

TopoAna: A generic tool for the event type 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 physics processes involved in the samples, including the types of processes and the number of processes in each type, is a great help to investigating signals and backgrounds. To help analysts obtain the physics process information from the truth information of the samples, we develop a physics process analysis program, TopoAna, with C++, ROOT, and LaTeX. The program implements the functionalities of component analysis and signal identification with many kinds of fine, customizable classification and matching algorithms. It tags physics processes in individual events accurately in the output root files, and exports the physics process information at the sample level clearly to the output plain text, tex source, and pdf files. 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 similar experiments is also prospective.

Keywords: event type; 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/generic Monte-Carlo (MC) samples 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

*Corresponding author.

E-mail address: zhoux@buaa.edu.cn, shenpc@fudan.edu.cn

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

Preprint submitted to Computer Physics Communications

September 30, 2020

23 optimization of preliminary criteria is often needed in the process. Under the circumstances, a
 24 comprehensive understanding of the samples is required. In particular, a clear knowledge of the
 25 physics processes, or event types, involved in the samples is quite helpful. To be specific, the
 26 physics process information includes the types of processes and the number of processes in each
 27 type, involved both in the entire samples and in the individual events. Here, the physics process
 28 could be a complete production and decay process involved in an event, or merely a part of it,
 29 such as the decay of an intermediate resonance. With the information, one can figure out the
 30 main backgrounds (especially the peaking ones), and optimize the selection criteria further by
 31 analyzing the differences between the main backgrounds and the signals. Even if it is difficult
 32 to further suppress these backgrounds, the knowledge of their types is beneficial to estimate the
 33 systematic uncertainties associated with them.

34 The analysis of the physics process information described above is a sort of component anal-
 35 ysis. It is complex since it has to classify physics processes actively and finely. Another sort
 36 of physics process analysis often required in practice is signal identification, which only aims
 37 to search for certain processes of interests. It is relatively simple because its core technique is
 38 merely pattern matching. Mostly, signal and background events coexist in inclusive MC samples.
 39 It is useful to differentiate them in such cases. The identified signal events can be used to make
 40 up a signal sample in the absence of specialized signal samples, or they can be removed to avoid
 41 repetition in the presence of specialized signal samples. Occasionally, we have to pick out some
 42 decay branches in order to re-weight them according to new theoretical predictions or updated
 43 experimental measurements. Signal identification also plays a part in this occasion.

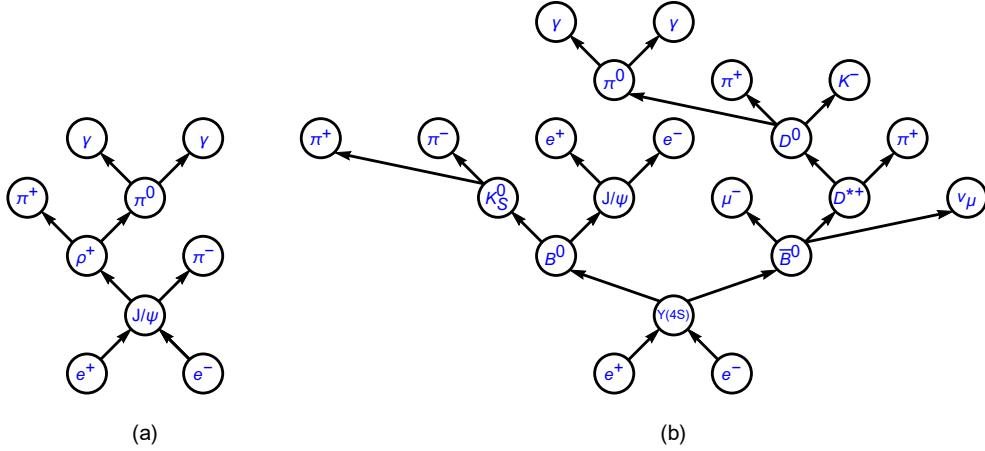


Figure 1: Topology diagrams of (a) $e^+e^- \rightarrow J/\psi$, $J/\psi \rightarrow \rho^+\pi^-$, $\rho^+ \rightarrow \pi^+\pi^0$, $\pi^0 \rightarrow \gamma\gamma$ and (b) $e^+e^- \rightarrow \Upsilon(4S)$, $\Upsilon(4S) \rightarrow B^0\bar{B}^0$, $B^0 \rightarrow K_S^0 J/\psi$, $\bar{B}^0 \rightarrow \mu^- D^{*+} \nu_\mu$, $K_S^0 \rightarrow \pi^+\pi^-$, $J/\psi \rightarrow e^+e^-$, $D^{*+} \rightarrow D^0\pi^+$, $D^0 \rightarrow \pi^0\pi^+K^-$, $\pi^0 \rightarrow \gamma\gamma$. As if trees grow, the diagrams are plotted from bottom to top.

44 Processes in high energy physics can be visualized with topology diagrams. As an example,
 45 Fig. 1 shows the topology diagrams of two typical physics processes occurring at e^+e^- colliders.
 46 From the figure, the hierarchies of the processes and the relationships among the particles are
 47 clearly illustrated with the diagrams. Though the complexities of topology diagrams vary with
 48 physics processes, there is only one diagram corresponding to each process. For this reason, we
 49 refer to the physics process information/analysis mentioned thereinbefore as topology informa-

50 tion/analysis hereinafter. The component analysis and signal identification introduced above are
51 exactly the two categories of topology analysis that will be discussed in this paper.

52 Since the raw topology truth information of inclusive MC samples is counter-intuitive, di-
53 verse, and overwhelming, it is difficult for analysts to check the topology information of the
54 samples directly. To help them do the checks quickly and easily, a topology analysis program
55 called TopoAna is developed with C++, ROOT [1], and LaTeX. Here, C++ is the programming
56 language, ROOT is the C++ based data analysis software universally used in modern high energy
57 physics experiments, and LaTeX is used for generating pdf documents containing the obtained
58 topology information. The program implements the functionalities of component analysis and
59 signal identification based on accurate pattern matching. To meet a variety of practical require-
60 ments, many kinds of fine, customizable classification and matching algorithms are implemented
61 in the program. Generally, the program recognizes, categorizes, and counts physics processes in
62 each event in the samples, and tags them in the corresponding entry of the output root (TFile [2])
63 files. After processing the events, the program exports the obtained topology information at the
64 sample level to the output plain text, tex source, and pdf files.

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

73 The program has a history of more than ten years. It has already gone through a series of
74 major upgrades. Prior to its development, analysts usually wrote some private codes to match
75 few signals and/or backgrounds for their own studies. The limited functions of these codes
76 do not satisfy the increasing demand for topology analysis. This motivates us to develop a
77 generic, powerful, and easy-to-use program. At first, the program was developed for the BESIII
78 experiment, an experiment in the τ -Charm energy region with abundant research topics under
79 study [3, 4]. Later, it was extended substantially for the Belle II experiment, which is primarily
80 dedicated to search for physics beyond the Standard Model in the flavor sector and has already
81 started data taking in the recent three years [5]. Besides, the program has also been tried and
82 used in the Belle experiment, the predecessor of the Belle II experiment, where some physics
83 studies are still ongoing [6]. Not relying on any specific software frameworks, the program now
84 applies to many high energy physics experiments.

85 This user guide gives a detailed description of TopoAna. It proceeds as follows: Section 2
86 introduces the basics of the program; Sections 3 and 4 expatiate the two categories of function-
87 alities of the program — component analysis and signal identification, respectively; Sections 5
88 and 6 present some common settings and auxiliary facilities for the executing of the program,
89 respectively; Section 7 summarizes the user guide. It is worth mentioning here that, aside from
90 the detailed description in the user guide, an essential description of the program has been written
91 into a paper, which has already been published by Computer Physics Communications. One can
92 find this paper and the preprint corresponding to it in the links [Comput. Phys. Commun. 258 \(2021\) 107540](#) and [arXiv:2001.04016](#), respectively. For your convenience, we provide the latest
93 version of the paper draft “paper_draft_v3.1.pdf” under the directory “share” of the package. If
94 the tool really helps your researches, we would appreciate it very much if you could cite the
95 paper in your publications.

97 **2. Basics of the program**

98 This section introduces the basics of the program, including the package, input, algorithm,
99 execution, performance, output, and validation of the program. The package implements the
100 program via a C++ class called “topoana” and a main function invoking the class. Compiling
101 the package creates the executable file of the program, that is, “topoana.exe”. To execute the
102 program, we have to first obtain the input data of the program, namely the raw topology truth
103 information of the inclusive MC samples, with some interfaces to the program in the software
104 systems of the corresponding experiments. Normally, the input data contain all the topology
105 information of the samples. With the data, all kinds of the topology analysis presented in the
106 user guide can be performed.

107 To carry out the topology analysis desired in our work, we have to provide some necessary in-
108 put, functionality, and output information to the program. The information is required to be filled
109 in the setting items designed and implemented in the program, and the items have to be put in a
110 plain text file named with a suffix “.card”. With the card file, one can execute the program with
111 the command line: “topoana.exe cardFileName”, where the argument “cardFileName” is option-
112 al and its default value is “topoana.card”. After the execution of the program, we can examine
113 the results of topology analysis in the output files and use them to analyze other experimental
114 quantities. The results help us gain a better understanding of the signals and backgrounds and
115 are conducive to carrying our work forward. Besides the package, input, execution, and output
116 of the program mentioned above, the algorithm, performance, and validation of the program will
117 also be discussed in this section, because they are also essential aspects of the program. In the
118 next seven subsections, we will present the package, input, algorithm, execution, performance,
119 output, and validation of the program in detail, with each part in one subsection.

120 *2.1. Package of the program*

121 The package consists of six directories — “include”, “src”, “bin”, “share”, “examples”, and
122 “utilities” — and five files — “LICENSE”, “README.md”, “Configure”, “Makefile”, and “Set-
123 up”. While the directory “include” only includes one header file “topoana.h”, the directory “src”
124 contains sixty source files “*.cpp” as well as a script file “topoana.C”. At present, only one class,
125 namely “topoana”, is defined in the program for all of its functionalities. The class is declared in
126 “topoana.h”, implemented in “*.cpp” files, and invoked in “topoana.C”.

127 The file “template_topoana.card” under the directory “share” saves all the items which are de-
128 veloped for users to specify information for the execution of the program. One can refer to the file
129 when filling in the cards for their own needs. Some plain text files “pid_3pchrg_txtpnm_txpnm
130 _iccp.dat_*” are also included in the directory “share”. They store the basic information of the
131 particles used in the program. The suffixes of their names indicate the experiments they apply
132 to. One of them will be copied to “pid_3pchrg_txtpnm_txpnm.iccp.dat” when we set up the
133 program. Besides, the directory “share” also contains three LaTeX style files “geometry.sty”,
134 “ifxetex.sty”, and “makecell.sty”, which are invoked by the program for generating pdf files. **The**
135 **directory “examples” includes plenty of detailed examples. Particularly, all the examples**
136 **involved in this user guide are under its sub-directory “in_the_user_guide”.** The directory
137 “utilities” contains some useful bash scripts.

138 The program is released under MIT license [7]. The file “README.md” briefly introduces
139 how to install and use the program. To set up the program, one should first set the package path
140 with the command “./Configure”. Standard outputs of the command are the guidelines for man-
141 ually adding the absolute path of “topoana.exe” to the environment variable “PATH”, in order to

142 execute it without any path. The second step is executing the command “make”. This command
 143 compiles the header, source, and script files into the executable file “topoana.exe” under the di-
 144 rectory “bin”, according to the rules specified in the “Makefile”. The last step is specifying the
 145 experiment name with the command line “./Setup experimentName”. Currently, the supported
 146 experiment names are “BESIII”, “Belle”, and “Belle_II”. Besides, “./Setup Example” is required
 147 for the execution of the examples in the user guide.

148 2.2. Input of the program

The input of the program is one or more root files including a TTree [8] object which contains raw topology truth information of the inclusive MC samples under study. To be specific, the information 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 [9] 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 input data.

Number of particles	:	63			(1)
PDG codes of particles	:	300553, -511, 511, -433, 421, 211, 22, -413, 111, 111, 113, 211, -431, 22, -323, 213, -421, -211, 22, 22, 22, 22, 211, -211, 333, 11, -12, 22, -311, -211, 211, 111, 221, 331, 321, -321, 310, 22, 22, 111, 111, 111, 111, 111, 221, 111, 111, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,			
		22, 22			
Mother indices of particles	:	-1, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4, 7, 7, 8, 8, 9, 9, 10, 10, 12, 12, 12, 12, 14, 14, 15, 15, 16, 16, 24, 24, 28, 31, 31, 32, 32, 32, 33, 33, 33, 36, 36, 39, 39, 40, 40, 41, 41, 42, 42, 43, 43, 44, 44, 45, 45, 46, 46			

149 The complete physics process contained in the data is displayed as follows.

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

150 Here, the decay branches in the process are placed into two blocks in order to make full use of
 151 the page space. In both blocks, the first, second, and third columns are the indices, symbolic
 152 expressions, and mother indices of the decay branches. Notably, all the decay branches of $\pi^0 \rightarrow$

¹⁵³ $\gamma\gamma$ are omitted in Eq. (2) in order to make the process look more concise. Since the topology
¹⁵⁴ diagram of such a process looks like a tree, we refer to the complete processes as decay trees.
¹⁵⁵ Obviously, the input data do not show the structure automatically. Thus, we need the program to
¹⁵⁶ do the topology analysis work.

From the first branch in Eq. (2), only one particle $\Upsilon(4S)$ is produced after the e^+e^- annihilation. Thus, $\Upsilon(4S)$ can be referred to as the root particle of the decay tree. Similarly, many other resonances with the quantum numbers $J^{PC} = 1^{--}$, such as J/ψ , can be solely produced at other proper energy points. Besides the cases with only one root particle, the program can deal with the cases with multiple root particles. For example, the program can recognize the following raw topology truth information

Number of particles	:	25	
PDG codes of particles	:	433, -321, 223, 211, -413, 431, 111, 211, -211, 111, -411, 111, 321, 113, 22, 22, 22, 22, 321, -211, -211, 22, 22, 211, -211	(3)
Mother indices of particles	:	-1, -1, -1, -1, -1, 0, 0, 2, 2, 2, 4, 4, 5, 5, 6, 6, 9, 9, 10, 10, 10, 11, 11, 13, 13	

as the following process

0	$e^+e^- \rightarrow \pi^+\omega K^-D^{*-}D_s^{*+}$	-1	4	$D^- \rightarrow \pi^-\pi^-K^+$	2	
1	$\omega \rightarrow \pi^0\pi^+\pi^-$	0	5	$D_s^+ \rightarrow \rho^0K^+$	3	
2	$D^{*-} \rightarrow \pi^0D^-$	0	6	$\rho^0 \rightarrow \pi^+\pi^-$	5	
3	$D_s^{*+} \rightarrow \pi^0D_s^+$	0				(4)

157 Here, the particles $\pi^+ \omega K^- D^{*-} D_s^{*+}$ in the first branch arise from hadronization processes, in which
 158 quark pairs produced from initial state particles turn into hadrons. The processes with hadronization-
 159 ignored have a tree structure and thus are easy to resolve. On the other hand, some hadronization-
 160 processes, particularly those in high energy regions, contain complicated loop structures that
 161 are difficult to resolve without sophisticated algorithms. Resolving these intricate hadronization
 162 processes is not involved in the program at present.

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 these 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 [10] object as a scalar integer, while the PDG codes of particles, as well as the mother indices of particles, can be stored in a TBranch object as an array of integers, in a TBranch object as a vector of integers, or in a group of TBranch objects 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 TBranch objects are not recommended to store arrays and vectors in order to use other tools such as NumPy [11] and pandas [12]. In such a situation, we have to store the number of particles in a TBranch object 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 TBranch objects 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

179 supported by the program, and their acronyms will be used in the related item of the card file (see
180 next subsection for details).

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

186 *2.3. Algorithm of the program*

The program resolves physics processes from the input data introduced above. Considering the diversity of the data, the program first sorts them before translating them into physics processes. Here, the diversity means that the data representing a process may have multiple permutations. For example, the data for the decay $\rho^0 \rightarrow \pi^+ \pi^-$ have the following two permutations.

Number of particles	:	3
PDG codes of particles	:	<u>113, 211, -211</u> or <u>113, -211, 211</u>
Mother indices of particles	:	-1, 0, 0

187 A decay tree can consist of many decay branches. As a consequence, the diversity issue is
188 complex. To avoid the different permutations of one group of data are identified as different pro-
189 cesses, the program first sorts the input data to adjust all the possible permutations to a unique
190 order, according to the PDG codes and electronic charges of the involved particles, and the num-
191 bers of their daughter particles in the case of identical particles present in the same decay branch.
192 For example, the two permutations above will be finally sorted into the first permutation (113,
193 211, -211) in the program. The sorting algorithm is implemented in the source file “sortPs.cpp”,
194 where some other settings are also involved. One can see the reference file “sortPs.cpp_core” for
195 the core of the sorting algorithm. After the sorting, the program can get the decay tree from the
196 sorted data into a vector of the type “vector< list<int> >” with the function implemented in the
197 source file “getDcyTr.cpp”.

198 As mentioned in the previous section, the program has two categories of functionalities: sig-
199 nal identification and component analysis. In this subsection, we introduce the basic algorithms
200 for signal identification and component analysis by taking the cases of decay trees as examples.
201 Figures 2 and 3 show the flow charts of these algorithms in detail. Dozens of lines of code, in-
202 cluding some using the ROOT classes TChain [13], TFile [2], and TTree [8], are involved in the
203 charts in order to express the algorithms explicitly. The flow chart of the signal identification for
204 decay trees is depicted in Fig. 2. Firstly, the program reads in the signal decay trees specified in
205 the user card file. Then, for each entry of the input root file, the program obtains the decay tree
206 from the sorted input data, matches the decay tree to the signal decay trees, records the index of
207 the matched signal decay tree, and increases the number of the matched signal decay tree. At
208 last, the program outputs the statistics of the signal decay trees.

209 The flow chart of the component analysis over decay trees is illustrated in Fig. 3. Despite
210 the similarity in their frameworks, the flow chart has significant differences from that of the
211 signal identification for decay trees in Fig. 2. In the signal identification algorithm, the signal
212 decay trees to be identified are specified beforehand in the user card file. On the contrary, in
213 the component analysis algorithm, the program has to classify decay trees by itself from scratch.
214 In the signal identification algorithm, the decay trees are matched by directly comparing the
215 vectors storing them. Since the number of specified signal decay trees is fixed and usually small,
216 the processing rate of the program is high and usually in constant. However, in the component

217 analysis algorithm, the number of decay tree types found in a sample can be quite large and tends
 218 to grow with the number of processed entries. On this occasion, if we still match the decay trees
 219 by comparing the vectors storing them, the processing rate of the program will decrease with
 220 the increase of the number of processed entries. To improve the processing rate, the unordered
 221 map [14], a kind of container template introduced since the C++ 11 standard, is employed for the
 222 fast matching of decay trees. Internally, the elements in the unordered maps are organized into
 223 buckets depending on their hash values, to allow for fast access to individual elements directly by
 224 their key values with a constant average time complexity [14]. This constant feature in average
 225 time complexity will be examined in Section 2.5.

226 2.4. Execution of the program

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

```

233
234     An example of the card file containing the essential items is shown as follows.
235
236     # The following six items set the input of the program.
237
238     % Names of input root files
239     {
240         ./input/jpsi1.root
241         ./input/jpsi2.root
242     }
243
244     % TTree name
245     {
246         evt
247     }
248
249     % Storage type of input raw topology truth information (Four options: AOI, VOI, MSI, and MSD. Default: AOI)
250     {
251         AOI
252     }
253
254     % TBranch name of the number of particles (Default: nMCGen)
255     {
256         Nmcps
257     }
258
259     % TBranch name of the PDG codes of particles (Default: MCGenPDG)
260     {
261         Pid
262     }
263
264     % TBranch name of the mother indices of particles (Default: MCGenMothIndex)
265     {
266         Midx
267     }
268
269     # The following item sets the basic functionality of the program.
  
```

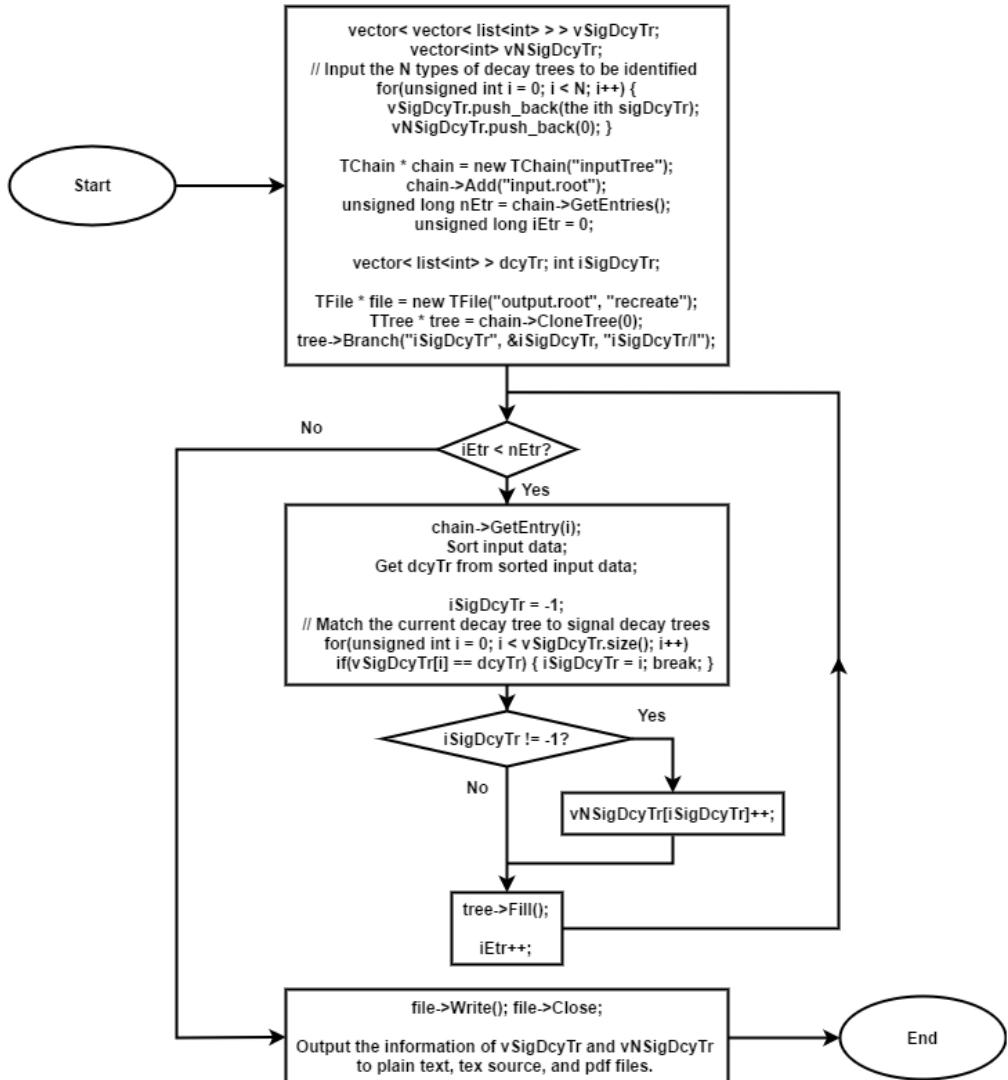


Figure 2: Basic flow chart of the signal identification for decay trees. The vectors “vSigDcyTr” and “vNSigDcyTr” are used to store the signal decay trees specified in the user card file and the numbers of these decay trees found in the input root file, respectively. The TBranch “iSigDcyTr” in the output root file is used to record the index of the signal decay tree involved in each entry of the input root file.

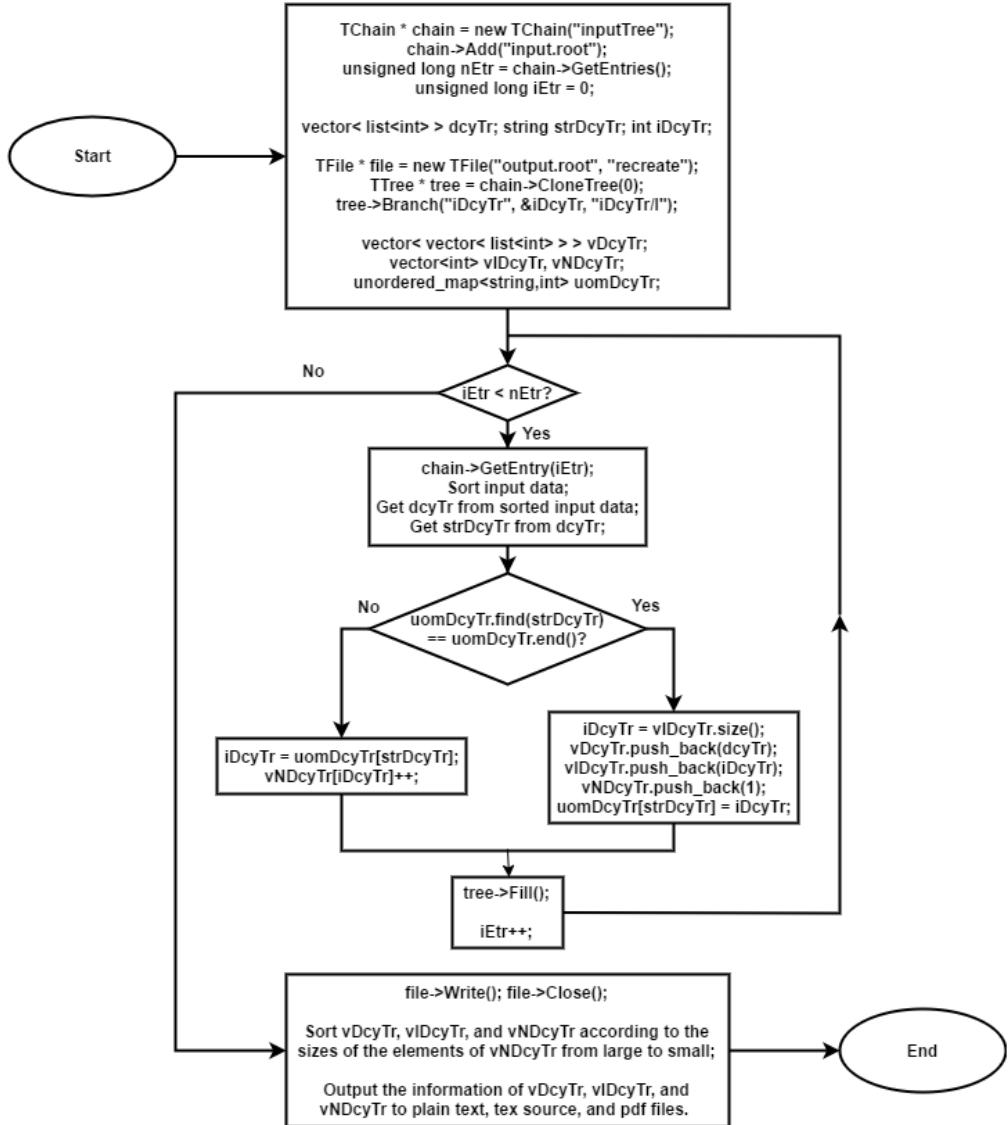


Figure 3: Basic flow chart of the component analysis over decay trees. The TBranch “iDcyTr” in the output root file is used to record the index of the decay tree involved in each entry of the input root file. The vectors “vDcyTr”, “vIDcyTr”, and “vNDcyTr” are used to store the decay trees found in the input root file, their individual indices, and their individual numbers, respectively. In addition, the unordered_map “uomDcyTr” is used for the fast matching of decay trees. Its key and value are the string “strDcyTr” and the index “iDcyTr”, respectively. Here, the string “strDcyTr” is constructed from the vector “dcyTr”; there is a one-to-one correspondence between them.

```

270      % Component analysis — decay trees
271      {
272          Y
273      }
274
275      # The following item sets the output of the program.
276
277      % Common name of output files (Default: Name of the card file)
278      {
279          jpsi_ta
280      }
281
282

```

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

286 The first item sets the names of the input root files. The names ought to be input one per
 287 line without tailing characters, such as comma, semicolon, and period. In the names, both the
 288 absolute and relative paths are allowed and wildcards “[]?*” are supported, just like those in the
 289 root file names input to the method Add() of the class TChain [13]. The second item specifies
 290 the TTree name. The third item tells the program the storage type of the input raw topology truth
 291 information, and the input should be one of the following four acronyms: AOI, VOI, MSI, and
 292 MSD, as we introduce in the previous subsection. The following three items set the TBranch
 293 names of the three ingredients of the input raw topology truth information. Of the first six items,
 294 the former two are indispensable, whereas the latter four can be removed or left empty if the
 295 input values are identical to the default values indicated in their prompts. Besides, the latter four
 296 items can be moved to the underlying card file, which is developed for frequently used items and
 297 will be introduced in Section 6.1, because the input values are usually fixed for a user or a group
 298 of users, though they might be different from the default values.

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

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

309 To provide a complete description, we list and explain all the essential items in the paragraphs
 310 above. However, in practical uses, we suggest removing the optional items if the input values
 311 are identical to the default ones, or moving them to the underlying card file if the input values
 312 are fixed for most of your use cases. In this way, the contents of the card file will become much
 313 more concise, making the use of the program easier and quicker. For example, unless otherwise
 314 stated, only the following two items are used to set the essential information in Sections 3, 4, and
 315 5.

```

316      % Names of input root files
317      {
318          ./input/mixed_1.root
319          ./input/mixed_2.root
320      }
321

```

```

322      % TTree name
323      {
324          evt
325      }
326
327

```

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

331 During the execution of the program, some standard output and error messages are printed to
332 the screen to provide some information on the input, progress, and output of the program, as well
333 as the possible problems and proposed solutions to them. The standard output messages include
334 the following four parts: (1) the values of the items with active inputs; (2) the total number of
335 entries contained in the input root files and the progress of the program to process these entries;
336 (3) the information output by the pdflatex command when it compiles the tex source file to
337 get the pdf file; (4) and the hints on the output of the program. The standard error messages
338 are prompted with “Error:” and “Infor:” in order to differentiate themselves from the standard
339 output messages. The messages started with “Error:” point out the problems encountered by the
340 program directly, while those started with “Infor:” give more information on the problems as
341 well as some guidelines on the solutions to the problems.

342 *2.5. Performance of the program*

343 Besides the performance of the used computing systems, the processing rate of the program is
344 largely related to the characteristics of the samples, particularly the average number of generated
345 particles in each event. Figure 4 shows the performance study of the program with the J/ψ sample
346 used in the example of this section as well as the $\tau^+\tau^-$, $d\bar{d}$, $u\bar{u}$, $s\bar{s}$, $c\bar{c}$, B^+B^- , and $B^0\bar{B}^0$ samples
347 generated at the peak energy of the $\Upsilon(4S)$ resonance. Each of the used samples consists of one
348 hundred thousand events. From the left plot in the figure, for all the samples, the number of
349 elapsed seconds grows linearly with the number of processed entries. This linear pattern is a nice
350 feature. It guarantees the program has a high rate even in the case of processing huge samples.
351 For example, the program can process one hundred thousand J/ψ events within five seconds.
352 Here, we note that the linear pattern is the result of fast searches with unordered maps [14], as
353 we discuss in Section 2.3. On the other hand, the processing rate of the program varies with
354 the processed samples. The right plot in Fig. 4 shows the relationship between the total number
355 of elapsed seconds over the whole sample and the average number of generated particles in an
356 event. Clearly, a linear pattern is also observed in the plot. To be specific, with the average
357 number of generated particles in an event increasing by one, the total number of elapsed seconds
358 over the whole sample increases by about 0.56.

359 *2.6. Output of the program*

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

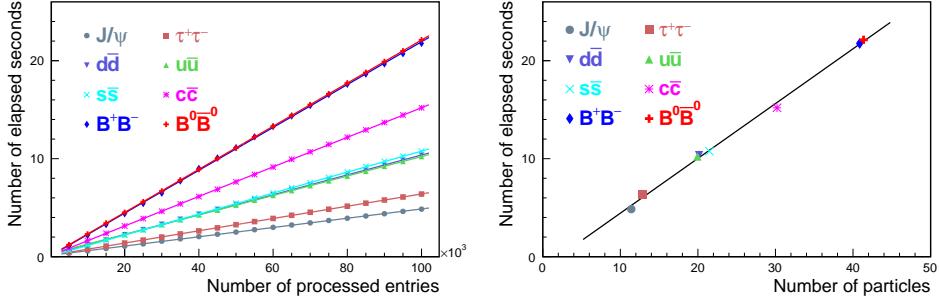


Figure 4: Performance study of the program with the J/ψ sample as well as the $\tau^+\tau^-$, $d\bar{d}$, $u\bar{u}$, $s\bar{s}$, $c\bar{c}$, B^+B^- , and $B^0\bar{B}^0$ samples generated at the peak energy of the $\Upsilon(4S)$ resonance. The left plot demonstrates the changing trends of the number of elapsed seconds with respect to the number of processed entries. The right plot illustrates the relationship between the total number of elapsed seconds over the whole sample and the average number of generated particles in an event. In both plots, the dots show the timing data from the standard output of the program, and the lines display the results of fitting linear functions to the data.

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

377 In addition to the three files for topology maps, one or more root files are output to save
 378 topology tags. The root files only include one TTree object, which is entirely the same as that in
 379 the input root files, except for the topology tags inserted in all of its entries. The number of root
 380 files depends on the size of output data. The program switches to one new root file whenever the
 381 size of the TTree object in memory exceeds 3 GB. In the case of the size less than 3 GB, only
 382 one root file is output. While the sole or first root file has the same name as the three files above,
 383 more possible root files are denominated with the suffix “_n” (n=1, 2, 3, and so on) appended to
 384 the name. In the example, the first root file is “jpsi_ta.root”, and more possible root files would
 385 be “jpsi_ta_1.root”, “jpsi_ta_2.root”, “jpsi_ta_3.root”, and so on.

386 In the example of the previous subsection, the program conducts its basic functionality, namely
 387 the component analysis over decay trees. From the 100000 events of the input sample, the
 388 program recognizes 17424 decay trees and outputs all of them to the plain text, tex source, and
 389 pdf files. Table 1 only shows the top ten decay trees and their respective final states listed in the
 390 output pdf file. With the help of the symbolic expressions, the components of the sample are
 391 clearly displayed in the table, which brings great convenience to us in examining the signals and
 392 backgrounds involved in the sample. In the table, “rowNo”, “iDcyTr”, “nEtr”, and “nCEtr” are
 393 abbreviations for the row number, index of decay tree, number of entries of decay tree, and num-

394 ber of the cumulative entries from the first to the current decay trees, respectively. The values of
 395 “iDcyTr” are assigned from small to large in the program but listed according to the values of
 396 “nEtr” from large to small in the table. This is the reason why they are not in natural order like
 397 the values of “rowNo”. Since J/ψ is the only root particle for the J/ψ sample, the production
 398 branch $e^+e^- \rightarrow J/\psi$ is omitted to save page space. Similar rules also apply to other samples with
 399 only one root particle. Considering π^0 has a very large production rate and approximatively 99%
 400 of it decays to $\gamma\gamma$, the program is designed to discard the decay $\pi^0 \rightarrow \gamma\gamma$ by default at the early
 401 phase of processing the input data (see Section 5.1.2 for the setting item to alter the behavior).
 402 As a result, $\pi^0 \rightarrow \gamma\gamma$ does not show itself in the table. Besides, the superscripts “ f ” and “ F ” in
 403 γ^f and γ^F indicate the final state radiation effect (see Section 5.1.3 for their difference).

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	$J/\psi \rightarrow \pi^0\pi^+\pi^+\pi^-\pi^-$	$\pi^0\pi^+\pi^+\pi^-\pi^-$	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	$J/\psi \rightarrow \pi^0\pi^0\pi^+\pi^+\pi^-\pi^-$	$\pi^0\pi^0\pi^+\pi^+\pi^-\pi^-$	127	1161	20061
9	$J/\psi \rightarrow \pi^0\pi^+\pi^+\pi^+\pi^-\pi^-\pi^-\pi^-$	$\pi^0\pi^+\pi^+\pi^+\pi^-\pi^-\pi^-\pi^-$	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

404 In the table, “iDcyTr” is the topology tag for decay trees. Thus, it is also saved in the TTree
 405 objects of the output root file, together with other quantities for physics analysis. Therefore, it
 406 can be used to pick out the entries of specific decay trees and then examine the distributions of
 407 the other quantities over the decay trees. In the example, besides the raw topology truth informa-
 408 tion, only a random variable following the standardized normal distribution, namely X, is stored
 409 in the input root files and thus copied by default to the output root file. Though not a genuine
 410 variable for physics analysis, X is quite good to illustrate the usage of the topology tag. Figure 5
 411 shows the distribution of X accumulated over the top ten decay trees. The figure is drawn with
 412 the root script

413 examples/in_the_user_guide/ex_for_tb_01/draw_X/v2/draw_X.C,

416 where, for example, a statement equivalent to

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

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

425 requirements on the displayed quantities.

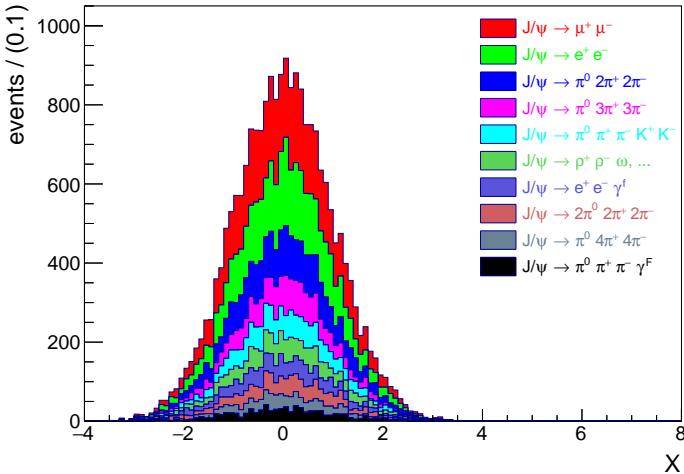


Figure 5: Distribution of X accumulated over the top ten decay trees. In the legend entry “ $J/\psi \rightarrow \rho^+ \rho^- \omega, \dots$ ”, the dots “ \dots ” represent the secondary decay branches: $\rho^+ \rightarrow \pi^0 \pi^+$, $\rho^- \rightarrow \pi^0 \pi^-$, $\omega \rightarrow \pi^0 \pi^+ \pi^-$.

426 2.7. Validation of the program

427 The decay trees displayed in Table 1 are relatively simple, and we can check their correctness
 428 by examining the input data directly. To validate the program generally, we need to do input and
 429 output checks, where some arbitrary physics processes are generated as the input of the program.
 430 The output has to be consistent with the input; otherwise, there must be some bugs in the program
 431 and we have to fix them. A large number of such checks have been performed in the develop-
 432 ment and application of the program, and some of them can be found under the sub-directory
 433 “examples/validation” of the package. These checks are divided into two groups: standalone and
 434 combined. In the standalone checks, forty exclusive J/ψ and $\Upsilon(4S)$ decays modeled with the
 435 EvtGen [15] generator are used to test the functionality of resolving decay trees. In the com-
 436 bined checks, randomly combined samples of these exclusive decays are used for verifying the
 437 functionalities of counting and tagging decay trees. The output agrees with the input in all the
 438 checks, which indicates the correctness of the program.

439 3. Component analysis

440 Component analysis is the primary functionality of the program. It is developed mainly for
 441 the background analysis involved in our physics studies. We perform it over decay trees in the
 442 previous example. Also, it can be carried out as follows: over decay initial-final states; with
 443 specified particles to check their decay branches, production branches, mothers, cascade decay
 444 branches, and decay final states; with specified inclusive decay branches to examine their exclu-
 445 sive components; and with specified intermediate-resonance-allowed (IRA) decay branches to
 446 investigate their inner structures. This section introduces the nine (five for specified particles)
 447 kinds of component analysis, with each in a subsection. For each kind of component analysis,

448 one item is designed and implemented in the program to set related parameters. In each subsection,
449 we take an example to demonstrate the corresponding setting item and show the resulting
450 topology map. For easy exposition, all of the essential topology tags involved in the component
451 analysis functionalities are presented in another separate subsection, namely the last subsection.

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

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

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

476 3.1. Decay trees

477 Component analysis over decay trees is the basic kind of topology analysis. It is quite useful
478 to study the backgrounds involved in our research works where the signals are the complete decay
479 trees fully reconstructed from final state particles. It has already been widely performed in the
480 BESIII experiment, as illustrated in the previous section with the J/ψ example. This subsection
481 introduces it further with the available optional settings using the $\Upsilon(4S)$ sample. The following
482 example shows the associated item with the maximum number of output components set to five.
483 In the item, a third parameter is also filled and set to “Y”. With the setting, the decay final states
484 in the output pdf file are put under their respective decay trees, rather than in a column next to
485 that for decay trees. It is recommended to use this optional parameter in cases there are too many
486 (about ten or more) particles in some final states. Here, we note that the symbol “–” can be used
487 as a placeholder for the maximum number of output components, if only the third parameter is
488 desired.

489
490
491 % Component analysis — decay trees
492 {
493 Y 5 Y
494 }

495

Component analysis over decay trees is one kind of the most time-consuming topology analysis tasks. To check further the efficiency of the program, the progress of running this example, in addition to the example in Section 2.4, is illustrated in the plots of Fig. 4 as well. In these plots, the timing data from this example are marked with the legend entry “ $B^0\bar{B}^0$ ”. Since the decay of the $\Upsilon(4S)$ resonance is more complex than that of the J/ψ resonance, it takes more than twenty seconds for the program to process one hundred thousand events in this example. Nonetheless, the program still has a high processing rate.

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. Here, we note that the events are not densely populated over the first five decay trees because the inclusive $\Upsilon(4S)$ sample used here is not selected beforehand with any requirements. In the symbolic expressions of decay initial-final states, the dashed right arrow ($--\rightarrow$) instead of the plain right arrow (\rightarrow) is used, in order to reflect that the initial states do 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 3.9.

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

rowNo	decay tree (decay initial-final states)	iDcyTr	nEtr	nCEtr
1	$\Upsilon(4S) \rightarrow B^0\bar{B}^0, B^0 \rightarrow e^+\nu_e D^{*-}\gamma^F, \bar{B}^0 \rightarrow \mu^-\bar{\nu}_\mu D^{*+}, D^{*-} \rightarrow \pi^-\bar{D}^0,$ $D^{*+} \rightarrow \pi^+ D^0, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) \dashrightarrow e^+\nu_e \mu^-\bar{\nu}_\mu \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- K^+ K^- \gamma^F)$	20870	3	3
2	$\Upsilon(4S) \rightarrow B^0\bar{B}^0, B^0 \rightarrow \mu^+\nu_\mu D^{*-}, \bar{B}^0 \rightarrow \rho^- D^{*+}, D^{*-} \rightarrow \pi^-\bar{D}^0,$ $\rho^- \rightarrow \pi^0 \pi^-, D^{*+} \rightarrow \pi^0 D^0, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^+ \rightarrow \pi^+ \pi^+ K^-$ $(\Upsilon(4S) \dashrightarrow \mu^+\nu_\mu \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- K^+ K^-)$	3648	2	5
3	$\Upsilon(4S) \rightarrow B^0\bar{B}^0, B^0 \rightarrow \pi^0 \pi^+ \pi^+ \rho^- D^-, \bar{B}^0 \rightarrow \mu^-\bar{\nu}_\mu D^{*+}, \rho^- \rightarrow \pi^0 \pi^-,$ $D^- \rightarrow \pi^- \pi^- K^+, D^{*+} \rightarrow \pi^+ D^0, D^0 \rightarrow K_L^0 \pi^+ \pi^-$ $(\Upsilon(4S) \dashrightarrow \mu^-\bar{\nu}_\mu \pi^0 \pi^0 K_L^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+)$	5295	2	7
4	$\Upsilon(4S) \rightarrow B^0\bar{B}^0, B^0 \rightarrow \mu^+\nu_\mu D^{*-}, \bar{B}^0 \rightarrow e^-\bar{\nu}_e D^+, D^{*-} \rightarrow \pi^-\bar{D}^0,$ $D^+ \rightarrow e^+\nu_e \tilde{K}^*, \bar{D}^0 \rightarrow \pi^0 \pi^+ \pi^- K_S^0, \tilde{K}^* \rightarrow \pi^0 \tilde{K}^0, K_S^0 \rightarrow \pi^+ \pi^-, \tilde{K}^0 \rightarrow K_L^0$ $(\Upsilon(4S) \dashrightarrow e^+\nu_e \bar{\nu}_e \mu^+\nu_\mu \pi^0 \pi^0 K_L^0 \pi^+ \pi^- \pi^- \pi^-)$	11954	2	9
5	$\Upsilon(4S) \rightarrow B^0\bar{B}^0, B^0 \rightarrow e^+\nu_e D^{*-}, \bar{B}^0 \rightarrow \pi^0 \pi^- \omega D^+, D^{*-} \rightarrow \pi^-\bar{D}^0,$ $\omega \rightarrow \pi^0 \pi^+ \pi^-, D^+ \rightarrow e^+\nu_e \pi^+ K^-, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+$ $(\Upsilon(4S) \dashrightarrow e^+\nu_e \nu_e \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+ K^-)$	14345	2	11
rest	$\Upsilon(4S) \rightarrow \text{others (99980 in total)}$ $(\Upsilon(4S) \dashrightarrow \text{corresponding to others})$	—	99989	100000

511 3.2. Decay initial-final states

512 On some occasions, we need to investigate the decay initial-final states of backgrounds for
513 some sophisticated physics analyses. Particularly, it is necessary to differentiate the following
514 two fundamental types of backgrounds: the one with the same initial-final states as the signal,
515 and the other with different initial-final states from the signal. While the latter type of back-
516 grounds needs to be suppressed as much as possible, the former type usually needs to be kept to
517 study more physical effects, for example, the interference effect. Besides, examining the decay
518 initial-final states of backgrounds sheds light on the misjudgment of final state particles at the
519 level of signal candidates. Below is an example demonstrating the related item with the maxi-
520 mum number of output components set to five.

521

522 % Component analysis — decay initial-final states

```

523   {
524     Y 5
525   }
526

```

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

Table 3: Decay initial-final states.

rowNo	decay initial-final states	iDcyIFsts	nEtr	nCEtr
1	$\Upsilon(4S) \rightarrow \mu^+ \nu_\mu \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+ K^-$	41	18	18
2	$\Upsilon(4S) \rightarrow \pi^0 \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-$	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	$\Upsilon(4S) \rightarrow \pi^0 \pi^0 \pi^0 \pi^0 \pi^0 K_L^0 \pi^+ \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- K^-$	1207	17	71
5	$\Upsilon(4S) \rightarrow \pi^0 \pi^0 \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-$	1215	17	88
rest	$\Upsilon(4S) \rightarrow$ others (78208 in total)	—	99912	100000

529 3.3. Decay branches of particles

530 The invariant mass constraint is one of the most frequently used event selection requirements
531 in high energy physics experiments. With the requirement applied to certain particle, the main
532 backgrounds (especially the peaking ones) to its signal decay mode are very likely to be its other
533 decay modes. In this case, it is significant to examine the decay branches of the particle. The
534 following example shows the associated item with the two particles D^{*+} and J/ψ set as research
535 objects. In the item, each row holds the information of a specified particle, and the first, sec-
536 ond and third columns are the textual expressions, aliases, and maximum numbers of output
537 components, respectively. As we introduce at the beginning part of this section, the aliases and
538 maximum numbers of output components are both optional. Here, we note that the symbol “–”
539 can be used as a placeholder for an unassigned alias, if only the maximum number of output
540 components is desired.

```

541
542 % Component analysis — decay branches of particles
543 {
544   D*+  Dsp  5
545   J/psi Jpsi 5
546 }
547

```

548 Table 4 shows the decay branches of D^{*+} . From the table, only four decay branches of D^{*+} are
549 found in the input inclusive MC sample. Since there is likely one or more cases of D^{*+} decays in
550 one input entry, “nCase” and “nCCase”, instead of “nEtr” and “nCEtr”, are used in the table in
551 order to accurately indicate what we are counting are the numbers of D^{*+} decays, rather than the
552 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

553 3.4. Production branches of particles

554 In some cases, we have interest in the production branches of certain particles. Below is an
 555 example demonstrating the related item also by taking the two particles D^{*+} and J/ψ as objects
 556 of study. The input to this item is the same as that to the above item.

```
557
558     % Component analysis — production branches of particles
559     {
560         D*+   Dsp   5
561         J/psi  Jpsi  5
562     }
```

563
 564 The production branches of D^{*+} are displayed in Table 5. In the production branches, D^{*+} is
 565 marked in blue so as to make it noticeable. From the table, the number of production branches
 566 of D^{*+} found in the input sample is 3277, much bigger than 4, which is the number of its decay
 567 branches.

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

568 3.5. Mothers of particles

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

```
572
573     % Component analysis — mothers of particles
574     {
575         D*+   Dsp   5
576         J/psi  Jpsi  5
577     }
```

578
 579 Table 6 shows the mothers of D^{*+} . Notably, the PDG codes of the mother particles, instead of
 580 additional indices, are listed in the table, since they are sufficient to tag the mother particles.
 581 From the table, six sources of D^{*+} are found in the input sample and the dominant one is the \bar{B}^0
 582 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

583 3.6. Cascade decay branches of particles

584 Sometimes, the invariant mass constraint is applied to certain particle and the signal pro-
 585 cess is its cascade decay branch. In this case, it is necessary to investigate the cascade decay
 586 branches of the particle, rather than its first decay branches, so as to analyze the background-
 587 s effectively. Below is an example demonstrating the related item by taking the two particles
 588 B^0 and D^0 as objects of study. While the first three columns of the input to this item have the
 589 same meanings as those to the three items above, the additional fourth column sets the maximum
 590 hierarchy of decay branches to be examined. Here, the hierarchy reflects the rank of a decay
 591 branch in a cascade decay branch of one specific particle. For instance, in the following cascade
 592 decay branch of B^0 : $B^0 \rightarrow \pi^0 \pi^0 \rho^0 \pi^+ D^{*-}$, $\rho^0 \rightarrow \pi^+ \pi^-$, $D^{*-} \rightarrow \pi^- \bar{D}^0$, $\bar{D}^0 \rightarrow \eta \eta'$, $\eta \rightarrow \pi^0 \pi^0 \pi^0$,
 593 $\eta' \rightarrow \pi^0 \pi^0 \eta$, $\eta \rightarrow \gamma\gamma$, the hierarchies of the seven individual decay branches are 1, 2, 2, 3, 4, 4,
 594 and 5, respectively. In the example, the maximum hierarchy of decay branches is set to two for
 595 both B^0 and D^0 , and hence only the first two hierarchies of branches in their cascade decays will
 596 be investigated. Without such settings, all the branches in their cascade decays will be examined.

597
 598 % Component analysis — cascade decay branches of particles
 599 {
 600 B0 B0 5 2
 601 D0 D0 5 2
 602 }

603 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

605 3.7. Decay final states of particles

606 When the invariant mass constraint is applied to certain particle reconstructed directly from
 607 a specific final state, it is significant to examine the decay final states of the particle, rather than
 608 its first or cascade decay branches, in order to study the backgrounds effectively. The following
 609 example shows the associated item also with the two particles B^0 and D^0 set as research objects.
 610 The format of the input to the item is the same as that to the above item, but the fourth parameters
 611 here are designed to restrict the numbers of final state particles. Without the fourth parameters,
 612 all the decay final states of the specified particles will be investigated. In the example, the pa-
 613 rameters are set to three for both B^0 and D^0 , and thus only the three-body decay final states of
 614 them will be examined.

615
 616 % Component analysis — decay final states of particles
 617 {
 618 B0 B0 5 3
 619 D0 D0 5 3
 620 }

621 Table 8 shows the three-body decay final states of D^0 . In the table, π^0 only decays to $\gamma\gamma$; oth-
 622 erwise, it will be replaced with its decay products, resulting in different decay final states of
 623 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	$D^0 \rightarrow$ others (24 in total)	—	2407	13620

625 3.8. Inclusive decay branches

626 In a few physics studies, we take inclusive decay branches as signals. In such cases, it is es-
 627 sential to have a basic knowledge of the exclusive components of these inclusive decay branches.
 628 Below is an example demonstrating the related item by investigating the exclusive components
 629 of the two inclusive decay branches $\bar{B}^0 \rightarrow D^{*+} + \text{anything}$ and $B^0 \rightarrow K_S^0 + \text{anything}$. In the
 630 item, each row holds the information of an inclusive decay branch, and the first, second, and
 631 third columns separated with the symbol “&” are the textual expressions, aliases, and maximum
 632 numbers of output components, respectively. As we introduce at the beginning part of this sec-
 633 tion, the aliases and maximum numbers of output components are both optional. Here, we note
 634 that the symbol “–” can be used as a placeholder for an unassigned alias, if only the maximum
 635 number of output components is desired.

```
636
637 % Component analysis — inclusive decay branches
638 {
639   B0 --> D*+ & B2Dsp & 5
640   B0 --> K_S0 & B2Ks & 5
641 }
```

642 The exclusive components of $B^0 \rightarrow K_S^0 + \text{anything}$ are displayed in Table 9. From the table,
 643 ten exclusive components of the inclusive decay branch are found in the input sample, and the
 644 particles denoted with *anything* are mainly the traditional charmonium states.
 645

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

646 3.9. Intermediate-resonance-allowed decay branches

647 In many research works, we take multi-body decay branches as signals. On such occasions,
 648 it is fundamental to investigate the intermediate resonances involved in these decay branches.
 649 In other words, we need to examine the exclusive components of these IRA decay branches.
 650 The following example shows the associated item with the two IRA decay branches $D^{*+} \rightarrow$
 651 $\pi^0 \pi^+ \pi^- K^-$ and $J/\psi \rightarrow \pi^0 \pi^+ \pi^-$ set as objects of study. Since IRA decay branches look like
 652 inclusive decay branches, the format of the input to the item for IRA decay branches is identical
 653 to that for inclusive decay branches, which is introduced in the previous subsection.
 654

```

655 % Component analysis — intermediate-resonance-allowed decay branches
656 {
657 D*+ --> K- pi+ pi+ pi0 & Dsp2K3Pi & 5
658 J/psi --> pi+ pi- pi0 & Jpsi23Pi & 5
659 }

```

660
661 Table 10 shows the exclusive components of $D^{*+} \rightarrow \pi^0\pi^+\pi^+K^-$. From the table, two intermediate-
662 particles D^0 and D^+ are found in the IRA decay branch, and they decay to $\pi^0\pi^+K^-$ and
663 $\pi^+\pi^+K^-$, respectively.

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	$D^{*+} \rightarrow \pi^0D^+, D^+ \rightarrow \pi^+\pi^+K^-$	1	1102	4971

664 3.10. Essential topology tags

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

Component type	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 _i s (or its decay branches)
	iDcyBrP_i_j	index of decay branch of the j th particle _i
Production branches of particles	nPProdBr_i	number of particle _i s (or its production branches)
	iProdBrP_i_j	index of production branch of the j th particle _i
Mothers of particles	nPMoth_i	number of particle _i s (or its mothers)
	PDGMothP_i_j	PDG code of mother of the j th particle _i
Cascade decay branches of particles	nPCascDcyBr_i	number of particle _i s (or its cascade decay branches)
	iCascDcyBrP_i_j	index of cascade decay branch of the j th particle _i
Decay final states of particles	nPDcyFSt_i	number of particle _i s (or its decay final states)
	iDcyFStP_i_j	index of decay final state of the j th particle _i
Inclusive decay branches	nIncDcyBr_i	number of inclusive decay branch _i s
	iDcyBrIncDcyBr_i_j	index of decay branch of the j th inclusive decay branch _i
IRA decay branches	nIRADcyBr_i	number of IRA decay branch _i s
	iDcyBrIRADcyBr_i_j	index of decay branch of the j th IRA decay branch _i

665 Table 11 lists and interprets all of the essential topology tags involved in the component
666 analysis functionalities. The topology tag for the component analysis over decay initial-final
667 states is iDcyIFsts. It has a similar interpretation as iDcyTr and is shown in the third column
668 of Table 3. For the latter seven kinds of component analysis, there are two sorts of topology
669 tags. The first sort, such as nPDcyBr_i, records the number of instances of the ith specified
670 particle or decay branch found in each event. The second sort, for example, iDcyBrP_i_j, keeps
671 the associated index of the jth found instance of the ith specified particle or decay branch. The
672 indices and the decays they stand for can be found in Tables 4 – 10.

673 In the topology tags, “i” in “_i” is the default index of the specified particle or decay branch,
674 and it ranges from 0 (included) to the number of specified particles or decay branches (excluded).
675 If the alias of the particle or decay branch is also specified, the index “i” will be replaced with

676 the alias. For example, since “Dsp” and “Jpsi” are set as the aliases of D^{*+} and J/ψ in the
677 component analysis over their decay branches, the specialized topology tags nPDcyBr.Dsp and
678 nPDcyBr.Jpsi, instead of the default ones nPDcyBr_0 and nPDcyBr_1, are used to store the
679 numbers of D^{*+} and J/ψ found in each event.

680 In addition, “j” in “_j” is the default index of the found instance of certain particle or decay
681 branch in an event, and it ranges from 0 (included) to the sample-level maximum of the number
682 of the particles or decay branches found in each event (excluded). For example, the maximum of
683 the number of D^{*+} found in each event is two for the whole sample, and thus two topology tags
684 iDcyBrP.Dsp.0 and iDcyBrP.Dsp.1 are employed to store the indices of D^{*+} decay branches.
685 These indices range from 0 (included) to the number of the types of D^{*+} decay branches found
686 in the samples (excluded). In the events with only one D^{*+} , iDcyBrP.Dsp.1 is assigned with
687 the default value -1; in the events that have no D^{*+} , the default value -1 is assigned to both
688 iDcyBrP.Dsp.0 and iDcyBrP.Dsp.1. We note that different from all other indices, PDGMoth_i.j
689 has the default value 0, instead of -1.

690 4. Signal identification

691 Signal identification is the other functionality of the program. Though relatively simple, it
692 can help us identify the “signals” we desire directly, quickly, and easily. Here, the “signals”
693 are not confined to the authentic signals in our research works but can be any physics processes
694 of interests, particularly some important backgrounds we concern. At present, the following
695 eight kinds of signals can be identified with the program: (1) decay trees, (2) decay initial-final
696 states, (3) particles, (4) (regular) decay branches, (5) cascade decay branches, (6) inclusive decay
697 branches, (7) inclusive cascade decay branches, and (8) IRA decay branches. For each kind of
698 signals, one item is developed to specify related parameters. This section introduces the eight
699 kinds of signal identification, with each in a subsection. In each subsection, we take an example
700 to demonstrate the related setting item and show the obtained topology map. For easy exposition,
701 all of the essential topology tags involved in the signal identification functionalities are presented
702 in another separate subsection, that is, the last subsection.

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

708 4.1. Decay trees

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

720 % Signal identification — decay trees

```

722 {
723   0 & Upsilon(4S) --> B0 anti-B0 & -1 & 1stDcyTrInTb2 & Y
724   1 & B0 --> e+ nu.e D*- gamma & 0
725   2 & anti-B0 --> mu- anti-nu.mu D*+ & 0
726   3 & D*- --> pi- anti-D0 & 1
727   4 & D*+ --> pi+ D0 & 2
728   5 & anti-D0 --> pi0 pi- K+ & 3
729   6 & D0 --> pi0 pi+ K- & 4
730
731   0 & Upsilon(4S) --> B0 anti-B0 & -1 & 2ndDcyTrInTb2
732   1 & B0 --> mu+ nu_mu D*- & 0
733   2 & anti-B0 --> rho- D*+ & 0
734   3 & D*- --> pi- anti-D0 & 1
735   4 & rho- --> pi0 pi- & 2
736   5 & D*+ --> pi0 D+ & 2
737   6 & anti-D0 --> pi0 pi- K+ & 3
738   7 & D+ --> pi+ pi+ K- & 5
739 }
740

```

Table 12 shows the resulting topology map. The results are the same as those displayed in the 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	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $D^{*+} \rightarrow \pi^+ D^0, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) \rightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu \pi^0 \pi^0 \pi^+ \pi^- \pi^- \pi^- K^+ K^- \gamma^F)$	0	3	3
2	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, \bar{B}^0 \rightarrow \rho^- D^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $\rho^- \rightarrow \pi^0 \pi^-, D^{*+} \rightarrow \pi^0 D^+, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^+ \rightarrow \pi^+ \pi^+ K^-$ $(\Upsilon(4S) \rightarrow \mu^+ \nu_\mu \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+ K^-)$	1	2	5

4.2. Decay initial-final states

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	$\Upsilon(4S) \rightarrow \pi^0 \pi^0 \pi^0 \pi^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ K^-$	1	18	36

In a few cases, we have an interest in some decay initial-final states. Below is an example demonstrating the related item by taking the first two decay initial-final states listed in Table 3 as signals. Similar to IRA decay branches, decay initial-final states look like inclusive decay branches. Hence, except that only two columns are involved in the item, the format of the input to the item for decay initial-final states is identical to that for the component analysis over inclusive decay branches, which is introduced in Section 3.8. As we can see from the example, the numbers of identical particles are supported to be written in front of their textual names in order to simplify the textual expressions of the final states. The obtained topology map is displayed in Table 13. The results are identical to those shown in the first two rows of Table 3.

```

753
754 % Signal identification — decay initial-final states
755 {
756   Y(4S) --> mu+ nu_mu 3 pi0 3 pi+ 4 pi- K+ K- & 2ndDcyIFStsInTb3
757   Y(4S) --> 5 pi0 5 pi+ 5 pi- K+ K- & 2ndDcyIFStsInTb3

```

758 }

759 4.3. Particles

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

```
764
765     % Signal identification — particles
766     {
767         D*+   Dsp
768         J/psi Jpsi
769     }
```

770
 771 Table 14 shows the resulting topology map. As a cross-check, the number of D^{*+} s in the table
 772 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/ψ	1	2654	48540

773 4.4. Decay branches

774 On some occasions, we have to identify certain regular decay branches. Below is an ex-
 775 ample demonstrating the related item by taking the two decay branches $\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$ and
 776 $B^0 \rightarrow K_S^0 J/\psi$ as signals. Since regular decay branches also look like inclusive decay branches,
 777 except that only two columns are involved in the item, the format of the input to the item for reg-
 778 ular decay branches is identical to that for the component analysis over inclusive decay branches,
 779 which is introduced in Section 3.8.

```
780
781     % Signal identification — decay branches
782     {
783         B0 --> mu- anti-nu_mu D*+ & B2munuDsp
784         B0 --> K_S0 J/psi & B2KsJpsi
785     }
```

786
 787 The obtained topology map is displayed Table 15. For cross-checks, we note that the number of
 788 $\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$ ($B^0 \rightarrow K_S^0 J/\psi$) in the table is equal to that in the first row of Table 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

789 4.5. Cascade decay branches

790 Sometimes, we are interested in certain cascade decay branches. The following example
 791 shows the associated item with the two cascade decay branches $B^0 \rightarrow D^{*-} D_s^{*+}$, $D^{*-} \rightarrow \pi^- \bar{D}^0$,
 792 $D_s^{*+} \rightarrow D_s^+ \gamma$ and $B^0 \rightarrow D^{*-} D_s^{*+}$, $D^{*-} \rightarrow \pi^- \bar{D}^0$ set as signals. While the first cascade decay

793 branch is identical to the fifth one in Table 7, the second is only part of it, which demonstrates
 794 that the cascade decay branches supported in the item are not necessarily fully specified at the
 795 level of certain hierarchy. Similar to decay trees, cascade decay branches are made up of regular
 796 decay branches. Hence, the format of the input to the item for cascade decay branches is identical
 797 to that for decay trees, which is introduced in Section 4.1.

```

798 % Signal identification — cascade decay branches
799 {
800   0 & B0 --> D*- D_s*+ & -1
801   1 & D*- --> pi- anti-D0 & 0
802   2 & D_s*+ --> D_s+ gamma & 0
803
804   0 & B0 --> D*- D_s*+ & -1
805   1 & D*- --> pi- anti-D0 & 0
806 }
807
808
809 Table 16 shows the resulting topology map. As a cross-check, the number of cases of the first
810 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

811 4.6. Inclusive decay branches

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

```

817 % Signal identification — inclusive decay branches
818 {
819   anti-B0 --> D*+ & B2Dsp
820   B0 --> K_S0 & B2Ks
821 }
822
823
824 The obtained topology map is displayed in Table 17. As a cross-check, the number of  $B^0 \rightarrow$ 
825  $K_S^0 + \text{anything}$  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

826 4.7. Inclusive cascade decay branches

827 Occasionally, we may have an interest in certain inclusive cascade decay branches. The
 828 following example shows the associated item with the two inclusive cascade decay branches
 829 $\bar{B}^0 \rightarrow D^{*+} + \text{anything}$, $D^{*+} \rightarrow \pi^+ D^0$ and $B^0 \rightarrow K_S^0 J/\psi$, $K_S^0 \rightarrow \pi^+ \pi^-$, $J/\psi \rightarrow \mu^+ + \text{anything}$ set

830 as signals. Similar to decay trees and cascade decay branches, inclusive cascade decay branches
 831 are made up of regular decay branches. Hence, the format of the input to the item for inclusive
 832 cascade decay branches is also identical to that for decay trees, which is introduced in Section
 833 [4.1](#). and the independent textual name “*” denotes anything.

```
834
835     % Signal identification — inclusive cascade decay branches
836     {
837         0 & anti-B0 --> D*+ * & -1
838         1 & D*+ --> pi+ D0 & 0
839
840         0 & B0 --> K_S0 J/psi & -1
841         1 & K_S0 --> pi+ pi- & 0
842         2 & J/psi --> mu+ * & 0
843     }
844
```

845 [Table 18](#) shows the resulting topology map.

Table 18: Signal inclusive cascade decay branches.

rowNo	signal inclusive cascade decay branch	iSigIncCascDcyBrs	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

846 4.8. Intermediate-resonance-allowed decay branches

847 On some occasions, we need to identify certain IRA decay branches. Below is an example
 848 demonstrating the related item by taking the two IRA decay branches $D^{*+} \rightarrow \pi^0 \pi^+ \pi^+ K^-$ and
 849 $J/\psi \rightarrow \pi^0 \pi^+ \pi^-$ as signals. Except that only two columns are involved in the item, the format
 850 of the input to the item is identical to that for the component analysis over IRA decay branches,
 851 which is introduced in Section [3.9](#).

```
852
853     % Signal identification — intermediate-resonance-allowed decay branches
854     {
855         D*+ --> K- pi+ pi+ pi0 & Dsp2K3Pi
856         J/psi --> pi+ pi- pi0 & Jpsi23Pi
857     }
```

858 The obtained topology map is displayed in [Table 19](#). For the purpose of cross-checks, we
 859 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

861 4.9. Essential topology tags

862 Table [20](#) summarizes and explains all of the essential topology tags involved in the signal
 863 identification functionalities. For signal decay trees and signal decay initial-final states, there are
 864 two sorts of topology tags. The first sort of tags, iSigDcyTr and iSigDcyIFsts, record the default
 865 indices of the specified signal decay trees and signal decay initial-final states. They have similar
 866 interpretations as iDcyTr and iDcyIFsts, and are shown in the third columns of Tables [12](#) and

867 13. The second sort of tags, nameSigDcyTr and nameSigDcyIFsts, save the specified aliases of
 868 the signal decay trees and signal decay initial-final states. In cases the aliases are not specified,
 869 empty strings will be stored.

870 For the latter six kinds of signal identification, there is only one sort of topology tags, which
 871 records the number of instances of certain specified particle or decay branch found in each event.
 872 Similar to the cases in the latter seven kinds of component analysis, in the topology tags, “i” in
 873 “_i” is the default index of the specified particle or decay branch, and it ranges from 0 (included)
 874 to the number of specified particles or decay branches (excluded). If the alias of the particle or
 875 decay 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 type	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</i>
Decay branches	nSigDcyBr_i	number of signal decay branch; <i>es</i>
Cascade decay branches	nSigCascDcyBr_i	number of signal cascade decay branch; <i>es</i>
Inclusive decay branches	nSigIncDcyBr_i	number of signal inclusive decay branch; <i>es</i>
Inclusive cascade decay branches	nSigIncCascDcyBr_i	number of signal inclusive cascade decay branch; <i>es</i>
IRA decay branches	nSigIRADcyBr_i	number of signal IRA decay branch; <i>es</i>

876 5. Common settings

877 From Sections 3 and 4, the optional parameters of the functionality items give us more choices
 878 and thus help us do our jobs quicker and better. In addition to these parameters, many optional
 879 items are designed and implemented to control the execution of the program in order to meet
 880 practical needs. Unlike the optional parameters, which only affect the individual functionalities
 881 to which they belong, the optional items have an impact on all of the functionalities, or at
 882 least most of the functionalities. The current version of the program contains 25 commonly used
 883 items, which can be divided into the following three groups: items on the input of the program,
 884 items on the functionalities of the program, and items on the output of the program. This section
 885 introduces these items in the three groups, with each group in one subsection.

886 5.1. Settings on the input of the program

887 5.1.1. Input entries

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

```
896
897    % Maximum number of entries to be processed
898    {
899       2000
```

```

900    }
901
902    On some occasions, especially in the course of optimizing selection criteria, we need to run
903    the program only over entries satisfying certain requirements. For this purpose, an item is devel-
904    oped to select entries. The following example shows the item with X set in the range (-1, 1).
905
906    % Cut to select entries
907    {
908        (X >-1) && (X <1)
909    }

```

910 Notably, in the old versions prior to 02-07-03, only a single-line selection requirement is support-
911 ed in the item, like the cases in the methods Draw() [16] and GetEntries() [17] of the class TTree.
912 Though such a requirement is able to express any condition with the help of the parentheses “()”
913 as well as the logical symbols “&&”, “||”, and “!”, it looks clumsy when it is used to express
914 a complicated condition. Starting from the version 02-07-03, the cuts supported in the item are
915 also allowed to be divided into two or more lines in order to make them clearer.

916 Occasionally, array variables are involved in the requirement. Under the circumstances, user-
917 s have to tell the program how to determine the total logical value with the individual logical
918 values. At present, two criteria are provided: (1) the total result is true as long as the result for
919 one instance is true; (2) the total result is false as long as the result for one instance is false. By
920 default, the second criterion is used in the program. One can alter it to the first one with the
921 following item.

```

923
924    % Method to apply cut to array variables (Two options: T and F. Default: F)
925    {
926        T
927    }

```

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

930 5.1.2. Input decay branches

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

931 Normally, the program deals with all of the decay branches in every decay tree. However,
932 examining all the branches is not always required in practice. Sometimes, we only concern the
933 first n hierarchies of the branches. Similar to that in cascade decay branches of particles (as we
934 introduce in Section 3.6), the hierarchy here reflects the rank of a decay branch in a decay tree.
935 For example, in the decay tree $\Upsilon(4S) \rightarrow B^0 \bar{B}^0$, $B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F$, $\bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}$, $D^{*-} \rightarrow \pi^- \bar{D}^0$,
936 $D^{*+} \rightarrow \pi^+ D^0$, $\bar{D}^0 \rightarrow \pi^0 \pi^- K^+$, $D^0 \rightarrow \pi^0 \pi^+ K^-$, the hierarchies of the seven individual branches
937 are 1, 2, 2, 3, 3, 4, and 4, respectively. The program provides an item to set the maximum hier-
938 archy. Below is an example showing the item with the maximum hierarchy set at one.

```

940      % Maximum hierarchy of heading decay branches to be processed in each event
941      {
942          1
943      }
944

```

945 With the setting, the decay branches with hierarchy larger than one will be ignored by the
946 program. For the component analysis over the decay trees of the $\Upsilon(4S)$ sample, only the first
947 hierarchy of $\Upsilon(4S)$ decay branches are analyzed, and the result is shown in Table 21. From the
948 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
949 B^0 - \bar{B}^0 mixing. Similarly, in the case of the maximum hierarchy set at two, we could get the result
950 of the component analysis over the first two hierarchies of $\Upsilon(4S)$ decay branches, as displayed
951 in Table 22.

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

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

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

```

955
956      % Ignore the decay of the following particles
957      {
958          B0
959          anti-B0
960      }
961

```

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

```

966
967      % Ignore the decay of the daughters of the following particles
968      {
969          B0
970          anti-B0
971      }
972

```

973 The two settings above have the same effects as those in the previous paragraph which set the

974 maximum hierarchy at one and two, and hence the corresponding results are identical to those
975 shown in Tables 21 and 22.

976 As mentioned in Section 2.6, the decay $\pi^0 \rightarrow \gamma\gamma$ is ignored by default. On the occasions
977 when we need to identify the signals involving the decay, we can make the program retain the
978 decay with the item below set to “Y”.

```
979 % Retain the decay of pi0 to gamma gamma (Two options: Y and N. Default: N)
980 {
981     Y
982 }
983 }
```

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

```
986 % Ignore the following final decay branches
987 {
988     eta --> gamma gamma
989     K_S0 --> pi+ pi-
990 }
991 }
```

993 5.1.3. Initial and final state radiation photons

994 Initial state radiation (ISR) and final state radiation (FSR) are inevitable physical effects in
995 e^+e^- colliding experiments. Therefore, ISR and FSR photons are often involved in inclusive
996 MC samples. The program processes them together with other particles in the default case. To
997 distinguish them from other photons, the program tries to label them in the output plain text, tex
998 source, and pdf files. Sometimes, these photons are marked out beforehand with special PDG
999 codes according to particle status information from generators. One can inform the program of
1000 these PDG codes by the following two items.

```
1001 % PDG code of ISR photons (Default: 222222222)
1002 {
1003     222222222
1004 }
1005 }

1006 %

1007 % PDG code of FSR photons (Default: -22)
1008 {
1009     -22
1010 }
1011 }
```

1012 In this case, the program is able to label the ISR and FSR photons as γ^i (gammai) and γ^f (gam-
1013 maf) in the output pdf (plain text) files, respectively.

1014 On other occasions, ISR and FSR photons are not marked out in advance due to some reasons.
1015 In such cases, the program has to identify them by itself according to the following rules: photons
1016 who have no mothers recorded in the arrays of the PDG codes and mother indices are considered
1017 as generalized ISR photons, while other photons who have at least one e^\pm , μ^\pm , π^\pm , K^\pm , p , or \bar{p}
1018 sister are taken as generalized FSR photons. Here, the modifier “generalized” is used because
1019 the rules can not determine the types of the photons in absolute accuracy. For example, photons
1020 from radiative decays might be mistaken as FSR photons. Despite this, generalized ISR and FSR
1021 photons are good concepts, particularly in cases where the sources of the photons are not required
1022 to be distinguished clearly. The program will label the generalized ISR and FSR photons as γ^i
1023 (gammaI) and γ^F (gammaF) in the output pdf (plain text) files, respectively.

1024 Notably, we are not concerned about these ISR and FSR photons in many cases, particularly
1025 when we want to identify our signals from some samples. If they have already been marked out

1027 beforehand, one can make the program ignore them accurately by setting the following two items
1028 to “Ys”.

```
1029 % Ignore ISR photons (Three options: Ys, Yg and N. Default: N)
1030 {
1031     Ys
1032 }
1033
1034
1035 % Ignore FSR photons (Three options: Ys, Yg and N. Default: N)
1036 {
1037     Ys
1038 }
1039
1040
```

1041 In cases that these photons are not marked in advance, the option “Yg” can be used to ignore the
1042 generalized ISR and FSR photons. In “Ys” and “Yg”, “s” and “g” are the initials of the words
1043 “strict” and “generalized”, respectively.

1044 *5.2. Settings on the functionalities of the program*

1045 *5.2.1. Candidate based analysis*

1046 According to the number of signal candidates in an event that are selected and retained to
1047 extract physics results, data analysis in high energy experiments can be divided into the following
1048 two categories: event based analysis and candidate based analysis. While at most one candidate
1049 in an event is kept in event based analysis, one or more candidates in an event can be retained in
1050 candidate based analysis. Generally, the quantities related to a candidate are stored in an entry of
1051 the TTree objects in the root files. Thus, one or more entries relate to an event in candidate based
1052 analysis, while only one entry corresponds to an event in event based analysis. Normally, the
1053 indices of candidates within an event are stored in the corresponding entries in candidate based
1054 analysis.

1055 By default, the program analyzes the input entries one by one. In this case, the events with
1056 multiple candidates will be processed repeatedly. Particularly, the number of physics processes at
1057 the sample level will be overcounted. One can make the program avoid the problem by inputting
1058 “Y” to the following item.

```
1059 % Avoid over counting for candidate based analysis (Two options: Y and N. Default: N)
1060 {
1061     Y
1062 }
1063
```

1064 Also, the indices of candidates within an event are required. We can tell the program the related
1065 TBranch name with the following item.

```
1066 % TBranch name of the indices of candidates in an event (Default: __candidate__)
1067 {
1068     iCandidate
1069 }
1070
1071 }
```

1072 With the settings, the program will process the first entry of each event in a normal way, including
1073 obtaining and storing the topology tags; it will not analyze the other entries of the same event,
1074 but only store the same topology tags to them.

1076 5.2.2. Charge conjugation

1077 Charge conjugation is an important concept in high energy physics. By default, charge con-
 1078 jugate objects (particles and decays) are processed separately in the program. However, we need
 1079 to handle them together in many physics studies because of the sameness between them. One
 1080 can have the program process them together with the item below set to “Y”.

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 index_{cc} are short for “charge conjugate” and “charge conjugate index”, respectively. For self-charge-conjugate objects (particles or decays), the charge conjugate indices have the value 0; for non-self-charge-conjugate objects, they have the value 1 or -1: while 1 tags the objects presented in the topology maps, -1 indicates their charge conjugate objects.

Component type	Topology tag	Interpretation
Decay trees	iCcDcyTr	index_{cc} of decay tree
Decay initial-final states	iCcDcyIFSts	index_{cc} of decay initial-final states
	iCpDcyBr.i	index_{cc} of particle _i
Decay branches of particles	(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)
	iCpProdBr.i	index_{cc} of particle _i
Production branches of particles	(1) iCcProdBrP.i.j	index_{cc} of production branch of the j th particle _i
	(2) nCcPProdBr.i	number of cc particle _i s (production branches)
	(2) iProdBrCcP.i.j	index of production branch of the j th cc particle _i
	(2) nAllPProdBr.i	number of all particle _i s (production branches)
	iCpMoth.i	index_{cc} of particle _i
Mothers of particles	(1) iCcMothP.i.j	index_{cc} of mother of the j th particle _i
	(2) nCcPMoth.i	number of cc particle _i s (mothers)
	(2) PDGMothCcP.i.j	PDG code of mother of the j th cc particle _i
	(2) nAllPMoth.i	number of all particle _i s (mothers)
	iCpCascDcyBr.i	index_{cc} of particle _i
Cascade decay branches of particles	(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)
	iCpDcyFSt.i	index_{cc} of particle _i
Decay final states of particles	(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)
	iCIncDcyBr.i	index_{cc} of inclusive decay branch _i
Inclusive decay branches	(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
	iCIRADcyBr.i	index_{cc} of IRA decay branch _i
IRA decay branches	(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

```

1081
1082
1083 % Process charge conjugate objects together (Two options: Y and N. Default: N)
1084 {
1085     Y
1086 }

```

1087 Performing topology analysis with this setting inserts new topology tags in the output root files
1088 and adds new counters to topology maps in the output plain text, tex source, and pdf files. Tables
1089 [23](#) and [24](#) list and interpret all of the topology tags related to charge conjugation involved in the
1090 component analysis and signal identification functionalities, respectively.

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 index_{cc} are short for “charge conjugate” and “charge conjugate index”, respectively. For self-charge-conjugate objects (particles or decays), the charge conjugate indices have the value 0; for non-self-charge-conjugate objects, they have the value 1 or -1 : while 1 tags the objects presented in the topology maps, -1 indicates their charge conjugate objects.

Signal type	Topology tag	Interpretation
Decay trees	$i\text{CcSigDcyTr}$	index_{cc} of signal decay tree
Decay initial-final states	$i\text{CcSigDcyIFSts}$	index_{cc} of signal decay initial-final states
Particles	$i\text{CcSigP}_i$ (*) $n\text{CcSigP}_i$ (*) $n\text{AllSigP}_i$	index_{cc} of signal particle _i s number of cc signal particle _i s number of all signal particle _i s
Decay branches	$i\text{CcSigDcyBr}_i$ (*) $n\text{CcSigDcyBr}_i$ (*) $n\text{AllSigDcyBr}_i$	index_{cc} of signal decay branch _i number of cc signal decay branch _i es number of all signal decay branch _i es
Cascade decay branches	$i\text{CcSigCascDcyBr}_i$ (*) $n\text{CcSigCascDcyBr}_i$ (*) $n\text{AllSigCascDcyBr}_i$	index_{cc} of signal cascade decay branch _i number of cc signal cascade decay branch _i es number of all signal cascade decay branch _i es
Inclusive decay branches	$i\text{CcSigIncDcyBr}_i$ (*) $n\text{CcSigIncDcyBr}_i$ (*) $n\text{AllSigIncDcyBr}_i$	index_{cc} of signal inclusive decay branch _i number of cc signal inclusive decay branch _i es number of all signal inclusive decay branch _i es
Inclusive cascade decay branches	$i\text{CcSigIncCascDcyBr}_i$ (*) $n\text{CcSigIncCascDcyBr}_i$ (*) $n\text{AllSigIncCascDcyBr}_i$	index_{cc} of signal inclusive cascade decay branch _i number of cc signal inclusive cascade decay branch _i es number of all signal inclusive cascade decay branch _i es
IRA decay branches	$i\text{CcSigIRADcyBr}_i$ (*) $n\text{CcSigIRADcyBr}_i$ (*) $n\text{AllSigIRADcyBr}_i$	index_{cc} of signal IRA decay branch _i number of cc signal IRA decay branch _i es number of all signal IRA decay branch _i es

1092 As an example, we perform the component analysis over decay trees with the charge con-
1093 jugate item. Table [25](#) shows the obtained topology map. Besides the columns in Table [2](#), two
1094 additional columns with the headers “nCcEtr” and “nAllEtr” are inserted in the table. Here, “nCc-
1095 cEtr” represents the number of entries involving the charge conjugate decay trees, and “nAllEtr”
1096 is the sum of “nEtr” and “nCcEtr”. In addition to “iDcyTr”, “iCcDcyTr” is also inserted in the
1097 output root files as a topology tag. It is short for charge conjugate index of decay tree. For self-
1098 charge-conjugate decay trees, it has the value 0; for non-self-charge-conjugate decay trees, it has
1099 the value 1 or -1 : while 1 tags the decay trees listed in the topology maps, -1 indicates their
1100 charge conjugate decay trees. Whereas the equal values of “iDcyTr” for each decay tree and its
1101 charge conjugate decay tree indicate their sameness, the opposite values of “iCcDcyTr” for them
1102 reflect their difference.

Table 25: Decay trees and their respective initial-final states (with the charge conjugation setting).

rowNo	decay tree (decay initial-final states)	iDcyTr	nEtr	nCcEtr	nAllEtr	nCEtr
1	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $D^{*+} \rightarrow \pi^+ D^0, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) \dashrightarrow e^+ \nu_e \mu^- \bar{\nu}_\mu \pi^0 \pi^0 \pi^+ \pi^- \pi^- K^+ K^- \gamma^F)$	20870	3	0	3	3
2	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, \bar{B}^0 \rightarrow \rho^- D^{*+}, D^{*-} \rightarrow \pi^- \bar{D}^0,$ $\rho^- \rightarrow \pi^0 \pi^-, D^{*+} \rightarrow \pi^0 D^+, \bar{D}^0 \rightarrow \pi^0 \pi^- K^+, D^+ \rightarrow \pi^+ \pi^+ K^-$ $(\Upsilon(4S) \dashrightarrow \mu^+ \nu_\mu \pi^0 \pi^0 \pi^0 \pi^+ \pi^- \pi^- K^+ K^-)$	3648	2	0	2	5
3	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \mu^+ \nu_\mu D^{*-}, \bar{B}^0 \rightarrow e^- \bar{\nu}_e D^{*+}, D^- \rightarrow e^- \bar{\nu}_e \pi^- K^+,$ $D^{*+} \rightarrow \pi^+ D^0, D^0 \rightarrow \pi^0 \pi^- K^-$ $(\Upsilon(4S) \dashrightarrow e^- e^- \bar{\nu}_e \bar{\nu}_e \mu^+ \nu_\mu \pi^0 \pi^+ \pi^+ \pi^- K^+ K^-)$	3722	1	1	2	7
4	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow \pi^0 \pi^+ \pi^+ \rho^- D^-, \bar{B}^0 \rightarrow \mu^- \bar{\nu}_\mu D^{*+}, \rho^- \rightarrow \pi^0 \pi^-,$ $D^- \rightarrow \pi^- \pi^- K^+, D^{*+} \rightarrow \pi^+ D^0, D^0 \rightarrow K_L^0 \pi^+ \pi^-$ $(\Upsilon(4S) \dashrightarrow \mu^- \bar{\nu}_\mu \pi^0 \pi^0 K_L^0 \pi^+ \pi^+ \pi^+ \pi^- \pi^- \pi^- K^+)$	5295	2	0	2	9
5	$\Upsilon(4S) \rightarrow B^0 \bar{B}^0, B^0 \rightarrow e^+ \nu_e D^{*-} \gamma^F, \bar{B}^0 \rightarrow \pi^0 \pi^+ \pi^- \pi^- D^{*+},$ $D^{*-} \rightarrow \pi^0 D^-, D^{*+} \rightarrow \pi^+ D^0, D^- \rightarrow \pi^- \pi^- K^+, D^0 \rightarrow \pi^0 \pi^+ K^-$ $(\Upsilon(4S) \dashrightarrow e^+ \nu_e \pi^0 \pi^0 \pi^+ \pi^+ \pi^- \pi^- \pi^- \pi^- K^+ \gamma^F)$	10206	1	1	2	11
rest	$\Upsilon(4S) \rightarrow \text{others (99969 in total)}$ $(\Upsilon(4S) \dashrightarrow \text{corresponding to others})$	—	—	—	99989	100000

As another example, we carry out the component analysis over the decay branches of D^{*+} and J/ψ . 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 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” tags whether the i^{th} particle is self-charge-conjugate. For self-charge-conjugate particles, it has the value 0; for non-self-charge-conjugate particles, it has the value 1.

Table 26: Decay branches of D^{*+} (with the charge conjugation setting).

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

The topology tag “iCcDcyBrP.i.j” records the charge conjugation property of the decay branch of the j^{th} instance of the i^{th} particle. It is to “iDcyBrP.i.j” what “iCcDcyTr” is to “iDcyTr”. The topology tag “iDcyBrCcP.i.j” is designed for the charge conjugate particle of the i^{th} particle (for D^{*-} in this example). It has a similar meaning as “iDcyBrP.i.j”. Particularly, the values of “iDcyBrP.i.j” and “iDcyBrCcP.i.j” tagging charge conjugate decay branches are equal to each other. The topology tag “nCcPDcyBr.i” stands for the number of the charge conjugate

1122 i^{th} particles (or their decay branches) found in each event, and “nAllPDcyBr_i” is the sum of
1123 “nPcyBr_i” and “nCcPDcyBr_i”.

1124 *5.2.3. Settings only on signal identification*

1125 Normally, the signals specified in the signal identification functionality items are both tagged
1126 and counted by executing the program one time. In the case of a huge sample that will take a long
1127 time, it is a good idea to first tag the signals with multiple jobs each running on one machine, and
1128 then count the tagged signals together. One can make the program carry out the idea by setting
1129 the following item to “T” and “C” in the first and second steps, respectively. Here, “T” and “C”
1130 stand for tagging and counting, respectively.

```
1131       % Analysis tasks for signal identifications (Three options: TC, T and C. Default: TC)
1132       {
1133         T
1134       }
```

1135 By default, the signals set in the signal identification functionality items are listed in the output
1136 plain text, tex source, and pdf files in the sequence they are specified. In cases of plenty of
1137 signals, there is probably a need to sort them according to the number of cases found in the input
1138 samples. One can have the program do the sorting by inputting “Y” to the item below.

```
1139       % Sort the signals in the topology maps related to signal identifications (Two options: Y and N. Default: N)
1140       {
1141         Y
1142       }
```

1143 *5.3. Settings on the output of the program*

1144 By default, decay objects (trees, initial-final states, and branches) are left-aligned in the output
1145 pdf files. If one likes it, he/she can request the program to center them by setting the following
1146 item to “Y”.

```
1147       % Center decay objects in output pdf files (Two options: Y and N. Default: N)
1148       {
1149         Y
1150       }
```

1151 As mentioned in Section 2.6, after the execution of the program, one or more root files will
1152 be output to save topology tags. By default, the program switches to a new output file whenever
1153 the size of the TTree object in memory exceeds 3 GB. In addition to this, the program provides
1154 an item to control the switch of output files by setting the maximum number of entries to be
1155 saved in a single output file. The following example shows the item with the maximum number
1156 set to 1 million.

```
1157       % Maximum number of entries to be saved in a single output root file
1158       {
1159         1000000
1160       }
```

1161 Besides, one can have the program generate one output file by one input file with the following
1162 item set to “Y”.

```
1163       % One output root file by one input root file (Two options: Y and N. Default: N)
1164       {
1165         Y
1166       }
```

1167 Notably, with the setting, the output root files will not be denominated according to the default or
1168 36

1177 specially specified common name of the output files. Instead, they will be named after the input
 1178 root files and with “_ta_n.root” (n=1, 2, 3 ...) as suffixex. Here, “ta” is short for topology analysis
 1179 and “n” is the corresponding file number. For example, with this setting, the names of the output
 1180 root files in the first example of the user guide will be “jpsi1_ta_1.root” and “jpsi2_ta_2.root”.

1181 In default cases, flat TBranch objects are used to store topology tags in the output root files.
 1182 This is necessary for the Belle II experiment, as array TBranch objects are not recommended to
 1183 use in physics analyses in order to use other tools such as NumPy [11] and pandas [12]. Howev-
 1184 er, since array TBranch objects are elegant and efficient in organizing and storing homogeneous
 1185 data, sometimes it is better to use them than flat TBranch objects in other experiments, such as
 1186 the BESIII experiment. One can make the program use array TBranch objects to store topology
 1187 tags by inputting “Y” to the item below.

```
1188 % Use array tbranches to store topology tags in output root files when possible (Two options: Y and N. Default: N)
1189 {
1190     Y
1191 }
1192 }
```

1193 By default, to facilitate the validation of topology analysis results, the input TBranch objects
 1194 are copied to the output root files along with other TBranch objects for physics analyses. How-
 1195 ever, they often occupy too much disk space and are useless for following physics analyses. In
 1196 the case of being flat, a massive amount of these TBranch objects also looks awkward. Thus,
 1197 after the validation with a small sample, it would be better to remove these TBranch objects from
 1198 the output root files. One can request the program to perform this removal operation before it
 1199 terminates by setting the following item to “Y”.

```
1200 % Remove the input tbranches from output root files (Two options: Y and N. Default: N)
1201 {
1202     Y
1203 }
1204 }
```

1205 On some occasions, besides the TTree object containing the raw topology truth information,
 1206 we may also want to clone some other TTree objects from the input root files to the output root
 1207 files. One can set the names of these TTree objects in the following item, with each in one line.

```
1208 % Other TTree names
1209 {
1210     abc
1211     xyz
1212 }
1213 }
```

1214 In the example, the two TTree names “abc” and “xyz” are specified. Notably, with such a setting,
 1215 the other TTree objects will only be cloned to the first output root file in cases that multiple output
 1216 root files are produced but they have no explicit one-by-one relationship to the input root files.

1217 Sometimes, we may only desire the topology maps. Under these circumstances, it would be
 1218 better to suppress the output root files, particularly in cases that they are large in file sizes. With
 1219 the item below, one can make the program do this automatically by first generating empty output
 1220 root files and then removing them after the corresponding entries are processed.

```
1221 % Suppress output root files (Two options: Y and N. Default: N)
1222 {
1223     Y
1224 }
1225 }
```

1226 In all of the previous examples, the program is applied to the inclusive MC samples in e^+e^-
 1227 colliding experiments. Besides, the program can also be used in other types of high energy ex-
 1228 periments, for example, the PANDA experiment [18], a $p\bar{p}$ annihilation experiment under con-

1233 struction at Darmstadt, Germany. On these occasions, we have to specify the right initial state
 1234 particles with the following item to obtain the proper topology maps.

```
1235
1236 % Initial state particles (Default: e- e+)
1237 {
1238   anti-p- p+
1239 }
```

1240 With the setting, the default initial state e^+e^- is replaced by $p\bar{p}$, as shown in Table 27, which
 1241 displays the results of a component analysis over decay trees of a small $p\bar{p}$ annihilation sample.

Table 27: Decay trees and their respective final states ($p\bar{p}$ annihilation).

rowNo	decay tree	decay final state	iDcyTr	nEtr	nCEtr
1	$p\bar{p} \rightarrow p\bar{p}$	$p\bar{p}$	1	232	232
2	$p\bar{p} \rightarrow \pi^+\pi^-p\bar{p}$	$\pi^+\pi^-p\bar{p}$	24	53	285
3	$p\bar{p} \rightarrow \pi^0p\bar{p}$	$\pi^0p\bar{p}$	5	35	320
4	$p\bar{p} \rightarrow \pi^0\pi^+\pi^-p\bar{p}$	$\pi^0\pi^+\pi^-p\bar{p}$	0	33	353
5	$p\bar{p} \rightarrow \pi^0\pi^0\pi^0\pi^+\pi^-\pi^-$	$\pi^0\pi^0\pi^0\pi^+\pi^+\pi^-$	39	31	384
rest	$p\bar{p} \rightarrow$ others (184 in total)	corresponding to others	—	616	1000

1243 6. Auxiliary facilities

1244 This section introduces some auxiliary facilities for the use of the program, including a card
 1245 file to preset frequently used items; some additional command line arguments to reset the names
 1246 of input root files, the common name of output files, and the maximum number of entries to be
 1247 processed; and two commands implemented in tex source files. Different from that presented in
 1248 the previous four sections, the content presented in this section is not the essential part of the
 1249 program. However, with these auxiliary facilities, we can make the program do our jobs better
 1250 and quicker on some occasions.

1251 6.1. The underlying card file

1252 A card file, namely “underlying_topoana.card” under the directory “share”, to preset fre-
 1253 quently used items is developed to assist the card file specified by the first argument of the
 1254 command “topoana.exe”. Here, we refer to the former and latter card files as underlying and
 1255 primary, respectively. In general, the primary card file is sufficient to set items for the execution
 1256 of the program. However, considering some items are frequently used with constant inputs by a
 1257 user or a group of users, it is better to move the items from the primary card file to the underlying
 1258 card file, in order to make the primary card file more concise and make us more focused on the
 1259 items specially set for the dedicated topology analysis.

1260 One can decide whether to set an item in the underlying card file according to his/her own
 1261 needs. Here, we introduce some frequently used items that are suitable to be put in the underlying
 1262 card file as follows. As mentioned in Section 2.4, the items related to the storage type and
 1263 TBranches names of the input data are usually fixed for a user or a group of users. Thus, it is
 1264 quite appropriate to move them to the underlying card file. We have to process charge conjugation
 1265 particles and decays together in many physics studies. In such studies, it is also a good practice
 1266 to put the item on charge conjugation in the underlying card file.

1267 The program first reads the items in the underlying card file and then reads those in the
1268 primary card file. The items set in the underlying card file can be reset in the primary card file.
1269 In such a case, the inputs in the underlying card file will be replaced by their counterparts in the
1270 primary card file.

1271 6.2. Additional command line arguments

1272 Normally, only the “cardFileName” is required to be passed as an argument of the command
1273 “topoana.exe”, and all of the necessary information can be configured via the setting items filled
1274 in the card file. On some occasions, we need to run the program over multiple samples separately,
1275 with identical settings except for the names of input root files and the common name of output
1276 files. A regular approach to do such a job requires multiple card files, each corresponding to
1277 one sample. This approach appears a bit tedious in cases of many samples. To avoid this, two
1278 additional command line arguments are designed and implemented to reset the names of input
1279 root files and the common name of output files. Similarly, an argument is also developed for the
1280 maximum number of entries to be processed.

1281 These optional arguments should be typed with prompts, which are listed and explained as
1282 follows.

- 1283 • –i: The names of input root files should be provided after the prompt. One or more names
1284 are allowed here. They will replace those set in the card file.
- 1285 • –t: The TTree name should be provided after the prompt. It will replace the one set in the
1286 card file.
- 1287 • –o: The common name of output files should be provided after the prompt. It will replace
1288 the one set in the card file or the default one, that is, the name of the card file.
- 1289 • –n: The maximum number of entries to be processed should be provided after the prompt.
1290 It will replace that set in the card file.

1291 6.3. Commands implemented in tex source files

1292 The output pdf files can be checked after the execution of the program. If their styles are not
1293 to our taste, we can edit the corresponding tex source files to get the desired styles, according
1294 to the regular LaTeX rules. Besides the rules, two commands are implemented in the tex source
1295 files to help us edit the files quickly and easily for two common desired styles.

1296 By default, topology tags are listed along with topology maps in the output plain text, tex
1297 source, and pdf files. However, only the topology maps are needed on some occasions, especially
1298 in presentations. In such cases, one can suppress the topology tags in the output tex source
1299 and pdf files by simply changing the definition of the cmtTopoTags command from the nominal
1300 one

1301
1302 \newcommand{\topoTags}[1]{#1}

1303
1304 to the alternative one

1305
1306 \newcommand{\topoTags}{}{}

1307
1308 in the preamble of the text source files. Here, “#1” is the formal parameter of the string for

1309 the topology tags. With the nominal definition, “\topoTags{\#1}” returns the string exactly, while
1310 with the alternative definition it only returns an empty string. That is why the definition below is
1311 able to suppress the topology tags.

1312 After the revision of the tex source files, one can re-compile them with the pdflatex command.
1313 Usually, the pdflatex command has to be executed two or three times for a fully compiled pdf
1314 file, and many undesired files in other formats are generated during the compilation. To execute
1315 the pdflatex command and remove the undesired files at one stroke, we develop a bash script,
1316 namely “getPdfFromTex.sh” under the directory “utilities”. The script should be executed with
1317 the following command line: getPDFFromTexFl.sh texFileName. Compiling the tex source
1318 files with the script is recommended.

1319 7. Summary

1320 We develop a program, namely TopoAna, with C++, ROOT, and LaTeX for the event type
1321 analysis of inclusive MC samples in high energy physics experiments. This user guide provides
1322 a detailed description of the program, including a basic introduction to it, two categories of its
1323 functionalities — component analysis and signal identification, and some common settings and
1324 auxiliary facilities for its execution. The program has rich functionalities and aims to solve all
1325 kinds of event type analysis tasks. Meanwhile, it is easy to use and has a high processing rate.
1326 These features make the program a powerful tool to analyze the backgrounds involved in our
1327 research works and to identify the physics processes of interests from the inclusive MC samples.

1328 Since it does not rely on any specific software frameworks, the program applies to many high
1329 energy physics experiments. Up to now, it has been put into use in three experiments at e^+e^-
1330 colliders: the BESIII, Belle, and Belle II experiments. Besides these experiments, it can also be
1331 used in other types of experiments, such as the PANDA experiment, a $p\bar{p}$ annihilation exper-
1332 iment. Also, the program is applicable to the future e^+e^- colliding experiments under research
1333 and development, such as the circular electron-positron collider (CEPC) [19, 20] experiment in
1334 China, the super Charm- τ factory (SCTF) experiment [21] in Russia, and the super τ -Charm fac-
1335 tory (STCF) experiment [22] in China. These experiments offer wide space for the application
1336 of the program.

1337 On the other hand, we note that the application of the program to some other experiments is
1338 limited. For example, thousands of particles can be produced from dozens of pp collisions in
1339 an event of the ATLAS [23] and CMS [24] experiments at the LHC [25]; in such cases, there
1340 is little point in performing the event type analysis of corresponding MC samples. Nonetheless,
1341 the application scope of the program is still broad. In particular, it applies to the e^+e^- colliding
1342 experiments where at most tens of particles are produced from the annihilation of a pair of e^+e^-
1343 in an event. With more user needs coming out in the future, we will further extend and perfect it
1344 to make it more powerful and well-rounded.

1345 Acknowledgements

1346 This work was supported by the National Natural Science Foundation of China [grant num-
1347 bers 11575017, 11661141008, 11761141009, 11875262, 11975076] and the CAS Center for
1348 Excellence in Particle Physics (CCEPP). In addition, we would like to thank all of the people
1349 who have helped us in the development of the program. We first thank Prof. Changzheng Yuan,
1350 Bo Xin, and Haixuan Chen for their help at the early stage of developing the program. We are

1351 particularly grateful to Prof. Xingtao Huang for his comments on the principles and styles of the
 1352 program, to Remco de Boer for his suggestions on the tex output and the use of GitHub, and to
 1353 Xi Chen for his discussions on the core algorithms. We are especially indebted to Prof. Xiqing
 1354 Hao, Longke Li, Xiaoping Qin, Ilya Komarov, Yubo Li, Guanda Gong, Suxian Li, Junhao Yin,
 1355 Prof. Xiaolong Wang, Yeqi Chen, Hannah Wakeling, Hongrong Qi, Ning Cao, and Sanjeeda B-
 1356 harati Das for their advice in extending and perfecting the program. Also, we thank Xi'an Xiong,
 1357 Runqiu Ma, Wencheng Yan, Sen Jia, Lu Cao, Dong Liu, Hongpeng Wang, Jiawei Zhang, Jiajun
 1358 Liu, Maoqiang Jing, Yi Zhang, Wei Shan, and Yadi Wang for their efforts in helping us test the
 1359 program.

1360 References

- 1361 [1] ROOT User's Guide, Available online: <https://root.cern/root/html/doc/guides/users-guide/ROOTUsersGuide.html>.
- 1362 [2] Documentation of the TFile class, Available online: <https://root.cern/root/html534/TFile.html>.
- 1363 [3] K.T. Chao, Y.F. Wang, et al., *Int. J. Mod. Phys. A* 24 (2009) S1-794.
- 1364 [4] M. Ablikim, et al. (BESIII Collaboration), *Chin. Phys. C* 44 (2020) 040001.
- 1365 [5] E. Kou, et al., *Prog. Theor. Exp. Phys.* 2019 (2019) 123C01.
- 1366 [6] J. Brodzicka, T. Browder, P. Chang, et al., *Prog. Theor. Exp. Phys.* 2012 (2012) 04D001.
- 1367 [7] Text of MIT license, Available online: <https://mit-license.org/>.
- 1368 [8] Documentation of the TTree class, Available online: <https://root.cern/root/html534/TTree.html>.
- 1369 [9] M. Tanabashi, et al. (Particle Data Group), *Phys. Rev. D* 98 (2018) 030001.
- 1370 [10] Documentation of the TBranch class, Available online: <https://root.cern/root/html534/TBranch.html>.
- 1371 [11] Documentation of NumPy, Available online: <https://numpy.org/devdocs/>.
- 1372 [12] Documentation of pandas, Available online: <https://pandas.pydata.org/pandas-docs/stable/>.
- 1373 [13] Documentation of the TChain class, Available online: <https://root.cern/root/html534/TChain.html>.
- 1374 [14] Reference of unordered maps, Available online: http://www.cplusplus.com/reference/unordered_map/unordered_map/.
- 1375 [15] D. J. Lange, *Nucl. Instrum. Meth. A* 462 (2001) 152.
- 1376 [16] Documentation of the Draw() method of the TTree class, Available online: <https://root.cern/root/html534/TTree.html#TTree:Draw@2>.
- 1377 [17] Documentation of the GetEntries() method of the TTree class, Available online: <https://root.cern/root/html534/TTree.html#TTree:GetEntries@1>.
- 1378 [18] W. Erni, et al. (PANDA Collaboration), Physics Performance Report for PANDA: Strong Interaction Studies with
1379 Antiprotons, arXiv:0903.3905.
- 1380 [19] CEPC CDR Volume 1 (Accelerator), Available online: http://cepc.ihep.ac.cn/CEPC_CDR_Vol1_Accelerator.pdf.
- 1381 [20] CEPC CDR Volume 2 (Physics & Detector), Available online: http://cepc.ihep.ac.cn/CEPC_CDR_Vol2_Physics-Detector.pdf.
- 1382 [21] A.E. Bondar, et al. (Charm-Tau Factory Collaboration), *Phys. Atom. Nucl.* 76 (2013) 1072.
- 1383 [22] Q. Luo, D. Xu, "Progress on Preliminary Conceptual Study of HIEPA, a Super Tau-Charm Factory in China", in
1384 Proc. 9th International Particle Accelerator Conf. (IPAC2018), Vancouver, BC, Canada, 422.
- 1385 [23] G. Aad, et al. (ATLAS Collaboration), *JINST* 3 (2008) S08003.
- 1386 [24] S. Chatrchyan, et al. (CMS Collaboration), *JINST* 3 (2008) S08004.
- 1387 [25] L. Evans (ed.), P. Bryant (ed.), *JINST* 3 (2008) S08001.