In such situations, the results of topology analysis are usually used together with other quantities for physics analysis. In spite of this, applying the program to the samples without undergoing any selection facilitates us to validate the generators and decay cards that produce the samples and helps novices get familiar with the topology of the samples.

Not relying on any specific software frameworks, the program applies to many high energy physics experiments. At first, the program was developed for the BESIII experiment, an experiment in the  $\tau$ -Charm energy region with abundant research topics under study [2, 3]. Then, it was extended substantially for the Belle II experiment, which is primarily dedicated to search for physics beyond the Standard Model in the flavor sector and has already started data taking in the recent two years [4]. Besides, the program has also been tried and used in the Belle experiment, the predecessor of the Belle II experiment, where some physics studies are still ongoing [5].

This user guide gives an detailed description of TopoAna. Its structure is organized as follows: Section 2 introduces the basics of the program; Sections 3 and 4 expatiate the two sorts of functionalities of the program — component analysis and signal identification, respectively; Sections 5 and 6 present some common settings and auxiliary facilities for the executing of the program, respectively; Section 7 summarizes the user guide. It is worth mentioning here that, aside from the detailed description in the user guide, an essential description of the program can

68 be found in the file “paper\_draft\_v\*.pdf” under the directory “share” of the package.

## 69 2. Basics of the program

70 This section introduces the basics of the program, including the package, input, execu-  
71 tion, and output of the program. The package implements the program via a C++ class called  
72 “topoana” and a main function invoking the class. Compiling the package creates the executable  
73 file of the program, that is, “topoana.exe”. To execute the program, we have to first obtain the  
74 input data of the program, namely the raw topology data of the inclusive MC samples, with some  
75 interfaces to the program in the software systems of the corresponding experiments. Normally,  
76 the input data contain all the topology information of the samples. With the data, all sorts of the  
77 topology analysis presented in the user guide can be performed.

78 To carry out the topology analysis desired in our work, we have to provide some necessary  
79 input, functionality, and output information to the program. The information is required to be  
80 filled in the setting items designed and implemented in the program, and the items have to be put  
81 in a plain text file named with a suffix “.card”. With the card file, one can execute the program  
82 with the command line: “topoana.exe cardFileName”, where the argument “cardFileName” is  
83 optional and its default value is “topoana.card”. After the execution of the program, we can  
84 examine the results of topology analysis in the output files and use them to analyze other experi-  
85 mental quantities. The results help us gain a better understanding of the signals and backgrounds  
86 and are conducive to carrying our work forward. In the next four subsections, we will present the  
87 package, input, execution, and output of the program in detail, with each part in one subsection.

### 88 2.1. Package of the program

89 The package consists of six directories — “include”, “src”, “bin”, “share”, “examples”, and  
90 “utilities” — and four files — “README.md”, “Configure”, “Makefile”, and “Setup”. While  
91 the directory “include” only includes one header file “topoana.h”, the directory “src” contains  
92 sixty source files “\*.cpp” as well as a script file “topoana.C”. Despite the largeness of the pro-  
93 gram, only one class “topoana” is defined for all of its functionalities. The class is declared in  
94 “topoana.h”, implemented in “\*.cpp” files, and invoked in “topoana.C”.

95 The file “template\_topoana.card” under the directory “share” saves all the items which are  
96 developed to set user specified information for the execution of the program. One can refer to the  
97 file when filling in the cards for their own needs. Some plain text files “pid\_3pchrg\_tntpnm\_texpnm  
98 \_iccp.dat.\*” are also included in the directory “share”. They store the basic information of the  
99 particles used in the program. The suffixes of their names indicate the experiments they apply  
100 to. One of them will be copied to “pid\_3pchrg\_tntpnm\_texpnm\_iccp.dat” when we set up the  
101 program. Besides, the directory “share” also contains three LaTeX style files “ geometry.sty”,  
102 “ifxetex.sty”, and “makecell.sty”, which are invoked by the program for generating pdf files.  
103 The directory “examples” includes plenty of detailed examples, particularly those involved in  
104 this paper, and the directory “utilities” contains some useful bash scripts.

105 The file “README.md” briefly introduces how to install and use the program. To set up  
106 the program, one should first set the package path with the command “./Configure”. Output  
107 of the command are the guidelines for manually adding the absolute path of “topoana.exe” to  
108 the environment variable “PATH”, in order to execute it without any path. The second step is  
109 executing the command “make”. This command compiles the header, source, and script files  
110 into the executable file “topoana.exe” under the directory “bin”, according to the rules specified

111 in the “Makefile”. Notably, executing the command once is sufficient, unless the header, source,  
 112 or script files are updated or revised, or the package is moved. The last step is specifying the  
 113 experiment name with the command line “./Setup experimentName”.

## 114 2.2. Input of the program

The input of the program is one or more root (TFile [6]) files including a TTree [7] object which contains raw topology data of the inclusive MC samples under study. To be specific, the data in each entry of the TTree object consists of the following three ingredients associated with the particles produced in an event of the samples: the number of particles, PDG [8] codes of particles, and mother indices of particles. Notably, the particles do not include the initial state particles ( $e^+$  and  $e^-$  in  $e^+e^-$  colliding experiments), which are default and thus omitted. Besides, the indices of particles are integers starting from zero (included) to the number of particles (excluded); they are obvious and hence not taken as an input ingredient for topology analysis. Equation (1) shows an example of the data.

$$\begin{aligned}
 \text{nMCGen} &= 63 \\
 \text{MCGenPDG} &= 300553, \\
 &\quad -511, 511, -433, 421, 211, 22, -413, 111, 111, 113, \\
 &\quad 211, -431, 22, -323, 213, -421, -211, 22, 22, 22, \\
 &\quad 22, 211, -211, 333, 11, -12, 22, -311, -211, 211, \\
 &\quad 111, 221, 331, 321, -321, 310, 22, 22, 111, 111, \\
 &\quad 111, 111, 111, 221, 111, 111, 22, 22, 22, 22, \\
 &\quad 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, \\
 &\quad 22, 22 \\
 \text{MCGenMothIndex} &= 0, \\
 &\quad 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, \\
 &\quad 2, 3, 3, 4, 4, 7, 7, 8, 8, 9, \\
 &\quad 9, 10, 10, 12, 12, 12, 12, 14, 14, 15, \\
 &\quad 15, 16, 16, 24, 24, 28, 31, 31, 32, 32, \\
 &\quad 32, 33, 33, 33, 36, 36, 39, 39, 40, 40, \\
 &\quad 41, 41, 42, 42, 43, 43, 44, 44, 45, 45, \\
 &\quad 46, 46
 \end{aligned} \tag{1}$$

In the example, nMCGen, MCGenPDG, and MCGenMothIndex are the names of the TBranches [9] used for storing the three ingredients. The complete physics process contained in the data is displayed in Eq. (2).

$$\begin{array}{lll}
 0 & e^+e^- \rightarrow \Upsilon(4S), & -1 \\
 1 & \Upsilon(4S) \rightarrow B^0\bar{B}^0, & 0 \\
 2 & B^0 \rightarrow \pi^0\pi^0\rho^0\pi^+D^{*-}, & 1 \\
 3 & \bar{B}^0 \rightarrow \pi^+D^0D_s^{*-}\gamma, & 1 \\
 4 & \rho^0 \rightarrow \pi^+\pi^-, & 2 \\
 5 & D^{*-} \rightarrow \pi^-\bar{D}^0, & 2 \\
 6 & D^0 \rightarrow \rho^+K^{*-}, & 3 \\
 7 & D_s^{*-} \rightarrow D_s^-\gamma, & 3 \\
 8 & \bar{D}^0 \rightarrow \eta\eta', & 5 \\
 9 & \rho^+ \rightarrow \pi^0\pi^+, & 6 \\
 10 & K^{*-} \rightarrow \pi^-\bar{K}^0, & 6 \\
 11 & D_s^- \rightarrow e^-\bar{\nu}_e\phi\gamma, & 7 \\
 12 & \eta \rightarrow \pi^0\pi^0\pi^0, & 8 \\
 13 & \eta' \rightarrow \pi^0\pi^0\eta, & 8 \\
 14 & \bar{K}^0 \rightarrow K_S^0, & 10 \\
 15 & \phi \rightarrow K^+K^-, & 11 \\
 16 & \eta \rightarrow \gamma\gamma, & 13 \\
 17 & K_S^0 \rightarrow \pi^0\pi^0 & 14
 \end{array} \tag{2}$$

115 Here, the decay branches in the process are placed into two blocks in order to make full use  
 116 of the page space. In both blocks, the first, second, and third columns are the indices, textual  
 117 expressions, and mother indices of the decay branches. Notably, all the decay branches of  $\pi^0 \rightarrow$   
 118  $\gamma\gamma$  are omitted in Eq. (2) in order to make the process look more concise. Considering  $\pi^0$  has a  
 119 very large production rate and approximatively 99% of it decays to  $\gamma\gamma$ , the program is designed

to discard the decay  $\pi^0 \rightarrow \gamma\gamma$  by default at the early phase of processing the input data (see Section 5.1.2 for the setting item to alter the behavior). Since the topology diagram of such a process looks like a tree, we refer to the complete processes as decay trees. Obviously, the data do not show the structure automatically. Thus, we need the program to do the topology analysis work.

It is recommended to save the input data in the TTree object together with other quantities for physics analyses, in order to facilitate the examination of the distributions of the quantities with the topology information. The input data can be stored in several types. Normally, the number of particles can be simply stored in a TBranch as a scalar integer, while the PDG codes of particles, as well as the mother indices of particles, can be stored in a TBranch as an array of integers, in a TBranch as a vector of integers, or in a group of TBranches as multiple scalar integers. In the analysis software of the Belle II experiment, double-precision variables are used uniformly to store all the quantities involved in the experiment, and TBranches are not recommended to store arrays and vectors in order to use other tools such as NumPy [10] and pandas [11]. In such a case, we have to store the number of particles in a TBranch as a scalar double-precision number, and store the PDG codes of particles, as well as the mother indices of particles, in a group of TBranches as multiple scalar double-precision numbers. Summing up the above, we have mentioned four storage types of the input information. For the sake of simplification, we refer to them with the following acronyms: AOI, VOI, MSI, and MSD, which are short for array of integers, vector of integers, multiple scalar integers, and multiple scalar double-precision numbers, respectively. All of the storage types are supported by the program, and their acronyms will be used in the related item of the card file (see next subsection for details).

It is easy to get the input of the program within the software framework of high energy physics experiments. In order to facilitate its use, we have developed the interfaces of the program to the software systems of the BESIII, Belle, and Belle II experiments. Similar interfaces for other experiments can also be implemented with ease. Beyond the scope of the user guide, we will not discuss the details of the interfaces here.

### 2.3. Execution of the program

To execute the program, we have to first configure some necessary setting items in a card file, and then run the program with the command line: “topoana.exe cardFileName”. This subsection introduces the essential items for the input, basic functionality, and output of the program. More items that can be set in the card file will be described in the following three sections. Sections 3 and 4 expatiate the available items for the functionalities of the program, and Section 5 presents the optional items for the common settings to control the execution of the program.

An example of the card file containing the essential items is shown as follows.

```
# The following six items set the input of the program.

% Names of input root files
{
  ../input/jpsi_1.root
  ../input/jpsi_2.root
}

% TTree name
{
  evt
```

```

168     }
169
170     % Storage type of input raw topology data (Four options: AOI, VOI, MSI, and MSD. Default: AOI)
171     {
172         AOI
173     }
174
175     % TBranch name of the number of particles (Default: nMCGen)
176     {
177         Nmcpes
178     }
179
180     % TBranch name of the PDG codes of particles (Default: MCGenPDG)
181     {
182         Pid
183     }
184
185     % TBranch name of the mother indices of particles (Default: MCGenMothIndex)
186     {
187         Midx
188     }
189
190     # The following item sets the basic functionality of the program.
191
192     % Component analysis — decay trees
193     {
194         Y
195     }
196
197     # The following item sets the output of the program.
198
199     % Main name of output files (Default: Main name of the card file)
200     {
201         jpsi_ta
202     }
203

```

In the card file, “#”, “%”, and the pair of “{” and “}”, are used for commenting, prompting, and grouping, respectively. The first six, seventh, and last one items are set for the input, basic functionality, and output of the program, respectively.

The first item sets the names of the input root files. The names ought to be input one per line without tailing characters, such as comma, semicolon, and period. In the names, both the absolute and relative paths are allowed and wildcards “[ ]?\*” are supported, just like those in the root file names input to the method Add() of the class TChain [12]. The second item specifies the TTree name. The third item tells the program the storage type of the raw input topology data, and the input should be one of the following four acronyms: AOI, VOI, MSI, and MSD, as we introduce in the previous subsection. The following three items set the TBranch names of the three ingredients of the raw input topology data. Of the first six items, the former two are indispensable, whereas the latter four can be removed or left empty if the input values are identical to the default values indicated in their prompts. Besides, the latter four items can be moved to the underlying card file, which is developed for frequently used items and will be introduced in Section 6.1, because the input values are usually fixed for a user or a group of users, though they might be different from the default values.

The seventh item sets the basic functionality of the program, namely the component analysis over decay trees. The item can be replaced or co-exist with other functionality items expatiated in

Sections 3 and 4. Here, we note that at least one functionality item has to be specified explicitly in the card file, otherwise the program will terminate soon after its start because no topology analysis to be performed is set up.

The last item specifies the main name of the output files. Though in different formats, the files are denominated with the same main name for the sake of uniformity. They will be introduced at length in the next subsection. This item is also optional, with the main name of the card file as its default input value. It is a good practice to first denominate the card file with the desired main name of the output files and then remove this item or leave it empty.

To provide a complete description, we list and explain all the essential items in the paragraphs above. However, in practical uses, we suggest removing the optional items if the input values are identical to the default ones, or moving them to the underlying card file if the input values are fixed for most of your use cases. In this way, the contents of the card file will become much more concise, making the use of the program easier and quicker. For example, only the following two items are used to set the essential information in Sections 3, 4, and 5.

```

237 % Names of input root files
238 {
239     ../input/mixed_1.root
240     ../input/mixed_2.root
241 }
242
243 % TTree name
244 {
245     evt
246 }

```

Besides, all the items in the program, also including those to be introduced in the following sections, are not required to be filled in the card files in a certain order. Nonetheless, we recommend filling them in a logical order for clearness.

During the execution of the program, some standard output and error messages are printed to the screen to provide some information on the input, progress, and output of the program, as well as the possible problems and proposed solutions to them. The standard output messages include the following four parts: (1) the values of the items with active inputs; (2) the total number of entries contained in the input root files and the progress of the program to process these entries; (3) the information output by the pdf<sub>l</sub>atex command when it compiles the tex source file to get the pdf file; (4) and the hints on the output of the program. The standard error messages are prompted with “ERROR” and “INFOR” in order to differentiate themselves from the standard output messages. The messages started with “ERROR” point out the problems encountered by the program directly, while those started with “INFOR” give more information on the problems as well as some guidelines on the solutions to the problems.

#### 2.4. Output of the program

The program gains the topology information from input data, and saves the information to output files. As mentioned in Section 1, the information includes the categories of physics processes and the number of processes in each category, involved both in entire samples and in individual events. We refer to the information at the sample level as topology maps. In the topology maps, we assign an integer to each category of physics processes as its index. We term the indices of processes as well as the numbers of processes involved in each category in the individual events as topology tags.

270 The program outputs topology maps to three different files: one plain text file, one tex source  
 271 file, and one pdf file, with the same main name specified in the card file. For instance, the  
 272 three files are “jpsi.ta.txt”, “jpsi.ta.tex”, and “jpsi.ta.pdf” in the example. Although in different  
 273 formats, the three files have the same information. The pdf file is the easiest to read. It is  
 274 converted from the tex source file with the command pdflatex. The tex source file is convenient  
 275 to us if we want to change the style of the pdf file to our taste and when we need to copy and  
 276 paste (parts of) the topology maps to our slides, papers, and so on. For example, all of the  
 277 tables displaying topology maps in this user guide are taken from associated tex source files. The  
 278 plain text file has its own advantage, because the topology maps in it can be checked with text  
 279 processing commands as well as text editors, and can be used on some occasions as input to the  
 280 functionality items (see Sections 3 and 4 for details) of another card file.

281 In addition to the three files for topology maps, one or more root files are output to save  
 282 topology tags. The root files only include one TTree object, which is entirely the same as that in  
 283 the input root files, except for the topology tags inserted in all of its entries. The number of root  
 284 files depends on the size of output data. The program switches to one new root file whenever  
 285 the size of the TTree object in memory exceeds 3 GB. In the case of the size less than 3 GB,  
 286 only one root file is output. While the sole or first root file has the same main name as the three  
 287 files above, more possible root files are denominated with the suffix “.n” (n=1, 2, 3, and so on)  
 288 appended to the main name. In the example, the first root file is “jpsi.ta.root”, and more possible  
 289 root files would be “jpsi.ta.1.root”, “jpsi.ta.2.root”, “jpsi.ta.3.root”, and so on.

290 In the example of the previous subsection, the program conducts its basic functionality, name-  
 291 ly the component analysis over decay trees. From the 100000 events of the input sample, the  
 292 program recognizes 17424 decay trees and outputs all of them to the txt, tex, and pdf files. Table  
 293 1 shows only the top ten decay trees and their respective final states in the obtained topology  
 294 map. In the table, “rowNo”, “iDcyTr”, “nEtr”, and “nCetr” are abbreviations for the row num-  
 295 ber, index of decay tree, number of entries of decay tree, and number of the cumulative entries  
 296 from the first to the current decay trees, respectively. The values of “iDcyTr” are assigned from  
 297 small to large in the program but listed according to the values of “nEtr” from large to small in  
 298 the table. This is the reason why they are not in natural order like the values of “rowNo”. In  
 299 Section 2.2, we mention that the decay  $\pi^0 \rightarrow \gamma\gamma$  is ignored by the program at the early phase of  
 300 processing the input data. As a result, it does not show itself in the table.

Table 1: Top ten decay trees and their respective final states.

rowNo	decay tree	decay final state	iDcyTr	nEtr	nCetr
1	$J/\psi \rightarrow \mu^+ \mu^-$	$\mu^+ \mu^-$	6	5269	5269
2	$J/\psi \rightarrow e^+ e^-$	$e^+ e^-$	4	4513	9782
3	$J/\psi \rightarrow \pi^0 \pi^+ \pi^- \pi^-$	$\pi^0 \pi^+ \pi^- \pi^-$	0	2850	12632
4	$J/\psi \rightarrow \pi^0 \pi^+ \pi^+ \pi^- \pi^-$	$\pi^0 \pi^+ \pi^+ \pi^- \pi^-$	2	1895	14527
5	$J/\psi \rightarrow \pi^0 \pi^+ \pi^- K^+ K^-$	$\pi^0 \pi^+ \pi^- K^+ K^-$	20	1698	16225
6	$J/\psi \rightarrow \rho^+ \rho^- \omega, \rho^+ \rightarrow \pi^0 \pi^+, \rho^- \rightarrow \pi^0 \pi^-, \omega \rightarrow \pi^0 \pi^+ \pi^-$	$\pi^0 \pi^0 \pi^0 \pi^+ \pi^- \pi^-$	19	1453	17678
7	$J/\psi \rightarrow e^+ e^- \gamma^f$	$e^+ e^- \gamma^f$	70	1222	18900
8	$J/\psi \rightarrow \pi^0 \pi^0 \pi^+ \pi^- \pi^-$	$\pi^0 \pi^0 \pi^+ \pi^- \pi^-$	127	1161	20061
9	$J/\psi \rightarrow \pi^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^-$	$\pi^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^-$	234	836	20897
10	$J/\psi \rightarrow \pi^0 \pi^0 \pi^+ \pi^- \gamma^F$	$\pi^0 \pi^0 \pi^+ \pi^- \gamma^F$	43	792	21689



301 In the table, “iDcyTr” is the topology tag for decay trees. Thus, it is also saved in the TTree  
 302 objects of the output root file, together with other quantities for physics analysis. Therefore, it  
 303 can be used to pick out the entries of specific decay trees and then examine the distributions of  
 304 the other quantities over the decay trees. In the example, besides the raw topology data, only  
 305 a random variable following standardized normal distribution, namely X, is stored in the input  
 306 root files and thus copied by default to the output root file. Though not a genuine variable for  
 307 physics analysis, X is quite good to illustrate the usage of the topology tag. Figure 1 shows the  
 308 distribution of X accumulated over the top ten decay trees. The figure is drawn with the root script

309  
 310 `examples/in_the_user_guide/ex_for_tb_01/draw_X/v2/draw_X.C,`

311 where, for example, a statement equivalent to

312  
 313  
 314 `chain->Draw(“X >>h0”, “iDcyTr==6”)`

315  
 316 is used to import X over the decay tree  $J/\psi \rightarrow \mu^+ \mu^-$  from the output root file to the histogram  
 317 named h0. With such a figure, we can clearly see the contribution of each decay tree. Particu-  
 318 larly, we can get to know whether a decay tree has a peak contribution or a contribution mainly  
 319 distributed in a different region. Based on these distributions, we can get a better understand-  
 320 ing of our signals and backgrounds, and thus optimize event selection criteria by applying new  
 321 requirements on the displayed quantities.

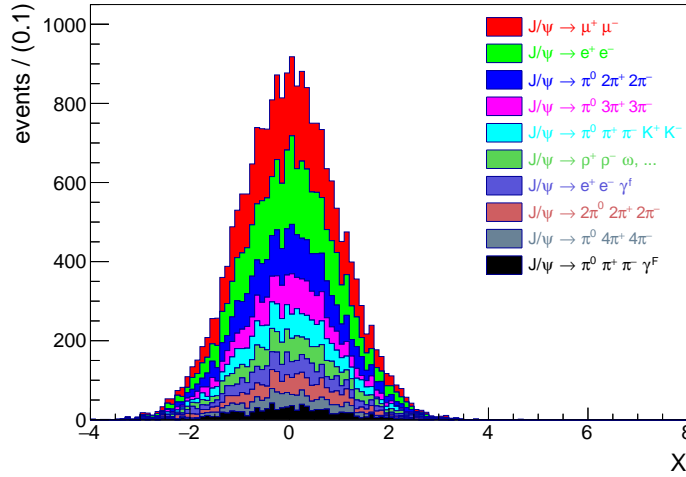


Figure 1: Distribution of X accumulated over the top ten decay trees. In the legend entry “ $J/\psi \rightarrow \rho^+ \rho^- \omega, \dots$ ”, the dots “...” represent the secondary decay branches:  $\rho^+ \rightarrow \pi^0 \pi^+$ ,  $\rho^- \rightarrow \pi^0 \pi^-$ ,  $\omega \rightarrow \pi^0 \pi^+ \pi^-$ .

### 322 3. Component analysis

323 Component analysis is the primary functionality of the program. It is performed over decay  
 324 trees in the previous example. In addition, it can be carried out as follows: over decay initial-

final states; with specified particles to check their decay branches, production branches, mothers, cascade decay branches, and decay final states; with specified inclusive decay branches to examine their exclusive components; and with specified intermediate-resonance-allowed (IRA) decay branches to investigate their inner structures. This section introduces the nine (five for specified particles) kinds of component analysis, with each in a subsection. For each kind of component analysis, one item is designed and implemented in the program to set related parameters. In each subsection, we take an example to demonstrate the corresponding setting item and show the resulting topology map. For easy exposition, all of the essential topology tags involved in the component analysis functionalities are presented in another separate subsection, namely the last subsection.

Similar to the case over decay trees, to perform the component analysis over decay initial-final states, we only need to input an positive option “Y” to the corresponding item. Different from the former two kinds, to carry out the latter seven kinds of component analysis, we have to explicitly specify one or more desired particles, inclusive decay branches, or IRA decay branches in the associated items. In the following examples, two particles or decay branches are set to illustrate the use of these items, but only the topology map related to one of them is shown to save space in the paper.

In addition to the indispensable parameters, two sorts of common optional parameters can be set in the items. The first sort is designed for all the nine kinds of component analysis to restrict the maximum number of components output to the txt, tex and pdf files. Without the optional parameters, all components will be output. This is fine if the number of components is not massive. In cases of too many (around ten thousand or more) components, it takes a long time for the program to output the components to the txt and tex files as well as to get the pdf file from the tex file. In such cases, it also takes up a large disk space to save these components in the output files. Considering further that the posterior components are generally unimportant and our time and energy to examine them are limited, it is better to set a maximum to the number of output components. To save space in the paper, we set the maximum number to five in the following examples.

The second sort of optional parameters are developed for the latter seven kinds of component analysis to assign meaningful aliases to the specified particles, inclusive decay branches, and IRA decay branches. By default, the indices 0, 1, 2, and so on are used to tag the particles and decay branches in the names of the TBranches appended in the TTree object of the output root files. This is fine, but it is significative to replace the indices with meaningful aliases, particularly in cases of many specified particles or decay branches.

### 3.1. Decay trees

Component analysis over decay trees is the basic kind of component analysis that has already been widely performed in the BESIII experiment. The following example shows the associated item with the maximum number of output components set to five. In the item, a third parameter is also filled and set to “Y”. With the setting, the decay final states in the output pdf file are put under their respective decay trees, rather than in a column next to that for decay trees. It is recommended to use this optional parameter in cases there are too many (about ten or more) particles in some final states. Here, we note that the symbol “—” can be used as a placeholder for the maximum number of output components, if only the third parameter is desired.

% Component analysis — decay trees

371 {  
372 Y 5 Y  
373 }  
374

375 Table 2 shows the decay trees. In the table, while the first five decay trees are listed exclusively  
376 in the main part, the rest decay trees are only summarized inclusively at the bottom row. In the  
377 symbolic expressions of decay initial-final states, the dashed right arrow ( $--\rightarrow$ ) instead of the  
378 plain right arrow ( $\rightarrow$ ) is used, in order to reflect that the initial state does not necessarily decay  
379 to the final states in a direct way. Similarly, it is also used in the symbolic expressions of IRA  
380 decay branches, which will be introduced in Section 3.9.

Table 2: Decay trees and their respective initial-final states.

rowNo	decay tree (decay initial-final states)	iDcyTr	nEtr	nCEtr
1	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $D^{*+} \rightarrow \pi^+ D^0, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) --\rightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu \pi^0 \pi^+ \pi^+ \pi^- \pi^- K^+ K^- \gamma^F)$	20870	3	3
2	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \pi^0 \pi^+ \pi^+ \rho^- D^-, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, \rho^- \rightarrow \pi^0 \pi^-,$ $D^- \rightarrow \pi^- \pi^- K^+, D^{*+} \rightarrow \pi^+ D^0, D^0 \rightarrow K_L^0 \pi^+ \pi^-$ $(\Upsilon(4S) --\rightarrow \mu^- \bar{\nu}_\mu \pi^0 \pi^0 K_L^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+)$	5295	2	5
3	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, \bar{B}^0 \rightarrow e^- \bar{\nu}_e D^+, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $D^+ \rightarrow e^+ \nu_e \bar{K}^*, \bar{D}^0 \rightarrow \pi^0 \pi^+ \pi^- K_S^0, \bar{K}^* \rightarrow \pi^0 \bar{K}^0, K_S^0 \rightarrow \pi^+ \pi^-, \bar{K}^0 \rightarrow K_L^0$ $(\Upsilon(4S) --\rightarrow e^+ e^- \nu_e \bar{\nu}_e \mu^+ \nu_\mu \pi^0 \pi^0 K_L^0 \pi^+ \pi^+ \pi^- \pi^- \pi^-)$	11954	2	7
4	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-}, \bar{B}^0 \rightarrow \pi^0 \pi^- \omega D^+, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $\omega \rightarrow \pi^0 \pi^+ \pi^-, D^+ \rightarrow e^+ \nu_e \pi^+ K^-, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+$ $(\Upsilon(4S) --\rightarrow e^+ e^+ \nu_e \nu_e \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+ K^-)$	14345	2	9
5	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, \bar{B}^0 \rightarrow e^- \bar{\nu}_e D^{*+} \gamma^F, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $D^{*+} \rightarrow \pi^0 D^+, \bar{D}^0 \rightarrow \pi^- K^+, D^+ \rightarrow e^+ \nu_e \bar{K}^*, \bar{K}^* \rightarrow \pi^+ K^-$ $(\Upsilon(4S) --\rightarrow e^+ e^- \nu_e \bar{\nu}_e \mu^+ \nu_\mu \pi^0 \pi^+ \pi^- \pi^- K^+ K^- \gamma^F)$	15332	2	11
rest	$\Upsilon(4S) \rightarrow \text{others (99980 in total)}$ $(\Upsilon(4S) --\rightarrow \text{corresponding to others})$	—	99989	100000

### 381 3.2. Decay initial-final states

Table 3: Decay initial-final states.

rowNo	decay initial-final states	iDcyIFSts	nEtr	nCEtr
1	$\Upsilon(4S) --\rightarrow \mu^+ \nu_\mu \pi^0 \pi^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+ K^-$	41	18	18
2	$\Upsilon(4S) --\rightarrow \pi^0 \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-$	887	18	36
3	$\Upsilon(4S) --\rightarrow \mu^- \bar{\nu}_\mu \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-$	3350	18	54
4	$\Upsilon(4S) --\rightarrow \pi^0 \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-$	1215	17	71
5	$\Upsilon(4S) --\rightarrow \pi^0 \pi^0 \pi^0 \pi^0 \pi^0 K_L^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^-$	1207	17	88
rest	$\Upsilon(4S) --\rightarrow \text{others (78208 in total)}$	—	99912	100000

382 On some occasions, we need to investigate decay initial-final states. Below is an example  
383 demonstrating the related item with the maximum number of output components set to five.

384  
385 % Component analysis — decay initial-final states  
386 {

```

387         Y    5
388     }

```

389  
390 The decay initial-final states are displayed in Table 3. The layout of the table is similar to that of  
391 Table 2, which shows the decay trees.

### 3.3. Decay branches of particles

392  
393 Sometimes, we have to examine the decay branches of certain particles. The following ex-  
394 ample shows the associated item with the two particles  $D^{*+}$  and  $J/\psi$  set as research objects.  
395 In the item, each row holds the information of a specified particle, and the first, second and  
396 third columns are the textual expressions, aliases, and maximum numbers of output components,  
397 respectively. As we introduce at the beginning part of this section, the aliases and maximum  
398 numbers of output components are both optional. Here, we note that the symbol “-” can be used  
399 as a placeholder for an unassigned alias, if only the maximum number of output components is  
400 desired.

```

401 % Component analysis — decay branches of particles
402 {
403     D*+   Dsp   5
404     J/psi  Jpsi  5
405 }

```

406  
407 Table 4 shows the decay branches of  $D^{*+}$ . From the table, only four decay branches of  $D^{*+}$  are  
408 found in the input inclusive MC sample. Since there is likely one or more cases of  $D^{*+}$  decays in  
409 one input entry, “nCase” and “nCCase”, instead of “nEtr” and “nCetr”, are used in the table in  
410 order to accurately indicate what we are counting are the numbers of  $D^{*+}$  decays, rather than the  
411 numbers of entries involving the  $D^{*+}$  decays.  
412

Table 4: Decay branches of  $D^{*+}$ .

rowNo	decay branch of $D^{*+}$	iDcyBrP	nCase	nCCase
1	$D^{*+} \rightarrow \pi^+ D^0$	0	31180	31180
2	$D^{*+} \rightarrow \pi^0 D^+$	1	13978	45158
3	$D^{*+} \rightarrow D^+ \gamma$	2	700	45858
4	$D^{*+} \rightarrow \pi^+ D^0 \gamma^F$	3	28	45886

### 3.4. Production branches of particles

413  
414 In some cases, we have interest in the production branches of certain particles. Below is an  
415 example demonstrating the related item also by taking the two particles  $D^{*+}$  and  $J/\psi$  as objects  
416 of study. The input to this item is the same as that to the above item.  
417

```

418 % Component analysis — production branches of particles
419 {
420     D*+   Dsp   5
421     J/psi  Jpsi  5
422 }

```

Table 5: Production branches of  $D^{*+}$ .

rowNo	production branch of $D^{*+}$	iProdBrP	nCase	nCCase
1	$\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$	9	4154	4154
2	$\bar{B}^0 \rightarrow e^- \bar{\nu}_e D^{*+}$	7	2886	7040
3	$\bar{B}^0 \rightarrow D^{*+} D_s^{*-}$	4	1691	8731
4	$\bar{B}^0 \rightarrow e^- \bar{\nu}_e D^{*+} \gamma^F$	10	1623	10354
5	$\bar{B}^0 \rightarrow \pi^0 \pi^+ \pi^- \pi^- D^{*+}$	40	1429	11783
rest	others (3272 in total)	—	34103	45886

The production branches of  $D^{*+}$  are displayed in Table 5. In the production branches,  $D^{*+}$  is marked in blue so as to make it noticeable. From the table, the number of production branches of  $D^{*+}$  found in the input sample is 3277, much bigger than 4, which is the number of its decay branches.

### 3.5. Mothers of particles

Occasionally, we may want to check the mothers of certain particles. The following example shows the associated item also with the two particles  $D^{*+}$  and  $J/\psi$  set as research objects. The input to this item is identical to those to the two items above.

```
% Component analysis — mothers of particles
{
  D*+   Dsp   5
  J/psi Jpsi   5
}
```

Table 6 shows the mothers of  $D^{*+}$ . Notably, the PDG codes of the mother particles, instead of additional indices, are listed in the table, since they are sufficient to tag the mother particles. From the table, six sources of  $D^{*+}$  are found in the input sample and the dominant one is the  $\bar{B}^0$  decay.

Table 6: Mothers of  $D^{*+}$ .

rowNo	mother of $D^{*+}$	PDGMoth	nCase	nCCase
1	$\bar{B}^0$	-511	41751	41751
2	$B^0$	511	2983	44734
3	$D_1^{*+}$	20413	455	45189
4	$D_1^+$	10413	368	45557
5	$D_2^{*+}$	415	247	45804
rest	others (1 in total)	—	82	45886

### 3.6. Cascade decay branches of particles

On some occasions, we need to investigate the cascade decay branches of certain particles. Below is an example demonstrating the related item by taking the two particles  $B^0$  and  $D^0$  as objects of study. While the first three columns of the input to this item have the same meanings as those to the three items above, the additional fourth column sets the maximum hierarchy of decay branches to be examined. Here, the hierarchy reflects the rank of a decay branch in a cascade decay branch of one specific particle. For instance, in the following cascade decay branch of  $B^0$ :  $B^0 \rightarrow \pi^0 \pi^0 \rho^0 \pi^+ D^{*-}$ ,  $\rho^0 \rightarrow \pi^+ \pi^-$ ,  $D^{*-} \rightarrow \pi^- \bar{D}^0$ ,  $\bar{D}^0 \rightarrow \eta \eta'$ ,  $\eta \rightarrow \pi^0 \pi^0 \pi^0$ ,  $\eta' \rightarrow \pi^0 \pi^0 \eta$ ,

$\eta \rightarrow \gamma\gamma$ , the hierarchies of the seven individual decay branches are 1, 2, 2, 3, 4, 4, and 5, respectively. In the example item, the maximum hierarchy of decay branches is set to two for both  $B^0$  and  $D^0$ , and hence only the first two hierarchies of branches in their cascade decays will be investigated. Without such settings, all the branches in their cascade decays will be examined.

```

457 % Component analysis — cascade decay branches of particles
458 {
459     B0  B0  5  2
460     D0  D0  5  2
461 }

```

The cascade decay branches of  $B^0$  are displayed in Table 7.

Table 7: Cascade decay branches of  $B^0$  (only the first two hierarchies are involved).

rowNo	cascade decay branch of $B^0$	iCascDcyBrsP	nCase	nCCase
1	$B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, D^{*-} \rightarrow \pi^- \bar{D}^0$	12	2912	2912
2	$B^0 \rightarrow e^+ \nu_e D^{*-}, D^{*-} \rightarrow \pi^- \bar{D}^0$	6	1991	4903
3	$B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, D^{*-} \rightarrow \pi^0 D^-$	70	1283	6186
4	$B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F, D^{*-} \rightarrow \pi^- \bar{D}^0$	18	1132	7318
5	$B^0 \rightarrow D^{*-} D_s^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0, D_s^{*+} \rightarrow D_s^+ \gamma$	20	1119	8437
rest	$B^0 \rightarrow \text{others (42074 in total)}$	—	91594	100031

### 3.7. Decay final states of particles

Sometimes, we have to examine the decay final states of certain particles. The following example shows the associated item also with the two particles  $B^0$  and  $D^0$  set as research objects. The format of the input to the item is the same as that to the above item, but the fourth parameters here are designed to restrict the numbers of final state particles. Without the fourth parameters, all the decay final states of the specified particles will be investigated. In the example, the parameters are set to three for both  $B^0$  and  $D^0$ , and thus only the three-body decay final states of them will be examined.

```

473 % Component analysis — decay final states of particles
474 {
475     B0  B0  5  3
476     D0  D0  5  3
477 }

```

Table 8 shows the three-body decay final states of  $D^0$ . In the table,  $\pi^0$  only decays to  $\gamma\gamma$ ; otherwise, it will be replaced with its decay products, resulting in different decay final states of  $D^0$ .

Table 8: Decay final states of  $D^0$  (only three-body final states are involved).

rowNo	decay final state of $D^0$	iDcyFStP	nCase	nCCase
1	$D^0 \rightarrow \pi^0 \pi^+ K^-$	2	6258	6258
2	$D^0 \rightarrow \mu^+ \nu_\mu K^-$	5	1487	7745
3	$D^0 \rightarrow \pi^0 \pi^+ \pi^-$	1	1162	8907
4	$D^0 \rightarrow K_L^0 \pi^+ \pi^-$	3	1158	10065
5	$D^0 \rightarrow e^+ \nu_e K^-$	11	1148	11213
rest	$D^0 \rightarrow \text{others (24 in total)}$	—	2407	13620

### 3.8. Inclusive decay branches

In some cases, we are interested in the exclusive components of certain inclusive decay branches. Below is an example demonstrating the related item by taking the two inclusive decay branches  $\bar{B}^0 \rightarrow D^{*+} + \text{anything}$  and  $B^0 \rightarrow K_S^0 + \text{anything}$  as objects of study. In the item, each row holds the information of a inclusive decay branch, and the first, second, and third columns separated with the symbol “&” are the textual expressions, aliases, and maximum numbers of output components, respectively. As we introduce at the beginning part of this section, the aliases and maximum numbers of output components are both optional. Here, we note that the symbol “\_” can be used as a placeholder for an unassigned alias, if only the maximum number of output components is desired.

```
% Component analysis — inclusive decay branches
{
  B0 --> D*+ & B2Dsp & 5
  B0 --> K_S0 & B2Ks & 5
}
```

The exclusive components of  $B^0 \rightarrow K_S^0 + \text{anything}$  are displayed in Table 9. From the table, ten exclusive components of the inclusive decay branch are found in the input sample, and the particles denoted with *anything* are mainly the traditional charmonium states.

Table 9: Exclusive components of  $B^0 \rightarrow K_S^0 + \text{anything}$ .

rowNo	exclusive component of $B^0 \rightarrow K_S^0 + \text{anything}$	iDcyBrIncDcyBr	nCase	nCCase
1	$B^0 \rightarrow K_S^0 J/\psi$	0	45	45
2	$B^0 \rightarrow K_S^0 \eta_c$	1	40	85
3	$B^0 \rightarrow K_S^0 \psi'$	3	33	118
4	$B^0 \rightarrow K_S^0 \chi_{c1}$	2	20	138
5	$B^0 \rightarrow K_S^0 \chi_{c0}$	4	6	144
rest	$B^0 \rightarrow K_S^0 + \text{others (5 in total)}$	—	9	153

### 3.9. Intermediate-resonance-allowed decay branches

Table 10: Exclusive components of  $D^{*+} \rightarrow \pi^0 \pi^+ \pi^- K^-$ .

rowNo	exclusive component of $D^{*+} \rightarrow \pi^0 \pi^+ \pi^- K^-$	iDcyBrIRADcyBr	nCase	nCCase
1	$D^{*+} \rightarrow \pi^+ D^0, D^0 \rightarrow \pi^0 \pi^+ K^-$	0	3869	3869
2	$D^{*+} \rightarrow \pi^0 D^+, D^+ \rightarrow \pi^+ \pi^+ K^-$	1	1102	4971

Occasionally, we may want to investigate the inner structures of certain IRA decay branches. The following example shows the associated item with the two IRA decay branches  $D^{*+} \rightarrow \pi^0 \pi^+ \pi^- K^-$  and  $J/\psi \rightarrow \pi^0 \pi^+ \pi^-$  set as research objects. Since IRA decay branches look like inclusive decay branches, the format of the input to the item for IRA decay branches is identical to that for inclusive decay branches, which is introduced in the previous subsection.

```
% Component analysis — intermediate-resonance-allowed decay branches
{
  D*+ --> K- pi+ pi+ pi0 & Dsp2K3Pi & 5
  J/psi --> pi+ pi- pi0 & Jpsi23Pi & 5
}
```

Table 10 shows the exclusive components of  $D^{*+} \rightarrow \pi^0 \pi^+ \pi^+ K^-$ . From the table, two intermediate particles  $D^0$  and  $D^+$  are found in the IRA decay branch, and they decay to  $\pi^0 \pi^+ K^-$  and  $\pi^+ \pi^+ K^-$ , respectively.

### 3.10. Essential topology tags

Table 11: Essential topology tags involved in each kind of component analysis.

Component analysis kind	Topology tag	Interpretation
Decay trees	iDcyTr	index of decay tree
Decay initial-final states	iDcyIFSts	index of decay initial-final states
Decay branches of particles	nPDcyBr.i	number of particle;s (or its decay branches)
	iDcyBrP.i.j	index of decay branch of the $j^{\text{th}}$ particle <sub>i</sub>
Production branches of particles	nPProdBr.i	number of particle;s (or its production branches)
	iProdBrP.i.j	index of production branch of the $j^{\text{th}}$ particle <sub>i</sub>
Mothers of particles	nPMoth.i	number of particle;s (or its mothers)
	PDGMothP.i.j	PDG code of mother of the $j^{\text{th}}$ particle <sub>i</sub>
Cascade decay branches of particles	nPCascDcyBr.i	number of particle;s (or its cascade decay branches)
	iCascDcyBrP.i.j	index of cascade decay branch of the $j^{\text{th}}$ particle <sub>i</sub>
Decay final states of particles	nPDcyFSt.i	number of particle;s (or its decay final states)
	iDcyFStP.i.j	index of decay final state of the $j^{\text{th}}$ particle <sub>i</sub>
Inclusive decay branches	nIncDcyBr.i	number of inclusive decay branch <sub>i</sub> es
	iDcyBrIncDcyBr.i.j	index of decay branch of the $j^{\text{th}}$ inclusive decay branch <sub>i</sub>
IRA decay branches	nIRADcyBr.i	number of IRA decay branch <sub>i</sub> es
	iDcyBrIRADcyBr.i.j	index of decay branch of the $j^{\text{th}}$ IRA decay branch <sub>i</sub>

Table 11 lists and interprets all of the essential topology tags involved in the component analysis functionalities. The topology tag for the component analysis over decay initial-final states is iDcyIFSts. It has the similar interpretation as iDcyTr and is shown in the third column of Table 3. For the latter seven kinds of component analysis, there are two sorts of topology tags. The first sort, such as nPDcyBr.i, records the number of the specified particles or decay branches found in each entry. The second sort, for example iDcyBrP.i.j, keeps the corresponding index of each instance of one specified particle or decay branch found in each entry. The indices are also listed in the third columns of Tables 4 – 10.

In the topology tags, “i” in “i” is the default index of the specified particle or decay branch, and it ranges from 0 (included) to the number of specified particles or decay branches (excluded). If the alias of the particle or decay branch is also specified, the index “i” will be replaced with the alias. For example, in the component analysis over decay branches of  $D^{*+}$  and  $J/\psi$ , since “Dsp” and “Jpsi” are set as their aliases, the specialised topology tags nPDcyBr.Dsp and nPDcyBr.Jpsi, instead of the default ones nPDcyBr.0 and nPDcyBr.1, are used to store the numbers of  $D^{*+}$  and  $J/\psi$  (or their decay branches) found in each entry.

In addition, “j” in “j” is the default index of the particle or decay branch found in an entry, and it ranges from 0 (included) to the sample-level maximum of the number of the particles or decay branches found in each entry (excluded). For example, in the component analysis over decay branches of  $D^{*+}$ , the sample-level maximum of the number of  $D^{*+}$  (or its decay branches) found in each entry is two, and thus the indices of the  $D^{*+}$  decay branches are stored in the topology tags iDcyBrP.Dsp.0 and iDcyBrP.Dsp.1. The indices range from 0 (include) to the number of the categories of the  $D^{*+}$  decay branches found in the samples. In the entries with



only one  $D^{*+}$ , iDcyBrP\_Dsp\_1 is assigned with the default value  $-1$ ; in the entries which have no  $D^{*+}$ , the default value  $-1$  is assigned to both iDcyBrP\_Dsp\_0 and iDcyBrP\_Dsp\_1. We note that different from all other indices, PDGMoth.i.j has the default value 0, instead of  $-1$ .

#### 4. Signal identification

Signal identification is the other functionality of the program. Though it is a relatively simple functionality, it can help us identify the signals we desire directly, quickly, and easily. At present, the following eight kinds of signals can be identified with the program: (1) decay trees, (2) decay initial-final states, (3) particles, (4) (regular) decay branches, (5) cascade decay branches, (6) inclusive decay branches, (7) inclusive cascade decay branches, and (8) IRA decay branches. For each kind of signals, one item is developed to specify related parameters. This section introduces the eight kinds of signal identification, with each in a subsection. In each subsection, we take an example to demonstrate the related setting item and show the obtained topology map. For easy exposition, all of the essential topology tags involved in the signal identification functionalities are presented in another separate subsection, that is, the last subsection.

Similar to the cases of the latter seven kinds of component analysis, one or more signals can be specified in each of the signal identification items, and two signals are set in the following examples to illustrate the use of the items. Besides, meaning aliases can also be optionally assigned to the specified signals so as to better tag them in the names of the TBranches appended in the TTree object of the output root files.

##### 4.1. Decay trees

Sometimes, we need to identify certain decay trees. The following example shows the associated item with the first two decay trees listed in Table 2 set as signals. In the item, each row holds a decay branch in the decay trees, and the first, second, and third columns separated with the symbol “&” are the indices, textual expressions, and mother indices of the decay branches, respectively. In addition, the decay branches with index 0 indicate the beginning of new decay trees, and their mother indices are equal to  $-1$ , suggesting they have no mother branches because they are the first decay branches of the decay trees. Besides, a name of each decay tree can be optionally filled in the fourth column of its first decay branch. Similar to the third parameter in the item for the component analysis over decay trees (see Section 3.1), a “Y” can be optionally filled in the fifth column of the first decay branch of the first decay tree, to adjust the positions of decay final states in the output pdf file.

```
% Signal identification — decay trees
{
  0 & Upsilon(4S) --> B0 anti-B0 & -1 & 1stDcyTrInTb2 & Y
  1 & B0 --> e+ nu_e D*- gamma & 0
  2 & anti-B0 --> mu- anti-nu_mu D*+ & 0
  3 & D*- --> pi- anti-D0 & 1
  4 & D*+ --> pi+ D0 & 2
  5 & anti-D0 --> pi0 pi- K+ & 3
  6 & D0 --> pi0 pi+ K- & 4

  0 & Upsilon(4S) --> B0 anti-B0 & -1 & 2ndDcyTrInTb2
  1 & B0 --> pi0 pi+ pi- rho- D- & 0
  2 & anti-B0 --> mu- anti-nu_mu D*+ & 0
  3 & rho- --> pi0 pi- & 1
  4 & D- --> pi- pi- K+ & 1
  5 & D*+ --> pi+ D0 & 2
  6 & D0 --> K_L0 pi+ pi- & 5
```

590 }

591

592 Table 12 shows the resulting topology map. The results are the same as those displayed in the  
593 first two rows of Table 2.

Table 12: Signal decay trees and their respective initial-final states.

rowNo	signal decay tree (signal decay initial-final states)	iSigDcyTr	nEtr	nCEtr
1	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $D^{*+} \rightarrow \pi^+ D^0, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) \rightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu \pi^0 \pi^+ \pi^- \pi^- K^+ K^- \gamma^F)$	0	3	3
2	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \pi^0 \pi^+ \pi^- \rho^- D^-, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, \rho^- \rightarrow \pi^0 \pi^-,$ $D^- \rightarrow \pi^- \pi^- K^+, D^{*+} \rightarrow \pi^+ D^0, D^0 \rightarrow K_S^0 \pi^+ \pi^-$ $(\Upsilon(4S) \rightarrow \mu^- \bar{\nu}_\mu \pi^0 \pi^0 K_S^0 \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-)$	1	2	5

#### 594 4.2. Decay initial-final states

595 In a few cases, we have interest in some decay initial-final states. Below is an example  
596 demonstrating the related item by taking the first two decay initial-final states listed in Table 3  
597 as signals. Similar to IRA decay branches, decay initial-final states look like inclusive decay  
598 branches. Hence, except that only two columns are involved in the item, the format of the input  
599 to the item for decay initial-final states is identical to that for the component analysis over inclu-  
600 sive decay branches, which is introduced in Section 3.8. As we can see from the example, the  
601 numbers of identical particles are supported to be written in front of their textual names in order  
602 to simplify the textual expressions of the final states.

```
603 % Signal identification — decay initial-final states
604 {
605     Y(4S) --> mu+ nu_mu 3 pi0 3 pi+ 4 pi- K+ K- & 2ndDcyIFStsInTb3
606     Y(4S) --> 5 pi0 5 pi+ 5 pi- K+ K- & 2ndDcyIFStsInTb3
607 }
608
```

610 The obtained topology map is displayed in Table 13. The results are identical to those shown in  
611 the first two rows of Table 3.

Table 13: Signal decay initial-final states.

rowNo	signal decay initial-final states	iSigDcyIFSts2	nEtr	nCEtr
1	$\Upsilon(4S) \rightarrow \mu^+ \nu_\mu \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-$	0	18	18
2	$\Upsilon(4S) \rightarrow \pi^0 \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-$	1	18	36

#### 612 4.3. Particles

613 Occasionally, we may want to identify some particles. The following example shows the  
614 associated item with the two particles  $D^{*+}$  and  $J/\psi$  set as signals. Except that only two columns  
615 are involved in the item, the format of the input to the item is identical to that for the component  
616 analysis over decay branches of particles, which is introduced in Section 3.3.

```
617 % Signal identification — particles
618 {
619     D*+   Dsp
620     J/psi  Jpsi
621 }
622
```

623

624 Table 14 shows the resulting topology map. As a cross-check, the number of  $D^{*+}$ s in the table  
 625 equals those in Tables 4, 5, and 6.

Table 14: Signal particles.

rowNo	signal particle	iSigP	nCase	nCCase
1	$D^{*+}$	0	45886	45886
2	$J/\psi$	1	2654	48540

#### 626 4.4. Decay branches

627 On some occasions, we have to identify certain regular decay branches. Below is an ex-  
 628 ample demonstrating the related item by taking the two decay branches  $\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$  and  
 629  $B^0 \rightarrow K_S^0 J/\psi$  as signals. Since regular decay branches also look like inclusive decay branches,  
 630 except that only two columns are involved in the item, the format of the input to the item for reg-  
 631 ular decay branches is identical to that for the component analysis over inclusive decay branches,  
 632 which is introduced in Section 3.8.

```
633 % Signal identification — decay branches
634 {
635   B0 --> mu- anti-nu_mu D*+ & B2munuDsp
636   B0 --> K_S0 J/psi & B2KsJpsi
637 }
638
```

639

640 The obtained topology map is displayed Table 15. For the purpose of cross-checks, we note that  
 641 the number of  $\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$  ( $B^0 \rightarrow K_S^0 J/\psi$ ) in the table is equal to that in the first row of Table  
 642 5 (9).

Table 15: Signal decay branches.

rowNo	signal decay branch	iSigDcyBr	nCase	nCCase
1	$\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$	0	4154	4154
2	$B^0 \rightarrow K_S^0 J/\psi$	1	45	4199

#### 643 4.5. Cascade decay branches

644 Sometimes, we are interested in certain cascade decay branches. The following example  
 645 shows the associated item with the two cascade decay branches  $B^0 \rightarrow D^{*-} D_s^{*+}$ ,  $D^{*-} \rightarrow \pi^- \bar{D}^0$ ,  
 646  $D_s^{*+} \rightarrow D_s^+ \gamma$  and  $B^0 \rightarrow D^{*-} D_s^{*+}$ ,  $D^{*-} \rightarrow \pi^- \bar{D}^0$  set as signals. While the first cascade decay  
 647 branch is identical to the fifth one in Table 7, the second is only part of it, which demonstrates  
 648 that the cascade decay branches supported in the item are not necessarily fully specified at the  
 649 level of certain hierarchy. Similar to decay trees, cascade decay branches are made up of regular  
 650 decay branches. Hence, the format of the input to the item for cascade decay branches is identical  
 651 to that for decay trees, which is introduced in Section 4.1.

652

```
653 % Signal identification — cascade decay branches
654 {
655   0 & B0 --> D*- D_s*+ & -1
656   1 & D*- --> pi- anti-D0 & 0
657   2 & D_s*+ --> D_s+ gamma & 0
658
659   0 & B0 --> D*- D_s*+ & -1

```

```

660      1 & D*- --> pi- anti-D0 & 0
661    }

```

Table 16 shows the resulting topology map. As a cross-check, the number of cases of the first cascade decay branch in the table equals that of the fifth cascade decay branch in Table 7.

Table 16: Signal cascade decay branches.

rowNo	signal cascade decay branch	iSigCascDcyBrs	nCase	nCCase
1	$B^0 \rightarrow D^{*-} D_s^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0, D_s^{*+} \rightarrow D_s^+ \gamma$	0	1119	1119
2	$B^0 \rightarrow D^{*-} D_s^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0$	1	1180	2299

#### 4.6. Inclusive decay branches

In a few cases, we have to identify some inclusive decay branches. Below is an example demonstrating the related item by taking the two inclusive decay branches  $\bar{B}^0 \rightarrow D^{*+} + \text{anything}$  and  $B^0 \rightarrow K_S^0 + \text{anything}$  as signals. Except that only two columns are involved in the item, the format of the input to the item is identical to that for the component analysis over inclusive decay branches, which is introduced in Section 3.8.

```

672    % Signal identification — inclusive decay branches
673    {
674      anti-B0 --> D*+ & B2Dsp
675      B0 --> K_S0 & B2Ks
676    }

```

The obtained topology map is displayed in Table 17. As a cross-check, the number of  $B^0 \rightarrow K_S^0 + \text{anything}$  in the table equals that in Table 9.

Table 17: Signal inclusive decay branches.

rowNo	signal inclusive decay branch	iSigIncDcyBr	nCase	nCCase
1	$\bar{B}^0 \rightarrow D^{*+} + \text{anything}$	0	41751	41751
2	$B^0 \rightarrow K_S^0 + \text{anything}$	1	153	41904

#### 4.7. Inclusive cascade decay branches

Occasionally, we may have interest in certain inclusive cascade decay branches. The following example shows the associated item with the two inclusive cascade decay branches  $\bar{B}^0 \rightarrow D^{*+} + \text{anything}$ ,  $D^{*+} \rightarrow \pi^+ D^0$  and  $B^0 \rightarrow K_S^0 J/\psi$ ,  $K_S^0 \rightarrow \pi^+ \pi^-$ ,  $J/\psi \rightarrow \mu^+ + \text{anything}$  set as signals. Similar to decay trees and cascade decay branches, inclusive cascade decay branches are made up of regular decay branches. Hence, the format of the input to the item for inclusive cascade decay branches is also identical to that for decay trees, which is introduced in Section 4.1. and the independent textual name “\*” denotes anything.

```

689    % Signal identification — inclusive cascade decay branches
690    {
691      0 & anti-B0 --> D*+ * & -1
692      1 & D*+ --> pi+ D0 & 0
693
694      0 & B0 --> K_S0 J/psi & -1
695      1 & K_S0 --> pi+ pi- & 0
696      2 & J/psi --> mu+ * & 0

```

697 }

698

699

Table 18 shows the resulting topology map.

Table 18: Signal inclusive cascade decay branches.

rowNo	signal inclusive cascade decay branch	iSigIncCascDcyBr	nCase	nCCase
1	$\bar{B}^0 \rightarrow D^{*+} + \text{anything}, D^{*+} \rightarrow \pi^+ D^0$	0	28367	28367
2	$B^0 \rightarrow K_S^0 J/\psi, K_S^0 \rightarrow \pi^+ \pi^-, J/\psi \rightarrow \mu^+ + \text{anything}$	1	1	28368

#### 700 4.8. Intermediate-resonance-allowed decay branches

701 On some occasions, we need to identify certain IRA decay branches. Below is an example  
 702 demonstrating the related item by taking the two IRA decay branches  $D^{*+} \rightarrow \pi^0 \pi^+ \pi^+ K^-$  and  
 703  $J/\psi \rightarrow \pi^0 \pi^+ \pi^-$  as signals. Except that only two columns are involved in the item, the format  
 704 of the input to the item is identical to that for the component analysis over IRA decay branches,  
 705 which is introduced in Section 3.9.

706 % Signal identification — intermediate-resonance-allowed decay branches

```
707 {
708   D*+ --> K- pi+ pi+ pi0 & Dsp2K3Pi
709   J/psi --> pi+ pi- pi0 & Jpsi23Pi
710 }
711
```

712

713 The obtained topology map is displayed in Table 19. For the purpose of cross-checks, we  
 714 note that the number of  $D^{*+} \rightarrow \pi^0 \pi^+ \pi^+ K^-$  in the table is equal to that in Table 10.

Table 19: Signal IRA decay branches.

rowNo	signal IRA decay branch	iSigIRADcyBr	nCase	nCCase
1	$D^{*+} \rightarrow \pi^0 \pi^+ \pi^+ K^-$	0	4971	4971
2	$J/\psi \rightarrow \pi^0 \pi^+ \pi^-$	1	59	5030

#### 715 4.9. Essential topology tags

716 Table 20 summarizes and explains all of the essential topology tags involved in the signal  
 717 identification functionalities. For signal decay trees and signal decay initial-final states, there  
 718 are two sorts of topology tags. The first sort of tags, iSigDcyTr and iSigDcyIFSts, record the  
 719 default indices of the specified signal decay trees and signal decay initial-final states. They have  
 720 the similar interpretation as iDcyTr and iDcyIFSts, and are shown in the third columns of Tables  
 721 12 and 13. The second sort of tags, nameSigDcyTr and nameSigDcyIFSts, save the specified  
 722 aliases of the signal decay trees and signal decay initial-final states. In cases the aliases are not  
 723 specified, empty strings will be stored.

724 For the latter six kinds of signal identification, there are only one sort of topology tags, which  
 725 record the number of the specified particles or decay branches found in each entry. Similar to  
 726 the cases in the latter seven kinds of component analysis, in the topology tags, “i” in “\_i” is the  
 727 default index of the specified particle or decay branch, and it ranges from 0 (included) to the  
 728 number of specified particles or decay branches (excluded). If the alias of the particle or decay  
 729 branch is also specified, the index “i” will be replaced with the alias.

Table 20: Essential topology tags involved in each kind of signal identification.

Signal identification kind	Topology tag	Interpretation
Decay trees	iSigDcyTr	index of signal decay tree
	nameSigDcyTr	name of signal decay tree
Decay initial-final states	iSigDcyIFSts	index of signal decay initial-final states
	nameSigDcyIFSts	name of signal decay initial-final states
Particles	nSigP_i	number of signal particle <sub>i</sub> s
Decay branches	nSigDcyBr_i	number of signal decay branch <sub>i</sub> es
Cascade decay branches	nSigCascDcyBr_i	number of signal cascade decay branch <sub>i</sub> es
Inclusive decay branches	nSigIncDcyBr_i	number of signal inclusive decay branch <sub>i</sub> es
Inclusive cascade decay branches	nSigIncCascDcyBr_i	number of signal inclusive cascade decay branch <sub>i</sub> es
IRA decay branches	nSigIRADcyBr_i	number of signal IRA decay branch <sub>i</sub> es

## 5. Common settings

From Sections 3 and 4, the optional parameters of the functionality items give us more choices and thus help us do our jobs quicker and better. In addition to these parameters, many optional items are designed and implemented to control the execution of the program in order to meet practical needs. Unlike the optional parameters, which only affect the individual functionalities to which they belong, the optional items have impact on all of the functionalities, or at least most of the functionalities. The current version of the program contains 24 commonly used items, which can be divided into the following three groups: items on the input of the program, items on the functionalities of the program, and items on the output of the program. This section introduces these items in the three groups, with each group in one subsection.

### 5.1. Settings on the input of the program

#### 5.1.1. Input entries

The program normally processes all of the entries in the input samples, but sometimes only a part of the entries are needed to be (first) processed. Running the program over a big sample usually takes a long time. In such a case, it is a good habit to run the program first over a small part of the sample to check possible exceptions, and then over the whole sample if no exceptions are found or after the found exceptions are handled. Besides, a small number of entries is usually sufficient to do tests in the developing of the program. For these reasons, an item is developed to set up the maximum number of entries to be processed. Below is an example showing the item with the maximum number set at two thousand.

```
% Maximum number of entries to be processed
{
    2000
}
```

On some occasions, especially in the course of optimizing selection criteria, we need to run the program only over entries satisfying certain requirements. For this purpose, an item is developed to select entries. The following example shows the item with X set in the range  $(-1, 1)$ .

```
% Cut to select entries
{
    (X > -1) && (X < 1)
}
```

Notably, only a single-line selection requirement is supported in the item, like the cases in the

766 methods Draw() and GetEntries() of the class TTree. In spite of this, such a requirement is able  
 767 to express any requirement with the help of the parentheses “()” as well as the logical symbols  
 768 “&&”, “||”, and “!”.

769 Occasionally, array variables are involved in the requirement. Under the circumstances, users  
 770 have to tell the program how to determine the total logical value with the individual logical  
 771 values. At present, two criteria are provided: (1) the total result is true as long as the result for  
 772 one instance is true; (2) the total result is false as long as the result for one instance is false. By  
 773 default, the second criterion is used in the program. One can alter it to the first one with the  
 774 following item.

```
775
776     % Method to apply cut to array variables (Two options: T and F. Default: F)
777     {
778         T
779     }
780
```

781 In the item, “T” and “F” stand for the first and second criteria, respectively.

### 782 5.1.2. Input decay branches

783 Normally, the program deals with all of the decay branches in every decay tree. However,  
 784 examining all the branches is not always required in practice. Sometimes, we only concern the  
 785 first  $n$  hierarchies of the branches. Similar to that in cascade decay branches of particles (as we  
 786 introduce in Section 3.6), the hierarchy here reflects the rank of a decay branch in a decay tree.  
 787 For example, in the decay tree  $\Upsilon(4S) \rightarrow B^0 \bar{B}^0$ ,  $B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F$ ,  $\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$ ,  $D^{*-} \rightarrow \pi^- \bar{D}^0$ ,  
 788  $D^{*+} \rightarrow \pi^+ D^0$ ,  $\bar{D}^0 \rightarrow \pi^0 \pi^- K^+$ ,  $D^0 \rightarrow \pi^0 \pi^+ K^-$ , the hierarchies of the seven individual branches  
 789 are 1, 2, 2, 3, 3, 4, and 4, respectively. The program provides an item to set the maximum hier-  
 790 archy. Below is an example showing the item with the maximum hierarchy set at one.

```
791
792     % Maximum hierarchy of heading decay branches to be processed in each event
793     {
794         1
795     }
796
```

797 With the setting, the decay branches with hierarchy larger than one will be ignored by the pro-  
 798 gram. For the component analysis over the decay trees of the  $\Upsilon(4S)$  sample, only the first hierar-  
 799 chy of  $\Upsilon(4S)$  decay branches are analyzed, and the result is shown in Table 21. From the table,  
 800 not only  $\Upsilon(4S) \rightarrow B^0 \bar{B}^0$  but also  $\Upsilon(4S) \rightarrow B^0 B^0$  and  $\Upsilon(4S) \rightarrow \bar{B}^0 \bar{B}^0$  are seen because of  $B^0$ - $\bar{B}^0$   
 801 mixing.

Table 21: Decay trees and their respective initial-final states.

rowNo	decay tree (decay initial-final states)	iDcyTr	nEtr	nCEtr
1	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0$ ( $\Upsilon(4S) \dashrightarrow B^0 \bar{B}^0$ )	0	81057	81057
2	$\Upsilon(4S) \rightarrow B^0 B^0$ ( $\Upsilon(4S) \dashrightarrow B^0 B^0$ )	1	9487	90544
3	$\Upsilon(4S) \rightarrow \bar{B}^0 \bar{B}^0$ ( $\Upsilon(4S) \dashrightarrow \bar{B}^0 \bar{B}^0$ )	2	9456	100000

802 Similarly, in the case of the maximum hierarchy set at two, we could get the result of the com-  
 803 ponent analysis over the first two hierarchies of  $\Upsilon(4S)$  decay branches, as displayed in Table  
 804 22.

Table 22: Decay trees and their respective initial-final states.

rowNo	decay tree (decay initial-final states)	iDcyTr	nEtr	nCEtr
1	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$ ( $\Upsilon(4S) \rightarrow \mu^+ \mu^- \nu_\mu \bar{\nu}_\mu D^{*+} D^{*-}$ )	936	136	136
2	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-}, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$ ( $\Upsilon(4S) \rightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu D^{*+} D^{*-}$ )	1188	112	248
3	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, \bar{B}^0 \rightarrow e^- \bar{\nu}_e D^{*+}$ ( $\Upsilon(4S) \rightarrow e^- \bar{\nu}_e \mu^+ \nu_\mu D^{*+} D^{*-}$ )	268	110	358
4	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow D^{*-} D_s^{*+}, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$ ( $\Upsilon(4S) \rightarrow \mu^- \bar{\nu}_\mu D^{*+} D^{*-} D_s^{*+}$ )	2063	72	430
5	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-}, \bar{B}^0 \rightarrow e^- \bar{\nu}_e D^{*+}$ ( $\Upsilon(4S) \rightarrow e^+ e^- \nu_e \bar{\nu}_e D^{*+} D^{*-}$ )	95	71	501
rest	$\Upsilon(4S) \rightarrow \text{others (81609 in total)}$ ( $\Upsilon(4S) \rightarrow \text{corresponding to others}$ )	—	99499	100000

Sometimes, we do not care about the decay of some particles. One can make the program ignore their decay branches with the following item. With the setting in the example, the decay of  $B^0$  and  $\bar{B}^0$  will be ignored by the program.

```

805
806
807
808
809 % Ignore the decay of the following particles
810 {
811     B0
812     anti-B0
813 }
814

```

At some other times, we have interest in the decay of some particles but not in the decay of their daughters. To handle this case, the following item is developed to make the program ignore the decay of their daughters. In the following example, the decay of the daughters of  $B^0$  and  $\bar{B}^0$  will be ignored by the program.

```

815
816
817
818
819
820 % Ignore the decay of the daughters of the following particles
821 {
822     B0
823     anti-B0
824 }
825

```

The two settings above have the same effects as those in the previous paragraph which set the maximum hierarchy at one and two, and hence the corresponding results are identical to those shown in Tables 21 and 22.

As mentioned in Section 2.4, the decay  $\pi^0 \rightarrow \gamma\gamma$  is ignored by default. On the occasions when we need to identify the signals involving the decay, we can make the program retain the decay with the item below set to “Y”.

```

832
833 % Retain the decay of pi0 to gamma gamma (Two options: Y and N. Default: N)
834 {
835     Y
836 }
837

```

Besides, if needed, one can make the program ignore other final decay branches, such as  $\eta \rightarrow \gamma\gamma$  and  $K_S^0 \rightarrow \pi^+ \pi^-$ , with the following item.

```

840
841 % Ignore the following final decay branches
842 {

```



```

843     eta --> gamma gamma
844     K_S0 --> pi+ pi-
845 }

```

### 846 5.1.3. Initial and final state radiation photons

847 Initial state radiation (ISR) and final state radiation (FSR) are inevitable physical effects in  
848  $e^+e^-$  colliding experiments. Therefore, ISR and FSR photons are often involved in the inclusive  
849 MC samples. The program processes them together with other particles in the default case. To  
850 distinguish them from other photons, the program tries to label them in the output txt, tex and pdf  
851 files. Sometimes, these photons are marked out beforehand with special PDG codes according  
852 to particle status information from generators. One can inform the program of these PDG codes  
853 by the following two items.

```

854
855 % PDG code of ISR photons (Default: 222222222)
856 {
857     222222222
858 }
859
860 % PDG code of FSR photons (Default: -22)
861 {
862     -22
863 }
864
865

```

866 In this case, the program is able to label the ISR and FSR photons as  $\gamma^i$  (gammai) and  $\gamma^f$  (gamma-  
867 maf) in the output pdf (txt) files, respectively.

868 On other occasions, ISR and FSR photons are not marked out in advance due to some reasons.  
869 In such cases, the program have to identify them by itself according to the following rules:  
870 photons which have no mothers recorded in the arrays of the PDG codes and mother indices  
871 are considered as generalized ISR photons, while other photons which have at least one  $e^\pm$ ,  $\mu^\pm$ ,  
872  $\pi^\pm$ ,  $K^\pm$ ,  $p$ , or  $\bar{p}$  sister are taken as generalized FSR photons. Here, the modifier “generalized”  
873 is used because the rules can not determine the types of the photons in absolute accuracy. For  
874 example, photons from radiative decays might be mistaken as ISR and FSR photons. Despite this,  
875 generalized ISR and FSR photons are good concepts, particularly in cases where the sources of  
876 the photons are not required to be distinguished clearly. The program will label the generalized  
877 ISR and FSR photons as  $\gamma^I$  (gammaI) and  $\gamma^F$  (gammaF) in the output pdf (txt) files, respectively.

878 Notably, we are not concerned about these ISR and FSR photons in many cases. If they have  
879 already been marked out beforehand, one can make the program ignore them accurately by set-  
880 ting the following two items to “Ys”.

```

881
882 % Ignore ISR photons (Three options: Ys, Yg and N. Default: N)
883 {
884     Ys
885 }
886
887
888 % Ignore FSR photons (Three options: Ys, Yg and N. Default: N)
889 {
890     Ys
891 }
892

```

893 In cases that these photons are not marked in advance, the option “Yg” can be used to ignore the  
894 generalized ISR and FSR photons. In “Ys” and “Yg”, “s” and “g” are the initials of the words  
895 “strict” and “generalized”, respectively.

## 896 5.2. Settings on the functionalities of the program

### 897 5.2.1. Candidate based analysis

898 According to the number of signal candidates in an event that are selected and retained to  
899 extract physics results, data analysis in high energy experiments can be divided into the following  
900 two categories: event based analysis and candidate based analysis. While at most one candidate  
901 in an event is kept in event based analysis, one or more candidates in an event can be retained in  
902 candidate based analysis. Generally, the quantities related to a candidate are stored in an entry of  
903 the TTree objects in the root files. Thus, one or more entries relate to an event in candidate based  
904 analysis, while only one entry corresponds to an event in event based analysis. Normally, the  
905 indices of candidates within an event are stored in the corresponding entries in candidate based  
906 analysis.

907 By default, the program analyzes the input entries one by one. In this case, the events with  
908 multiple candidates will be processed repeatedly. Particularly, the number of physics processes  
909 at the sample level will be over counted. One can make the program avoid the problem by in-  
910 putting “Y” to the following item.

```
911  
912 % Avoid over counting for candidate based analysis (Two options: Y and N. Default: N)  
913 {  
914     Y  
915 }
```

916 Also, the indices of candidates within an event are required. We can tell the program the related  
917 TBranch name with the following item.

```
918  
919  
920 % TBranch name of the indices of candidates in an event (Default: __candidate__)  
921 {  
922     iCandidate  
923 }
```

924  
925 With the settings, the program will process the first entry of each event in a normal way, including  
926 obtaining and storing the topology tags; it will not analyze the other entries of the same event,  
927 but only store the same topology tags to them.

### 928 5.2.2. Charge conjugation

929 Charge conjugation is an important concept in high energy physics. By default, charge con-  
930 jugate objects (particles and decays) are processed separately in the program. However, we need  
931 to handle them together in many physics studies because of the sameness between them. One  
932 can have the program process them together with the item below set to “Y”.

```
933  
934 % Process charge conjugate objects together (Two options: Y and N. Default: N)  
935 {  
936     Y  
937 }
```

938  
939 Performing topology analysis with the setting inserts new topology tags in the output root files  
940 and adds new counters to topology maps in the output txt, tex, and pdf files. Tables 23 and 24  
941 list and interpret all of the topology tags related to charge conjugation involved in the component  
942 analysis and signal identification functionalities, respectively.

943 As an example, we perform the component analysis over decay trees with the charge con-  
944 jugate item. Table 25 shows the obtained topology map. Besides the columns in Table 2, two  
945 additional columns with the headers “nCCEtr” and “nAllEtr” are inserted in the table. Here, “nC-  
946 cEtr” represents the number of entries involving the charge conjugate decay trees, and “nAllEtr”

Table 23: Topology tags related to charge conjugation involved in each kind of component analysis. For the latter seven kinds of component analysis, the topology tags in the 1) and 2) groups are only designed for the self-charge-conjugate and non-self-charge-conjugate particles and decay branches, respectively. The acronyms “cc” and  $\text{index}_{\text{cc}}$  are short for “charge conjugate” and “charge conjugate index”, respectively.

Component analysis kind	Topology tag	Interpretation
Decay trees	iCcDcyTr	$\text{index}_{\text{cc}}$ of decay tree
Decay initial-final states	iCcDcyIFSts	$\text{index}_{\text{cc}}$ of decay initial-final states
Decay branches of particles	iCcPDcyBr.i	$\text{index}_{\text{cc}}$ of particle <sub>i</sub>
	1) iCcDcyBrP.i.j	$\text{index}_{\text{cc}}$ of decay branch of the j <sup>th</sup> particle <sub>i</sub>
	2) nCcPDcyBr.i	number of cc particle <sub>i</sub> s (decay branches)
	2) iDcyBrCcP.i.j	index of decay branch of the j <sup>th</sup> cc particle <sub>i</sub>
	2) nAllPDcyBr.i	number of all particle <sub>i</sub> s (decay branches)
Production branches of particles	iCcPProdBr.i	$\text{index}_{\text{cc}}$ of particle <sub>i</sub>
	1) iCcProdBrP.i.j	$\text{index}_{\text{cc}}$ of production branch of the j <sup>th</sup> particle <sub>i</sub>
	2) nCcPProdBr.i	number of cc particle <sub>i</sub> s (production branches)
	2) iProdBrCcP.i.j	index of production branch of the j <sup>th</sup> cc particle <sub>i</sub>
	2) nAllPProdBr.i	number of all particle <sub>i</sub> s (production branches)
Mothers of particles	iCcPMoth.i	$\text{index}_{\text{cc}}$ of particle <sub>i</sub>
	1) iCcMothP.i.j	$\text{index}_{\text{cc}}$ of mother of the j <sup>th</sup> particle <sub>i</sub>
	2) nCcPMoth.i	number of cc particle <sub>i</sub> s (mothers)
	2) PDGMothCcP.i.j	PDG code of mother of the j <sup>th</sup> cc particle <sub>i</sub>
	2) nAllPMoth.i	number of all particle <sub>i</sub> s (mothers)
Cascade decay branches of particles	iCcPCascDcyBr.i	$\text{index}_{\text{cc}}$ of particle <sub>i</sub>
	1) iCcCascDcyBrP.i.j	$\text{index}_{\text{cc}}$ of cascade decay branch of the j <sup>th</sup> particle <sub>i</sub>
	2) nCcPCascDcyBr.i	number of cc particle <sub>i</sub> s (cascade decay branches)
	2) iCascDcyBrCcP.i.j	index of cascade decay branch of the j <sup>th</sup> cc particle <sub>i</sub>
	2) nAllPCascDcyBr.i	number of all particle <sub>i</sub> s (cascade decay branches)
Decay final states of particles	iCcPDcyFSt.i	$\text{index}_{\text{cc}}$ of particle <sub>i</sub>
	1) iCcDcyFStP.i.j	$\text{index}_{\text{cc}}$ of decay final state of the j <sup>th</sup> particle <sub>i</sub>
	2) nCcPDcyFSt.i	number of cc particle <sub>i</sub> s (decay final states)
	2) iDcyFStCcP.i.j	index of decay final state of the j <sup>th</sup> cc particle <sub>i</sub>
	2) nAllPDcyFSt.i	number of all particle <sub>i</sub> s (decay final states)
Inclusive decay branches	iCcIncDcyBr.i	$\text{index}_{\text{cc}}$ of inclusive decay branch <sub>i</sub>
	1) iCcDcyBrIncDcyBr.i.j	$\text{index}_{\text{cc}}$ of decay branch of the j <sup>th</sup> inclusive decay branch <sub>i</sub>
	2) nCcIncDcyBr.i	number of cc inclusive decay branch <sub>i</sub> es
	2) iDcyBrCcIncDcyBr.i.j	index of decay branch of the j <sup>th</sup> cc inclusive decay branch <sub>i</sub>
	2) nAllIncDcyBr.i	number of all inclusive decay branch <sub>i</sub> es
IRA decay branches	iCcIRADcyBr.i	$\text{index}_{\text{cc}}$ of IRA decay branch <sub>i</sub>
	1) iCcDcyBrIRADcyBr.i.j	$\text{index}_{\text{cc}}$ of decay branch of the j <sup>th</sup> IRA decay branch <sub>i</sub>
	2) nCcIRADcyBr.i	number of cc IRA decay branch <sub>i</sub> es
	2) iDcyBrCcIRADcyBr.i.j	index of decay branch of the j <sup>th</sup> cc IRA decay branch <sub>i</sub>
	2) nAllIRADcyBr.i	number of all IRA decay branch <sub>i</sub> es

Table 24: Topology tags related to charge conjugation involved in each kind of signal identification. For the latter six kinds of signal identification, the topology tags in the \*) groups are only designed for the non-self-charge-conjugate particles and decay branches. The acronyms “cc” and  $\text{index}_{\text{cc}}$  are short for “charge conjugate” and “charge conjugate index”, respectively.

Signal identification kind	Topology tag	Interpretation
Decay trees	iCcSigDcyTr	$\text{index}_{\text{cc}}$ of signal decay tree
Decay initial-final states	iCcSigDcyIFSts	$\text{index}_{\text{cc}}$ of signal decay initial-final states
Particles	iCcSigP.i	$\text{index}_{\text{cc}}$ of signal particle <sub>i</sub>
	*) nCcSigP.i	number of cc signal particle <sub>i</sub> s
	*) nAllSigP.i	number of all signal particle <sub>i</sub> s
Decay branches	iCcSigDcyBr.i	$\text{index}_{\text{cc}}$ of signal decay branch <sub>i</sub>
	*) nCcSigDcyBr.i	number of cc signal decay branch <sub>i</sub> es
	*) nAllSigDcyBr.i	number of all signal decay branch <sub>i</sub> es
Cascade decay branches	iCcSigCascDcyBr.i	$\text{index}_{\text{cc}}$ of signal cascade decay branch <sub>i</sub>
	*) nCcSigCascDcyBr.i	number of cc signal cascade decay branch <sub>i</sub> es
	*) nAllSigCascDcyBr.i	number of all signal cascade decay branch <sub>i</sub> es
Inclusive decay branches	iCcSigIncDcyBr.i	$\text{index}_{\text{cc}}$ of signal inclusive decay branch <sub>i</sub>
	*) nCcSigIncDcyBr.i	number of cc signal inclusive decay branch <sub>i</sub> es
	*) nAllSigIncDcyBr.i	number of all signal inclusive decay branch <sub>i</sub> es
Inclusive cascade decay branches	iCcSigIncCascDcyBr.i	$\text{index}_{\text{cc}}$ of signal inclusive cascade decay branch <sub>i</sub>
	*) nCcSigIncCascDcyBr.i	number of cc signal inclusive cascade decay branch <sub>i</sub> es
	*) nAllSigIncCascDcyBr.i	number of all signal inclusive cascade decay branch <sub>i</sub> es
IRA decay branches	iCcSigIRADcyBr.i	$\text{index}_{\text{cc}}$ of signal IRA decay branch <sub>i</sub>
	*) nCcSigIRADcyBr.i	number of cc signal IRA decay branch <sub>i</sub> es
	*) nAllSigIRADcyBr.i	number of all signal IRA decay branch <sub>i</sub> es

is the sum of “nEtr” and “nCcEtr”. In addition to “iDcyTr”, “iCcDcyTr” is also inserted in the output root files as a topology tag. It is short for charge conjugate index of decay tree. For self-charge-conjugate decay trees, it has the value 0; for non-self-charge-conjugate decay trees, it has the value 1 or  $-1$ : while 1 tags the decay trees listed in the topology maps,  $-1$  indicates their charge conjugate decay trees. Whereas the values of “iDcyTr” for each decay tree and its charge conjugate decay tree are equal in order to indicate their sameness, the values of “iCcDcyTr” for them are opposite so as to reflect their difference.

As another example, we carry out the component analysis over the decay branches of  $D^{*+}$ . The resulting topology map of  $D^{*+}$  is displayed in Table 26. Compared with Table 4, two new columns are added to the table, and their headers “nCcCase” and “nAllCase” have the similar meanings as “nCcEtr” and “nAllEtr” in Table 25. For a specified particle, what we want to further record with topology tags are as follows: (1) whether it is self-charge-conjugate; (2) whether its decay branches are self-charge-conjugate, if it is self-charge-conjugate; (3) the number and the indices of the decay branches of its charge-conjugate particle, if it is not self-charge-conjugate. Hence, in addition to “nPDcyBr.i” and “iDcyBrP.i.j”, the following topology tags are also inserted in the output root files: “iCcPDcyBr.i” for all specified particles; “iCcDcyBrP.i.j” for self-charge-conjugate particles only; and “nCcPDcyBr.i”, “iDcyBrCcP.i.j”, and “nAllPDcyBr.i” for non-self-charge-conjugate particles only. Here, “iCcPDcyBr.i” is short for charge conjugate index of the  $i^{\text{th}}$  particle specified for its decay branches. For self-charge-conjugate particles, it has the value 0; for non-self-charge-conjugate particles, it has the value 1.

Table 25: Decay trees and their respective initial-final states.

rowNo	decay tree (decay initial-final states)	iDcyTr	nEtr	nCcEtr	nAllEtr	nCEtr
1	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $D^{*+} \rightarrow \pi^+ D^0, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) \rightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu \pi^0 \pi^0 \pi^+ \pi^- \pi^- K^+ K^- \gamma^F)$	20870	3	0	3	3
2	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^-, \bar{B}^0 \rightarrow e^- \bar{\nu}_e D^{*+}, D^- \rightarrow e^- \bar{\nu}_e \pi^- K^+,$ $D^{*+} \rightarrow \pi^+ D^0, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) \rightarrow e^- e^+ \bar{\nu}_e \nu_e \mu^+ \nu_\mu \pi^0 \pi^+ \pi^+ \pi^- K^+ K^-)$	3722	1	1	2	5
3	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \pi^0 \pi^+ \pi^+ \rho^- D^-, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, \rho^- \rightarrow \pi^0 \pi^-,$ $D^- \rightarrow \pi^- \pi^- K^+, D^{*+} \rightarrow \pi^+ D^0, D^0 \rightarrow K_L^0 \pi^+ \pi^-$ $(\Upsilon(4S) \rightarrow \mu^- \bar{\nu}_\mu \pi^0 \pi^0 K_L^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+ K^-)$	5295	2	0	2	7
4	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F, \bar{B}^0 \rightarrow \pi^0 \pi^+ \pi^- \pi^- D^{*+},$ $D^{*-} \rightarrow \pi^0 D^-, D^{*+} \rightarrow \pi^+ D^0, D^- \rightarrow \pi^- \pi^- K^+, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) \rightarrow e^+ \nu_e \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+ K^- \gamma^F)$	10206	1	1	2	9
5	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, \bar{B}^0 \rightarrow \mu^+ \nu_\mu D^{*+}, D^{*-} \rightarrow \pi^0 D^-,$ $D^{*+} \rightarrow \pi^+ \bar{D}^0, D^- \rightarrow \pi^0 \pi^- K_S^0, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, K_S^0 \rightarrow \pi^+ \pi^-$ $(\Upsilon(4S) \rightarrow e^+ \mu^+ \nu_\mu \nu_\mu \pi^0 \pi^0 \pi^0 \pi^+ \pi^- \pi^- \pi^- K^+)$	11916	1	1	2	11
rest	$\Upsilon(4S) \rightarrow \text{others (99969 in total)}$ $(\Upsilon(4S) \rightarrow \text{corresponding to others})$	—	—	—	99989	100000

The topology tag “iCcDcyBrP.i.j” denotes charge conjugate index of decay branch of the  $j^{\text{th}}$  instance of the  $i^{\text{th}}$  particle. It is to “iDcyBrP.i.j” what “iCcDcyTr” is to “iDcyTr”. The topology tag “iDcyBrCcP.i.j” has the similar meaning as “iDcyBrP.i.j”, but it is designed for the charge conjugate particle of the  $i^{\text{th}}$  particle. Particularly, it ranges from 0 (included) to the number of the categories of decay branches of the  $i^{\text{th}}$  particle found in the sample (excluded). The topology tag “nCcPDcyBr.i” stands for the number of the charge conjugate  $i^{\text{th}}$  particles (or their decay branches) found in each entry, and “nAllPDcyBr.i” is the sum of “nPDcyBr.i” and “nCcPDcyBr.i”.

Table 26: Decay branches of  $D^{*+}$ .

rowNo	decay branch of $D^{*+}$	iDcyBrP	nCase	nCcCase	nAllCase	nCCCase
1	$D^{*+} \rightarrow \pi^+ D^0$	0	31180	31291	62471	62471
2	$D^{*+} \rightarrow \pi^0 D^+$	1	13978	14166	28144	90615
3	$D^{*+} \rightarrow D^+ \gamma$	2	700	721	1421	92036
4	$D^{*+} \rightarrow \pi^+ D^0 \gamma^F$	3	28	36	64	92100
5	$D^{*+} \rightarrow \pi^0 D^+ \gamma$	4	0	1	1	92101

### 5.2.3. Settings only on signal identification

Normally, the signals specified in the signal identification functionality items are both tagged and counted by executing the program one time. In the case of a huge sample that will take a long time, it is a good idea to first tag the signals with multiple jobs each running on one machine, and then count the tagged signals together. One can make the program carry out the idea by setting the following item to “T” and “C” in the first and second steps, respectively. Here, “T” and “C” stand for tagging and counting, respectively.

```
% Analysis tasks for signal identifications (Three options: TC, T and C. Default: TC)
{
  T
}
```

By default, the signals set in the signal identification functionality items are listed in the out-

989 put txt, tex, and pdf files in the sequence they are specified. In cases of plenty of signals, there is  
 990 probably a need to sort them according to the number of cases found in the input samples. One  
 991 can have the program do the sorting by inputting “Y” to the item below.

```
992
993     % Sort the signals in the topology maps related to signal identifications (Two options: Y and N. Default: N)
994     {
995         Y
996     }
```

### 997 5.3. Settings on the output of the program

998 By default, decay objects (trees, initial-final states, and branches) are left-aligned in the out-  
 999 put pdf files. If one likes it, he/she can request the program to center them by setting the following  
 1000 item to “Y”.

```
1001
1002     % Center decay objects in output pdf files (Two options: Y and N. Default: N)
1003     {
1004         Y
1005     }
```

1006  
 1007 As mentioned in Section 2.4, after the execution of the program, one or more root files will  
 1008 be output to save topology tags. By default, the program switches to a new output file whenever  
 1009 the size of the TTree object in memory exceeds 3 GB. In addition to this, the program provides  
 1010 an item to control the switch of output files by setting the maximum number of entries to be  
 1011 saved in a single output file. The following example shows the item with the maximum number  
 1012 set to 1 million.

```
1013
1014     % Maximum number of entries to be saved in a single output root file
1015     {
1016         1000000
1017     }
```

1018  
 1019 Besides, one can have the program generate one output file by one input file with the following  
 1020 item set to “Y”.

```
1021
1022     % One output root file by one input root file (Two options: Y and N. Default: N)
1023     {
1024         Y
1025     }
```

1026  
 1027 In default cases, flat TBranches are used to store topology tags in the output root files. This is  
 1028 necessary for the Belle II experiment, as array TBranches are not recommended to use in physics  
 1029 analyses in order to use other tools such as NumPy [10] and pandas [11]. However, since array  
 1030 TBranches are elegant and efficient in organizing and storing homogeneous data, sometimes it is  
 1031 better to use them than flat TBranches in other experiments, such as the BESIII experiment. One  
 1032 can make the program use array TBranches to store topology tags by inputting “Y” to the item  
 1033 below.

```
1034
1035     % Use array TBranches to store topology tags in output root files when possible (Two options: Y and N. Default: N)
1036     {
1037         Y
1038     }
```

1039  
 1040 By default, to facilitate the validation of topology analysis results, the input TBranches are  
 1041 copied to the output root files along with other TBranches for physics analyses. However, they  
 1042 often occupy too much disk space and are useless for following physics analyses. In the case of  
 1043 being flat, a massive amount of these TBranches also looks awkward. Thus, after the validation  
 1044 with a small sample, it would be better to remove these TBranches from the output root files.  
 1045 One can request the program to perform this removal operation before it terminates by setting

1046 the following item to “Y”.

```
1047
1048 % Remove the input tbranches from output root files (Two options: Y and N. Default: N)
1049 {
1050     Y
1051 }
```

## 1052 6. Auxiliary facilities

1053 This section introduces some auxiliary facilities for the use of the program, including a card  
1054 file to preset frequently used items and two commands implemented in tex source files. Different  
1055 from that presented in the previous four sections, the content presented in this section is not the  
1056 essential part of the program. However, with these auxiliary facilities, we can make the program  
1057 do our jobs better and quicker on some occasions.

### 1058 6.1. The underlying card file

1059 A card file, namely “underlying\_topoana.card” under the directory “share”, to preset fre-  
1060 quently used items is developed to assist the card file specified by the first argument of the  
1061 command “topoana.exe”. Here, we refer to the former and latter card files as underlying and  
1062 primary, respectively. In general, the primary card file is sufficient to set items for the execution  
1063 of the program. However, considering some items are frequently used with constant inputs by a  
1064 user or a group of users, it is better to move the items from the primary card file to the underlying  
1065 card file, in order to make the primary card file more concise and make us more focused on the  
1066 items specially set for the dedicated topology analysis.

1067 One can decide whether to set an item in the underlying card file according to his/her own  
1068 needs. Here, we introduce some frequently used items that are suitable to be put in the underlying  
1069 card file as follows. As mentioned in Section 2.3, the items related to the storage type and  
1070 TBranches names of the input data are usually fixed for a user or a group of users. Thus, it is  
1071 quite appropriate to move them to the underlying card file. We have to process charge conjugation  
1072 particles and decays together in many physics studies. In such studies, it is also a good practice  
1073 to put the item on charge conjugation in the underlying card file.

1074 The program first reads the items in the underlying card file and then reads those in the  
1075 primary card file. The items set in the underlying card file can be reset in the primary card file.  
1076 In such a case, the inputs in the underlying card file will be replaced by their counterparts in the  
1077 primary card file.

### 1078 6.2. Commands implemented in tex source files

1079 The output pdf files can be checked after the execution of the program. If their styles are not  
1080 to our taste, we can edit the corresponding tex source files to get the desired styles, according  
1081 to the regular LaTeX rules. Besides the rules, two commands are implemented in the tex source  
1082 files to help us edit the files quickly and easily for two common desired styles.

1083 By default, topology tags are listed along with topology maps in the output txt, tex, and pdf  
1084 files. However, only the topology maps are needed on some occasions, especially in presenta-  
1085 tions. In such cases, one can suppress the topology tags in the output tex and pdf files by simply  
1086 changing the definition of the `cmtTopoTags` command from the nominal one

```
1087
1088 \newcommand{\topoTags}[1]{#1}
```

1089

1090 to the alternative one

1091

1092

`\newcommand{\topoTags}[1]{}{}`

1093

1094 in the preamble of the text source files. Here, “#1” is the formal parameter of the string for  
 1095 the topology tags. With the nominal definition, “\topoTags{#1}” returns the string exactly, while  
 1096 with the alternative definition it only returns an empty string. That is why the definition below is  
 1097 able to suppress the topology tags.

1098

1099 After the revision of the tex source files, one can re-compile them with the pdflatex command.  
 1099 Usually, the pdflatex command has to be executed two or three times for a fully compiled pdf  
 1100 file, and many undesired files in other formats are generated during the compilation. To execute  
 1101 the pdflatex command and remove the undesired files at one stroke, we develop a bash script,  
 1102 namely “getPdfFromTex.sh” under the directory “utilities”. The script should be executed with  
 1103 the following command line: `getPdfFromTexFl.sh texFileName`. Compiling the tex source  
 1104 files with the script is recommended.

## 1105 7. Summary

1106 We develop a program, namely TopoAna, with C++, ROOT and LaTeX for the topology  
 1107 analysis of inclusive MC samples in high energy physics experiments at  $e^+e^-$  colliders. This  
 1108 paper provides an essential description of the program, including a basic introduction of the pro-  
 1109 gram, two sorts of functionalities of the program — component analysis and signal identification,  
 1110 and some common settings and auxiliary facilities for the executing of the program.

1111 Since it does not rely on any specific software frameworks, the program applies to many  
 1112 high energy physics experiments. Up to now, it has been put into use in three experiments at  
 1113  $e^+e^-$  colliders: the BESIII, Belle, and Belle II experiments. Besides these experiments, it can  
 1114 also be used in the PANDA experiment [13], which is an anti-proton annihilation experiment  
 1115 under construction at Darmstadt, Germany. In addition, the program is also applicable to the  
 1116 pre-research of future  $e^+e^-$  colliding experiments, such as the circular electron positron collider  
 1117 (CEPC) [14, 15] experiment in China, the super charm-tau factory (SCTF) experiment [16] in  
 1118 Russia, and the super tau-charm factory (STCF) experiment [17] in China. These experiments  
 1119 offer a wide range of potential uses of the program. With more user needs coming out in the  
 1120 future, we will further extend and perfect it to make it more powerful and well-rounded.

## 1121 Acknowledgements

1122 This project is supported by the National Natural Science Foundation of China under Grants  
 1123 No. 11575017, No. 11661141008, No. 11761141009, and No. 11975076; the CAS Center for  
 1124 Excellence in Particle Physics (CCEPP). In addition, we would like to thank all of the people  
 1125 who have helped us in the development of the program. We first thank Prof. Changzheng Yuan,  
 1126 Bo Xin, and Haixuan Chen for their help at the early stage of developing the program. We are  
 1127 particularly grateful to Prof. Xingtao Huang for his comments on the principles and styles of the  
 1128 program, to Remco de Boer for his suggestions on the tex output and the use of GitHub, and to Xi  
 1129 Chen for his discussions on the core algorithms. We are especially indebted to Prof. Xiqing Hao,  
 1130 Longke Li, Xiaoping Qin, Ilya Komarov, Yubo Li, Guanda Gong, Suxian Li, Junhao Yin, Prof.



1131 Xiaolong Wang, and Yeqi Chen for their advice in extending and perfecting the program. Also,  
 1132 we thank Xi'an Xiong, Runqiu Ma, Wencheng Yan, Sen Jia, Hongpeng Wang, Jiawei Zhang,  
 1133 Hongrong Qi, Jiajun Liu, Maoqiang Jing, Yi Zhang, Wei Shan, and Yadi Wang for their efforts  
 1134 in helping us test the program.

## 1135 References

- 1136 [1] ROOT User's Guide, Available online: <https://root.cern/root/html/doc/guides/users-guide/ROOTUsersGuide.html>.
- 1137 [2] K.T. Chao, Y.F. Wang, et al., Physics at BES-III, *Int. J. Mod. Phys. A* 24 (2009) S1-794.
- 1138 [3] M. Ablikim, et al. (BESIII Collaboration), White Paper on the Future Physics Programme of BESIII, [arXiv:1912.05983](https://arxiv.org/abs/1912.05983).
- 1139 [4] E. Kou, et al., *Prog. Theor. Exp. Phys.* 2019 (2019) 123C01.
- 1140 [5] J. Brodzicka, T. Browder, P. Chang, et al., *Prog. Theor. Exp. Phys.* 2012 (2012) 04D001.
- 1141 [6] Documentation of the TFile class, Available online: <https://root.cern/root/html534/TFile.html>.
- 1142 [7] Documentation of the TTree class, Available online: <https://root.cern/root/html534/TTree.html>.
- 1143 [8] M. Tanabashi, et al. (Particle Data Group), *Phys. Rev. D* 98 (2018) 030001.
- 1144 [9] Documentation of the TBranch class, Available online: <https://root.cern/root/html534/TBranch.html>.
- 1145 [10] Documentation of NumPy, Available online: <https://numpy.org/devdocs/>.
- 1146 [11] Documentation of pandas, Available online: <https://pandas.pydata.org/pandas-docs/stable/>.
- 1147 [12] Documentation of the TChain class, Available online: <https://root.cern/root/html534/TChain.html>.
- 1148 [13] W. Erni, et al. (PANDA Collaboration), Physics Performance Report for PANDA: Strong Interaction Studies with  
 1149 Antiprotons, [arXiv:0903.3905](https://arxiv.org/abs/0903.3905).
- 1150 [14] CEPC CDR Volume 1 (Accelerator), Available online: [http://cepc.ihep.ac.cn/CEPC\\_CDR\\_Vol1\\_Accelerator.pdf](http://cepc.ihep.ac.cn/CEPC_CDR_Vol1_Accelerator.pdf).
- 1151 [15] CEPC CDR Volume 2 (Physics & Detector), Available online: [http://cepc.ihep.ac.cn/CEPC\\_CDR\\_Vol2\\_Physics-Detector.pdf](http://cepc.ihep.ac.cn/CEPC_CDR_Vol2_Physics-Detector.pdf).
- 1152 [16] A.E. Bondar, et al. (Charm-Tau Factory Collaboration), *Phys. Atom. Nucl.* 76 (2013) 1072.
- 1153 [17] Q. Luo, D. Xu, "Progress on Preliminary Conceptual Study of HIEPA, a Super Tau-Charm Factory in China", in  
 1154 Proc. 9th International Particle Accelerator Conf. (IPAC2018), Vancouver, BC, Canada, 422.
- 1155