Описание программной компоненты TIQMax

1. Аннотация

Настоящий документ содержит описание программной компоненты TIQMax и предназначен для пользователей и программистов. Описывается назначение программы, перечень реализуемых функций, порядок ее применения и имеющиеся ограничения. Изложена логика работы. Описан порядок запуска программ и параметры, передаваемые посредством интерфейса командной строки. Представлены форматы входных и выходных данных.

2. Содержание

3. Основной раздел

3.1. Вводная часть

3.1.1. полное наименование программы

«Программа анализа эволюции набора белковых семейств и построения на их основе потенциального дерева эволюции видов».

3.1.2. обозначение

TIQMax

3.1.3. применение программы

Для использования программы требуется операционная система, обеспечивающая компиляцию программы на языке C++ (детали см. в разделе 3.3.3 ниже), запуск полученного исполняемого файла с передачей ему параметров в командной строке и последовательное чтение/запись файлов программой. На практике подойдёт любая операционная система семейства UNIX/Linux или Windows.

Программа функционирует в пакетном режиме как консольное приложение и в общем случае не требует диалога с оператором или наблюдения за ходом счета, хотя с учетом его возможной продолжительности предусмотрен вывод на консоль оператора сообщений, информирующих о ходе работы.

Для окончательного формирования результатов одной из функций программы (построения дерева видов) желательно использование программы построения консенсуса деревьев с одинаковым набором таксонов; для этой цели подходит, например, программа *consense* из пакета *Phylip*.

3.2. Функциональное назначение

3.2.1. назначение программы

Программа предназначена для массового анализа белковых семейств, кластеров ортологичных групп белков и их деревьев и обеспечивает построение потенциального дерева эволюции видов по набору деревьев эволюции белковых семейств. При этом реконструиру-

ются эволюционные события (дупликации и потери генов, горизонтальные переносы), вычисляются их характеристики и оцениваются качества белковых семейств.

3.2.2. общее описание функционирования программы

Программа обладает достаточно широкой и разнообразной функциональностью. В наборе её возможностей можно выделить три основных функции, на которые далее будем ссылаться по сокращённым наименованиям:

- ***Функция 1***: Построение на основе набора деревьев эволюции белковых семейств потенциального дерева эволюции видов.
- ***Функция 2:*** Реконструкция эволюционных событий (дупликаций и потерь генов) в белковых семействах.
- ***Функция 3:*** Вычисление характеристик эволюционных событий и качества белковых семейств.

Программа реализует квазиоптимальный оптимизационно-комбинаторный алгоритм решения следующей задачи. Пусть заданы n исходных деревьев белковых семейств (деревья генов), каждый лист которых помечен названием гена и соответствующего ему вида. Требуется построить такое дерево видов (его листья помечены видами), которое бы наиболее полно отразило усреднённые по данным деревьям генов топологические отношения между видами. Математически задача ставится как задача оптимизации заданного функционала, описывающего степень соответствия дерева видов набору деревьев генов.

Основой алгоритма является каноническое вложение α дерева генов G в дерево видов S (которое предполагается бинарным). Обозначим $c(G,S)$ стоимость («цену») этого вложения, показывающую, насколько вложение α отличается от тождественного отображения одного дерева в другое, т.е. насколько дерево G отличается от совпадения с деревом S. Дупликация – это пара (g,s), где g – вершина в дереве генов G и s – вершина в дереве видов S, для которой выполняется $\alpha(g) = \alpha(g') = s$ для одного или обоих непосредственных потомков g' гена g (левый из этих потомков обозначим cg, а правый – Cg). Вершина s (в дереве видов S) называется g-промежуточной, если она расположена в точности между вершинами $\alpha(g)$ и $\alpha(pg)$, где pg обозначает непосредственного предка («отца») вершины g. Обозначим $M(G,S)$ множество всех g-промежуточных вершин для всех g из G. Элемент множества $M(G,S)$ называется еще пропуском. Пропуску соответствует в дереве генов ребро (g,pg), его длину обозначим $l_{(g,pg)}$. Число всех потерь (в содержательном смысле) генов вычисляется как сумма числа всех односторонних дупликаций и числа всех пропусков. Дупликация (g,s) называется односторонней, если выполняется ровно одно из соотношений $\alpha(g) = \alpha(cg)$ или $\alpha(g) = \alpha(Cg)$. Односторонней дупликации (g,s) соответствует одно из ребер (g,cg) или (g,Cg) в дереве генов. Мно-

жество всех односторонних дупликаций обозначается $O(G,S)$. Дупликация (g,s) называется двусторонней, если выполняется оба соотношения $\alpha(g) = \alpha(cg)$ и $\alpha(g) = \alpha(Cg)$.

Дупликация (g,s) считается отнесенной к вершине s. Число дупликаций *для* (фиксированного) *белкового семейства* (КОГа) в вершине s дерева видов равно числу дупликаций при фиксированном s. *Суммарное число дупликаций по геному* в вершине s дерева видов – это сумма всех односторонних дупликаций в этой вершине по всем белковым семействам из некоторого фиксированного их набора. (Выражение «по геному» означает, что этот набор подбирается по возможности более представительным). Для отдельного белкового семейства также подсчитывается *суммарное число дупликаций по всем потомкам* в вершине s дерева видов как сумма чисел односторонних дупликаций по всем вершинам филогенетической группы, состоящей из потомков предка s. Несколько более сложным образом подсчитывается число потерь, отнесенных к вершине s. Напомним, что потеря гена g в вершине s это пара (g,s), у которой в g имела место дупликация и s – потомок $\alpha(g)$, а клада s не содержит ни одного гена из клады g', где g' – один из двух непосредственных потомков g, но в тоже время клада ps содержит гены из обеих клад g'. (Это определение иногда заменяется на более сложное с дополнительными условиями на пару (g,s)). Число потерь в вершине s для фиксированного белкового семейства определяется числом всех таких пар (g,s) при фиксированном s. Алгоритм обеспечивает подсчет дупликаций и потерь генов.

Для каждого дерева генов строится его вложение α в дерево видов. Стоимость (цена) этого вложения определяется как

$$c(G,S) = \left|O(G,S)\right| + \gamma \cdot \left|M(G,S)\right|,$$

где $|\{\cdot\}|$ обозначает мощность множества $\{\cdot\}$, т.е. число элементов в нем. В варианте с длинами ветвей она определяется через две суммы

$$c(G,S) = \sum_{g \in O(G,S)} l_{(g,cg)} + \gamma \cdot \sum_{g \in M(G,S)} l_{(g,pg)}.$$

Величина γ является параметром алгоритма. Минимизируя величину $c(G,S)$ при γ=1, минимизируем общее число всех потерь генов. При γ<1 значение этой цены отражает предпочтение событий дупликации по сравнению с пропусками. Дерево видов S ищется путем минимизации величины

$$c = c(S) = c(G_1,S) + c(G_2,S) + \ldots + c(G_n,S),$$

где все деревья G_i генов даны в исходных данных, а ищется неизвестное дерево видов S. Эту величину будем называть также стоимостью (ценой) вложения семейства $\{G_i\}$ деревьев генов. С математической точки зрения поиск минимума величины $c(S)$ – весьма нетривиальная задача, в некотором смысле она содержит проблематику всей теории сложности вычислений.

Существенно, что более надежные ребра деревьев генов G_i оказывают большее влияние на результат минимизации функционала $c(S)$. Длины (надежность или время), приписанные ребрам деревьев G_i, можно индуцировать на результирующее дерево видов S (вопрос о биологическом смысле этих чисел не решается в рамках проекта).

Алгоритм ищет оптимальное дерево S^* как специальный локальный минимум. Поскольку получаемый алгоритмом локальный минимум зависит от начального дерева видов S_0, в программе реализован метод построения особого множества исходных деревьев S_0. Он основывается на распределении вероятностей на множестве всех начальных деревьев видов, которое автоматически определяется по семейству $\{G_i\}$ деревьев генов следующим образом. Для любых видов a и b, определяется распределение $p(b|a)$ как вероятность того, что b и a сформируют элементарное дерево (т.е. будут расположены на расстоянии, не большем, чем r=2). Пусть N_a – число деревьев генов, содержащих вид a, и пусть $N_{a,b}$ – число этих деревьев, содержащих a и b на расстоянии r. Тогда $p(b|a) = N_{a,b} / N_a$, и $1 - \sum_b p(b|a)$ – это вероятность события, что не существует $b \neq a$ в элементарном дереве, включающем a (т.е. никакие виды не расположены на расстоянии r от a). Порожденное этим распределением случайное бинарное дерево S_0 служит начальным деревом для алгоритма поиска. По множеству таких начальных деревьев алгоритм выдает множество результирующих деревьев. Алгоритм строит консенсусное дерево по подмножеству так полученных алгоритмом поиска результирующих деревьев с достаточно малым значением функционала c на них.

Для каждого (полученного вышеописанным образом) начального дерева видов алгоритм поиска состоит в последовательных проходах по всем вершинам текущего дерева видов до достижения стабилизации цены вложения. Элементарная операция при этом состоит в переборе всевозможных топологий поддерева глубины 2 с корнем в данной вершине и замене его на оптимальное. В результате оптимизации полученное дерево видов удовлетворяет следующему условию: любое локальное изменение его топологии вблизи любой его вершины не уменьшает цены вложения.

Описанный алгоритм квазиоптимальный, так как перебираются лишь локальные топологии вблизи каждой вершины текущего дерева видов и результат оптимизации зависит от выбора начального дерева видов. Для устранения этого недостатка используется описанное выше стохастическое варьирование начального дерева S_0. Чем больше число испытываемых начальных деревьев, тем ближе результирующее дерево видов к абсолютному оптимуму, но тем дольше работает алгоритм.

В целях частичного решения известной проблемы, возникающей при построении дерева видов, – проблемы притяжения длинных ветвей, до минимизации функционала $c(S)$ вы-

полняется нормализация длин ребер в деревьях генов из семейства $\{G_i\}$, полученного в исходных данных. Длины пересчитываются по формуле

$$l(g) = (l(g) - l_{cp}) \cdot (1 + \mu)^{-(l(g)/l_{cp})} + l_{cp},$$

где l_{cp} – средняя длина ребра в дереве генов, μ – параметр алгоритма. Нормализация уменьшает влияние наиболее длинных ветвей в дереве генов G_i на результирующее дерево S.

Следующим этапом алгоритма является поиск множества концевых вершин дерева генов G, вызывающих рассогласование его и дерева видов S. Этот этап состоит из двух нижеописанных частей.

Первая часть. Ген g (концевая вершина в дереве генов G) рассматривается как потенциально возникший в результате горизонтального переноса, если все близкие к нему (в дереве G) гены $g_1, g_2, \ldots, g_n$, кроме самого g, отображаются с помощью α в виды $s_1, s_2, \ldots, s_n$, далекие от вида $s = \alpha(g)$; дополнительно мы хотим, чтобы множество $\{s_1, s_2, \ldots, s_n\}$ видов располагалось в дереве S достаточно кучно, т.е. его предок s_0 находился достаточно близко к листьям, а расстояние в S между s_0 и s было достаточно велико. При этом гены $g_1, g_2, \ldots, g_n$ определяются как все концевые гены кроме самого g, расположенные в G на расстоянии меньше r, где расстояние измеряется длиной пути, соединяющего g и g_i. Множество генов $\{g_1, g_2, \ldots, g_n\}$ называется еще *выколотой окрестностью* гена g радиуса r.

Если два концевых гена g и g_1 расположены на маленьком расстоянии в G, но виды $\alpha(g)$ и $\alpha(g_1)$, в которые они отображаются с помощью α, находятся на большом расстоянии в S, то это может указывать на аномальное расположение одного из этих генов. Таким образом, расстояния $r(g,g_i)$ в дереве генов и расстояния $r(s,s_i)$ в дереве видов вычисляются при $i = 1, \ldots, n$, и затем берутся средние значения

$$r(g) = \frac{1}{n}\sum_i r(g, g_i) \text{ и } r(s) = \frac{1}{n}\sum_i r(s, s_i).$$

Величина $R_g = r(s)/r(g)$ отражает, насколько размер множества $\{s_1, s_2, \ldots, s_n\}$ видов больше размера множества $\{g_1, g_2, \ldots, g_n\}$ генов. Большое значение R_g может интерпретироваться как указание на неправильное расположение гена g в дереве видов. Также используются величины p-values для статистики $p(\cdot)$, рассчитываемые по формуле

$$p(g) = \frac{1}{m}\left|\left\{g' \middle| R_{g'} \geq R_g\right\}\right|,$$

где m – число терминальных вершин. Алгоритм отбирает все гены g, для которых $p(g) \leq p_0$, где p_0 – порог. Эти гены рассматриваются как аномально расположенные.

Вторая часть. Предполагается, что любой аномально расположенный ген порождает серию ложных дупликаций и потерь при отображении α, которые нужны, чтобы объяснить

несоответствие между деревом генов G и деревом видов S в предлагаемой модели. Тогда временное удаление гена g из G и затем обновление α (после этого удаления) позволит существенно снизить стоимость $c(G,S)$ вложения G в S. Поэтому вычисляется $c(G,S)$ до и после удаления каждого гена g из G, при этом получается уменьшенное дерево генов G_g и вычисляется стоимость c_g нового вложения нового дерева генов G_g в то же самое дерево видов S. Относительное изменение стоимости вложения равно

$$F_g = \frac{c_g - c}{c}.$$

Также используются p-values для статистики F_g для всех генов g из данного КОГа G. Аналогично, алгоритм отбирает все гены g, для которых $p(g) \le p_0$.

Гены, отобранные во второй части алгоритма, интерпретируются как приобретенные (может быть, не только в результате горизонтального переноса – источник горизонтального переноса не всегда определяется алгоритмом). Гены, отобранные обеими частями алгоритма, рассматриваются как полученные в результате эволюционного события (возможно, горизонтального переноса).

Характеристику качества КОГа определим как среднее арифметическое по всем генам КОГа величин, характеризующих степень артефактности в положении гена в дереве генов относительно дерева видов. Такими величинами являются определённые в предыдущих двух пунктах величины R_g и F_g. Таким образом,

$$< R_g >= \frac{1}{m} \cdot \sum_{g \in COG} R_g ,$$

где m – число генов в КОГе и аналогично определяется $<F_g>$. Большие значения этих характеристик указывают, вообще говоря, на низкое качество данного КОГа (и его дерева G).

Опишем реализованную в программе схему реконструкции эволюционных событий. Сначала тестируются все гены из каждого КОГа из заданного семейства КОГов и отбираются те гены, которые в большей степени ответственны за рассогласование дерева генов и дерева видов. Затем отобранные гены удаляются из соответствующих деревьев генов, и вложение α каждого из этих деревьев генов в дерево видов пересчитывается заново. Для каждого КОГа до и после удаления рассчитываются числа дупликаций генов и потерь генов. Первый случай соответствует non-GAIN сценарию, а второй – GAIN-сценарию. Деля разность между числом потерь в первом сценарии и числом потерь во втором сценарии на число приобретений (т.е. на число отобранных генов), рассчитываем, на сколько в среднем уменьшает число потерь гипотеза об одном событии приобретения, что является важной характеристикой эволюции этих белковых семейств. Рассчитываем распределение суммарного числа дупликаций

и потерь в обоих сценариях по подсемействам и выдаем соответствующие выводы эволюционного характера. Например, большое значение (сравнимое с числом белковых семейств) суммарного числа дупликаций генов, отнесенное к какой-либо вершине дерева видов, интерпретируется как результат геномной дупликации.

3.2.3. сведения об ограничениях на применение

Основным фактором, ограничивающими возможность применения программы TIQMax при осуществлении функции 1, является длительное время счёта, заметно растущее с увеличением размера используемых деревьев. Функции 2 и 3 не требуют длительного счёта и на практике работают секунды даже для больших объёмов данных. При любом варианте использования программа не предъявляет заметных требований к объёму доступной памяти: объём хранимых в памяти данных примерно равен по порядку объёму входных и выходных данных, не превышающему на практике единицы мегабайт.

3.3. Описание логики программы

3.3.1. Описание структуры программы и её основных частей

Программа TIQMax состоит из 35 файлов с исполняемым кодом на языке C++ и 41 заголовочного файла, в результате трансляции и сборки которых образуется один исполняемый модуль TIQMax. Соответственно, межпрограммное взаимодействие отсутствует.

3.3.2. Описание функций составных частей и связей между ними

Исходный код программы может быть довольно чётко разделён на следующие 4 части, которые могут пониматься как «слои»:

– основные алгоритмы программы;
– механизмы чтения деревьев из файлов;
– библиотека классов деревьев;
– библиотека классов общего назначения.

Исходные коды каждого из трёх «верхних» слоёв существенно используют «нижележащие» слои (расположенные ниже их в приведённом списке), но никаким образом не зависят от вышележащих слоёв и могут быть использованы в других программах отдельно от них. Распределение файлов исходного кода программы по этим четырём слоём приведено в табл. 1.

Таблица 1. Функции составных частей программы.

Файл с исполняемым кодом	**Заголовочный файл**	**Выполняемая функция**
Основные алгоритмы программы		

Файл с испол-няемым кодом	Заголовочный файл	Выполняемая функция
Analyse.cpp	Analyse.h	функция анализа эволюционных событий
CmdLnPrm.cpp	CmdLnPrm.h	функция ввода параметров из командной строки
DefGlSet.cpp	DefGlSet.h	функция установки значений по умолчанию глобальных параметров программы
DupTree.cpp	DupTree.h	класс DuplicationsTree, реализующий дерево дупликаций
GeneTabl.cpp	GeneTabl.h	класс GenesTable, реализующий таблицу принадлежности генов видам
GlSttngs.cpp	GlSttngs.h	массив глобальных параметров программы
IntMat.cpp	IntMat.h	класс IntMatrix, реализующий целочисленную матрицу
LogOut.cpp	LogOut.h	функции вывода в файл протокола
	OpenFile.h	функции открытия файла
Optimize.cpp	Optimize.h	функции верхнего уровня управления оптимизацией вложения
ReadSets.cpp	ReadSets.h	функция чтения глобальных параметров программы из файла настроек
ReadTree.cpp	ReadTree.h	функции чтения деревьев из файлов
Settings.cpp	Settings.h	классы, реализующие работу с глобальными параметрами программы
	SpTrAr.h	класс, реализующий массив деревьев видов
	StrCnsts.h	глобальные строковые константы интерфейса пользователя
TInject.cpp	TInject.h	классы, реализующие вложение деревьев
TIQMax.cpp	TIQMax.h	класс, реализующий алгоритмы оптимизации вложения
TIQMMain.cpp		функция main( ), организующая деятельность программы на самом верхнем уровне
TModify.cpp	TModify.h	класс TreeModifier, реализующий локальную модификацию дерева видов
TreeOut.cpp	TreeOut.h	функции вывода деревьев пользователю
TUtils.cpp	TUtils.h	различные вспомогательные функции работы с деревьями
механизмы чтения деревьев из файлов		
GParser.cpp		файлы этого слоя автоматически генерируются программами синтаксического анализа Flex и Bison; в программе используется лишь два метода класса TreeParser, определяемого в файле ‘TParser.h’: – TreeParser::getGeneTrees( ) – TreeParser::getSpeciesTrees( ) это единственные возможные точки входа в данный слой.
L_YYwrap.cpp		
LexYY.cpp	LexYY.h	
Parser.cpp	Parser.h	
SParser.cpp	TParser.h	
Variable.cpp	Variable.h	
	YYDefs.h	
библиотека классов деревьев		
BioTree.cpp	BioTree.h	базовый класс BioTree для филогенетических деревьев
GeneTree.cpp	GeneTree.h	класс GeneTree, реализующий дерево генов (белков)
SpecTree.cpp	SpecTree.h	класс SpeciesTree, реализующий дерево видов
Tree.cpp	Tree.h	базовый класс Tree, реализующий дерево
TreeBase.cpp	TreeBase.h	абстрактный базовый класс TreeBase для древовидных структур
TreeMan.cpp	TreeMan.h	класс TreeManipulator, реализующий манипуляцию

Файл с исполняемым кодом	Заголовочный файл	Выполняемая функция
		деревьями
библиотека классов общего назначения		
ClArBase.cpp	ClArBase.h	базовый класс ClArrayBase для динамических массивов элементов типа Clonable
ClArray.cpp	ClArray.h	динамический массив элементов типа Clonable (ClArray)
	Clonable.h	«интерфейс» Clonable (объекты с функцией клонирования)
	ClsDefs.h	типы, константы и макроопределения для препроцессора
ClString.cpp	ClString.h	класс String, реализующий текстовую строку
	ClsTypes.h	список идентификаторов типов используемых классов
NewHndlr.cpp	NewHndlr.h	обработчик нехватки динамической памяти
	Object.h	абстрактный базовый класс Object (корень иерархии классов)
	Sortable.h	«интерфейс» Sortable (объекты с возможностью сравнения / сортировки)
SrtArray.cpp	SrtArray.h	отсортированный динамический массив объектов Sortable (SrtArray)

3.3.3. Сведения о языке программирования

Все функции запрограммированы на языке C++ с соблюдением стандарта ANSI C++ и спецификации ISO/IEC от 1998 г. Тестовая сборка программы производилась компилятором gcc (g++) в среде операционной системы Linux и портами gcc на платформу MS Windows (в средах разработки DJGPP и CygWin).

3.3.4. Описание входных и выходных данных

Входными данными программы TIQMax являются:

- *файл деревьев генов*, содержащий список используемых деревьев эволюции белковых семейств; необходим при любом варианте использования программы;
- *таблица принадлежности генов видам*, содержащая сведения о принадлежности каждого гена (белка) из каждого дерева генов какому-либо виду; также необходима при любом варианте использования программы;
- необязательный *файл дерева видов*, содержащий полученное ранее с помощью программы или взятое из независимых источников эталонное дерево видов, с которым будет производиться сравнительный анализ деревьев генов; файл используется в функциях 2 и 3;
- *файл настроек*, содержащий все необходимые параметры алгоритма, имена входных/выходных файлов и прочие настройки программы.

Формат входных данных приведен в приложении 1.

Выходными данными программы TIQMax являются:

- *файл деревьев видов*, являющийся основным выходным файлом программы при выполнении функции 1 (когда параметру `NumberOfRandomSpeciesTrees` задано положительное значение) и содержащий набор полученных в результате оптимизации деревьев видов, имеющих минимальную цену вложения в него заданных деревьев генов (при заданных параметрах), вместе с соответствующими им ценами вложения; этот файл также является входным в режиме продолжения работы с использованием результатов предыдущего запуска (когда параметру `ResumeCalculations` задано значение true); в этом случае в начале работы программа читает из данного файла список лучших деревьев видов, полученных в ходе предыдущих запусков, и работает с ними так же, как если бы они были получены в ходе текущего запуска;
- *файлы эволюционных событий*, являющиеся основными выходными файлами программы при выполнении функции 2 (когда параметр `AnalyseEvolutionaryEvents` = true) и содержащие распределение числа различных эволюционных событий по дереву видов;
- *файл качества генов и КОГов*, являющийся основным выходным файлом программы при выполнении функции 3 (когда параметр `LookForBadGenes` установлен в true);
- *файл протокола*, детально описывающий ход исполнения программы;
- сообщения о ходе работы программы и нештатных ситуациях, выдаваемые на консоль оператора.

Описание форматов выходных данных программы TIQMax приведено в приложении 2.

3.3.5. Описание логики составных частей

Функция main( ) реализует логику программы на самом верхнем уровне, управляя выполнением основных этапов и алгоритмов программы. Функция состоит из следующих основных этапов (в скобках указаны номера строк файла TIQMMain.cpp):

– подготовка к началу работы (:19-87);
– ввод данных (:90-118);
– проверка и предобработка данных (:121-198);
– основная часть (:201-218).

Вначале производится обработка запроса подсказки в параметрах командной строки запуска программы (:25-32). Если первым параметром командной строки является одна из комбинаций символов “--help”, “-h”, “/?” или “?”, то на консоль оператора выводится краткая справка о формате и возможных параметрах командной строки. После этого производится установка значений глобальных параметров конфигурации программы в соответствие с заданными аргументами командной строки и содержимым файла настроек (см. приложения 1 и 3), для чего выполняется следующая последовательность действий (:37-60):

– набор настроек программы, хранящийся в глобальной переменной *globalSettings* (объекте класса Settings) устанавливается в значения по умолчанию посредством вызова функции setDefaultGlobalSettings( );
– в локальную переменную *commandLineParameters* (объект класса Settings) считываются посредством вызова функции readCommandLineParameters( ) необязательные параметры командной строки;
– значения набора глобальных настроек *globalSettings* корректируются с учётом информации в *commandLineParameters*;
– из набора глобальных настроек *globalSettings* в переменную *settingsFileName* извлекается значение пользовательского параметра с именем SettingsFileName;
– производится попытка открытия файла с именем *settingsFileName* на чтение; если файл существует, хранящиеся в нём пользовательские настройки считываются в массив *globalSettings* посредством вызова функции readGlobalSettings( ), в противном случае набор всех настроек программы из массива *globalSettings* с их текущими значениями выводится в файл с именем *settingsFileName* (в виде, допускающем корректировку значений параметров пользователем и считывание его программой в качестве файла настроек);
– значения глобальных настроек в *globalSettings* повторно корректируются с учётом информации в *commandLineParameters*, что гарантирует приоритет значений параметров, заданных в командной строке, перед значениями, указанными в файле конфигурации.

Далее, если включён пользовательский параметр LogAllowed, производится открытие файла протокола (:62-76). В завершение подготовительного этапа производится взаимная корректировка параметров (в том случае, если их значения вступают в противоречие) (:78-87).

Этап ввода данных начинается с загрузки в локальную переменную *gTable* (объект класса GenesTable) таблицы принадлежности генов видам из соответствующего файла (см. приложение 2), которая выполняется с помощью функции readGenesTable( ) (:94-96). Затем в локальную переменную *gTrees* (объект класса ClArray) посредством вызова функции readGeneTrees( ) производится загрузка набора деревьев генов (:98-102). В заключение данного этапа, если значение пользовательского параметра NumberOfRandomSpeciesTrees положительно, в локальную переменную *sTree* (объект класса SpeciesTree) посредством вызова функции readSpeciesTree( ) загружается дерево видов (:104-113).

Этап проверки и предобработки данных начинается с приписывания соответствующих видов листьям всех деревьев генов путем вызова функции setSpeciesOnGeneTrees(*gTrees*, *gTable*) (:125-126). Затем, если не задано дерево видов (пользовательский параметр NumberOfRandomSpeciesTrees имеет положительное значение), посредством вызова функции generateSpeciesTree(*gTrees*, *sTree*) (128-130) в переменной *sTree* создаётся «заготовка» для

дерева видов (дерево с тривиальной структурой, обладающее листьями, соответствующими всем видам, гены из которых имеются в заданном наборе деревьев генов). Затем, если включён пользовательский параметр CheckGeneTrees, с помощью обращения к функции removeAbsentGenes(*gTrees*, *sTree*) (:132-146) из деревьев генов удаляются все листья, соответствующие генам, не связанным (посредством таблицы принадлежности генов видам) ни с одним из видов, приписанных листьям деревьев видов (т.е. удаляются все гены, не упомянутые в *gTable*, и все гены, для которых нет соответствующего вида в *sTree*). После этого, если включён пользовательский параметр MergeOneSpeciesSubtrees, для каждого дерева генов из *gTrees* применением метода mergeOneSpeciesSubrees( ) (:148-155) производится слияние в единый лист каждого поддерева, всем листьям которого приписаны гены, принадлежащие одному и тому же виду. Если включён пользовательский параметр IgnoreLengths, посредством вызова функции equalizeLengths(*gTrees*) всем ветвям всех деревьев генов присваивается длина 1; в противном случае посредством вызова функции normalizeGeneTrees(*gTrees*) длины всех ветвей всех деревьев генов делятся на число, приписанное корню их дерева (это даёт возможность пользователю менять степень вклада каждого дерева генов в результат оптимизации или анализа эволюционных событий), после чего проводится процедура нормализации длин рёбер, описанная в разделе 3.2.2 выше, с параметром μ, равным значению пользовательского параметра GeneTreeNormFactor (:157-161). Создаётся локальный объект *proximity* класса IntMatrix – квадратная матрица с размерностью, равной числу видов в таблице принадлежности генов видам; если пользовательский параметр NumberOfRandomSpeciesTrees имеет положительное значение, и включён параметр UseEmpiricalProbability, посредством вызова функции calculateProximity(*proximity*, *gTrees*, *gTable*) матрица *proximity* заполняется частотами соседства пар видов в данном наборе деревьев генов (:163-167). В заключение этапа проверки и предобработки данных, если включён пользовательский параметр ProduceGeneTreesFiles, каждое обработанное вышеизложенным образом дерево генов выводится в отдельный файл, имя которого образуется путём добавления к имени дерева генов (метке, приписанной его корневой вершине), расширения ".ph" (:183-196).

Основная часть функции main( ) заключается в оптимизации заданного или построении нового дерева видов и/или проведения анализа эволюционных вложений. Если пользовательский параметр NumberOfRandomSpeciesTrees имеет положительное значение, то посредством вызова функции optimizeRandomSpeciesTrees(*gTrees*, *sTree*, *gTable*, *proximity*) запускается процедура построения дерева видов путём генерации и оптимизации начальных деревьев видов (функция 1 программы), в противном случае посредством вызова функции optimizeGivenSpeciesTree(*gTrees*, *sTree*) запускается процедура оптимизации заданного дерева видов (*sTree*), которая, однако, не меняет дерева видов в случае, если включён пользо-

вательский параметр InjectOnly (что является предпочтительным вариантом для использования функций 2 и 3 программы) (:203-208). После этого, если включён хотя бы один из пользовательских параметров LookForBadGenes или “AnalyseEvolutionaryEvents”, выполняется вызов функции analyseInjections(*gTrees*, *sTree*, *gTable*), выполняющей анализ эволюционных событий и качества белковых семейств (функции 2 и 3 программы) (:210-218).

Функция optimizeRandomSpeciesTrees( ) обеспечивает управление процедурой построения дерева видов путём многократного порождения начальных деревьев видов, их локальной оптимизации и отбора подмножества полученных деревьев, в наибольшей степени согласующегося с заданным набором деревьев генов. Прототип функции находится в файле ‘Optimize.h’:

```
OK_ERROR optimizeRandomSpeciesTrees( const ClArray & gTrees,
  SpeciesTree & sTree, const GenesTable & gTable,
  const IntMatrix & proximity );
```

Определение функции находится в файле ‘Optimize.cpp’. Параметры имеют следующий смысл: *gTrees* – набор деревьев генов, используемый для построения дерева видов; *sTree* – дерево видов, структура которого подвергается многократной оптимизации и по завершении работы функции принимает вид, наиболее согласующийся с заданным набором деревьев генов (начальная структура дерева игнорируется, но набор листьев всегда остаётся одинаковым); *gTable* – таблица принадлежности генов видам; *proximity* – матрица частоты соседства пар видов в множестве деревьев генов. Функция возвращает значение, равное одной из констант: *OK* (нормальное завершение работы) или *ERROR* (признак неудачи). В начале работы функции в локальные переменные *numOfBestSTrees* и *resultFileName* импортируются значения пользовательских параметров NumberOfResultSpeciesTrees и ResultFileName соответственно (:152-154). Создаются локальные объекты *bestSTree* (объект класса SpeciesTree для хранения лучшего полученного варианта дерева видов) и *bestSTrees* (объект класса SpeciesTreesArray для хранения *numOfBestSTrees* лучших деревьев видов) (:156-157). Затем (:159-228), если файл с именем *resultFileName* существует, производится либо его удаление (если пользовательский параметр ResumeCalculations выключен), либо (в противном случае) попытка загрузить из него в массив *bestSTrees* варианты деревьев видов, полученные в ходе предыдущих сеансов работы программы; последнее производится в следующем порядке:

– создаётся временный локальный массив *sTrees* (объект класса ClArray), в который посредством вызова функции readSpeciesTrees(*sTrees*) считываются все деревья видов, имеющиеся в файле с именем, заданным пользовательским параметром ResultFileName;
– каждое дерево видов добавляется в массив *bestSTrees*; если метка его корня содержит информацию о цене вложения и числе появлений данного дерева в ходе предыдущих запусков, эта информация фиксируется и цена вложения проверяется на соответствие фактиче-

скому её значению; при этом, если цены вложения совпали для первых трёх деревьев, для остальных вложение уже не производится, а цена их вложения напрямую копируется из метки их корня (при условии, что все они содержат информацию о цене вложения).

После (возможного) считывания результатов предыдущих запусков начинается непосредственно процедура итерационного построения дерева видов, реализуемая циклом, в теле которого *numOfRandSTrees* раз производятся следующие действия:

- рандомизация текущего дерева видов с учётом матрицы близости пар видов в наборе деревьев генов посредством вызова функции randomizeSpeciesTree(*sTree*, *proximity*, *gTable*);
- создаётся локальный объект *injector* класса TreeInjector с инициализацией ссылками на *gTrees* и *sTree*; с помощью его метода inject( ) производится начальное вложение деревьев генов в дерево видов;
- применением к паре (*gTrees*, *sTree*) метода maximize( ) неименованного временного объекта класса TreeInjectionQualityMaximizer дерево видов приводится к локально-оптимальному строению;
- если включён пользовательский параметр DefineLengthsOnSpeciesTree, на полученное дерево видов индуцируются длины ветвей деревьев генов применением к паре (*gTrees*, *sTree*) метода defineLengths( ) неименованного временного объекта класса TreeInjectionQualityMaximizer;
- оптимизированное дерево видов помещается в массив *bestSTrees* (с автоматическим соблюдением упорядоченности по возрастанию цены вложения и возможным вытеснением наименее качественного дерева);
- если значение пользовательского параметра AutosavePeriod отлично от нуля, и порядковый номер прохода цикла ему кратен, производится запись текущего списка лучших деревьев видов в файл вызовом функции writeResults(*bestSTrees*).

По окончании цикла производится безусловная запись списка лучших деревьев видов в файл вызовом функции writeResults(*bestSTrees*).

Функция analyseInjections( ) реализует анализ вложений, реконструкцию эволюционных событий, вычисление их характеристик и качества белковых семейств. Прототип функции находится в файле ‘Analyse.h’:

```
analyseInjections( const ClArray & gTrees, const SpeciesTree & sTree,
  const GenesTable & gTable );
```

Определение функции находится в файле ‘Analyse.cpp’. Параметры имеют следующий смысл: *gTrees* – набор деревьев генов, *sTree* – дерево видов, *gTable* – таблица принадлежности генов видам; возвращаемое значение равно одной из констант *OK* (нормальное заверше-

ние работы) или *ERROR* (признак неудачи). Функция может быть разделена на 2 основные этапа:

– выявление аномально расположенных генов, и оценка качества КОГов (:238-979);
– анализ распределения эволюционных событий по дереве видов (:938-1120).

Основной используемой в функции структурой данных является класс GeneWithCost, хранящий информацию об отдельном гене в аспекте рассматриваемой задачи. Основные члены-данные класса GeneWithCost приведены в таблице 2.

Таблица 2. Члены класса GeneWithCost.

Имя	**Тип**	**Описание**
name	String	имя гена
spec	String	имя вида
tNum	int	порядковый номер дерева генов
tName	String	имя КОГа (дерева генов)
cost	double	(относительная) величина приращения цены вложения (F_g)
leng	double	(относительная) длина ветви листа-гена в дереве генов
stgr	double	отношение среднего расстояния соседей на дереве видов к их среднему расстоянию на дереве генов (R_g)
cmdT	double	отклонение F_g от среднего по дереву (в долях среднеквадратичного отклонения σ)
cost_p	double	p-value по F_g
stgr_p	double	p-value по R_g

В начале работы функции значение глобальной настройки CostRatio устанавливается равным значению пользовательского параметра CostRatio2, а в конце возвращается в исходное состояние. Это даёт возможность пользователю задавать специфическое значения этого параметра (γ) для задачи анализа эволюционных событий.

Этап выявления аномально расположенных генов и оценки качества КОГов начинается с выполнения «большого» цикла обработки вложений (:360-530), в котором проводится анализ характеристик вложений отдельных деревьев генов. В этот цикл входят:

– цикл первичной обработки генов данного дерева с подсчётом F_g (:377-414);
– цикл вторичной обработки генов данного дерева с подсчётом R_g (:421-483);
– цикл подсчёта величин p-value для F_g и R_g (:493-508);

По завершении «большого» цикла выполняется цикл подсчёта значений p-value ранее вычисленных средних величин $\langle F_g \rangle$ и $\langle R_g \rangle$ (:535-551). Вычисленные характеристики выводятся в виде таблиц в файл качества генов и КОГов, при этом характер выводимых данных определяется значением пользовательского параметра SimpleInjectionAnalysisOutput: в случае, когда этот параметр включён, в «файл качества генов и КОГов» выводится две таблицы,

описанные в приложении 2, содержащие небольшой объём самой важной информации; в противном случае в файл выводится существенно больший объём данных, предназначенных в основном для углубленного анализа результатов и отладки программы.

Этап анализа распределения эволюционных событий по дереву видов выполняется только если включён пользовательский параметр AnalyseEvolutionaryEvents. Он начинается с создания локального объекта *dTree* класса DuplicationsTree (:1016), в который с помощью объекта класса TreeInjector выполняется вложение деревьев генов (:1019-1020). С помощью методов defineNumberOfDuplications( ), defineSumOfDuplications( ) и defineSumOfLosses( ) класса DuplicationsTree на объекте *dTree* собирается три вида статистики эволюционных событий, соответствующие non-GAIN сценарию, выводимые в соответствующие выходные файлы (см. приложение 2) (:1022-1044). Затем из копии деревьев генов удаляются все найденные выше аномально расположенные гены и вложение повторяется, за счёт чего осуществляется переход к GAIN-сценарию (:1046-1061), после чего повторяется сбор тех же трёх видов статистических данных, выводимых в соответствующие выходные файлы (:1071-1093). В заключение проводится анализ распределения суммы числа потерь генов по вершинам дерева видов, эти числа выводятся в последний из семи выходных файлов (:1095-1112).

4. Приложение 1. Формат входных данных

4.1. Файл деревьев генов

Текстовый файл деревьев генов (его имя задаётся параметром `GeneTreesFileName`) хранит набор деревьев белковых семейств в формате Newick (см. http://evolution.genetics.washington.edu /phylip/newicktree.html). Пример файла деревьев генов приведён на рисунке 1.

```
(((((((((ATP6_DROSOP:100.0,ATP6_ANOPH:100.0):78.3,
ATP6_PENAEUS:100.0):50.7,ATP6_LOCUST:100.0):96.0,ATP6_ARTEM:100.0):78.8,
ATP6_IXODE:100.0):96.2,((((((ATP6_GALLUS:100.0,ATP6_ALLIGA:100.0):41.2,
((ATP6_SALMO:100.0,ATP6_XENOR:100.0):49.2,ATP6_SQUAL:100.0):78.3):58.5,
ATP6_PETR:100.0):43.2,(ATP6_HOMO:100.0,ATP6_DIDEL:100.0):100.0):80.5,
ATP6_BRANC:100.0):91.5,(ATP6_FLORO:100.0,
ATP6_ASTER:100.0):100.0):41.8):36.5,((ATP6_LUMBR:100.0,
ATP6_PLATY:100.0):88.5,ATP6_KATHA:100.0):99.0):100.0,
ATP6_METRI:100.0):98.0,(((ATP6_RECLINOM:100.0,ATP6_MARCH:100.0):68.7,
((ATP6_CAFETER:100.0,ATP6_CHRY:100.0):49.0,ATP6_PHYT:100.0):45.0):37.7,
(ATP6_PORPH:100.0,ATP6_CYAND:100.0):97.5):75.0):100.0,ATP6_ALLOM:100.0);
...
```

Рисунок 1. Фрагмент файла деревьев генов.

4.2. Таблица принадлежности генов видам

Текстовый файл таблицы принадлежности генов видам (его имя задаётся параметром `GeneTableFileName`) состоит из записей переменной длины со следующим форматом. Каждая запись начинается идентификатором вида, за которым следует знак равенства и список идентификаторов соответствующих генов, разделённых запятыми, (возможно, но не обязательно) заключённый в фигурные скобки и завершается точкой с запятой (‘;’). Пробельные символы (пробел, табуляция, символ новой строки) могут присутствовать в любом числе и в любых местах, кроме как внутри идентификаторов. Пример таблицы принадлежности генов видам приведён на рисунке 2.

```
ALLIGA = { ND1_ALLIGA, ND2_ALLIGA, COX1_ALLIGA, COX2_ALLIGA, ATP8_ALLIGA,
ATP6_ALLIGA, COX3_ALLIGA, ND3_ALLIGA, ND4L_ALLIGA, ND4_ALLIGA, ND5_ALLIGA,
ND6_ALLIGA, CYTB_ALLIGA};

ALLOM = { COB_ALLOM, COX3_ALLOM, COX2_ALLOM, NAD5_ALLOM, ATP6_ALLOM,
COX1_ALLOM, NAD6_ALLOM, ATP8_ALLOM, NAD3_ALLOM, NAD2_ALLOM, NAD4_ALLOM,
NAD1_ALLOM, NAD4L_ALLOM, ATP9_ALLOM };

ANOPH = { ND2_ANOPH, COX1_ANOPH, COX2_ANOPH, ATP8_ANOPH, ATP6_ANOPH,
COX3_ANOPH, ND3_ANOPH, ND5_ANOPH, ND4_ANOPH, ND4L_ANOPH, ND6_ANOPH,
CYTB_ANOPH, ND1_ANOPH };
```

Рисунок 2. Пример таблицы принадлежности генов видам.

4.3. Входной файл дерева видов

Текстовый входной файл деревьев видов (его имя задаётся параметром `SpeciesTreesFileName`) имеет стандартный формат Newick (см. раздел 4.1 выше), но не должен содержать длин ветвей. Пример файла дерева видов приведён на рисунке 3.

```
((((Afu,(Tac,Tvo)),Hbs),((Ape,Sso),((Mja,Mth),(Pab,Pho)))),
((((Bbu,Tpa),(Cpn,Ctr)),((((((Buc,((Eco,(Hin,Pmu)),Vch)),Pae),Xfa),Nme),
((Ccr,Mlo),Rpr)),(Cje,Hpy))),(((((Bha,Bsu),Sau),(Lla,Spy)),
((Mge,Mpn),Uur)),((Dra,Mtu),Syn))),(Aae,Tma));
```

Рисунок 3. Пример файла дерева видов.

4.4. Файл настроек

Имя файла настроек передаётся программе в командной строке с помощью ключа ‘`-s`’; например, если файл настроек имеет имя ‘Settings.txt’, то командная строка выглядит следующим образом:

```
TIQMax -sSettings.txt
```

(если имя файла настроек не указано – оно считается равным ‘TIQMax.set’).

Файл настроек программы представляет собой текстовый файл, каждая рабочая строка которого содержит ровно одно присваивание значения одному из параметров программы. Присваивание имеет вид "SettingName = Setting Value", где `SettingName' - идентификатор, являющийся именем одной из глобальных настроек программы, 'Setting Value' – текстовая строка, содержание которой трактуется в зависимости от типа данной настройки. Символы пробела и табуляции в рабочих строках игнорируются (за исключением того случая, когда они составляют часть значения). Помимо рабочих строк в файле установок допускается наличие пустых строк и строк комментариев. Пустой строкой считается строка, не содержащая никаких символов кроме, возможно, пробельных. Строкой комментария считается строка, первый символ которой, не являющийся пробельным, – точка с запятой (';'). Пустые строки и строки комментариев игнорируются программой.

Основные настройки программы с указанием их типа, значения по умолчанию и кратким описанием значения приведены в приложении 3.

5. Приложение 2. Формат выходных данных

5.1. Выходной файл деревьев видов

Текстовый выходной файл деревьев видов (его имя задаётся параметром `ResultFileName`) имеет стандартный формат Newick (см. раздел 4.1 выше). В качестве метки корню дерева приписана запись вида ‘cost_XXX’ или ‘cost_XXXxNNN’, означающая, что цена вложения в данное дерево видов заданного семейства деревьев генов (с заданными параметрами) равняется XXX, и (в случае второй формы записи) данное дерево явилось результатом оптимизации NNN случайных деревьев видов. Пример файла деревьев видов приведён на рисунке 4.

Для построения по данному файлу консенсусного дерева видов рекомендуется использовать программу *consense* из пакета *Phylip*.

```
(((((Aae,Tma),((Bbu,Tpa),(Cpn,Ctr))),(((((Bha,Bsu),Sau),
(Lla,Spy)),((Mge,Mpn),Uur)),(Cje,Hpy))),(((((((Buc,(Eco,
(Hin,Pmu))),Vch),Pae),Xfa),Nme),((Ccr,Mlo),Rpr)),
((Dra,Mtu),Syn))),(((Afu,Hbs),(Mja,Mth)),((Ape,Sso),
((Pab,Tvo),(Pho,Tac)))))cost_66627.87x23;

(((((Aae,Tma),((Bbu,Tpa),(Cpn,Ctr))),
(((((Bha,Bsu),Sau),(Lla,Spy)),((Mge,Mpn),Uur)),(Cje,Hpy))),
(((((((Buc,(Eco,(Hin,Pmu))),Vch),Pae),Xfa),Nme),((Ccr,Mlo),Rpr)),
((Dra,Mtu),Syn))),(((Afu,Hbs),(Mja,Mth)),((Ape,Sso),((Pab,Tac),
(Pho,Tvo)))))cost_66627.87x29;

(((((Aae,Tma),((((Bha,Bsu),Sau),(Lla,Spy)),
((Mge,Mpn),Uur))),(((Bbu,Tpa),(Cje,Hpy)),(Cpn,Ctr))),
(((((((Buc,(Eco,(Hin,Pmu))),Vch),Pae),Xfa),Nme),
((Ccr,Mlo),Rpr)),((Dra,Mtu),Syn))),(((Afu,Hbs),(Mja,Mth)),
((Ape,Sso),((Pab,Tac),(Pho,Tvo)))))cost_66648.79x12;
```

Рисунок 4. Пример файла деревьев видов.

5.2. Выходные файлы эволюционных событий

Текстовые выходные файлы эволюционных событий (7 штук, их имена задаются параметром `ResultFileName` с добавлением суффиксов, описанных ниже) имеют вид копий заданного дерева видов, вершинам которых приписаны различные числовые характеристики. Деревья хранятся в стандартном формате Newick (см. раздел 4.1 выше). Имена файлов образуются посредством добавления к имени файла результатов (но до его расширения, если таковое имеется) семи фиксированных суффиксов, кратко выражающих смысл чисел, приписанных вершинам дерева видов в данном файле. Используемые суффиксы и описание содержимого соответствующих файлов приведены в следующей таблице.

№	Суффикс	Содержимое файла
1	1a-non-gain-dup	Вершинам дерева видов приписано *число дупликаций*, равное сумме дупликаций, возникающих в данной вершине дерева видов при вложении каждого дерева генов.
2	1b-non-gain-sum-dup	Вершинам дерева видов приписано *суммарное число дупликаций*, равное сумме дупликаций, возникающих в данной вершине дерева видов и её потомках при вложении каждого дерева генов.
3	1c-non-gain-losses	Вершинам дерева видов приписано *суммарное число потерь*, равное сумме потерь, возникающих в данной вершине дерева видов и её потомках при вложении каждого дерева генов.
4	2a-gain-dup	То же, что 1, но после удаления аномально расположенных генов (GAIN-сценарий).
5	2b-gain-sum-dup	То же, что 2, но после удаления аномально расположенных генов (GAIN-сценарий).
6	2c-gain-losses	То же, что 3, но после удаления аномально расположенных генов (GAIN-сценарий).
7	3-gains	Вершинам дерева видов приписаны *суммарное число приобретений*, равное сумме аномально расположенных генов, принадлежащих видам из определяемых ими поддеревьев.

5.3. Выходной файл качества генов и КОГов

Текстовый выходной файл качества генов и КОГов (его имя задаётся параметром `BadGenesFileName`) имеет формат HTML и содержит две таблицы:

1. В таблице 1 для каждого КОГа (с ненулевым числом артефактных генов) приведён список артефактных генов из него. Таблица упорядочена в соответствии с порядком деревьев во входном файле и содержит следующие столбцы: 1 – идентификатор КОГа, 2 – идентификатор гена (белка), идентификатор вида (организма), значение F_g для данного гена, значение R_g для данного гена.
2. В таблице 2 приводятся две характеристики качества каждого КОГа: два средних арифметических по всем генам КОГа величин R_g и F_g. Таблица упорядочена по убыванию среднего значения R_g и содержит следующие столбцы: 1 – номер КОГа (в порядке следования деревьев во входном файле), идентификатор КОГа, среднее значение R_g для белков данного КОГа, среднее значение F_g для белков данного КОГа.

5.4. Файл протокола программы

Текстовый файл протокола содержит сообщения на русском языке о ходе исполнения программы и фрагменты данных в формате Newick и других соответствующих форматах (рис. 5).

```
*******************************************************************************
* TIQMax.log: Log-файл программы TIQMax.                                      *
*******************************************************************************

Версия 0.610 beta (Build May  3 2006 14:55:31).

Командная строка:
c:/usrfiles/work/iitp/devel/tiqmax/src/tiqmax/bin/tiqmax.exe -sSettings.txt

--- Значения глобальных устоновок программы :
CostDeltaPValueThreshold = 0.1
CostDeltaThreshold = -7
CostRatio = 1
CostRatio2 = 0.1
DispersionPValueThreshold = 0.1
DispersionThreshold = 2
GeneTreeNormFactor = 0.7
AutosavePeriod = 100
LocalOptimizationsOrder = 0
NeighbourhoodRadius = 4
NumberOfRandomSpeciesTrees = 0
NumberOfResultSpeciesTrees = 0
RandomNumberSeed = 0
BadGenesFileName = _uc2/BadGenes.html
GeneTreesFileName = _uc2/GTrees.gt
GenesTableFileName = _uc2/GTable.gtb
<...>

 * Файл '_uc2/GTable.gtb' успешно открыт на чтение.
 * Читаю таблицу принадлежности генов видам.
 * Чтение таблицы генов успешно завершено.
 * Файл '_uc2/GTrees.gt' успешно открыт на чтение.
 * Читаю деревья генов.
 * Чтение деревьев генов успешно завершено.
 * Прочтено 132 деревьев генов.
 * Файл '_uc2/STree.ph' успешно открыт на чтение.
 * Читаю дерево видов.
 * Чтение дерева видов успешно завершено.

 * Проверяю деревья генов.
 * Проверка деревьев генов завершена.
 * Деревья генов -- в порядке.
 * Объединяю поддеревья принадлежащие одному виду.

Деревья генов после предобработки:
1-e:
     (([Pab]PAB1357:4.423,[Pho]PH0525:6.164):28.174,((((((((((((([Cpn]CPn0321:20.086,[Ctr]CT092
:11.414):34.644,[Buc]BU191:34.4192):6.857,([Hpy]HP0569:24.901,[Cje]Cj0930:24.287):16.927):6.93
6,([Nme]NMB1838:22.312,(((((Vch]VC2185:13.946,([Hin]HI0393:2.208,[Pmu]PM0163:4.515):6.518):3.4
31,[Eco]ychF:11.893):7.373,[Pae]PA4673:15.575):5.545,[Xfa]XF2641:26.4616):4.999):10.407):5.703
,(([Dra]DR1386:27.9689,[Mtu]Rv1112:33.3927):10.818,(([Ccr]CC0479:23.218,[Rpr]RP604:31.9914):8.
772,[Mlo]mlr2695:23.736):13.906):5.468):3.803,(((((Mge]MG024:11.881,[Mpn]MPN026:25.587):30.699
,[Uur]UU595:27.1923):20.513,(([Lla]L0161:7.328,[Spy]SPy0006:6.161):18.243,([Sau]SA0351:22.815,
([Bsu]BS_yyaF:8.603,[Bha]BH4051:11.63):6.053):3.714):10.142):7.499,([Syn]sll0245:28.0466,([Tpa
]TP0124:32.2812,[Bbu]BB0235:33.2423):13.521):4.371):6.596):13.626,[Aae]aq_609:26.319):10.892,[
Tma]TM1240:34.9675):89.98,([Tac]Ta0905:20.12,[Tvo]TVN1009:18.769):48.272):25.769,[Hbs]VNG1749G
+VNG0504G:35.1544):7.5,[Mth]MTH1515:28.6864):7.475,(([Sso]SSO0743:27.3964,[Ape]APE1164:32.6486
):12.493,[Afu]AF1364:28.6903):2.977):6.135,[Mja]MJ1332:26.3798)COG0012;
2-e:
     ((((((((((((((([Mja]MJ0564:35.7292,[Mth]MTH1683:37.9466):13.093,([Pho]PH0297:6.876,[Pab]P
AB1245:5.884):37.297):6.289,((((Tac]Ta0849:20.802,[Tvo]TVN0819:22.939):59.615,[Hbs]VNG2283G:39
.2639):9.408,[Afu]AF2255:39.1193):6.093):12.02,([Sso]SSO0341:35.0198,[Ape]APE2166:38.0749):29.
554):68.378,[Dra]DR2300:41.5789):9.191,(([Tma]TM1396:35.1468,([Bbu]BB0220:34.0333,[Tpa]TP1017:
32.4247):41.687):13.011,[Aae]aq_1293:38.5068):9.973):6.959,([Syn]sll0362:39.2296,[Mtu]Rv2555c:
43.5461):11.206):6.57,(((((Mpn]MPN418:17.901,[Mge]MG292:23.575):45.859,[Uur]UU369:36.2882):53.
782,([Spy]SPy1389:18.052,[Lla]L0343:17.736):25.488):10.001,([Sau]SA1446:31.9881,([Bha]BH1267:1
7.388,[Bsu]BS_alaS:18.866):13.513):9.308):18.857):10.041,(([Cje]Cj0506:26.778,[Hpy]HP1241:35.5
294):32.383,([Cpn]CPn0892:18.203,[Ctr]CT749:15.685):65.704):7.248):9.984,(([Mlo]mlr0032:25.982
,[Ccr]CC2529:30.966):10.962,[Rpr]RP856:39.9203):16.61):7.972,[Xfa]XF0124:37.9851):7.572,([Nme]
NMB1595:33.7316,[Pae]PA0903:24.94):6.649):5.826,[Buc]BU403:43.2224):5.042,([Pmu]PM1287:8.905,[
Hin]HI0814:9.417):17.461):5.115,[Vch]VC0545:24.598,[Eco]alaS:15.734)COG0013;
<...>


Цена вложения исходных деревьев : 258205


 * Оптимизация вложения выключена.

 * Начинаю анализ вложений.
```

```
 * Коэффициент стоимости заменён на 0.1.
 * Файл '_uc2/BadGenes.html' успешно открыт на запись.
 * Пишу список ППЦВ в файл "_uc2/BadGenes.html".
 * Файл '_uc2/Result.ph' успешно открыт на запись.
 * Пишу деревья со статистикой в файл "_uc2/Result.ph".

 * Non-Gain сценарий:
 1) Дерево дупликаций:
((((((Nme,(Xfa,(Pae,(Vch,((Eco,Buc),(Hin,Pmu))_11_)_83_)_11_)_20_)_82_,(Rpr,(Ccr,Mlo))_7_)_17_
,(Hpy,Cje))_12_,((Bbu,Tpa),(Cpn,Ctr))_2_)_26_,((((Sau,(Bsu,Bha)_1_)_10_,(Lla,Spy)_1_)_42_,(Uur
,(Mpn,Mge)))_8_,(Dra,(Mtu,Syn))_4_)_12_)_171_,((Ape,Sso),((Mth,(Mja,(Pho,Pab)))_9_,(Hbs,(Afu,(
Tac,Tvo)))_9_)_48_)_92_,(Aae,Tma))_887_;
 2) Дерево сумм дупликаций:
((((((Nme,(Xfa,(Pae,(Vch,((Eco,Buc),(Hin,Pmu))_11_)_94_)_105_)_125_)_207_,(Rpr,(Ccr,Mlo))_7_)_
231_,(Hpy,Cje))_243_,((Bbu,Tpa),(Cpn,Ctr))_2_)_271_,((((Sau,(Bsu,Bha)_1_)_11_,(Lla,Spy)_1_)_54
_,(Uur,(Mpn,Mge)))_62_,(Dra,(Mtu,Syn))_4_)_78_)_520_,((Ape,Sso),((Mth,(Mja,(Pho,Pab)))_9_,(Hbs
,(Afu,(Tac,Tvo)))_9_)_66_)_158_,(Aae,Tma))_1565_;
 3) Дерево сумм потерь:
((((((Nme,(Xfa,(Pae,(Vch,((Eco,Buc)_152_,(Hin,Pmu)_16_)_379_)_542_)_753_)_1019_)_1247_,(Rpr,(C
cr,Mlo)_68_)_227_)_1810_,(Hpy,Cje)_45_)_2330_,((Bbu,Tpa)_72_,(Cpn,Ctr)_10_)_308_)_3282_,((((Sa
u,(Bsu,Bha)_99_)_280_,(Lla,Spy)_40_)_538_,(Uur,(Mpn,Mge)_8_)_46_)_925_,(Dra,(Mtu,Syn)_232_)_52
9_)_2077_)_6321_,((Ape,Sso)_75_,((Mth,(Mja,(Pho,Pab)_17_)_173_)_370_,(Hbs,(Afu,(Tac,Tvo)_29_)_
216_)_441_)_1173_)_1604_,(Aae,Tma)_197_)_9009_;

 * Удаляю аномальные гены.
 * Аномальные гены удалены из ДдГ (78).

 * Gain сценарий:
 1) Дерево дупликаций:
((((((Nme,(Xfa,(Pae,(Vch,((Eco,Buc),(Hin,Pmu))_11_)_83_)_11_)_20_)_82_,(Rpr,(Ccr,Mlo))_7_)_13_
,(Hpy,Cje))_12_,((Bbu,Tpa),(Cpn,Ctr))_2_)_26_,((((Sau,(Bsu,Bha)_1_)_10_,(Lla,Spy)_1_)_42_,(Uur
,(Mpn,Mge)))_8_,(Dra,(Mtu,Syn))_5_)_12_)_139_,((Ape,Sso),((Mth,(Mja,(Pho,Pab)))_9_,(Hbs,(Afu,(
Tac,Tvo)))_9_)_47_)_92_,(Aae,Tma))_834_;
 2) Дерево сумм дупликаций:
((((((Nme,(Xfa,(Pae,(Vch,((Eco,Buc),(Hin,Pmu))_11_)_94_)_105_)_125_)_207_,(Rpr,(Ccr,Mlo))_7_)_
227_,(Hpy,Cje))_239_,((Bbu,Tpa),(Cpn,Ctr))_2_)_267_,((((Sau,(Bsu,Bha)_1_)_11_,(Lla,Spy)_1_)_54
_,(Uur,(Mpn,Mge)))_62_,(Dra,(Mtu,Syn))_5_)_79_)_485_,((Ape,Sso),((Mth,(Mja,(Pho,Pab)))_9_,(Hbs
,(Afu,(Tac,Tvo)))_9_)_65_)_157_,(Aae,Tma))_1476_;
 3) Дерево сумм потерь:
((((((Nme,(Xfa,(Pae,(Vch,((Eco,Buc)_144_,(Hin,Pmu)_15_)_361_)_515_)_715_)_970_)_1183_,(Rpr,(Cc
r,Mlo)_59_)_200_)_1681_,(Hpy,Cje)_42_)_2162_,((Bbu,Tpa)_61_,(Cpn,Ctr)_10_)_286_)_3036_,((((Sau
,(Bsu,Bha)_96_)_273_,(Lla,Spy)_38_)_522_,(Uur,(Mpn,Mge)_8_)_46_)_901_,(Dra,(Mtu,Syn)_208_)_467
_)_1944_)_5868_,((Ape,Sso)_74_,((Mth,(Mja,(Pho,Pab)_16_)_169_)_362_,(Hbs,(Afu,(Tac,Tvo)_29_)_2
14_)_433_)_1148_)_1567_,(Aae,Tma)_185_)_8454_;
 * Дерево приобретений:
((((((Nme,(Xfa,(Pae_2_,(Vch,((Eco_3_,Buc_5_)_8_,(Hin,Pmu_1_)_1_)_9_)_9_)_11_)_11_)_11_,(Rpr_5_
,(Ccr_1_,Mlo_4_)_5_)_10_)_21_,(Hpy,Cje_3_)_3_)_24_,((Bbu_5_,Tpa_2_)_7_,(Cpn,Ctr))_7_)_31_,((((
Sau_1_,(Bsu_2_,Bha_1_)_3_)_4_,(Lla,Spy))_4_,(Uur,(Mpn,Mge)))_4_,(Dra_7_,(Mtu_9_,Syn_9_)_18_)_2
5_)_29_)_60_,((Ape_1_,Sso)_1_,((Mth,(Mja,(Pho,Pab_1_)_1_)_1_)_1_,(Hbs_4_,(Afu_2_,(Tac,Tvo))_2_
)_6_)_7_)_8_,(Aae_6_,Tma_4_)_10_)_78_;
 ! Доля потерь на 1 аномальный ген: 7.1

 * Работа программы успешно завершена.

 * Программа работала < 2.81319 секунд.

*******************************************************************************
* TIQMax.log: Log-файл программы TIQMax.                                      *
*******************************************************************************
```

Рисунок 5. Пример (фрагментов) файла протокола.

5.5. Сообщения на консоль оператора

Сообщения на консоль оператора выдаются на русском языке и отображают текущий статус работающей программы (рис. 6).

```
TIQMax: Программа оптимизации и анализа вложения деревьев генов в дерево
видов.
Авторские права (C) 2000-2006 Лаборатория математических методов и моделей
в биоинформатике ИППИ РАН.
Версия 0.610 beta (Build May  3 2006 14:55:31).

Чтение данных... Ok.
Предобработка данных... Ok.
Вложение исходных деревьев... Ok.

Поиск генов, мешающих вложению... (7.1) Ok.

Работа программы успешно завершена.
Коллектив разработчиков благодарит Вас за её использование.
```

Рисунок 6. Пример выдачи на консоль оператора.

6. Приложение 3. Настройки программы

6.1. Управление выполнением программы

Параметры этой группы управляют общим ходом работой программы.

Имя параметра и номера использующих его функций	**Значение по умолчанию, тип**	**Описание и комментарии**
AnalyseEvolutionaryEvents (3)	false (булевский)	Включить **функцию 3**. Если параметр включён, программа будет выводить в выходной файл данные, описанные в разделе 5.2.
AutosavePeriod (1)	100 (целый)	Период автосохранения списка лучших деревьев видов. Если параметр не равен нулю, программа периодически сохраняет список лучших деревьев видов в файл результатов; сохранение производится после обработки группы деревьев видов, число которых равно значению параметра. Значение 0 понимается как "плюс бесконечность", т.е. автосохранение не производится, и результаты сохраняются в конце работы. Автосохранение удобно для просмотра текущих результатов в ходе работы программы, а также для минимизации ущерба от возможных программно-аппаратных сбоев во время работы. См. также *ResumeCalculations*.
InjectOnly (2,3)	false (булевский)	Выключить оптимизацию вложения. Если параметр включён, программа по окончании чтения входных данных переходит к анализу вложения (без проведения его оптимизации).
LookForBadGenes (2)	false (булевский)	Включить **функцию 2**. Если параметр включён, программа будет выводить в соответствующий выходной файл различную статистическую информацию о качестве КОГов и отдельных генов относительно финального вложения.
NumberOfRandomSpeciesTrees (1)	0 (целый)	Число генерируемых начальных деревьев видов (0 – работать с заданным деревом видов); отличное от нуля значение включает **функцию 1**. Этот параметр управляет генерацией деревьев видов. Если он равен нулю, случайные деревья видов не генерируются, вместо этого используется дерево видов из файла. Если значение параметра отлично от нуля, то программа производит *NumberOfRandomSpeciesTrees* повторений цикла, в котором генерируется случайное дерево видов, оптимизируется и добавляется в список лучших деревьев видов.

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
NumberOfResultSpeciesTrees (1)	0 (целый)	Число выводимых в файл результатов деревьев видов (0 - неограниченное). Этот параметр ограничивает число результирующих деревьев видов, хранимых в памяти во время работы программы и выводимых в файл результатов. Сохраняется "верхняя" часть списка, т.е. деревья видов с худшим качеством вытесняются лучшими. Значение 0 понимается как "плюс бесконечность", т.е. сохраняется и выводится весь список результирующих деревьев видов.
ResumeCalculations (1)	false (булевский)	Использовать результаты предыдущего запуска программы. Если этот параметр включен, и программа используется в режиме функции 1, то перед порождением новых деревьев видов программа попытается считать результаты своей работы во время предыдущего запуска из файла результатов и (в случае успешного чтения) добавит их в список лучших деревьев видов.

6.2. Имена используемых входных/выходных файлов

Эта группа установок содержит имена входных/выходных файлов программы. Под именем файла понимается полное (включающее путь) или относительное (относительно текущей директории) имя файла, доступного на чтение в случае входного файла / на запись или создание в случае выходного файла. (Если для выходного файла указывается имя существующего файла, доступного на запись, то старое содержимое файла теряется.) Допустимость имени файла определяется соглашениями, принятыми в конкретной операционной системе и не проверяется самой программой.

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
BadGenesFileName (2)	BadGenes.txt (строка)	Имя выходного файла функции 2. В этот файл выводится различная статистическая информация о вложениях, вкладе отдельных генов в цену вложений и т.п. Файл используется только в случае *LookForBadGenes* = true. См. также: *LookForBadGenes*.
GenesTableFileName (1,2,3)	GTable (строка)	Имя входного файла, содержащего таблицу принадлежности генов видам. Из этого файла программа узнаёт, из какого вида выделен каждый ген. Эти сведения необходимы для осуществления вложения листьев деревьев генов в соответствующие листья дерева видов. См. также: *CheckGeneTrees*.
GeneTreesFileName (1,2,3)	GTrees.gt (строка)	Имя файла, содержащего набор деревьев генов. Из этого файла программа читает набор деревьев генов – основной материал исследования.
LogFileName (1,2,3)	TIQMax.log (строка)	Имя файла протокола. В этот файл записывается дополнительная и служебная информация о деталях хода работы программы. См. также: *LogAllowed*.
ResultFileName (1,3)	Result.ph (строка)	Имя файла для вывода результатов работы (оптимизированных деревьев видов или статистики эволюционных событий). Это основной файл результатов. В зависимости от решаемой задачи, сюда может помещаться либо набор лучших (по цене вложения) деревьев видов (в функции 1), полученных

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
		оптимизацией сгенерированных деревьев видов, либо статистика эволюционных событий, отнесённых к вершинам дерева видов (в функции 3). Этот файл также используется для чтения из него результатов работы программы во время предыдущего запуска (если *ResumeCalculations* = true).
SpeciesTreeFileName (2,3)	STree.st (строка)	Имя файла, содержащего исходное дерево видов. Из этого файла программа считывает исходное дерево видов, используемое для произведения анализа деревьев генов (относительно заданного дерева видов) или дерева видов (относительно заданных деревьев генов).

6.3. Настройка алгоритма оптимизации и вложения

Параметры этой группы служат для настройки алгоритмов оптимизации дерева видов и вложения. В основном они носят экспериментальный характер.

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
CostRatio (1)	1.0 (вещественный)	Отношение λ цены пропуска к цене дупликации. Цена вложения складывается из цен пропусков и цен дупликаций с некоторыми коэффициентами "значимости". Данный параметр выражает отношение "веса" одного пропуска к "весу" одной дупликации при подсчёте цены вложения. См. также: *CostRatio2*.
LocalOptimizationsOrder (1)	0 (целый)	Порядок обхода поддеревьев дерева видов при проведении локальных оптимизаций. Параметр может принимать одно из следующих значений: 0 - прямой рекурсивный обход, 1 – концевой рекурсивный обход, 2 - обход "по ярусам" сверху вниз, 3 - обход "по ярусам" снизу вверх, 4 - движение в направлении максимального улучшения цены вложения. Данный параметр введён в экспериментальных целях и, как показала практика, не оказывает существенного влияния на результаты оптимизации.
RandomNumberSeed (1)	0 (целый)	Инициализирующее значение для датчика случайных чисел (0 – использовать таймер). Если параметру задано ненулевое значение, внутренний датчик псевдослучайных чисел будет генерировать при каждом запуске повторяющуюся последовательность (зависящую от указанного значения), что позволяет воспроизводить однажды проделанные вычисления. Если задано значение 0 – алгоритм использует для построения начального значения системный таймер, что позволяет получать при каждом запуске разные последовательности и, как следствия, разные наборы начальных деревьев видов.
UseEmpiricalProbability (1)	true (булевский)	Использовать при построении случайных деревьев видов эмпирические вероятности соседства листьев. Если параметр включён, при построении начальных деревьев видов используется процедура, описанная в разделе 3.2.2, в противном случае деревья строятся случайным образом без учёта статистической близости видов в наборе деревьев белковых семейств.

6.4. Проверка, пред- и пост-обработка данных

Параметры этой группы управляют первичной обработкой данных перед началом применения основных алгоритмов, а также обработкой полученных в результате работы программы результатов.

Имя параметра и номера использующих его функций	**Значение по умолчанию, тип**	**Описание и комментарии**
CheckGeneTrees (1,2,3)	true (булевский)	Проверять деревья генов на наличие неиспользуемых генов (и удалять последние). Если этот параметр включён, программа после считывания исходных данных проверит, все ли гены в деревьях генов имеют информацию о себе в таблице принадлежности генов видам и соответствующий вид в исходном дереве видов (если последнее задано); гены, не удовлетворяющие данному условию, удаляются из деревьев генов (с соблюдением правил, позволяющих сохранить осмысленность обработанного таким образом дерева). Если в каком-нибудь дереве генов в результате удаления осталось не более двух генов, это дерево исключается из рассмотрения и в дальнейшей работе программы не участвует. Если параметр отключен, проверка не производится и в случае существования генов с неопределённой принадлежностью при выполнении вложения работа программы аварийно завершается с сообщением об ошибке.
DefineLengthsOnSpeciesTree (1)	false (булевский)	Приписывать длины рёбрам результирующих деревьев видов по результатам вложения. Включает процедуру, снабжающую результирующие деревья видов длинами рёбер, отражающими степень "поддержки" отдельных его ветвей ветвями деревьев генов.
GenesToExclude (1,2,3)	*[пустая строка]* (строка)	Список генов, исключаемых из рассмотрения. Данный параметр позволяет задать список генов, которые нужно исключить из рассмотрения. Такие гены не будут прочитаны из таблицы принадлежности генов видам и в дальнейшем будут удалены из деревьев генов процедурой проверки последних (см. *CheckGeneTrees*). Список оформляется в виде последовательности имён генов, заключённых в кавычки (") и (возможно) разделённых запятыми.
GeneTreeNormFactor (1,2,3)	0 (вещественный)	Коэффициент μ нормировки длин ветвей деревьев генов. Параметр задаёт коэффициент μ, используемый при нормализации длин деревьев генов методом, описанным в разделе 3.2.2.
IgnoreLengths (1,2,3)	false (булевский)	Не использовать длины ветвей деревьев генов. Если параметр включён, программа после чтения исходных данных установит длины всех ветвей деревьев генов равными единице. В результате, при расчётах будет учтена только топология деревьев генов без учёта длин их ветвей.
MergeOneSpeciesSubtrees (1,2,3)	false (булевский)	Объединять в листы поддеревья деревьев генов, отображающиеся в листы дерева видов. Для исходных данных нередки случаи, когда небольшое поддерево дерева генов образовано исключительно генами из одного вида; это не мешает выполнению программы, но несколько замедляет его и усложняет визуальное восприятие результатов работы программы. В этой связи можно использовать данный параметр для замены подобных поддеревьев на новые листья, имена которых складываются из имён листьев поддерева. Такая замена никак не сказывается на цене вложения, и, значит, на результатах оптимизации, но, разумеется, отражается на списке аномально расположенных генов.

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
OrderSpeciesTreesBranches (1)	true (булевский)	Упорядочивать ветви результирующих деревьев видов в соответствии с именами видов. Если параметр включён, каждое новое полученное оптимизацией дерево видов будет подвергаться упорядочиванию ветвей, гарантирующему, что в файле результатов одинаковые (с точностью до перестановки поддеревьев-"братьев") деревья видов будут выглядеть одинаково. Эта процедура законна, т.к. порядок потомков не несёт никакой информации по смыслу задачи.
ProduceGeneTreesFiles (0)	false (булевский)	Вывести после предобработки деревья генов в файлы (по одному дереву в файл).

6.5. Анализ генов и КОГов

Параметры этой группы управляют подсчётом и выводом статистических данных, позволяющих анализировать качество КОГов и отдельных генов в них.

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
CostDeltaPValueThreshold (2,3)	0.1 (вещественный)	Верхний порог p-value F_g для аномальных генов. См. также: *LookForBadGenes*.
CostDeltaThreshold (2,3)	-7 (вещественный)	Верхний порог F_g для аномальных генов. См. также: *LookForBadGenes*.
CostRatio2 (2,3)	0.1 (вещественный)	Отношение цены пропуска к цене дупликации при составлении списка приращений цены вложения. В связи с некоторыми соображениями теоретического характера, желательно иметь возможность использовать отдельное значение отношения цены пропуска к цене дупликации (см. *CostRatio*) в функциях 2, 3. Данный параметр позволяет установить значение *CostRatio*, используемое во втором случае. См. также: *CostRatio*.
DispersionThreshold (2,3)	2 (вещественный)	Верхний порог R_g для аномальных генов. См. также: *LookForBadGenes*.
DispersionPValueThreshold (2,3)	0.1 (вещественный)	Верхний порог p-value R_g для аномальных генов. См. также: *LookForBadGenes*.
NeighbourhoodRadius (2,3)	5 (целый)	Радиус окрестности при исследовании "рассеивания" генов. Задаёт радиус окрестности гена (на дереве генов), используемой при исследовании "рассеивания" R_g генов при вложении, т.е. увеличения попарного расстояния (в топологии соответствующих деревьев) между данным геном и генами из его окрестности. См. также: *LookForBadGenes*.
SimpleInjectionAnalysisOutput (3)	false (булевский)	Упрощённый вывод результатов анализа вложения. Если параметр включён, программа использует при выводе результатов анализа вложения (функция 3) формат, описанный выше в разделе 0, в противном случае выводится намного более подробная и объёмистая информация. См. также: *LookForBadGenes*.

6.6. Настройки интерфейса

Параметры этой группы управляют деталями пользовательского интерфейса программы.

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
BeepOnNewMinimum (1)	false (булевский)	Выдавать звуковой сигнал при достижении нового минимума цены вложения. Если параметр включён, программа будет выводить в устройство *stderr* символ с ASCII-кодом 7 каждый раз, когда достигнут новый минимум цены вложения, что должно вызывать воспроизведение звукового сигнала (тип которого зависит от используемой операционной системы и её настроек).
ShowProgressBar (1)	true (булевский)	Выводить индикатор процесса оптимизации. Выводит на экран во время оптимизации дерева видов полоску из псевдографических символов, каждый символ которой отображает окончание выполнения одного эффективного обхода дерева видов (т.е. обхода, в результате которого качество вложения улучшилось).

6.7. Общие установки вывода

Параметры этой группы включают/выключают разные варианты отображения данных в выходных файлах. Они предназначены для самостоятельного анализа вложений пользователем и не используются ни в одной из описываемых в данном документе функций.

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
NormalizeGeneTrees (0)	false (булевский)	Делить длины рёбер исходных деревьев генов на число генов в них при подсчёте статистики вложения. Задумывалось для уравнивания вкладов различных по размерам деревьев генов в цену вложения. Имеет значение только при подсчёте статистики вложения и никак не сказывается на результатах работы программы.
ShowInjections (0)	true (булевский)	Выводить вложения (и информацию по индивидуальным вложениям).
ShowInjectionsInfo (0)	true (булевский)	Выводить цены индивидуальных вложений для каждого дерева генов.
ShowInjectionStatistics (0)	true (булевский)	Выводить статистику по вложению в целом. Отображает среднюю цену вложения отдельных деревьев и нормальное отклонение (и то же самое отдельно для пропусков и дупликаций).
ShowInjectionTables (0)	true (булевский)	Выводить вложения в табличном виде. Каждая строка таблицы соответствует одной вершине дерева генов и отображает имя (или LR-последовательность) для данной вершины дерева генов, имя (или LR-последовательность) вершины дерева видов, в которую она отображается, а также цену пропусков и дупликаций для этой вершины.
ShowInjectionTrees (0)	true (булевский)	Выводить вложения в виде деревьев. Отображает вложения в виде деревьев генов, каждой вершине которых приписано обозначение вершины дерева видов, в которую она отображается.
ShowPhylipTrees (0)	true (булевский)	Выводить деревья в PHYLIP-формате. Этот формат скобочной записи деревьев является распространённым стандартом хранения деревьев и может быть прочитан многими программами.
ShowPseudographicsTrees (0)	true (булевский)	Выводить деревья в псевдографическом формате. Этот способ записи деревьев является более удобным для визуального восприятия (но требует некоторой привычки).

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
ShowTreeInfo (0)	true (булевский)	Выводить статистическую информацию о деревьях (число вершин и т.п.). Малополезен и может быть удалён из следующих версий программы.
SortInjections (0)	true (булевский)	Сортировать индивидуальные вложения по цене. При выводе вложений (или информации о них) обычно сохраняется порядок, в котором деревья генов записаны во входном файле. Данный параметр позволяет изменить этот порядок, выводя вложения в порядке возрастания их цены.

6.8. Управление выводом файла протокола

Параметры этой группы включают/выключают вывод различной дополнительной информации в файл протокола.

Имя параметра и номера использующих его функций	Значение по умолчанию, тип	Описание и комментарии
LogAllowed (1,2,3)	false (булевский)	Разрешить ведение файла протокола. Разрешает/запрещает ведение файла протокола. Запрет ведения протокола позволяет сократить время работы программы не только за счёт уменьшения дисковых операций, но также и за счёт отказа от выполнения некоторых дополнительных вычислений (например, выполнения дополнительных вложений).
LogOutAbsentGenes (1,2,3)	true (булевский)	Выводить имена генов, отсутствующих в дереве видов. Заставляет программу отображать в протоколе список генов с неизвестной видовой принадлежностью, удалённых из деревьев генов. См. также: *CheckGeneTrees*, *LogAllowed*.
LogOutCutGeneTrees (1,2,3)	true (булевский)	Выводить "урезанные" деревья генов (без "лишних" генов). Выводит в файл протокола деревья генов, полученные удалением из исходных деревьев генов, информация о которых отсутствует в таблице принадлежности генов видам. Полученный таким образом список деревьев генов можно использовать в качестве списка деревьев генов, подаваемого на вход программы. См. также: *CheckGeneTrees*, *LogAllowed*.
LogOutInitialInjections (0)	false (булевский)	Выводить исходные вложения (для каждого сгенерированного дерева видов). Можно использовать для заполнения значительного объёма дискового пространства информацией, которая (ввиду своего объёма) физически не может быть прочтена одним человеком. Кандидат на удаление в следующих версиях программы. См. также: *LogAllowed*.
LogOutInitialTrees (1,2,3)	true (булевский)	Выводить исходные деревья (для проверки и визуализации). Определяет, выводить ли прочитанные из файлов деревья в файл протокола. См. также: *LogAllowed*.
LogOutOptimizedInjections (0)	false (булевский)	Выводить результирующие вложения (для каждого сгенерированного дерева видов). Для этого параметра справедливо всё, сказанное в комментарии к *LogOutInitialInjections*. См. также: *LogAllowed*.
LogOutPreparedTrees (1,2,3)	false (булевский)	Выводить предобработанные деревья генов. Вывод в файл протокола деревьев генов, прошедших процедуру предварительной обработки (удаление "лишних" генов, "навешивание" видов и т.п.). См. также: *CheckGeneTrees*, *LogAllowed*.

Имя параметра и номера использующих его функций	**Значение по умолчанию, тип**	**Описание и комментарии**
LogOutProgramSettings (1,2,3)	true (булевский)	Выводить глобальные установки программы. Разрешает вывод в начале файла протокола списка глобальных установок программы с их текущими значениями. См. также: *LogAllowed*.
LogOutProximityMatrix (1)	false (булевский)	Выводить матрицу статистической близости видов. Разрешает вывод в файл протокола матрицы распределения эмпирических вероятностей близости видов на деревьях генов. См. также: *LogAllowed*, *UseEmpiricalProbability*.
LogOutRandomSpeciesTrees (1)	true (булевский)	Выводить сгенерированные исходные деревья видов. Не представляет интереса и, возможно, будет удалён из следующих версий программы. См. также: *LogAllowed*.
WarnAmbiguousTransitions (0)	false (булевский)	Предупреждать о сомнительных переходах при локальных оптимизациях. При выборе локального устройства дерева видов в окрестности одной из его вершин возможны случаи, когда несколько вариантов дают одну и ту же цену вложения (при отказе от использования длин рёбер такие ситуации становятся типичными). В данной версии программы выбирается "первый попавшийся" из "равноценных" вариантов, что явно не является оптимальным решением. Данный экспериментальный параметр позволяет оценить частоту появления описанных ситуаций (эта частота сильно зависит от входных данных).