

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

Xingyu Zhou^{a,*}, Shuxian Du^b, Gang Li^c, Chengping Shen^{d,*}

^a*School of Physics, Beihang University, Beijing 100191, China*

^b*School of Physics and Microelectronics, Zhengzhou University, Zhengzhou 450000, China*

^c*Institute of High Energy Physics, Chinese Academy of Sciences, Beijing 100049, China*

^d*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) samples¹ 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>.

¹The 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 τ -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 paper gives an essential 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; Section 5 presents some common settings for the executing of the program, respectively; Section 6 summarizes the paper. It is worth mentioning here that, aside from the essential description in the paper, a detailed description of the program can be found in the file “user_guide_v*.pdf” under

68 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 paper 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 three files — “README.md”, “Configure”, and “Makefile”. While the direc-
91 tory “include” only includes one header file “topoana.h”, the directory “src” contains sixty source
92 files “*.cpp” as well as a script file “topoana.C”. Despite the largeness of the program, only one
93 class “topoana” is defined for all of its functionalities. The class is declared in “topoana.h”,
94 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_txtpnm_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_txtpnm_texpnm_iccp.dat” when we configure 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 the
106 program, one should first set the experiment name and the package path with the command line
107 “./Configure expNm”, where “expNm” is short for experiment name. Output of the command
108 are the guidelines for manually adding the absolute path of “topoana.exe” to the environment
109 variable “PATH”, in order to execute it without any path. The second step to set up the program
110 is executing the command “make”. This command compiles the header, source, and script files

into the executable file “topoana.exe” under the directory “bin”, according to the rules specified in the “Makefile”. Notably, executing the command once is sufficient, unless the header, source, or script files are updated or revised, or the package is moved.

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).

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

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

120 to discard the decay $\pi^0 \rightarrow \gamma\gamma$ by default at the early phase of processing the input data. Since the
 121 topology diagram of such a process looks like a tree, we refer to the complete processes as decay
 122 trees. Obviously, the data do not show the structure automatically. Thus, we need the program
 123 to do the topology analysis work.

124 The input data are recommended to be saved in the TTree object together with other quantities
 125 for physics analyses, in order to facilitate the examination of the distributions of the quantities
 126 with the topology information. It is easy to get the input of the program within the software
 127 framework of high energy physics experiments. In order to facilitate its use, we have developed
 128 the interfaces of the program to the software systems of the BESIII, Belle, and Belle II experi-
 129 ments. Similar interfaces for other experiments can also be implemented with ease. Beyond the
 130 scope of the paper, we will not discuss the details of the interfaces here.

131 2.3. Execution of the program

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

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

```

139 # The following five items set the input of the program.
140
141 % Names of input root files
142 {
143   ../input/jpsi_1.root
144   ../input/jpsi_2.root
145 }
146
147 % TTree name
148 {
149   evt
150 }
151
152 % TBranch name of the number of particles (Default: nMCGen)
153 {
154   Nmcp
155 }
156
157 % TBranch name of the PDG codes of particles (Default: MCGenPDG)
158 {
159   Pid
160 }
161
162 % TBranch name of the mother indices of particles (Default: MCGenMothIndex)
163 {
164   Midx
165 }
166
167 # The following item sets the basic functionality of the program.
168
169 % Component analysis — decay trees
170
171
```

```

172     {
173         Y
174     }
175
176     # The following item sets the output of the program.
177
178     % Main name of output files (Default: Main name of the card file)
179     {
180         jpsi.ta
181     }
182

```

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

186 The first item sets the names of the input root files. The names ought to be input one per
187 line without tailing characters, such as comma, semicolon, and period. In the names, both the
188 absolute and relative paths are allowed and wildcards “[]?*” are supported, just like those in the
189 root file names input to the method Add() of the class TChain [10]. The second item specifies
190 the TTree name. The following three items set the TBranch names of the three ingredients of
191 the raw input topology data. Of the first five items, the former two are indispensable, whereas
192 the latter three can be removed or left empty if the input values are identical to the default values
193 indicated in their prompts.

194 The sixth item sets the basic functionality of the program, namely the component analysis
195 over decay trees. The item can be replaced or co-exist with other functionality items expatiated in
196 Sections 3 and 4. Here, we note that at least one functionality item has to be specified explicitly
197 in the card file, otherwise the program will terminate soon after its start because no topology
198 analysis to be performed is set up.

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

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

```

209
210     % Names of input root files
211     {
212         ../input/mixed.1.root
213         ../input/mixed.2.root
214     }
215
216     % TTree name
217     {
218         evt
219     }

```

220 2.4. Output of the program

221 The program gains the topology information from input data, and saves the information to
222 output files. As mentioned in Section 1, the information includes the categories of physics pro-

cesses 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.

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

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

In the example of the previous subsection, the program conducts its basic functionality, namely the component analysis over decay trees. From the 100000 events of the input sample, the program recognizes 17424 decay trees and outputs all of them to the txt, tex, and pdf files. Table 1 shows only the top ten decay trees and their respective final states in the obtained topology map.

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	$J/\psi \rightarrow \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^-$	$\pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^-$	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

In the table, “rowNo”, “iDcyTr”, “nEtr”, and “nCEtr” are abbreviations for the row number,

index of decay tree, number of entries of decay tree, and number of the cumulative entries from the first to the current decay trees, respectively. The values of “iDcyTr” are assigned from small to large in the program but listed according to the values of “nEtr” from large to small in the table. This is the reason why they are not in natural order like the values of “rowNo”. In Section 2.2, we mention that the decay $\pi^0 \rightarrow \gamma\gamma$ is ignored by the program at the early phase of processing the input data. As a result, it does not show itself in the table.

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

```
examples/in_the_paper/ex_for_tb.01/draw_X/v2/draw_X.C,
```

where, for example, a statement equivalent to

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

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

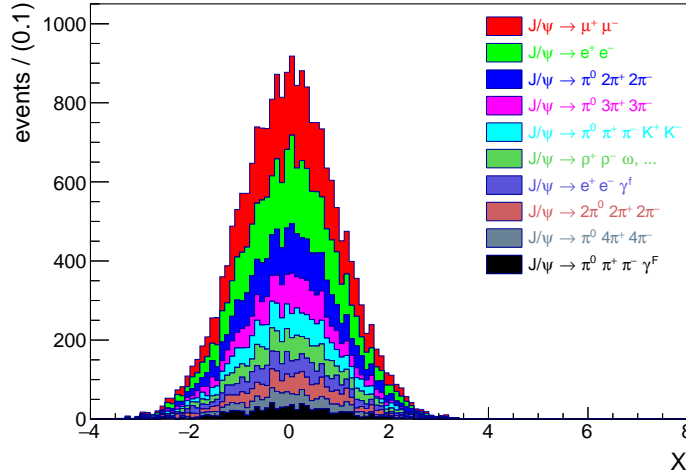


Figure 1: Distribution of X accumulated over the top ten decay trees. In the legend entry “ $J/\psi \rightarrow \rho^+\rho^-\omega, \dots$ ”, the dots “...” represent the secondary decay branches: $\rho^+ \rightarrow \pi^0\pi^+$, $\rho^- \rightarrow \pi^0\pi^-$, $\omega \rightarrow \pi^0\pi^+\pi^-$.

3. Component analysis

Component analysis is the primary functionality of the program. It is performed over decay 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, 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 seven (three 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 five 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 seven 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 five 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.

326
327
328
329
330
331
332
333
334
335
336
337
338

% Component analysis — decay trees
{
 Y 5 Y
}

Table 2 shows the decay trees. In the table, while the first five decay trees are listed exclusively in the main part, the rest decay trees are only summarized inclusively at the bottom row. In the symbolic expressions of decay initial-final states, the dashed right arrow (\dashrightarrow) instead of the plain right arrow (\rightarrow) is used, in order to reflect that the initial state does not necessarily decay to the final states in a direct way. Similarly, it is also used in the symbolic expressions of IRA decay branches, which will be introduced in Section 3.7.

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

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

3.2. Decay initial-final states

Table 3: Decay initial-final states.

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

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

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

```

342
343
344
345     % Component analysis — decay initial-final states
346     {
347         Y    5
348     }

```

3.3. Decay branches of particles

Sometimes, we have to examine the decay branches of certain particles. The following example shows the associated item with the two particles D^{*+} and J/ψ set as research objects. In the item, each row holds the information of a specified particle, and the first, second and third columns 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 also be used as a placeholder for an unassigned alias, if only the maximum number of output components is desired.

```

358
359     % Component analysis — decay branches of particles
360     {
361         D*+   Dsp   5
362         J/psi  Jpsi  5
363     }
364

```

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

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

It is worth mentioning here that, in addition to decay branches, production branches and mothers of specified particles can be examined with the program. One can make the program execute the two functionalities by replacing “decay branches” in the prompt of the item with “production branches” and “mothers”, respectively.

3.4. 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.

```

387 % Component analysis — cascade decay branches of particles
388 {
389     B0  B0  5  2
390     D0  D0  5  2
391 }

```

The cascade decay branches of B^0 are displayed in Table 5.

Table 5: 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

3.5. 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.

```

403 % Component analysis — decay final states of particles
404 {
405     B0  B0  5  3
406     D0  D0  5  3
407 }

```

Table 6: 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

Table 6 shows the three-body decay final states of D^0 . In the table, π^0 only decays to $\gamma\gamma$; otherwise, it will be replaced with its decay products, resulting in different decay final states of

410 D^0 .

411 3.6. Inclusive decay branches

412 In some cases, we are interested in the exclusive components of certain inclusive decay
 413 branches. Below is an example demonstrating the related item by taking the two inclusive decay
 414 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
 415 row holds the information of an inclusive decay branch, and the first, second, and third column-
 416 s separated with the symbol “&” are the textual expressions, aliases, and maximum numbers of
 417 output components, respectively. As we introduce at the beginning part of this section, the aliases
 418 and maximum numbers of output components are both optional. Here, we note that the symbol
 419 “-” can be used as a placeholder for an unassigned alias, if only the maximum number of output
 420 components is desired.

```
421 % Component analysis — inclusive decay branches
422 {
423   B0 --> D*+ & B2Dsp & 5
424   B0 --> K_S0 & B2Ks & 5
425 }
426
```

427
 428 The exclusive components of $B^0 \rightarrow K_S^0 + \text{anything}$ are displayed in Table 7. From the table,
 429 ten exclusive components of the inclusive decay branch are found in the input sample, and the
 430 particles denoted with *anything* are mainly the traditional charmonium states.

Table 7: 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

431 3.7. Intermediate-resonance-allowed decay branches

432 Occasionally, we may want to investigate the inner structures of certain IRA decay branches.
 433 The following example shows the associated item with the two IRA decay branches $D^{*+} \rightarrow$
 434 $\pi^0 \pi^+ \pi^+ K^-$ and $J/\psi \rightarrow \pi^0 \pi^+ \pi^-$ set as research objects. Since IRA decay branches look like
 435 inclusive decay branches, the format of the input to the item for IRA decay branches is identical
 436 to that for inclusive decay branches, which is introduced in the previous subsection.

```
437 % Component analysis — intermediate-resonance-allowed decay branches
438 {
439   D*+ --> K- pi+ pi+ pi0 & Dsp2K3Pi & 5
440   J/psi --> pi+ pi- pi0 & Jpsi23Pi & 5
441 }
442
```

443
 444 Table 8 shows the exclusive components of $D^{*+} \rightarrow \pi^0 \pi^+ \pi^+ K^-$. From the table, two intermediate
 445 particles D^0 and D^+ are found in the IRA decay branch, and they decay to $\pi^0 \pi^+ K^-$ and $\pi^+ \pi^+ K^-$,
 446 respectively.

Table 8: 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

3.8. Essential topology tags

Table 9: 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)
Cascade decay branches of particles	iDcyBrP.i.j	index of decay branch of the j^{th} particle _i
	nPCascDcyBr.i	number of particle;s (or its cascade decay branches)
Decay final states of particles	iCascDcyBrP.i.j	index of cascade decay branch of the j^{th} particle _i
	nPDcyFSt.i	number of particle;s (or its decay final states)
Inclusive decay branches	iDcyFStP.i.j	index of decay final state of the j^{th} particle _i
	nIncDcyBr.i	number of inclusive decay branch;es
IRA decay branches	iDcyBrIncDcyBr.i.j	index of decay branch of the j^{th} inclusive decay branch _i
	nIRADcyBr.i	number of IRA decay branch;es
	iDcyBrIRADcyBr.i.j	index of decay branch of the j^{th} IRA decay branch _i

Table 9 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 five 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 – 8.

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/ψ , 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/ψ (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

471 D^{*+} , the default value -1 is assigned to both iDcyBrP_Dsp_0 and iDcyBrP_Dsp_1. We note that
 472 different from all other indices, PDGMoth.i.j has the default value 0, instead of -1 .

473 4. Signal identification

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

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

489 4.1. Decay trees

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

```

501 % Signal identification — decay trees
502 {
503   0 & Upsilon(4S) --> B0 anti-B0 & -1 & 1stDcyTrInTb2 & Y
504   1 & B0 --> e+ nu_e D*- gamma & 0
505   2 & anti-B0 --> mu- anti-nu_mu D*+ & 0
506   3 & D*- --> pi- anti-D0 & 1
507   4 & D*+ --> pi+ D0 & 2
508   5 & anti-D0 --> pi0 pi- K+ & 3
509   6 & D0 --> pi0 pi+ K- & 4
510
511   0 & Upsilon(4S) --> B0 anti-B0 & -1 & 2ndDcyTrInTb2
512   1 & B0 --> pi0 pi+ pi- rho- D- & 0
513   2 & anti-B0 --> mu- anti-nu_mu D*+ & 0
514   3 & rho- --> pi0 pi- & 1
515   4 & D- --> pi- pi- K+ & 1
516   5 & D*+ --> pi+ D0 & 2
517   6 & D0 --> K_L0 pi+ pi- & 5
518 }
519
```

520

521 Table 10 shows the resulting topology map. The results are the same as those displayed in the
 522 first two rows of Table 2.

Table 10: Signal decay trees and their respective initial-final states.

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

523 4.2. Decay initial-final states

524 In a few cases, we have interest in some decay initial-final states. Below is an example
 525 demonstrating the related item by taking the first two decay initial-final states listed in Table 3
 526 as signals. Similar to IRA decay branches, decay initial-final states look like inclusive decay
 527 branches. Hence, except that only two columns are involved in the item, the format of the input
 528 to the item for decay initial-final states is identical to that for the component analysis over inclu-
 529 sive decay branches, which is introduced in Section 3.6. As we can see from the example, the
 530 numbers of identical particles are supported to be written in front of their textual names in order
 531 to simplify the textual expressions of the final states.

```

532 % Signal identification — decay initial-final states
533 {
534   Y(4S) --> mu+ nu_mu 3 pi0 3 pi+ 4 pi- K+ K- & 2ndDcyIFStsInTb3
535   Y(4S) --> 5 pi0 5 pi+ 5 pi- K+ K- & 2ndDcyIFStsInTb3
536 }
537
538

```

539 The obtained topology map is displayed in Table 11. The results are identical to those shown in
 540 the first two rows of Table 3.

Table 11: 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

541 4.3. Particles

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

```

546 % Signal identification — particles
547 {
548   D*+   Dsp
549   J/psi  Jpsi
550 }
551
552

```

553 Table 12 shows the resulting topology map. As a cross-check, the number of D^{*+} s in the table
 554 equals that in Table 4.

Table 12: Signal particles.

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

4.4. Decay branches

On some occasions, we have to identify certain regular decay branches. Below is an example demonstrating the related item by taking the two decay branches $\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$ and $B^0 \rightarrow K_S^0 J/\psi$ as signals. Since regular decay branches also look like inclusive decay branches, except that only two columns are involved in the item, the format of the input to the item for regular decay branches is identical to that for the component analysis over inclusive decay branches, which is introduced in Section 3.6.

```

562 % Signal identification — decay branches
563 {
564     B0 --> mu- anti-nu_mu D*+ & B2munuDsp
565     B0 --> K_S0 J/psi & B2KsJpsi
566 }

```

The obtained topology map is displayed Table 13. For the purpose of cross-checks, we note that the number of $B^0 \rightarrow K_S^0 J/\psi$ in the table is equal to that in the first row of Table 7.

Table 13: 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

4.5. Cascade decay branches

Sometimes, we are interested in certain cascade decay branches. The following example shows the associated item with the two cascade decay branches $B^0 \rightarrow D^{*-} D_s^{*+}$, $D^{*-} \rightarrow \pi^- \bar{D}^0$, $D_s^{*+} \rightarrow D_s^+ \gamma$ and $B^0 \rightarrow D^{*-} D_s^{*+}$, $D^{*-} \rightarrow \pi^- \bar{D}^0$ set as signals. While the first cascade decay branch is identical to the fifth one in Table 5, the second is only part of it, which demonstrates that the cascade decay branches supported in the item are not necessarily fully specified at the level of certain hierarchy. Similar to decay trees, cascade decay branches are made up of regular decay branches. Hence, the format of the input to the item for cascade decay branches is identical to that for decay trees, which is introduced in Section 4.1.

```

580 % Signal identification — cascade decay branches
581 {
582     0 & B0 --> D*- D_s*+ & -1
583     1 & D*- --> pi- anti-D0 & 0
584     2 & D_s*+ --> D_s+ gamma & 0
585 }
586
587     0 & B0 --> D*- D_s*+ & -1
588     1 & D*- --> pi- anti-D0 & 0
589 }

```

Table 14 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 5.

Table 14: Signal cascade decay branches.

rowNo	signal cascade decay branch	iSigCascDcyBr	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 3.6.

```
% Signal identification — inclusive decay branches
{
  anti-B0 --> D*+ & B2Dsp
  B0 --> K_S0 & B2Ks
}
```

The obtained topology map is displayed in Table 15. As a cross-check, the number of $B^0 \rightarrow K_S^0 + \text{anything}$ in the table equals that in Table 7.

Table 15: 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. Intermediate-resonance-allowed decay branches

On some occasions, we need to identify 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 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 IRA decay branches, which is introduced in Section 3.7.

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

Table 16 shows the resulting topology map. For the purpose of cross-checks, we note that the number of $D^{*+} \rightarrow \pi^0 \pi^+ \pi^+ K^-$ in the table is equal to that in Table 8.

Table 16: 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

4.8. Essential topology tags

Table 17 summarizes and explains all of the essential topology tags involved in the signal identification functionalities. For signal decay trees and signal decay initial-final states, there are two sorts of topology tags. The first sort of tags, iSigDcyTr and iSigDcyIFSts, record the default indices of the specified signal decay trees and signal decay initial-final states. They have the similar interpretation as iDcyTr and iDcyIFSts, and are shown in the third columns of Tables 10 and 11. The second sort of tags, nameSigDcyTr and nameSigDcyIFSts, save the specified aliases of the signal decay trees and signal decay initial-final states. In cases the aliases are not specified, empty strings will be stored.

For the latter five kinds of signal identification, there are only one sort of topology tags, which record the number of the specified particles or decay branches found in each entry. Similar to the cases in the latter five kinds of component analysis, 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.

Table 17: 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 _i s
Decay branches	nSigDcyBr_i	number of signal decay branch _i es
Cascade decay branches	nSigCascDcyBr_i	number of signal cascade decay branch _i es
Inclusive decay branches	nSigIncDcyBr_i	number of signal inclusive decay branch _i es
IRA decay branches	nSigIRADcyBr_i	number of signal IRA decay branch _i 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 two dozen common setting items on its input, functionalities, and output. In this paper, we only introduce a part of the items that are crucial to our physics studies.

5.1. Settings on 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.

```

655
656
657     % Maximum number of entries to be processed
658     {
659         2000
660     }

```

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 $(-1, 1)$.

```

665
666     % Cut to select entries
667     {
668         (X > -1) && (X < 1)
669     }

```

Notably, only a single-line selection requirement is supported in the item, like the cases in the methods Draw() and GetEntries() of the class TTree. In spite of this, such a requirement is able to express any requirement with the help of the parentheses “()” as well as the logical symbols “&&”, “||”, and “!”.

5.2. Setting on input decay branches

Normally, the program deals with all of the decay branches in every decay tree. However, examining all the branches is not always required in practice. Sometimes, we only concern the first n hierarchies of the branches. Similar to that in cascade decay branches of particles (as we introduce in Section 3.4), the hierarchy here reflects the rank of a decay branch in a decay tree. For example, in the decay tree $\Upsilon(4S) \rightarrow B^0 \bar{B}^0$, $B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F$, $\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$, $D^{*-} \rightarrow \pi^- \bar{D}^0$, $D^{*+} \rightarrow \pi^+ D^0$, $\bar{D}^0 \rightarrow \pi^0 \pi^- K^+$, $D^0 \rightarrow \pi^0 \pi^+ K^-$, the hierarchies of the seven individual branches are 1, 2, 2, 3, 3, 4, and 4, respectively. The program provides an item to set the maximum hierarchy. Below is an example showing the item with the maximum hierarchy set at one.

```

684
685     % Maximum hierarchy of heading decay branches to be processed in each event
686     {
687         1
688     }

```

With the setting, the decay branches with hierarchy larger than one will be ignored by the program. For the component analysis over the decay trees of the $\Upsilon(4S)$ sample, only the first hierarchy of $\Upsilon(4S)$ decay branches are analyzed, and the result is shown in Table 18. From the table, 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 mixing.

Table 18: 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

Similarly, in the case of the maximum hierarchy set at two, we could get the result of the component analysis over the first two hierarchies of $\Upsilon(4S)$ decay branches, as displayed in Table 19.

Table 19: 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

5.3. Settings on initial and final state radiation photons

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

```
% PDG code of ISR photons (Default: 222222222)
{
  222222222
}

% PDG code of FSR photons (Default: -22)
{
  -22
}
```

In this case, the program is able to label the ISR and FSR photons as γ^i (gammai) and γ^f (gammaf) in the output pdf (txt) files, respectively.

On other occasions, ISR and FSR photons are not marked out in advance due to some reasons. In such cases, the program have to identify them by itself according to the following rules: photons which have no mothers recorded in the arrays of the PDG codes and mother indices are considered as generalized ISR photons, while other photons which have at least one e^\pm , μ^\pm , π^\pm , K^\pm , p , or \bar{p} sister are taken as generalized FSR photons. Here, the modifier “generalized” is used because the rules can not determine the types of the photons in absolute accuracy. For example, photons from radiative decays might be mistaken as ISR and FSR photons. Despite this, generalized ISR and FSR photons are good concepts, particularly in cases where the sources of

the photons are not required to be distinguished clearly. The program will label the generalized ISR and FSR photons as γ^I (gammaI) and γ^F (gammaF) in the output pdf (txt) files, respectively. Notably, we are not concerned about these ISR and FSR photons in many cases. If they have already been marked out beforehand, one can make the program ignore them accurately by setting the following two items to “Ys”.

```

733
734     % Ignore ISR photons (Three options: Ys, Yg and N. Default: N)
735     {
736         Ys
737     }
738
739
740     % Ignore FSR photons (Three options: Ys, Yg and N. Default: N)
741     {
742         Ys
743     }
744

```

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

5.4. Settings on candidate based analysis

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

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

```

762
763     % Avoid over counting for candidate based analysis (Two options: Y and N. Default: N)
764     {
765         Y
766     }
767

```

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

```

770
771     % TBranch name of the indices of candidates in an event (Default: __candidate__)
772     {
773         iCandidate
774     }
775

```

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

5.5. Setting on charge conjugation

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

```
% Process charge conjugate objects together (Two options: Y and N. Default: N)
{
  Y
}
```

Performing topology analysis with the setting inserts new topology tags in the output root files and adds new counters to topology maps in the output txt, tex, and pdf files. Tables 20 and 21 list and interpret all of the topology tags related to charge conjugation involved in the component analysis and signal identification functionalities, respectively.

Table 20: Topology tags related to charge conjugation involved in each kind of component analysis. For the latter five 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 index_{cc} are short for “charge conjugate” and “charge conjugate index”, respectively.

Component analysis kind	Topology tag	Interpretation
Decay trees	iCcDcyTr	index_{cc} of decay tree
Decay initial-final states	iCcDcyIFSts	index_{cc} of decay initial-final states
Decay branches of particles	iCcPDcyBr.i	index_{cc} of particle _i
	1) iCcDcyBrP.i.j	index_{cc} of decay branch of the j th particle _i
	2) nCcPDcyBr.i	number of cc particle _i s (decay branches)
	2) iDcyBrCcP.i.j	index of decay branch of the j th cc particle _i
	2) nAllPDcyBr.i	number of all particle _i s (decay branches)
Cascade decay branches of particles	iCcPCascDcyBr.i	index_{cc} of particle _i
	1) iCcCascDcyBrP.i.j	index_{cc} of cascade decay branch of the j th particle _i
	2) nCcPCascDcyBr.i	number of cc particle _i s (cascade decay branches)
	2) iCascDcyBrCcP.i.j	index of cascade decay branch of the j th cc particle _i
	2) nAllPCascDcyBr.i	number of all particle _i s (cascade decay branches)
Decay final states of particles	iCcPDcyFSt.i	index_{cc} of particle _i
	1) iCcDcyFStP.i.j	index_{cc} of decay final state of the j th particle _i
	2) nCcPDcyFSt.i	number of cc particle _i s (decay final states)
	2) iDcyFStCcP.i.j	index of decay final state of the j th cc particle _i
	2) nAllPDcyFSt.i	number of all particle _i s (decay final states)
Inclusive decay branches	iCcIncDcyBr.i	index_{cc} of inclusive decay branch _i
	1) iCcDcyBrIncDcyBr.i.j	index_{cc} of decay branch of the j th inclusive decay branch _i
	2) nCcIncDcyBr.i	number of cc inclusive decay branch _i s
	2) iDcyBrCcIncDcyBr.i.j	index of decay branch of the j th cc inclusive decay branch _i
	2) nAllIncDcyBr.i	number of all inclusive decay branch _i s
IRA decay branches	iCcIRADcyBr.i	index_{cc} of IRA decay branch _i
	1) iCcDcyBrIRADcyBr.i.j	index_{cc} of decay branch of the j th IRA decay branch _i
	2) nCcIRADcyBr.i	number of cc IRA decay branch _i s
	2) iDcyBrCcIRADcyBr.i.j	index of decay branch of the j th cc IRA decay branch _i
	2) nAllIRADcyBr.i	number of all IRA decay branch _i s

As an example, we carry out the component analysis over the decay branches of D^{*+} . The resulting topology map of D^{*+} is displayed in Table 22. Besides the columns in Table 4, two

Table 21: Topology tags related to charge conjugation involved in each kind of signal identification. For the latter five 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 index_{cc} are short for “charge conjugate” and “charge conjugate index”, respectively.

Signal identification kind	Topology tag	Interpretation
Decay trees	iCcSigDcyTr	index_{cc} of signal decay tree
Decay initial-final states	iCcSigDcyIFSts	index_{cc} of signal decay initial-final states
Particles	iCcSigP_i	index_{cc} of signal particle _i
	*) nCcSigP_i	number of cc signal particle _i s
	*) nAllSigP_i	number of all signal particle _i s
Decay branches	iCcSigDcyBr_i	index_{cc} of signal decay branch _i
	*) nCcSigDcyBr_i	number of cc signal decay branch _i es
	*) nAllSigDcyBr_i	number of all signal decay branch _i es
Cascade decay branches	iCcSigCascDcyBr_i	index_{cc} of signal cascade decay branch _i
	*) nCcSigCascDcyBr_i	number of cc signal cascade decay branch _i es
	*) nAllSigCascDcyBr_i	number of all signal cascade decay branch _i es
Inclusive decay branches	iCcSigIncDcyBr_i	index_{cc} of signal inclusive decay branch _i
	*) nCcSigIncDcyBr_i	number of cc signal inclusive decay branch _i es
	*) nAllSigIncDcyBr_i	number of all signal inclusive decay branch _i es
IRA decay branches	iCcSigIRADcyBr_i	index_{cc} of signal IRA decay branch _i
	*) nCcSigIRADcyBr_i	number of cc signal IRA decay branch _i es
	*) nAllSigIRADcyBr_i	number of all signal IRA decay branch _i es

796 additional columns with the headers “nCcCase” and “nAllCase” are inserted in the table. Here,
797 “nCcCase” represents the number of cases involving the charge conjugate particles, and “nAll-
798 Case” is the sum of “nCase” and “nCcCase”.

Table 22: Decay branches of D^{*+} .

rowNo	decay branch of D^{*+}	iDcyBrP	nCase	nCcCase	nAllCase	nCCase
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

799 For a specified particle, what we want to further record with topology tags are as follows: (1)
800 whether it is self-charge-conjugate; (2) whether its decay branches are self-charge-conjugate, if
801 it is self-charge-conjugate; (3) the number and the indices of the decay branches of its charge-
802 conjugate particle, if it is not self-charge-conjugate. Hence, in addition to “nPDcyBr_i” and
803 “iDcyBrP_i_j”, the following topology tags are also inserted in the output root files: “iCcPDcy-
804 Br_i” for all specified particles; “iCcDcyBrP_i_j” for self-charge-conjugate particles only; and
805 “nCcPDcyBr_i”, “iDcyBrCcP_i_j”, and “nAllPDcyBr_i” for non-self-charge-conjugate particles
806 only. Here, “iCcPDcyBr_i” is short for charge conjugate index of the i^{th} particle specified for
807 its decay branches. For self-charge-conjugate particles, it has the value 0; for non-self-charge-
808 conjugate particles, it has the value 1.

809 The topology tag “iCcDcyBrP_i_j” denotes charge conjugate index of decay branch of the j^{th}
810 instance of the i^{th} particle. For self-charge-conjugate decay branches, it has the value 0; for non-
811 self-charge-conjugate decay branches, it has the value 1 or -1 : while 1 tags the decay branches

812 listed in the topology maps, -1 indicates their charge conjugate decay branches. Whereas the
 813 values of “iDcyBrP.i.j” for each decay branch and its charge conjugate decay branch are equal
 814 in order to indicate their sameness, the values of “iCcDcyBrP.i.j” for them are opposite so as to
 815 reflect their difference.

816 The topology tag “iDcyBrCcP.i.j” has the similar meaning as “iDcyBrP.i.j”, but it is de-
 817 signed for the charge conjugate particle of the i^{th} particle. Particularly, it ranges from 0 (included)
 818 to the number of the categories of decay branches of the i^{th} particle found in the sample (exclud-
 819 ed). The topology tag “nCcpDcyBr.i” stands for the number of the charge conjugate i^{th} particles
 820 (or their decay branches) found in each entry, and “nAllPDcyBr.i” is the sum of “nPDcyBr.i”
 821 and “nCcpDcyBr.i”.

822 6. Summary

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

828 Since it does not rely on any specific software frameworks, the program applies to many
 829 high energy physics experiments. Up to now, it has been put into use in three experiments at
 830 e^+e^- colliders: the BESIII, Belle, and Belle II experiments. Besides these experiments, it can
 831 also be used in the PANDA experiment [11], which is an anti-proton annihilation experiment
 832 under construction at Darmstadt, Germany. In addition, the program is also applicable to the
 833 pre-research of future e^+e^- colliding experiments, such as the circular electron positron collider
 834 (CEPC) [12, 13] experiment in China, the super charm-tau factory (SCTF) experiment [14] in
 835 Russia, and the super tau-charm factory (STCF) experiment [15] in China. These experiments
 836 offer a wide range of potential uses of the program. With more user needs coming out in the
 837 future, we will further extend and perfect it to make it more powerful and well-rounded.

838 Acknowledgements

839 This project is supported by the National Natural Science Foundation of China under Grants
 840 No. 11575017, No. 11661141008, No. 11761141009, and No. 11975076; the CAS Center for
 841 Excellence in Particle Physics (CCEPP). In addition, we would like to thank all of the people
 842 who have helped us in the development of the program. We first thank Prof. Changzheng Yuan,
 843 Bo Xin, and Haixuan Chen for their help at the early stage of developing the program. We are
 844 particularly grateful to Prof. Xingtao Huang for his comments on the principles and styles of the
 845 program, to Remco de Boer for his suggestions on the tex output and the use of GitHub, and to Xi
 846 Chen for his discussions on the core algorithms. We are especially indebted to Prof. Xiqing Hao,
 847 Longke Li, Xiaoping Qin, Ilya Komarov, Yubo Li, Guanda Gong, Suxian Li, Junhao Yin, Prof.
 848 Xiaolong Wang, and Yeqi Chen for their advice in extending and perfecting the program. Also,
 849 we thank Xi’an Xiong, Runqiu Ma, Wencheng Yan, Sen Jia, Hongpeng Wang, Jiawei Zhang,
 850 Hongrong Qi, Jiajun Liu, Maoqiang Jing, Yi Zhang, Wei Shan, and Yadi Wang for their efforts
 851 in helping us test the program.

References

- [1] ROOT User's Guide, Available online: <https://root.cern/root/html/doc/guides/users-guide/ROOTUsersGuide.html>.
- [2] K.T. Chao, Y.F. Wang, et al., Physics at BES-III, *Int. J. Mod. Phys. A* 24 (2009) S1-794.
- [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).
- [4] E. Kou, et al., *Prog. Theor. Exp. Phys.* 2019 (2019) 123C01.
- [5] J. Brodzicka, T. Browder, P. Chang, et al., *Prog. Theor. Exp. Phys.* 2012 (2012) 04D001.
- [6] Documentation of the TFile class, Available online: <https://root.cern/root/html534/TFile.html>.
- [7] Documentation of the TTree class, Available online: <https://root.cern/root/html534/TTree.html>.
- [8] M. Tanabashi, et al. (Particle Data Group), *Phys. Rev. D* 98 (2018) 030001.
- [9] Documentation of the TBranch class, Available online: <https://root.cern/root/html534/TBranch.html>.
- [10] Documentation of the TChain class, Available online: <https://root.cern/root/html534/TChain.html>.
- [11] W. Erni, et al. (PANDA Collaboration), Physics Performance Report for PANDA: Strong Interaction Studies with Antiprotons, [arXiv:0903.3905](https://arxiv.org/abs/0903.3905).
- [12] CEPC CDR Volume 1 (Accelerator), Available online: http://cepc.ihep.ac.cn/CEPC_CDR_Vol1_Accelerator.pdf.
- [13] CEPC CDR Volume 2 (Physics & Detector), Available online: http://cepc.ihep.ac.cn/CEPC_CDR_Vol2_Physics-Detector.pdf.
- [14] A.E. Bondar, et al. (Charm-Tau Factory Collaboration), *Phys. Atom. Nucl.* 76 (2013) 1072.
- [15] Q. Luo, D. Xu, "Progress on Preliminary Conceptual Study of HIEPA, a Super Tau-Charm Factory in China", in *Proc. 9th International Particle Accelerator Conf. (IPAC2018)*, Vancouver, BC, Canada, 422.