# Построение полного решающего дерева с использованием гетерогенной системы на основе технологии CUDA*

***И. Е. Генрихов***
ingvar1485@rambler.ru
«Мобайл парк ИТ», г. Химки, ул. Панфилова, 21/1

Статья посвящена исследованию алгоритмов классификации на основе полных решающих деревьев (ПРД). Рассматриваемая конструкция решающего дерева (РД) позволяет в каждой специальной вершине дерева учитывать все признаки, удовлетворяющие критерию ветвления. Основным недостатком ПРД является существенно большее время синтеза дерева по сравнению с классическим РД. Рассмотрены вопросы снижения времени построения ПРД с использованием технологии CUDA (Compute Unified Device Architecture). Данная технология позволяет использовать большое число ядер графического процессора для ускорения выполнения сложных вычислений. Приведены результаты тестирования на модельных и реальных задачах. Показано, что применение технологии CUDA позволяет заметно снизить время синтеза ПРД (более чем в 10 раз) только в тех случаях, когда обучающая выборка содержит большое число признаков и/или обучающих объектов и при этом информация либо вещественнозначная, либо целочисленная большой значности.

**Ключевые слова**: *задача распознавания по прецедентам; полное решающее дерево; CUDA; гетерогенная система*

**DOI:** 10.21469/22233792.2.1.05

## 1 Введение

Одним из известных инструментов для решения задач обучения по прецедентам [1] являются РД. Процедура построения классического РД представляет собой итерационный процесс, на каждом шаге которого для построения очередной внутренней вершины РД выбирается признак, наилучшим образом удовлетворяющий некоторому критерию ветвления. По каждой ветви, исходящей из построенной внутренней вершины, осуществляется спуск и строится либо лист дерева, либо новая внутренняя вершина. Каждому листу в РД приписан один из классов и, как правило, в листе содержится вся информация, позволяющая сделать вывод о принадлежности распознаваемого объекта классу, который приписан данному листу.

Очевидным недостатком классической модели РД является то, что на очередной итерации для построения внутренней вершины среди всех признаков, удовлетворяющих выбранному критерию ветвления в равной или почти равной мере, выбирается только один признак (и выбирается этот признак фактически случайным образом). При этом в зависимости от выбранного признака построенные деревья могут существенно отличаться по своим распознающим качествам. Для решения данной проблемы в [2] предложен новый подход к синтезу РД. При возникновении ситуации, когда два или более признака удовлетворяют критерию ветвления в равной или почти равной мере, предлагается проводить ветвление по каждому из этих признаков независимо. Полученная в результате конструкция названа ПРД.

*Работа выполнена при финансовой поддержке РФФИ, проект № 13-01-00787.

Таким образом, в отличие от классического РД в ПРД на каждой итерации строится специальная вершина, называемая полной вершиной, которой соответствует набор признаков $X$. Далее по аналогии с классическим РД проводится ветвление по каждому из признаков, входящих в $X$. Конструкция ПРД позволяет более существенно использовать имеющуюся информацию, при этом описание распознаваемого объекта может порождаться не одной ветвью, как в классическом РД, а несколькими ветвями. Каждая такая ветвь участвует в процедуре голосования (является голосующей).

Первоначальная модель ПРД [2] предназначалась для обработки целочисленной информации, при этом использовались логические критерии ветвления и самый простой вид коллективного голосования (голосования по большинству). Более совершенные модели ПРД с энтропийным критерием ветвления, использующие взвешенное голосование по голосующим ветвям дерева, были построены и исследованы в [3, 4]. На их основе были построены алгоритмы синтеза ПРД для обработки вещественнозначной информации с наличием пропусков в признаковых описаниях объектов и неравномерного распределения объектов по классам в обучающей выборке (в этом случае можно указать пару классов таких, что число обучающих объектов в одном из них существенно больше числа обучающих объектов в другом). В [3, 4] получены теоретические и экспериментальные оценки, характеризующие высокую обобщающую способность ПРД по сравнению с обобщающей способностью классического РД. Счет на реальных задачах показал, что классификаторы на основе ПРД не уступают по качеству другим современным классификаторам на основе РД, например таким, как «бустинг» над РД и «баггинг» над РД, а иногда показывают и более высокое качество.

Основным недостатком ПРД является существенно большее время синтеза дерева по сравнению с классическим РД. Теоретические оценки времени синтеза ПРД в худшем и в некоторых важных частных случаях были получены в [3]. Поэтому актуальной задачей является снижение временно́й сложности построения ПРД.

В данной работе рассмотрены вопросы снижения времени синтеза ПРД с применением параллельных вычислений на основе технологии CUDA — программно-аппаратная архитектура унифицированных вычислений [5, 6] от компании NVIDIA. Указанная технология позволяет выполнить операции, не требующие длительного времени, на центральном процессоре (CPU) компьютера, а все сложные операции (в вычислительном плане) — на графическом процессоре компьютера (GPU). При таком способе реализации алгоритма принято говорить о применении гетерогенной системы, так как используются ресурсы CPU и GPU для ее выполнения. Графический процессор состоит из однородных вычислительных элементов (мультипроцессоров) с общей памятью. Каждый мультипроцессор способен исполнять параллельно тысячи вычислительных «нитей». Нити могут быть сгруппированы в вычислительные потоки, имеющие общий кэш и быструю разделяемую память для обмена данными между нитями потока. Применение гетерогенных вычислений с использованием GPU наиболее эффективно при решении задач, обладающих параллелизмом по данным, число арифметических операций в которых велико по сравнению с операциями над памятью. Существует много таких задач в различных областях: обработка сигналов, физика, data mining, deep learning, machine learning, вычислительная биология, медицина, биоинформатика, вычислительная гидродинамика, компьютерное видение и работа с изображениями, медиа и развлечения, медицинская визуализация, молекулярная динамика, численный анализ, квантовая химия и др. Также основными достоинствами технологии CUDA является свободный доступ к программным инструментам, позволяющим

реализовать алгоритмы с применением технологии CUDA, и доступность графических ускорителей.

В разд. 2 введены основные понятия и описана общая схема построения ПРД.

В разд. 3 описаны разработанные алгоритмы синтеза ПРД с применением технологии CUDA.

В разд. 4 на модельных данных и на реальных задачах из репозитория UCI [7] и коллекции задач [1], собранной в отделе математических проблем распознавания и методов комбинаторного анализа Вычислительного центра им. А. А. Дородницына РАН Федерального исследовательского центра «Информатика и управление» РАН (ВЦ РАН), протестированы разработанные алгоритмы синтеза ПРД на основе гетерогенных вычислений и выявлены особенности их применения на различных типах задач по сравнению с алгоритмом синтеза ПРД, полностью исполняемом на CPU.

## 2 Основные понятия

Рассматривается задача распознавания по прецедентам с системой признаков $\{x_1, \ldots, x_n\}$, с непересекающимися классами $K_i,\ i \in I = \{1, \ldots, l\},$ и множеством обучающих объектов $T = \{S_1, \ldots, S_m\}$, где $S_r = (a_{r1}, \ldots, a_{rn}),\ a_{rj} \in \{\mathbb{R}, \text{«*»}\},\ r \in \{1, \ldots, m\},$ $j = 1, \ldots, n$. Если $a_{rj} =$«*», то значение признака $x_j$ для объекта $S_r$ не определено. Пусть далее $S = (b_1, \ldots, b_n)$ — распознаваемый объект и $b_j \in \{\mathbb{R}, \text{«*»}\},\ j = 1, \ldots, n.$

Опишем структуру ПРД. Пусть $\hat{T}$ — подмножество обучающих объектов и $X(\hat{T})$ — подмножество признаков, рассматриваемых на текущем шаге построения дерева. На первом шаге $\hat{T} = T,\ X(\hat{T}) = \{x_1, \ldots, x_n\}$.

При построении ПРД могут встречаться три типа вершин: висячие, полные и обычные вершины.

**Определение 1.** *Ветвью в ПРД называется путь, начинающийся в корне дерева и заканчивающийся в вершине ПРД.*

**Определение 2.** *Вершины ПРД, не имеющие выходящих дуг, называются висячими вершинами или листьями.*

**Определение 3.** *Обычной вершиной в ПРД называется внутренняя вершина ПРД, для которой выполняются следующие условия:*

1. *Данной вершине соответствует ровно один признак $x \in \{x_1, \ldots, x_n\}$.*
2. *В данную вершину входит одна дуга и выходит не менее двух дуг, помеченных разными числами.*
3. *Каждая дуга, выходящая из данной вершины, входит либо в висячую вершину, либо в полную вершину ПРД.*

**Определение 4.** *Полной вершиной в ПРД называется внутренняя вершина ПРД, для которой выполняются следующие условия:*

1. *Данной вершине $\nu$ соответствует набор различных признаков $X_\nu = \{x_{j_1}, \ldots, x_{j_q}\}, X_\nu \subseteq$ $\subseteq X, q \geqslant 1$.*
2. *В данную вершину $\nu$ входит одна дуга и выходит ровно $q$ дуг с метками, равными номерам признаков из $X_\nu$.*
3. *Каждая дуга с меткой $t, t \in \{j_1, \ldots, j_q\}$, выходящая из данной вершины, входит в обычную вершину, соответствующую признаку $x_t, x_t \in X_\nu$.*

**Определение 5.** *Глубиной ветви в ПРД называется число обычных вершин, которые содержит эта ветвь, исключая концевую вершину ветви.*

**Определение 6.** *Ярусом i-го уровня (i-м ярусом) в ПРД называется совокупность полных и обычных вершин, порожденных ветвями с глубиной $i-1$, а также совокупность листьев дерева, порожденных ветвями с глубиной $i$.*

На каждом шаге синтеза ПРД строится либо висячая вершина дерева, либо формируется набор из различных признаков $X_\nu, X_\nu \subseteq X$, образующий полную вершину $\nu$. Далее из полной вершины $\nu$ строится ровно $|X_\nu|$ дуг с метками $j_1, \ldots, j_q$. Дуга с меткой $t, t \in \{j_1, \ldots, j_q\}$, входит в обычную вершину, соответствующую признаку $x_t, x_t \in X_\nu$. При ветвлении из обычной вершины, соответствующей признаку $x_t$, происходит удаление признака $x_t$ из $X(\hat{T})$ и удаление некоторых объектов из $\hat{T}$.

В данной работе рассматривается задача распознавания с веществennозначными признаками, поэтому используется следующий способ ветвления из обычной вершины [3]. Для ветвления из обычной вершины, соответствующей признаку $x_t, x_t \in X(\hat{T})$, осуществляется бинарная перекодировка текущих значений признака $x_t$ с помощью «оптимального» порога $d(x_t)$. Рассматриваемая вершина помечается парой $(x_t, d(x_t))$. Спуск из вершины $(x_t, d(x_t))$ происходит по двум ветвям, при этом левая ветвь помечается 0, а правая — 1. При спуске из вершины $(x_t, d(x_t))$ по левой (правой) ветви происходит удаление признака $x_t$ из $X(\hat{T})$ и удаляются те объекты из $\hat{T}$, для которых значение признака $x_t$ больше (не больше) $d(x_t)$.

Рассмотрим решающее правило при классификации распознаваемого объекта $S$ с помощью ПРД.

Пусть $v$ — висячая вершина, ей может быть приписана пара $\left(B_v, \left\{\omega_v^1, \ldots, \omega_v^l\right\}\right)$ [3], где $B_v$ — элементарная конъюнкция (э.к.) над переменными $x_1, \ldots, x_n$; $\omega_v^i$ — оценка принадлежности объекта $S$ классу $K_i, i \in I$, вносимая вершиной $v$.

В данной работе используется следующий способ вычисления вектора оценок в висячей вершине $v$ [3]. Пусть $m_v^i$ — число объектов класса $K_i$, описание которых попадает в интервал истинности конъюнкции $B_v$; $m^i$ — число объектов класса $K_i$ в исходной обучающей выборке. Тогда $\omega_v^i = (m_v^i + 1)/(m^i + l), i \in I$.

**Замечание 1.** Причина применения указанного способа вычисления вектора оценок в висячей вершине $v$ заключается в том, что в этом случае повышается качество распознавания [3, 4].

Пусть висячая вершина $v$ порождена ветвью дерева с обычными вершинами $x_{j_1}, \ldots, x_{j_r}$ и $\sigma_i, i \in 1, \ldots, r,$ — метка дуги, выходящая из вершины $x_{j_i}$. Под э.к. $B_v$ для висячей вершины $v$ подразумевается конъюнкция вида $[x_{j_1} > d(x_{j_1})]^{\sigma_1} \cdots [x_{j_r} > d(x_{j_r})]^{\sigma_r}$, где $[x_{j_i} > d(x_{j_i})]^{\sigma_i} = 1$, если $x_{j_i} > d(x_{j_i})$ при $\sigma_i = 1$ или $x_{j_i} \leqslant d(x_{j_i})$ при $\sigma_i = 0$, иначе $[x_{j_i} > d(x_{j_i})]^{\sigma_i} = 0,\ i \in 1, \ldots, r$.

Под интервалом истинности $N_v$ э.к. $B_v$ будем понимать множество наборов вида $(\alpha_1, \ldots, \alpha_n)$, где $\alpha_{j_i} = \sigma_i$ при $i = 1, \ldots, r$, и $\alpha_j \in \{0, 1\}, j \notin \{j_1, \ldots, j_r\}$.

Описанием объекта $S = (b_1, \ldots, b_n)$ в вершине $v$ будем называть вектор $S(v) = (\beta_1, \ldots, \beta_n)$, в котором $\beta_{j_i} = 1$, если $b_{j_i} > d(x_{j_i})$, иначе $\beta_{j_i} = 0$ при $i = 1, \ldots, r$, и $\beta_j = 0$ при $j \notin \{j_1, \ldots, j_r\}$.

**Определение 7.** *Висячая вершина $v$ называется голосующей для $S$, если $S(v) \in N_v$.*

При синтезе ПРД, в отличие от классического РД, описание распознаваемого объекта $S$ может попасть в разные листья дерева, т. е. $S(v_1) \in N_{v_1}$ и $S(v_2) \in N_{v_2}$ при $v_1 \neq v_2$.

Пусть $Q(S)$ — множество всех голосующих висячих вершин для $S$. Для каждого $i \in I$ вычисляется оценка принадлежности объекта $S$ классу $K_i$, имеющая вид:

$$\Gamma(S, K_i) = \sum_{v \in Q(S)} \omega_v^i, \;\; i \in I.$$

Объект $S$ зачисляется в класс $K_i$, если $\Gamma(S, K_i) = \max\limits_{j \in I} \Gamma(S, K_j)$, $i \in I$, $\Gamma(S, K_i) \neq \neq \Gamma(S, K_j)$ при $i \neq j$, $j \in I$.

Если классов с максимальной оценкой несколько, то среди них выбирается только один, а именно тот, который имеет наибольшее число объектов в обучающей выборке, иначе происходит отказ алгоритма от классификации объекта $S$.

В случае вещественнозначной информации важной является задача нахождения такого порога $d(x_t)$, который наилучшим образом разделяет объекты из $\hat{T}$ по признаку $x_t$, принадлежащие разным классам. Опишем способ выбора порога для перекодировки текущих значений признака $x_t \in X(\hat{T})$, применяемый в данной работе.

Пусть $\{c_1, \ldots, c_u\}$, $u \leqslant m$, — множество различных значений по признаку $x_t$, $c_{i+1} > c_i$, $1 \leqslant i \leqslant u-1$. Пусть объекты $S_{i_1} = (a_{i_1 1}, \ldots, a_{i_1 n})$, $S_{i_2} = (a_{i_2 1}, \ldots, a_{i_2 n})$ из $\hat{T}$ принадлежат разным классам. Если $a_{i_1 t} = c_i$ и $a_{i_2 t} = c_{i+1}$, тогда число $k_{t_i} = (c_i + c_{i+1})/2$, $1 \leqslant i \leqslant u-1$, является порогом признака $x_t$.

Обозначим через $G_t = \{k_{t_1}, \ldots, k_{t_j}\}$ множество порогов признака $x_t$. Порог $k \in G_t$ разбивает множество $\hat{T}$ на два подмножества $\left\{T_k^{(1)}, T_k^{(2)}\right\}$, где $T_k^{(1)}$ $\left(T_k^{(2)}\right)$ состоит из объектов множества $\hat{T}$, для которых $a_{rt} \leqslant k$ $(a_{rt} > k)$, $r = 1, \ldots, m$.

Здесь применена идея корректного перекодирования вещественнозначной информации, предложенная Ю. И. Журавлевым и используемая при построении логических процедур распознавания для дискретизации исходной информации и понижения значности целочисленных данных [8]. Данный способ определения порога для признака $x_t \in X(\hat{T})$, позволяет сократить число порогов и делает эту процедуру более корректной по сравнению со способом определения порога для признака в алгоритме C4.5 [9].

Для каждого найденного порога признака $x_t \in X(\hat{T})$ определяется «информативность», и в качестве оптимального порога $d(x_t)$ берется тот порог, для которого эта информативность максимальна.

Опишем критерий выбора оптимального порога для признака $x_t \in X(\hat{T})$.

Обозначим через $f\left(K_i, \hat{T}\right)$, $i \in I$, число объектов из множества $\hat{T}$, относящихся к классу $K_i$, и через $R_t$ — множество объектов из $\hat{T}$, для которых значение признака $x_t$ не определено. Вероятность $P_t^i(\hat{T})$ того, что случайно выбранный объект из множества $\hat{T}$ будет принадлежать классу $K_i$, равна $f(K_t, \hat{T} \backslash R_t) / \left|\hat{T} \backslash R_t\right|$.

Величина, вычисляемая по формуле

$$\mathrm{Info}(\hat{T})_t = -\sum_{i=1}^{l} P_t^i(\hat{T}) \log_2 P_t^i(\hat{T}),$$

называется количеством информации (энтропией) по признаку $x_t$, необходимое для определения класса, которому принадлежит объект из множества $\hat{T}$.

Величина, вычисляемая по формуле

$$\mathrm{Info}(x_t)_k = \frac{\left|T_k^{(1)}\right|}{\left|\hat{T} \setminus R_t\right|} \mathrm{Info}\left(T_k^{(1)}\right)_t + \frac{\left|T_k^{(2)}\right|}{\left|\hat{T} \setminus R_t\right|} \mathrm{Info}\left(T_k^{(2)}\right)_t,$$

называется количеством информации, необходимым для определения класса, которому принадлежит объект из множества $\hat{T}$ после разбиения $\hat{T}$ по порогу $k$ признака $x_t$.

Информационный выигрыш (information gain) после выбора порога $k$ признака $x_t$ вычисляется по формуле $\mathrm{Gain}(x_t)_k = \mathrm{Info}(\hat{T})_t - \mathrm{Info}(x_t)_k$.

Величина, вычисляемая по формуле

$$\mathrm{SplitInfo}(x_t)_k = -\frac{\left|T_k^{(1)}\right|}{\left|\hat{T} \setminus R_t\right|} \log_2 \frac{\left|T_k^{(1)}\right|}{\left|\hat{T} \setminus R_t\right|} - \frac{\left|T_k^{(2)}\right|}{\left|\hat{T} \setminus R_t\right|} \log_2 \frac{\left|T_k^{(2)}\right|}{\left|\hat{T} \setminus R_t\right|},$$

определяет потенциальную информацию, получаемую при разбиении множества $\hat{T}$ по порогу $k$ признака $x_t$.

Оптимальным порогом в $G_t$ для признака $x_t$ считается порог $k$, для которого нормированный информационный выигрыш

$$\mathrm{GainRatio}(x_t)_k = \frac{\mathrm{Gain}(x_t)_k}{\mathrm{SplitInfo}(x_t)_k}$$

принимает свое наибольшее значение.

Таким образом, для каждого найденного текущего порога определяется информативность признака $x_t \in X(\hat{T})$ по описанному выше критерию и в качестве оптимального порога $d(x_t)$ берется тот порог, для которого эта информативность максимальна. Данная процедура повторяется для каждого признака из $X(\hat{T})$. Далее вызывается процедура выбора набора признаков для ветвления $X_\nu \in \{x_{j_1}, \ldots, x_{j_q}\}$, $X_\nu \subseteq X(\hat{T})$ (см. разд. 3) и осуществляется ветвление из полной вершины $\left(\left\{x_{j_1}, \ldots, x_{j_q}\right\}, \left\{d(x_{j_1}), \ldots, d(x_{j_q})\right\}\right)$.

**Замечание 2.** Описанная выше модификация энтропийного критерия была применена в работах [3, 4]. Отличие от аналогичного критерия, применяемого в алгоритме C4.5, заключается в используемой методике учета пропущенных данных в признаковых описаниях обучающих объектов при ветвлении из обычной вершины дерева. Различие методик учета пропусков описано ниже.

В описанных критериях при вычислении информативности разбиения текущего множества $\hat{T}$ по порогу $k$ признака $x_t$ пропущенные значения признака $x_t$ для объектов из $\hat{T}$ не принимаются во внимание. Если в описании обучающего объекта значение признака $x_t$ пропущено, то при ветвлении из обычной вершины, соответствующей признаку $x_t$, этот объект удаляется. Применяемая методика обработки пропусков направлена на сохранение исходной информации в полном объеме.

В алгоритме C4.5 применяется другая методика: предполагается, что пропущенные значения признака $x_t$ вероятностно распределены пропорционально частоте появления встречающихся значений. Поэтому в алгоритме C4.5 если в описании обучающего объекта значение признака $x_t$ пропущено, то такой объект не удаляется и при ветвлении из обычной вершины, соответствующей признаку $x_t$, его описание попадает и в левую, и в правую ветвь с определенными весами, которые учитываются при классификации. Использование методики алгоритма C4.5 вносит шум в обучающие данные. Если бы на месте пропущенного значения признака $x_t$ находилось какое-либо реальное число, полученное в процессе сбора данных, то оно могло бы существенно повлиять на выбор оптимального порога признака $x_t$, что могло бы изменить структуру и качество решающего дерева.

Описание других методик, используемых при решении задачи классификации с пропусками, представлено в работах [10, 11]. Большинство из них основано на замене

пропущенного значения одним из допустимых. Это значение может быть вычислено различными способами: как среднее по существующим значениям признака; как наиболее вероятное значение для признака; случайно выбрано из существующих значений; получено с помощью методов $k$-ближайших соседей, регрессионного или кластерного анализа. Также существует методика, основанная на удалении объектов с пропущенными значениями из обучающей выборки до начала построения дерева. Такой подход может применяться в случае, когда число объектов с пропущенными значениями невелико по сравнению с числом всех обучающих объектов. Недостаток данного подхода состоит в том, что теряется полезная информация, содержащаяся в удаленных объектах. Иногда применяется методика, заключающаяся в построении дополнительной ветви, выходящей из обычной вершины, соответствующей признаку $x_t$, в которую «попадают» все обучающие объекты, в описании которых значение признака $x_t$ не определено [12].

В случае если значение признака $x_t$ для распознаваемого объекта $S$ не определено, то признак $x_t$ исключается из исходного набора признаков. Далее строится ПРД для объекта $S$, т. е. при ветвлении из обычной вершины строится только та ветвь, по которой будет осуществлен «спуск» описания объекта $S$. Таким образом, при построении ПРД для классификации объекта $S$ учитываются только те признаки, для которых значения в $S$ определены. Данный способ учета пропусков в распознаваемом объекте был применен в [3, 4].

## 3 Алгоритмы синтеза полного решающего дерева с помощью гетерогенных вычислений

В данной работе в качестве базового алгоритма синтеза ПРД применяется алгоритм AGI.Bias [3].

Опишем алгоритм AGI.Bias.

Алгоритм AGI.Bias является рекурсивным. Пусть $T(a_{rj})$ — матрица, задаваемая обучающей выборкой $T$, где $r = 1, \ldots, m,\ j = 1, \ldots, n$, $a_{rj}$ — значение признака $x_j$ для обучающего объекта $S_r$. Обозначим через $\tilde{T}$ матрицу, рассматриваемую на текущем шаге алгоритма, $\tilde{X} = \{x_j \in X_T\}$ — множество всех признаков на текущем шаге. На первом шаге $\tilde{T} = T(a_{rj}),\ X_T = \{x_1, \ldots, x_n\}$. Шаг рекурсии в алгоритме AGI.Bias представляет собой последовательность действий 1–3, описанных ниже.

1. Просматриваются все столбцы матрицы $\tilde{T}$. Если в столбце нет хотя бы двух различных значений, то этот столбец вычеркивается из $\tilde{T}$ и признак, соответствующий данному столбцу, удаляется из $\tilde{X}$. Если $\tilde{X} = \varnothing$, то переходим к третьему действию, иначе осуществляется переход к следующему действию.
2. Для каждого признака $x_j \in \tilde{X}$ вычисляется значение критерия $\mathrm{GainRatio}(x_j)_k$. Если $G_j = \varnothing$, то $\mathrm{GainRatio}(x_j)_k = 0$. Если значение $\mathrm{GainRatio}(x_j)_k = 0$ для всех признаков из $\tilde{X}$, то переходим к третьему действию. Иначе вызывается процедура формирования набора признаков для ветвления, описанная ниже. Пусть в результате сформирован набор признаков $X_\nu = \{x_{j_1}, \ldots, x_{j_q}\}, 1 \leqslant q \leqslant n$. Создается полная вершина с меткой $\left(\{x_{j_1}, \ldots, x_{j_q}\}, \{d(x_{j_1}), \ldots, d(x_{j_q})\}\right)$. Далее осуществляется ветвление по каждому признаку $x_t \in X_\nu, t \in \{j_1, \ldots, j_q\}$.

   Для каждого признака $x_t \in X_\nu, t \in \{j_1, \ldots, j_q\}$, по оптимальному порогу $k = d(x_t)$ строятся две дуги, выходящие из вершины $(x_t, k)$. Если матрица $\tilde{T}$ состоит из одного столбца, то при построении подматриц $\tilde{T}_k^{(1)}$ и $\tilde{T}_k^{(2)}$ этот столбец не удаляется. Для левой (правой) дуги вершины $(x_t, k)$ строится подматрица $\tilde{T}_k^{(1)}$ ($\tilde{T}_k^{(2)}$) матрицы $\tilde{T}$, полученная удалением столбца, соответствующего признаку $x_t$, и строк $S_r$, в которых

$a_{rt} > k$ ($a_{rt} \leqslant k$), $r = 1, \ldots, m$. Если подматрица $\tilde{T}_k^{(1)}$ ($\tilde{T}_k^{(2)}$) содержит объекты одного класса или состоит из одного столбца, то переходим к третьему шагу, иначе $\tilde{T} = \tilde{T}_k^{(1)}$ ($\tilde{T} = \tilde{T}_k^{(2)}$), $\tilde{X} = \tilde{X} \backslash \{x_t\}$ и осуществляется рекурсивный переход к первому действию.

3. Пусть $\tilde{T}$ содержит $m_v^i$ объектов класса $K_i, i \in I$. Строится висячая вершина $v$ с меткой $\left(B_v, \{\omega_v^1, \ldots, \omega_v^l\}\right)$, $B_v$ — конъюнкция соответствующая данной вершине, $\omega_v^i = (m_v^i + 1)/(m^i + l)$, где $m^i$ — число объектов класса $K_i$ в исходной обучающей выборке, $i \in I$.

Для того чтобы построить ПРД для классификации объекта $S$ в случае наличия пропусков в описании объекта $S = (b_1, \ldots, b_n)$, достаточно положить на первом рекурсивном шаге $\tilde{X} = \{x_j \in X_T | b_j \neq «*»\}$.

**Замечание 3.** Для сокращения времени классификации объекта $S$ строятся только голосующие за $S$ листья ПРД [3]. Поэтому при спуске из обычной вершины $(x_t, d(x_t))$ если $b_t \leqslant d(x_t)$, то строится левая дуга, иначе строится правая дуга.

Процедура выбора набора признаков $X_\nu$ для ветвления представляет собой следующую последовательность шагов.

1. Пусть $\tilde{T}$ содержит $w$ столбцов, соответствующих признакам $x_{j_1}, \ldots, x_{j_w}$. Тогда $Y = \{x_{j_1}, \ldots, x_{j_w}\}$.
2. Вычисляется средний информационный выигрыш $q = \sum\limits_{i=1,\ldots,w} \text{GainRatio}(x_{j_i})_k / w$.
3. Определяется число признаков, для которых информационный выигрыш выше среднего или равен ему $n = \sum\limits_{i=1,\ldots,w} c_{j_i}$, где $c_{j_i} = 1$, если $\text{GainRatio}(x_{j_i})_k \geqslant q$, иначе $c_{j_i} = 0$. Признаки $x_{j_i}$, для которых $\text{GainRatio}(x_{j_i})_k < q$, $i = 1, \ldots, w$, удаляются из $Y$.
4. Вычисляется $h = \min\limits_{x_i \in Y} \text{GainRatio}(x_i)_k$.
5. Если $(q/n) + h \geqslant \max\limits_{x_i \in Y} \text{GainRatio}(x_i)_k$, то происходит выход из процедуры и возвращается итоговый набор признаков $Y$, иначе осуществляется переход к третьему шагу процедуры, положив $q := (q/n) + h$.

Смысл указанной процедуры формирования набора признаков $X_\nu$ для ветвления на текущем шаге синтеза дерева заключается в поиске признаков, информативность которых совпадает с максимальным значением информативности среди признаков из $Y$, а также признаки, информативность которых близка к максимальному значению информативности. Близость признаков из $Y$ по информативности вычисляется на основе среднего нормированного информационного выигрыша.

Таким образом, в процессе синтеза ПРД наибольшее время (в основном от 93%–99% от всего времени синтеза ПРД) (см. разд. 4) тратится на поиск оптимальных порогов для признаков на каждом шаге синтеза дерева, так как для каждого признака из $\tilde{X}$ требуется выделить все различные значения признака, далее отсортировать их, после чего найти возможные пороги для бинарной перекодировки, по описанному в разд. 2 модифицированному энтропийному критерию вычислить информативность для каждого найденного порога и выбрать порог с максимальной информативностью. Поэтому в приведенных ниже алгоритмах синтеза ПРД с использованием технологии CUDA указанные вычисления были полностью «перенесены» на GPU.

В первом из разработанных алгоритмов — PAGI.Bias (Parallel AGI.Bias) — реализована функция поиска оптимального порога для признака $x_t \in \tilde{X}$ на GPU — FindOptimalThreshold. На вход данной функции передаются: массив значений признака $x_t$, текущий массив меток обучающих объектов и начальная информативность при-

знака $x_t$ ($\mathrm{Info}(\hat{T})_t$ (см. разд. 2)). На выходе функции FindOptimalThreshold: оптимальный порог и значение максимальной информативности для признака $x_t$. При инициализации функции FindOptimalThreshold указывается величина $p$ — максимальное число вычислительных «нитей», которые могут быть использованы GPU для вычислений. В описанной функции последовательно выполняются следующие шаги:

1. Определяются все «уникальные» (различные) значения признака $x_t$. Данный блок выполняется параллельно $p$ вычислительными нитями GPU. Пусть $a_i^j$ — $i$-е значение признака $x_t$, которое проверяется на уникальность $j$-й нитью. Просматриваются все значения признака $x_t$, индекс которых меньше $i$, и если нет ни одного значения, равного $a_i^j$, то $a_i^j$ считается уникальным для признака $x_t$. После проверки на уникальность одного значения вычислительная нить переходит к проверке следующего значения. При этом каждая нить анализирует значения признака $x_t$ на уникальность, индексы которых во входном массиве не совпадают со значениями индексов, анализируемыми другими нитями GPU.
2. Сортируется массив уникальных значений признака $x_t$, полученный на шаге 1. В качестве метода сортировки используется вариант сортировки подсчетом, суть которой заключается в определении числа значений, которые меньше текущего значения. Данный блок выполняется параллельно $p$ вычислительными нитями. Пусть $c_i^j$ — $i$-е уникальное значение признака $x_t$, которое сортируется $j$-й нитью. Определяется $s_i^j$ — число уникальных значений признака $x_t$, меньшее $c_i^j$. Число $s_i^j$ однозначно определяет индекс значения $c_i^j$ в результирующем массиве. После определения индекса для одного уникального значения признака $x_t$ вычислительная нить переходит к поиску индекса для следующего уникального значения. Здесь так же, как и на первом шаге, каждая нить определяет индексы для набора уникальных значений признака $x_t$, который не пересекается ни с одним набором, анализируемым другими нитями.
3. Выделяются возможные пороги на основе отсортированного массива уникальных значений признака, полученного на шаге 2. Для проверки того, что полусумма двух соседних значений из упорядоченного множества уникальных значений является порогом, необходимо просмотреть метки обучающих объектов, в описании которых встречается одно из двух значений, и если найдутся хотя бы два таких объекта, принадлежащих разным классам, то данное число является порогом (см. разд. 2). Данный блок также выполняется $p$ вычислительными нитями GPU. Это значит, что каждая нить осуществляет описанную проверку только по одной полусумме соседних значений из входного массива для данного шага. Аналогично предыдущим шагам после проверки одной полусуммы нить переходит к проверке следующей полусуммы соседних значений. Наборы анализируемых значений полусумм для вычислительных нитей GPU не пересекаются между собой.
4. Вычисляются значения информативности для каждого порога, полученного на шаге 3. Данный блок также выполняется параллельно $p$ вычислительными нитями. Пусть $k_i^j$ — $i$-й порог признака $x_t$, для которого вычисляется информативность $j$-й нитью. Для порога $k_i^j$ нитью с индексом $j$ определяется значение модифицированного энтропийного критерия $\mathrm{GainRatio}(x_t)_{k_i^j}$ (см. разд. 2). После расчета информативности порога вычислительная нить переходит к вычислению информативности следующего порога из входного массива для данного шага. Наборы порогов, для которых вычисляется информативность нитями GPU, не пересекаются между собой.
5. Для признака $x_t$ вычисляется оптимальный порог $d(x_t)$ — порог с максимальной информативностью. Поиск оптимального порога осуществляется редукцией [5, 6] масси-

ва информативности порогов, полученного на шаге 4, с помощью $p$ вычислительных нитей.

Таким образом, из описания алгоритма PAGI.Bias следует, что реализованный способ вычислений на GPU «позволяет» с линейной сложностью отсортировать массив текущих значений признака, построить все пороги и вычислить их информативность (для входного массива значений признака из $m$ элементов, если число параллельно работающих нитей больше $m$), а затем с логарифмической сложностью осуществить поиск оптимального порога. Для лучшей загрузки ядер GPU на каждом шаге синтеза ПРД динамически определяется максимальное число вычислительных нитей GPU, которые могут быть задействованы при поиске оптимального порога для признака.

**Замечание 4.** В описанном выше алгоритме PAGI.Bias говорится, что блок вычислений на каждом шаге функции FindOptimalThreshold будет выполняться параллельно $p$ вычислительными нитями. Здесь следует понимать, что выполнение не подразумевает, что данный блок будет действительно выполнен параллельно $p$ нитями, так как следует учитывать программно-аппаратные ограничения максимального числа параллельно работающих нитей и особенности работы вычислительных нитей с учетом задержек, связанных с операциями доступа к глобальной памяти GPU [5, 6].

Второй разработанный алгоритм — Dynamic PAGI.Bias — отличается от алгоритма PAGI.Bias тем, что поиск оптимального порога осуществляется параллельно для всех признаков на текущем шаге синтеза ПРД. Для этого применяется динамический параллелизм (возможность динамически порождать новые вычислительные потоки без возврата к коду, исполняемом на CPU) [6]. Если на текущем шаге синтеза ПРД имеется $m$ признаков, то с помощью динамического параллелизма создается $m$ вычислительных потоков, где поток с индексом $i$ осуществляет поиск оптимального порога для признака $x_i$, т. е. внутри потока инициализируется и вызывается функция FindOptimalThreshold для соответствующего признака. Описанное отличие позволяет еще больше распараллелить вычисления и уменьшить временны́е издержки, связанные с необходимостью копировать данные между памятью GPU и оперативной памятью CPU. В алгоритме PAGI.Bias на одном шаге синтеза дерева при расчете оптимальных порогов для $m$ признаков и $n$ обучающих объектов требуется передать $4(mn + n + 2)$ байт и получить $8m$ байт данных. В алгоритме Dynamic PAGI.Bias требуется передать $4(2n + m + 2)$ байт и получить $8m$ байт данных.

Третий разработанный алгоритм — Deep Dynamic PAGI.Bias — отличается от алгоритма Dynamic PAGI.Bias тем, что динамический параллелизм применяется и на более низких вычислительных уровнях — при реализации блоков вычислений 1–4 функции FindOptimalThreshold. Например, в блоке поиска уникальных значений признака $x_t$ формируются вычислительные потоки, каждый из которых с помощью $z$ вычислительных нитей параллельно проверяет на уникальность не более $z$ значений признака, после анализа выбранного набора значений вычислительный поток прекращает свою работу. Отличие заключается в том, что описанные потоки независимы друг от друга (за исключением того, что один поток может начать свое выполнение раньше (если потоки попали в одну очередь потоков GPU) или чуть раньше другого потока (если потоки попали в разные очереди потоков GPU) [5, 6]), а в алгоритме Dynamic PAGI.Bias данный блок выполняется в одном вычислительном потоке, тем самым накладывая определенные ограничения на параллелизм вычислений нитями этого потока. Для лучшей загрузки мультипроцессоров GPU на каждом шаге синтеза ПРД в алгоритме Deep Dynamic PAGI.Bias динамически определяется $z$ — число вычислительных нитей GPU, которые могут быть

задействованы при работе каждого вычислительного потока, выполняемых в блоках 1–4 функции FindOptimalThreshold.

## 4 Результаты численного эксперимента

Исследование времени поиска оптимальных порогов для признаков в процессе построения ПРД алгоритмами AGI.Bias, PAGI.Bias, Dynamic PAGI.Bias и Deep Dynamic PAGI.Bias, описанных в разд. 3, осуществлялось на модельных данных. Каждая модель представляет собой один шаг синтеза ПРД указанными алгоритмами, на котором рассматриваются разные наборы значений по одному или нескольким признакам. Отличие моделей друг от друга заключается в распределении значений признака $x$ и в числе обучающих объектов. Число классов равно двум, число обучающих объектов каждого класса одинаково, т. е. $|K_1| = |K_2|$. Для каждой модели рассматривались варианты с $1, 2, 4, 6, 8, 12, 16$ и 20 признаками. При этом значения всех признаков модели совпадают, т. е. если в описании обучающего объекта значение признака $j$ равно 1, то и значение признака $k$ равно $1, \forall k \neq j$. Таким образом, исследуется влияние размерности и типа задачи на время расчета оптимальных порогов разработанными алгоритмами.

Опишем модели и полученные результаты.

Модель 1 — имеется 2500 объектов первого класса и 2500 объектов второго класса. Значения признака $x$ представляют собой массив последовательных значений от 0 до $m - 1$ включительно, где $m$ — общее число обучающих объектов. Метки обучающих объектов чередуются, т. е. первый объект имеет метку класса 1, второй объект — метку класса 2, третий объект — метку класса 1 и т. п.

Модель 2 и модель 3 аналогичны модели 1 за исключением того, что в этих случаях рассматривается ситуация, когда в первом классе и во втором классе по 250 объектов и 25 объектов соответственно.

Модели 1–3 позволяют исследовать время расчета оптимальных порогов для признаков алгоритмами, приведенных в разд. 3, в худшей ситуации (целочисленная информация большой значности). В этом случае для каждого признака с описанным распределением $m$ значений по $m$ обучающим объектам будет найдено $m$ различных значений, отсортировано ровно $m$ значений, будет определено $m - 1$ порогов, для каждого из найденного порога будет вычислена информативность по модифицированному энтропийному критерию (см. разд. 2), и будет найден оптимальный порог редукцией $m - 1$ значения информативности найденных порогов.

Модель 4 по сути является моделью 1, но только в этой модели значения признака для $i$-ого объекта равно $(i - 1) \mod 15$, т. е. первые 15 значений последовательно заполнены числами от 0 до 14 включительно, следующие 15 значений тоже последовательно заполнены числами от 0 до 14 включительно и т. п.

Модель 5 и модель 6 аналогичны модели 4, но только в этих случаях рассматривается ситуация, когда в первом классе и во втором классе по 250 объектов и 25 объектов соответственно.

Модели 4–6 позволяют проанализировать время поиска оптимальных порогов для признаков алгоритмами AGI.Bias, PAGI.Bias, Dynamic PAGI.Bias и Deep Dynamic PAGI.Bias в лучшей ситуации (целочисленная информация малой значности). В этом случае для каждого признака с таким распределением 15-ти значений по $m$ обучающим объектам будет найдено 15 различных значений, отсортировано ровно 15 значений, будет определено 14 порогов, для каждого из этих порогов будет вычислена информативность по модифицированному энтропийному критерию и будет найден оптимальный порог редукцией 14 значений информативности найденных порогов.

Значения времени поиска оптимальных порогов для всех рассматриваемых наборов признаков каждой модели и каждого алгоритма приведены на рис. 3. По оси абсцисс — 75-й процентиль времени поиска оптимальных порогов для данного набора признаков, по оси ординат — число признаков. Время указано в миллисекундах. Для расчета 75-го процентиля вычисление оптимальных порогов для каждого набора признаков каждой модели и каждого алгоритма осуществлялось 40 раз. На графиках введены обозначения: CPU — AGI.Bias, GPU 1 — PAGI.Bias, GPU 2 — Dynamic PAGI.Bias, GPU 3 — Deep Dynamic PAGI.Bias.

Из приведенных графиков на рис. 1, *г*–1, *е* видно, что в моделях с небольшим числом различных значений признака (модели 4–6) параллельные варианты алгоритма AGI.Bias показывают плохие результаты по сравнению с алгоритмом AGI.Bias, за исключением ситуации, когда имеется большое число признаков и большое число обучающих объектов — алгоритм Deep Dynamic PAGI.Bias показал сопоставимые с алгоритмом AGI.Bias результаты. Также следует отметить, что при увеличении размерности по числу обучающих объектов (при переходе от модели 5 к модели 4) скорость роста времени расчета оптимальных порогов алгоритмом AGI.Bias в несколько раз больше по сравнению с разработанными параллельными вариантами этого алгоритма (в среднем в 2,5 раза больше).

Сравнение параллельных алгоритмов между собой на графиках рис. 1, *г*–1, *е* показывает следующее.

1. Алгоритм PAGI.Bias медленнее (в среднем в 7,5 раза) алгоритмов Dynamic PAGI.Bias и Deep Dynamic PAGI.Bias за счет того, что во время расчета оптимальных порогов требуется передать больше данных в память GPU, и за счет того, что поиск оптимальных порогов осуществляется последовательно по одному признаку.
2. Алгоритм Dynamic PAGI.Bias быстрее (в среднем в 1,9 раза) алгоритма Deep Dynamic PAGI.Bias на моделях с небольшим и малым числом обучающих объектов (модели 5 и 6) за счет того, что алгоритму Deep Dynamic PAGI.Bias требуется больше данных (различных значений признака и порогов признака) для лучшей загрузки вычислительных потоков на более низких вычислительных уровнях (см. разд. 3), иначе время, затрачиваемое на инфраструктурные расходы, оказывается больше выигрыша по времени, получаемого от распараллеливания вычислений на этих уровнях.
3. Алгоритм Deep Dynamic PAGI.Bias быстрее (в среднем в 1,5 раза) алгоритма Dynamic PAGI.Bias на модели с большим числом обучающих объектов (модель 4).

Из графиков на рис. 1, *а*–1, *в* следует, что в моделях с большим числом обучающих объектов и с большим числом различных значений признака (модель 1) параллельные варианты алгоритма AGI.Bias показывают существенное ускорение расчета оптимальных порогов для признаков (до 20 раз) по сравнению с алгоритмом AGI.Bias. Если имеется меньшее число обучающих объектов с большим числом различных значений признака (модель 2), то параллельные алгоритмы показывают ускорение до 5 раз. Если осуществляется расчет оптимальных порогов для признаков с малым числом обучающих объектов и с большим числом различных значений признака (модель 3), то параллельные алгоритмы оказываются медленнее алгоритма AGI.Bias. Следует также отметить, что при увеличении размерности по числу обучающих объектов (при переходе от модели 2 к модели 1) скорость роста времени расчета оптимальных порогов алгоритмом AGI.Bias в несколько раз больше по сравнению с разработанными параллельными вариантами этого алгоритма (в среднем в 4 раза больше).

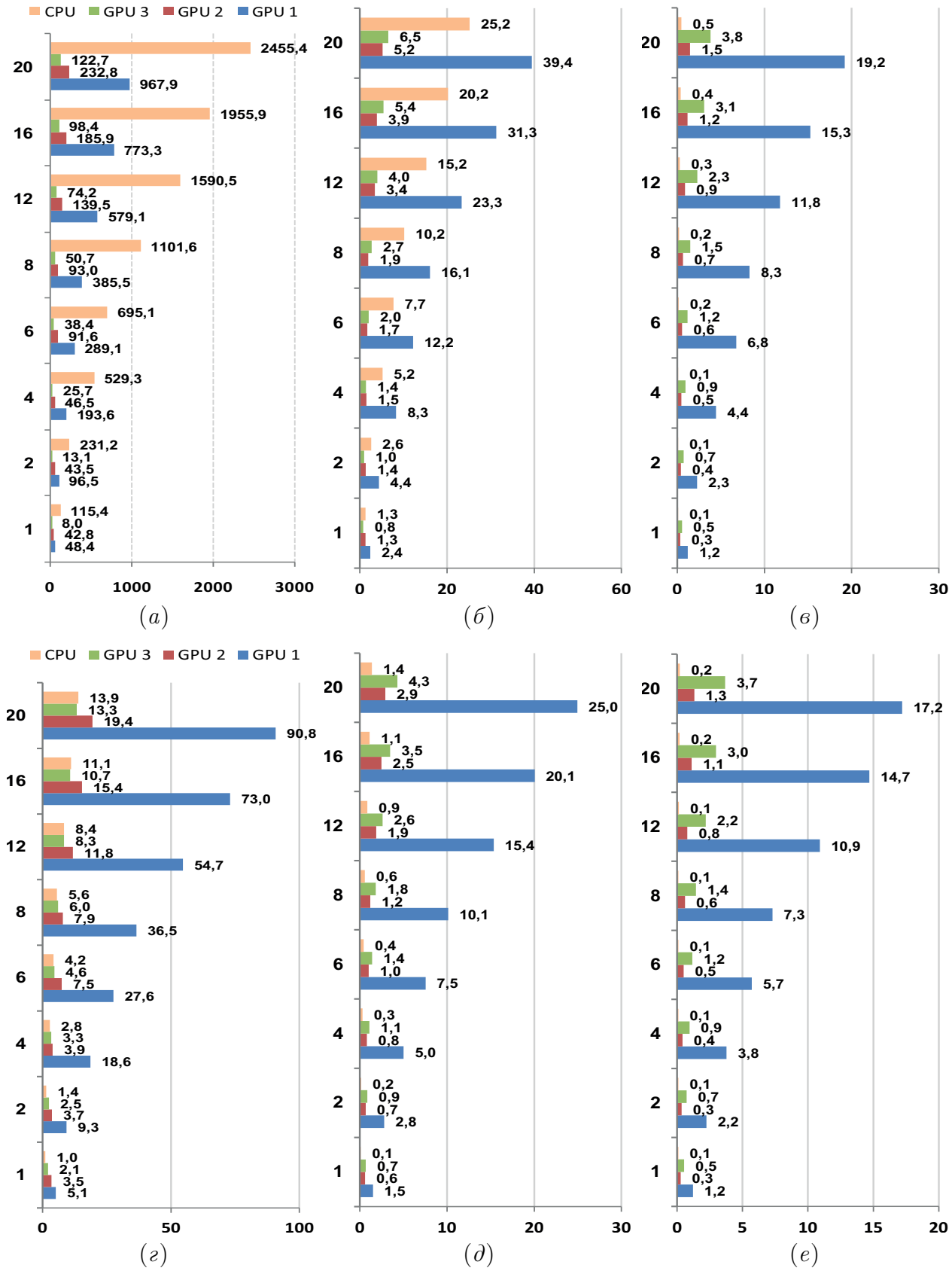


**Рис. 1** Изменение времени вычисления информативности признаков от числа признаков для различных алгоритмов в моделях 1 (*а*), 2 (*б*), 3 (*в*), 4 (*г*), 5 (*д*) и 6 (*е*)

Сравнение параллельных алгоритмов между собой на графиках рис. 1, *а*–1, *в* показывает следующее.

1. Алгоритм PAGI.Bias медленнее (в среднем в 8 раз) алгоритмов Dynamic PAGI.Bias и Deep Dynamic PAGI.Bias за счет того, что во время расчета оптимальных порогов требуется передать больше данных из памяти CPU в память GPU, и за счет того, что поиск оптимальных порогов осуществляется последовательно по одному признаку.
2. Алгоритм Dynamic PAGI.Bias быстрее алгоритма Deep Dynamic PAGI.Bias (в среднем в 1,6 раза) на моделях с небольшим и малым числом обучающих объектов (модели 2 и 3) за счет того, что алгоритму Deep Dynamic PAGI.Bias требуется больше данных (различных значений признака и порогов для признака), чтобы время, затрачиваемое на инфраструктурные расходы дополнительного распараллеливания вычислений на более низких вычислительных уровнях (см. разд. 3), было меньше выигрыша по времени, получаемого от этого дополнительного распараллеливания.
3. Алгоритм Deep Dynamic PAGI.Bias быстрее алгоритма Dynamic PAGI.Bias (в среднем в 2,6 раз) на модели с большим числом обучающих объектов (модель 1).

Таким образом, на модельных данных наилучшие результаты в случае малого числа обучающих объектов или признаков с малой значностью показывает алгоритм AGI.Bias, в случае большого числа обучающих объектов с большой значностью признаков наилучшие результаты показывает алгоритм Deep Dynamic PAGI.Bias (до 20 раз быстрее алгоритма AGI.Bias). В ситуации небольшого числа обучающих объектов с большой значностью признаков наиболее быстрым оказался алгоритм Dynamic PAGI.Bias (до 5 раз быстрее алгоритма AGI.Bias и до 1,4 раза быстрее алгоритма Deep Dynamic PAGI.Bias).

Исследование времени синтеза ПРД алгоритмами AGI.Bias, PAGI.Bias, Dynamic PAGI.Bias и Deep Dynamic PAGI.Bias, описанных в разд. 3, осуществлялось на 24 реальных задачах: 5 задач из репозитория ВЦ РАН [1] и 19 задач (Abalone (задача № 12), EEG Eye State (задача № 13), Adult (задача № 14), Bank Marketing (задача № 15), Image Segmentation (задача № 16), MAGIC Gamma Telescope (задача № 17), Statlog Image Segmentation(задача № 19), Seismic-Bumps (задача № 20), Statlog Shuttle (задача № 21), Wine Quality Red and White (задачи № 22 и № 23), LED Display (задача № 24), Wilt (задача № 11), Pima Indians Diabetes (задача № 8), Credit Approval (задача № 9), Banknote Authentication (задача № 7), Hepatitis (задача № 6), Glass Identification (задача № 5), Wine (задача № 3)) из репозитория UCI [7].

На задачах №№ 12–24 каждым алгоритмом полностью строилось по одному ПРД (без применения процедуры, описанной в замечании 3). На задачах №№ 1–11 рассматривалась процедура скользящего контроля («leave-one-out» (LOO)) [13]. Для этой процедуры один шаг алгоритма заключается в удалении одного из объектов обучающей выборки, построении классификатора (синтез ПРД с применением процедуры, описанной в замечании 3) и распознавании удаленного объекта.

Опишем полученные результаты.

В табл. 1 представлены результаты алгоритмов для задач №№ 1–11. Для алгоритма AGI.Bias и каждой задачи приведены две величины: общее время работы процедуры LOO и количество процентов от общего времени, затраченное на вычисление оптимальных порогов. Для параллельных версий алгоритма AGI.Bias в табл. 1 приведены значения величины

$$q = \begin{cases} -\dfrac{t_G}{t_C}, & \text{если } t_G > t_C; \\ \dfrac{t_C}{t_G}, & \text{если } t_C \geqslant t_G, \end{cases}$$

**Таблица 1** Эффективность алгоритмов по времени синтеза ПРД в процедуре LOO

| Описание задачи | | AGI.BIAS | PAGI.BIAS | Dynamic PAGI.BIAS | Deep Dynamic PAGI.Bias |
|---|---|---|---|---|---|
| № | $(\|K_1\|,\ldots,\|K_l\|,n)$ | | | | |
| 1 | $(48,23,8)*$ | 245,0 (92,91%) | −45,6 | −20,9 | −15,0 |
| 2 | $(47,30,7)$ | 54,2 (93,57%) | −37,2 | −11,1 | −8,2 |
| 3 | $(59,71,48,13)*$ | 2374,2 (97,94%) | −24,9 | −4,5 | −4,1 |
| 4 | $(51,218,24)$ | 3755,4 (86,99%) | −106,1 | −26,4 | −31,2 |
| 5 | $(70,76,17,13,9,29,9)*$ | 46915,6 (98,69%) | −9,3 | −4,6 | −2,2 |
| 6 | $(32,123,19)$ | 397,4 (93%) | −69,1 | −12,6 | −14,1 |
| 7 | $(762,610,4)*$ | 85248,1 (92,91%) | −2,2 | 1,3 | 15,2 |
| 8 | $(500,268,8)*$ | 58757,9 (99,88%) | −5,3 | −2,1 | 1,8 |
| 9 | $(307,383,15)$ | 12414,4 (96,3%) | −24,9 | −8,2 | −6,3 |
| 10 | $(11,47,15)*$ | 91,5 (95,79%) | −29,6 | −6,2 | −14,6 |
| 11 | $(74,4265,5)*$ | 3649297,0 (99,95%) | 1,8 | 5,0 | 14,3 |

**Таблица 2** Эффективность алгоритмов по времени синтеза одного ПРД

| Описание задачи | | AGI.BIAS | PAGI.BIAS | Dynamic PAGI.BIAS | Deep Dynamic PAGI.Bias |
|---|---|---|---|---|---|
| № | $(\|K_1\|,\ldots,\|K_l\|,n)$ | | | | |
| 12 | $(1528,1307,1342,8)*$ | 36488,6 (99,69%) | 1,2 | 2,0 | 5,4 |
| 13 | $(8257,6723,14)*$ | 3375061,3 (99,39%) | −1,2 | 1,6 | 3,2 |
| 14 | $(24720,7841,14)$ | 486958,7 (99,17%) | −1,8 | −1,5 | 7,6 |
| 15 | $(39922,5289,16)$ | 59926,0 (96,67%) | −4,3 | −3,0 | 1,3 |
| 16 | $(30,30,30,30,30,30,30,14)*$ | 77797,3 (87,81%) | −77,9 | −35,1 | −29,5 |
| 17 | $(12332,6688,10)*$ | 17322443,8 (99,97%) | 3,1 | 5,3 | 26,3 |
| 18 | $(96,104,963)*$ | 18398853,7 (99,97%) | 3,3 | 5,6 | 27,5 |
| 19 | $(330,330,330,330,330,330,330,19)*$ | 862045,5 (97,54%) | −12,1 | −6,1 | −3,5 |
| 20 | $(2414,170,18)$ | 2185,8 (92,83%) | −14,9 | −8,1 | −4,2 |
| 21 | $(34108,37,132,6748,2458,6,11,9)$ | 3609,9 (95,88%) | −10,6 | −4,0 | 1,6 |
| 22 | $(20,163,1457,2198,880,175,5,11)*$ | 5170,2 (99,69%) | −1,9 | 1,1 | 2,5 |
| 23 | $(10,53,681,638,199,18,11)*$ | 2650,1 (99,23%) | −2,4 | −1,1 | 1,5 |
| 24 | $(300,330,309,315,310,269,302,304,276,285,7)$ | 70,3 (98,95%) | −17,7 | −15,7 | −12,3 |

где $t_C$ — общее время процедуры LOO при использовании алгоритма AGI.Bias; $t_G$ — общее время процедуры LOO для соответствующего варианта параллельной версии алгоритма AGI.Bias. В табл. 2 представлены результаты алгоритмов для задач №№ 12–24. Смысл значений, приведенных в табл. 2, аналогичен смыслу значений табл. 1 за исключением того, что вместо процедуры LOO для задачи полностью строилось одно ПРД. Звездочкой помечены задачи, в которых большинство признаков с вещественнозначной или целочисленной информацией большой значности. Коричневым цветом выделены значения, когда достигается ускорение алгоритма PAGI.Bias относительно алгоритма AGI.Bias, бирюзовым — значения ускорения алгоритма Dynamic PAGI.Bias относительно AGI.Bias, зеленым — значения ускорения алгоритма Deep Dynamic PAGI.Bias относительно AGI.Bias.

На рис. 2 и 3 для некоторых задач приведены нормированные графики изменения времени расчета оптимальных порогов на каждом ярусе дерева при построении ПРД исследуемыми алгоритмами.

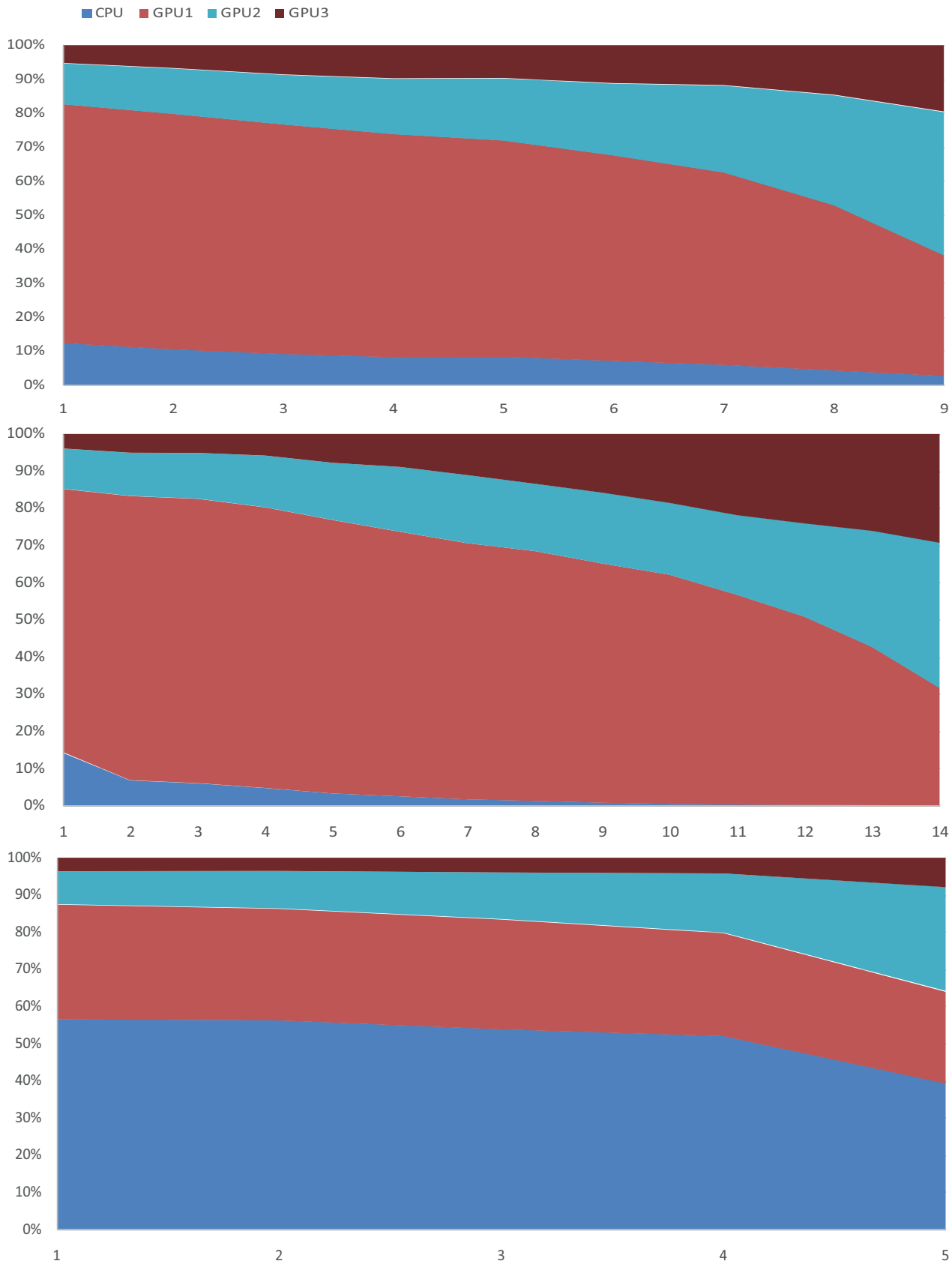


**Рис. 2** Нормированные графики изменения времени расчета оптимальных порогов в процедуре LOO для каждого яруса и каждого алгоритма в задачах № 5 (сверху), № 8 (по центру) и № 11 (снизу)

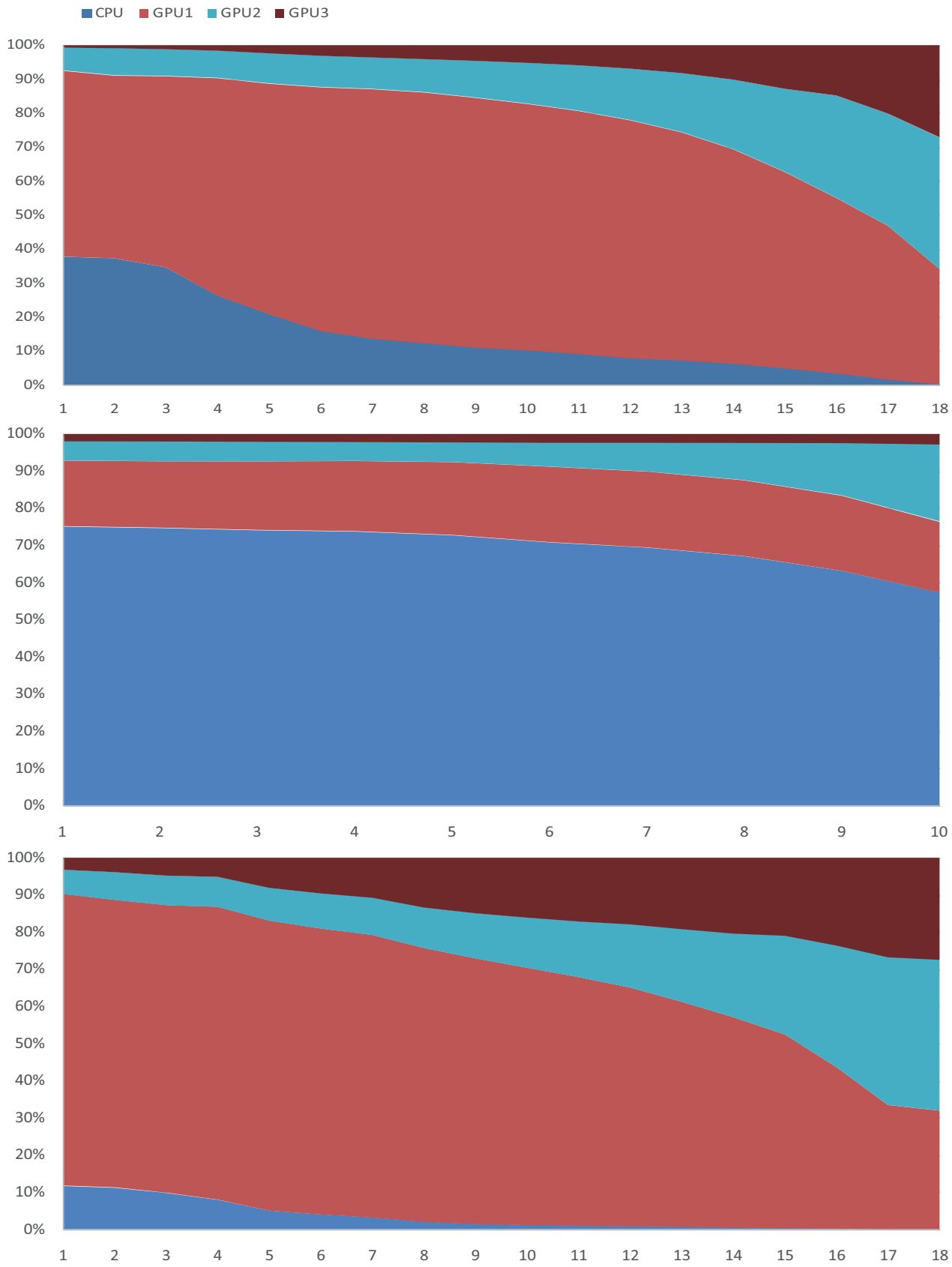


**Рис. 3** Нормированные графики изменения времени расчета оптимальных порогов в процессе синтеза ПРД для каждого яруса и каждого алгоритма в задачах №19 (сверху), №18 (по центру) и №16 (снизу)

Наиболее трудоемким этапом синтеза ПРД алгоритмом AGI.Bias является поиск оптимальных порогов для бинарной перекодировки значений признаков. На указанные вычисления приходится в среднем 96,3% от всего времени синтеза ПРД.

Анализ полученных результатов по табл. 1 и 2 показал, что наиболее быстрым из описанных в разд. 3 параллельных алгоритмов оказался алгоритм Deep Dynamic PAGI.Bias, который позволяет сократить время построения ПРД (до 27 раз) по сравнению с алгоритмом AGI.Bias только в тех случаях, когда обучающая выборка содержит большое число признаков и/или обучающих объектов и при этом информация либо вещественнозначная, либо целочисленная большой значности (задачи №№ 7, 8, 11–15, 17, 18, 21–23). В случае небольших обучающих выборок с вещественнозначной или целочисленной информацией небольшой значности уменьшение времени (в среднем в 8,5 раз) наблюдалось только при построении первых трех–пяти ярусов ПРД (задачи №№ 5, 9, 16, 19 и 20). При построении остальных ярусов ПРД происходило увеличение времени за счет того, что копирование данных между GPU и CPU занимало большее время, чем получаемый выигрыш в вычислениях на GPU по сравнению с CPU. Данный факт наглядно показан на рис. 2 и 3. На задачах с малой значностью признаков или малой обучающей выборкой наблюдалось существенное увеличение времени при построении ПРД параллельными версиями алгоритма AGI.Bias. Данный факт объясняется, во-первых, тем, что выигрыша в вычислениях на GPU по сравнению с CPU практически нет. Во-вторых, тем, что на перенос необходимой информации в память GPU требуется намного больше времени, чем получаемый выигрыш в вычислениях на GPU.

Сравнение параллельных версий алгоритма AGI.Bias между собой показало следующее:

1. Применение в алгоритме Dynamic PAGI.Bias динамического параллелизма для поиска оптимальных порогов по нескольким признакам параллельно и передача меньшего объема данных между CPU и GPU позволило уменьшить время построения ПРД до 5,5 раз (в среднем в 2,7 раза) по сравнению с PAGI.Bias.
2. Применение динамического параллелизма на более «низких» вычислительных уровнях (см. разд. 3) в алгоритме Deep Dynamic PAGI.Bias позволило ускорить синтез ПРД до 11 раз (в среднем в 3 раза) по сравнению с алгоритмом Dynamic PAGI.Bias.

## 5 Заключение

В данной работе разработаны алгоритмы синтеза ПРД с применением параллельных вычислений на основе технологии CUDA — PAGI.Bias, Dynamic PAGI.Bias и Deep Dynamic PAGI.Bias. В качестве базового алгоритма был использован алгоритм AGI.Bias.

Алгоритм PAGI.Bias отличается от алгоритма AGI.Bias тем, что поиск оптимального порога для бинарной перекодировки текущих значений признака при синтезе ПРД полностью перенесен на GPU. Указанное изменение ускорило построение ПРД алгоритмом PAGI.Bias (до 3,3 раз на реальных задачах) по сравнению с алгоритмом AGI.Bias.

Алгоритм Dynamic PAGI.Bias отличается от алгоритма PAGI.Bias в применении динамического параллелизма (поиск оптимального порога осуществляется по нескольким признакам параллельно) и в уменьшении объема передаваемых данных между памятью GPU и оперативной памятью CPU. Данные изменения позволили снизить время синтеза дерева алгоритмом Dynamic PAGI.Bias (до 5,5 раз на реальных задачах) по сравнению с алгоритмом PAGI.Bias.

Алгоритм Deep Dynamic PAGI.Bias отличается от алгоритма Dynamic PAGI.Bias тем, что динамический параллелизм применяется и на более «низких» вычислительных уров-

нях (например, при расчете информативности порогов признака по различным наборам порогов параллельно). Показано, что применение алгоритма Deep Dynamic PAGI.Bias позволяет заметно снизить время синтеза ПРД (до 11 раз на реальных задачах) по сравнению с алгоритмом Dynamic PAGI.Bias.

На реальных задачах показано, что наиболее быстрым из разработанных алгоритмов оказался алгоритм Deep Dynamic PAGI.Bias, который позволяет сократить время построения ПРД (до 27 раз) по сравнению с алгоритмом AGI.Bias только в тех случаях, когда обучающая выборка содержит большое число признаков и/или обучающих объектов и при этом информация либо вещественнозначная, либо целочисленная большой значности. В случае небольших обучающих выборок с вещественнозначной или целочисленной информацией небольшой значности уменьшение времени (в среднем в 8,5 раз) наблюдалось только при построении первых трех–пяти ярусов ПРД. При построении остальных ярусов ПРД происходило увеличение времени за счет того, что копирование данных между GPU и CPU занимало большее время, чем получаемый выигрыш в вычислениях на GPU по сравнению с CPU. На задачах с малой значностью признаков или малой обучающей выборкой наблюдалось увеличение времени на всех ярусах ПРД по той же самой причине.

Таким образом, разработанные алгоритмы синтеза ПРД с применением параллельных вычислений на основе технологии CUDA позволяют существенно снизить время синтеза дерева (более чем в 10 раз) только для задач распознавания по прецедентам, содержащим большое число признаков и/или обучающих объектов с вещественнозначной информацией или целочисленной информацией большой значности.

## Литература


[1] *Журавлев Ю. И., Рязанов В. В., Сенько О. В.* Распознавание. Математические методы. Программная система. Практические применения. — М.: ФАЗИС, 2006. 176 с.

[2] *Djukova E. V., Peskov N. V.* A classification algorithm based on the complete decision tree // J. Pattern Recogn. Image Anal., 2007. Vol. 17. No. 3. P. 363–367.

[3] *Генрихов И. Е.* Построение и исследование полных решающих деревьев для задач классификации по прецедентам: Дисс. ... канд. физ.–мат. наук. — М., 2013. 169 с.

[4] *Генрихов И. Е.* Исследование обобщающей способности полного решающего дерева // Ж. вычисл. мат. мат. физ., 2014. Т. 54. № 6. С. 1033–1047.

[5] *Боресков А. В., Харламов А. А., Марковский Н. Д.* Параллельные вычисления на GPU. Архитектура и программная модель CUDA. — М.: Изд-во Московского ун-та, 2012. 336 с.

[6] *Cheng J., Grossman M., McKercher T.* Professional CUDA C programming. — New York, NY, USA: Wrox, 2014. 528 p.

[7] *Lichman M.* UCI Machine Learning Repository. Irvine, CA, USA: University of California, School of Information and Computer Science, 2013. `http://archive.ics.uci.edu/ml`.

[8] *Дюкова Е. В., Журавлев Ю. И., Песков Н. В., Сахаров А. А.* Обработка вещественнозначной информации логическими процедурами распознавания // Искусственный интеллект, 2004. № 2, С. 80–85.

[9] *Quinlan J. R.* C4.5: Programs for machine learning. San Mateo, CA, USA: Morgan Kaufmann, 1993. 302 p.

[10] *Peng L., Lei L.* A review of missing data treatment methods // Int. J. Intelligent Information Management Syst. Technol., 2005. Vol. 1. No. 3. P. 412–419.

[11] *Marlin B. M.* Missing data problems in machine learning. Department of Computer Science, University of Toronto, 2008. PhD Thesis. 156 p.

[12] *Kohavi R., Kunz C.* Option decision trees with majority votes // Conference (International) on Machine Learning, 1997. P. 161–169.

[13] *Kuncheva L. I.* Combining pattern classifiers methods and algorithms. — Hoboken, NJ, USA: John Wiley & Sons, Inc., 2004. 350 p.

*Поступила в редакцию 20.12.2015*

# Synthesis of full decision tree with using heterogeneous systems on the basis of CUDA technology

***I. E. Genrikhov***

ingvar1485@rambler.ru

Mobile park IT, 21/1 Panfilova st., Khimki, Moscow Region, Russia

The article is devoted to classification algorithms based on full decision trees. Due to the decision tree construction under consideration, all the features satisfying a branching criterion are taken into account at each special vertex. The main drawback of full decision tree is significantly larger the time for synthesis of tree compared with the classical decision tree. The issues are considered of reducing the time for synthesis of full decision tree using CUDA technology. This technology allows one to use a large number of GPU cores to speed up the execution of complex computing. The results of the testing are given on model and real tasks. It is shown that the use of CUDA technology allows one to significantly reduce the time for synthesis of the full decision tree (more than 10 times) only in cases where the training set contains large number of attributes and/or learning objects and the information are either real-valued or integer large complexity.

**Keywords**: *precedent-based pattern recognition problem; full decision tree; CUDA; heterogeneous system*

**DOI:** 10.21469/22233792.2.1.05

## References

[1] Zhuravlev, Yu. I., V. V. Ryazanov, and O. V. Senko. 2006. *Raspoznavanie. Matematicheskie metody. Programmnaya sistema. Prakticheskoe primenenie* [Recognition. Mathematical methods. Software system. Practical applications]. Moscow: Phasis. 176 p.

[2] Djukova, E. V., and N. V. Peskov. 2007. A classification algorithm based on the complete decision tree. *J. Pattern Recogn. Image Anal.* 17(3):363–367.

[3] Genrikhov, I. E. 2013. Postroenie i issledovanie polnykh reshayuschcikh derev'ev dlya zadach klassifikatsii po pretsedentam [Construction and research the full decision trees for classification problems by precedents]. Moscow. PhD Thesis. 169 p.

[4] Genrikhov, I. E. 2014. Issledovanie oboshchayushchey sposobnosti polnogo reshayushchego dereva [Analysis of the generalization ability of a full decision tree]. *J. Comput. Math. Math. Phys.* 54(6):1046–1059.

[5] Boreskov, A. V., A. A. Harlamov, and N. D. Markovskii. 2012. *Parallel'nye vychisleniya na GPU. Arkhitektura i programmnaya model' CUDA* [Parallel computing on the GPU. The architecture and programming model of CUDA]. Moscow: Moscow University Publs. 336 p.

[6] Cheng, J., M. Grossman, and T. McKercher. 2014. *Professional CUDA C programming*. New York, NY: Wrox. 528 p.

[7] Lichman, M. 2013. UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. Available at: `http://archive.ics.uci.edu/ml` (accessed August 14, 2016).

[8] Djukova, E. V., Yu. I. Zhuravlev, N. V. Peskov, and A. A. Saharov. 2004. Obrabotka veshchestvennoznachnoy informatsii logicheskimi protsdurami raspoznavaniya [Processing a real-valued information with logical recognition procedures]. *Artificial Intelligence* 2: 80–85.

[9] Quinlan, J. R. 1993. *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann. 302 p.

[10] Peng, L., and L. Lei. 2005. A review of missing data treatment methods. *Int. J. Intelligent Information Management Syst. Technol.* 1(3):412–419.

[11] Marlin, B. M. 2008. *Missing data problems in machine learning*. Department of Computer Science University of Toronto. PhD Thesis. 156 p.

[12] Kohavi, R., and C. Kunz. 1997. Option decision trees with majority votes. *Conference (International) on Machine Learning*. 161–169.

[13] Kuncheva, L. I. 2004. *Combining pattern classifiers methods and algorithms*. Hoboken, NJ: John Wiley & Sons, Inc. 350 p.

*Received December 20, 2015*