3. Назаренко Н. А., Падерно П. И. Влияние интерфейса на состояние и здоровье оператора // Биотехносфера. 2009. № 6. С. 45–52.

4. Падерно П. И. Метод комплексирования мнений экспертов внутри группы при использовании метода анализа иерархий // Изв. Санкт-Петербургской лесотехнической академии. 2009. № 189. С. 238–245.

5. Падерно П. И. Комплексирование мнений групп экспертов при оценке значимости показателей // Изв. Санкт-Петербургской лесотехнической академии. 2010. № 190. С. 207–211.

6. Бурков Е. А., Падерно П. И., Пахарьков Г. Н. Эргономическая экспертиза: системные проблемы и пути их решения при выборе медицинской техники // Биотехносфера. 2010. № 2. С. 6–14.

E. A. Burkov, S. S. Nasser

SUPPORT OF IDENTIFICATION AND EVALUATION OF WORKPLACE RISKS (ON THE EXAMPLE OF THE OPERATOR OF CONTROL BOARDS)

In this article considered the method of identification and evaluation of workplace hazards on the example of the power industry workers. The main groups of the risks connected with professional activity of operators of control boards are given.

Workplace risks, analytic hierarchy process, ergonomics

УДК: 20.53.19, 28.23.13

И. И. Холод, З. А. Каршиев

МЕТОДИКА РАСПАРАЛЛЕЛИВАНИЯ АЛГОРИТМОВ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА

Описывается методика распараллеливания алгоритмов интеллектуального анализа данных на основе блочной структуры. Данная методика учитывает характеристики среды выполнения параллельных алгоритмов, типа распределения анализируемых данных, а также особенности распараллеливаемого алгоритма.

Интеллектуальный анализ, параллельные алгоритмы, интеллектуальный анализ распределенных данных

В последние годы повышение производительности вычислительной техники связано с развитием многоядерных процессоров. Такие процессоры уже давно не являются прерогативой мощных вычислительных серверов. Процессорами с четырьмя, шестью или восемью ядрами оснащаются обычные домашние, офисные и тем более рабочие места аналитиков. Однако современное программное обеспечение значительно отстает от аппаратной части и часто неэффективно использует предоставляемые аппаратные ресурсы. Как правило, на компьютерах с многоядерными процессорами используется 2 или в лучшем случае 4 ядра. Данная проблема в первую очередь связана с трудоемкостью задачи распараллеливания вычислительных алгоритмов. К сожалению, не являются исключением и алгоритмы интеллектуального анализа данных (ИАД). Большинство усилий исследователей в области параллельных алгоритмов ИАД тратятся на распараллеливание отдельных алгоритмов анализа и их дальнейшую оптимизацию [1], [2]. Ситуацию усугубляет и то, что эти усилия прикладываются исходя из определенной среды вычисления, следовательно, при переносе такого решения в другие условия оно становится неэффективным.

В данной статье предлагается методика распараллеливания алгоритмов ИАД. Она описывает последовательность шагов для распараллеливания любого алгоритма ИАД, реализованного в блочной структуре [3]. При этом учитываются как среда выполнения алгоритма и тип распределения данных, так и особенности таких алгоритмов.

Исходными данными для методики распараллеливания алгоритма ИАД являются:

• описание среды выполнения параллельного алгоритма:

– количество процессоров (потоков), на которых возможно параллельное выполнение;

– производительность процессоров;

– наличие/отсутствие общей памяти у процессоров и/или средств взаимодействия;

• тип распределения анализируемых данных [2]:

– единое или распределенное хранилище;

– в случае распределенного хранилища тип распределения: горизонтальное или вертикальное;

• описание алгоритма ИАД:

– формальное описание ИАД;

– модель знаний, получаемая в результате работы алгоритма.

В статье [4] описана классификация параллельных алгоритмов ИАД. В соответствии с ней выделяют 2 уровня характеристик таких алгоритмов:

• структурный уровень, предполагающий изменение структуры параллельного алгоритма;

• настраиваемый уровень, предполагающий изменение параметров выполнения параллельного алгоритма путем его настройки без изменения структуры алгоритма.

В соответствии с такой классификацией разделим методику на 2 основных этапа:

I. Определение структуры параллельного алгоритма ИАД.

II. Настройка параллельного алгоритма ИАД.

На первом этапе выполняется анализ структуры алгоритма ИАД с целью выявления типовых элементов и других особенностей, присущих алгоритму. На основании результатов анализа формируется структура параллельного алгоритма.

Для определения структуры параллельного алгоритма необходимо выполнить следующие действия:

I.1. Декомпозировать алгоритм ИАД на элементарные операции.

I.2. Сгруппировать в функциональные блоки действия, выполняющие дискретное изменение модели.

I.3. Выполнить программную реализацию блочной структуры последовательного алгоритма.

I.4. Выполнить анализ полученной структуры с учетом оставшихся циклов и условных переходов и определить структуру параллельного алгоритма.

I.5. Выполнить программную реализацию параллельной структуры алгоритма ИАД на основе ранее реализованных функциональных блоков.

На первом шаге структура алгоритма должна быть декомпозирована на элементарные операции, выполняющиеся над моделью знаний: добавление, удаление и изменение отдельных элементов модели (в соответствии с формальной моделью алгоритма, описанной в статье [5]). При этом в структуре алгоритма должны быть явно выделены основные управляющие конструкции: циклы, условные переходы и др. Декомпозиция должна выполняться с учетом следующих условий:

– после выполнения каждой операции сохраняется целостность модели знаний;

– количество переменных, используемых различными операциями, должно быть минимизировано и локализовано (переменные должны использоваться только для следующих друг за другом операций).

На данном шаге допускается наличие элементарных операций, не изменяющих модель знаний.

Примером такой декомпозиции может служить следующий псевдокод алгоритма Naive Bayes (для набора данных с одним целевым атрибутом):

1. for i=1 to |W| // для всех векторов

2. p = D(a_t).getIndex($v_{t.i}$) // получить индекс значения целевого атрибута a_t текущего вектора из его области определения D(a_t)

3. Mnb.k_p= Mnb.k_p+1

4. for j=1 to |A| // для всех независимых атрибутов

5. u = Mnb.R.getIndex(($a_j = v_{j.i}$) $\rightarrow$ ($a_t = v_{t.p}$)) // получить индекс правила с соответствующей условной и целевой частями

6. Mnb.n_u = n_u + 1

7. end for j;

8. end for i;

Приведенный алгоритм Naive Bayes вычисляет элементы модели Mnb, соответствующей модели NaveBayesModel стандарта PMML*, формально описанной в статье [6] и включающей в себя следующие элементы:

– Mnb.R – множество классификационных правил вида $(a_j = v_{j.i}) \rightarrow (a_t = v_{t.p})$;

– Mnb.k_p – количество векторов, у которых значение целевого атрибута $v_{t.i}$;

– Mnb.n_u – количество векторов, у которых независимый атрибут a_j имеет значение $v_{j.i}$ и целевой атрибут a_t имеет значение $v_{t.p}$.

Описанный алгоритм работает с исходной моделью знаний, в которой все элементы уже созданы, исходя из метаданных об обрабатываемых данных (значениях целевых и нецелевых атрибутов). В связи с этим в алгоритме нет операций, связанных с добавлением элементов в модель, а присутствуют только операции по их изменению в строках 3 и 6. Действия в строках 2 и 5 не изменяют модель. Кроме того алгоритм включает в себя 2 вложенных друг в друга цикла: внешний – по всем векторам W и внутренний – по всем независимым атрибутам A.

На втором шаге действия группируются в функциональные блоки, выполняющие дискретное изменение модели знаний. В процессе группировки в функциональный блок выделяются операции, связанные между собой общими локальными переменными. Базовыми в таких блоках являются операции, изменяющие состояние модели знаний, к которым добавляются предшествующие и/или последующие операции, не меняющие модель. Группировка операций, не меняющих модель, со смежными базовыми операциями выполняется исходя из их логической принадлежности к ним. Признаком такой логической связи может служить использование промежуточных (локальных) переменных. Такие переменные должны быть локализованы внутри создаваемого функционального блока.

Управляющие конструкции (циклы, условные переходы и т. п.) не изменяют состояния модели, однако они должны быть выделены в отдельные управляющие блоки.

В рассматриваемом примере есть две операции, меняющие модель знаний (строки 3 и 6). Они должны образовывать 2 функциональных блока:

---

* PMML Specification.: Data Mining Group. http://www.dmg.org/PMML-4_0.

– IncForTargetValue – инкрементирование количества векторов с определенным значением целевого атрибута $Mnb.k_p$;

– IncForRule – инкрементирование количества векторов, соответствующих определенному правилу $Mnb.n_u$.

В блок IncForTargetValue должна быть добавлена операция из строки 2, так как она использует общую промежуточную переменную p с базовой операцией в строке 3. Операция из строки 5 по тем же причинам должна быть добавлена в блок IncForRule.

Кроме функциональных блоков в алгоритме можно выделить 2 управляющих блока:

– CycleByVector – цикл по всем векторам набора данных;

– CycleByAttribute – цикл по всем независимым атрибутам.

Таким образом, ранее описанный алгоритм Naive Bayes можно представить в виде блок-схемы (рис. 1).

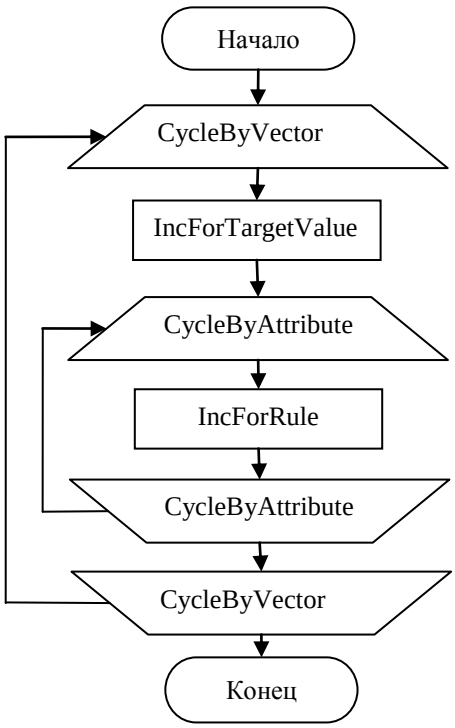

Рис. 1

На третьем шаге необходимо реализовать полученные функциональные блоки. Данная реализации должна быть выполнена путем наследования от базовых классов, описанных в [3]. Полученные функциональные блоки объединяются в единый последовательный алгоритм.

На предпоследнем шаге первого этапа методики необходимо выполнить анализ полученной структуры алгоритма. Результатом данного анализа должна стать структура параллельного алгоритма, отнесенная к одному из классов в соответствии с предложенной классификацией [4]. Можно дать следующие рекомендации:

– если в структуре присутствует цикл по векторам, то возможно распараллеливание по данным, при этом параллельными ветвями алгоритма будут сам цикл с выделенной порцией векторов и блоки, выполняющиеся внутри цикла и обрабатывающие их;

– при выделении в алгоритме параллельных ветвей, работу которых необходимо координировать (например, распределять векторы между ветвями в зависимости от значения целевого атрибута), в структуру может быть добавлен поток-диспетчер;

– если в структуре присутствует блок условного перехода, то возможна параллелизация по задачам, кроме того в поток-диспетчер может быть выделен как сам условный переход, так и блоки, предшествующие ему, а параллельными ветвями блоки, выполняющиеся в зависимости от выполнения условия;

– если в алгоритме присутствуют блоки, требующие обработки всех данных (например, сравнения каждого вектора с каждым или расчет функции по всем векторам), то возможна реализация взаимодействия между потоками или добавление потока-диспетчера, содержащего блоки по маршрутизации векторов;

– если алгоритм строит модель знаний, которая не может быть корректно объединена после ее завершения и требуется согласование на промежуточных стадиях (например, для построения деревьев решений, строящихся для разных наборов данных, должно выполняться промежуточное согласование об информативности атрибутов), то в структуру алгоритма может быть добавлен поток-диспетчер.

В рассматриваемом примере при анализе структуры алгоритма, представленного на рис. 1, можно сделать вывод о возможности его распараллеливания по данным в связи с наличием в нем цикла по векторам – CycleByVector. При этом в параллельных ветвях алгоритма будет выполняться блок CycleByVector и все блоки, выполняющиеся внутри этого цикла.

Полученная структура параллельного алгоритма представлена на рис. 2.

На последнем шаге выполняется реализация параллельного алгоритма, выбранной структуры добавлением в последовательную версию алгоритма (полученную на шаге I.3) соответствующих блоков для параллельного выполнения, описанных в [3].

На втором этапе распараллеливания алгоритма производится настройка выполнения параллельного алгоритма и анализ его эффективности с последующей отладкой. Этап включает в себя следующие шаги:

II.1. Настройка выполнения параллельного алгоритма ИАД в соответствии с параметрами среды и типом распределения данных.

II.2. Анализ параллельного выполнения алгоритма ИАД на тестовых данных путем оценки эффективности и ускорения.

В случае неудовлетворительных результатов осуществляется возврат на шаг II.1, если не все настройки были испытаны, и на шаг I.4 – в противном случае.

На шаге II.1:

– указываются средства выполнения параллельного алгоритма (потоки, сервис-ориентированная архитектура и др.) исходя из существующих возможностей и среды выполнения;

– указывается количество обработчиков параллельных ветвей алгоритма в зависимости от среды выполнения (например, от количества ядер у процессора);

– указывается тип параллельного выполнения алгоритма ИАД [4]:

SDSM – один поток данных, одна модель;

MDSM – много потоков данных, одна модель;

SDMM – один поток данных, много моделей;

MDMM – много потоков данных, много моделей.

Настройка типа параллельного выполнения алгоритма может выполняться в зависимости от типа распределения данных, особенностей строящейся модели знаний и среды выполнения. На этом шаге можно дать следующие рекомендации:

– если источник информации единый, то предпочтительнее выбирать тип SD*, если источников несколько, то MD*;

– если модель знаний не объединяется из нескольких независимо построенных и средства выполнения имеют общую память, то лучшим вариантом будет выбор типа *SM, в противном случае может быть использован любой тип.

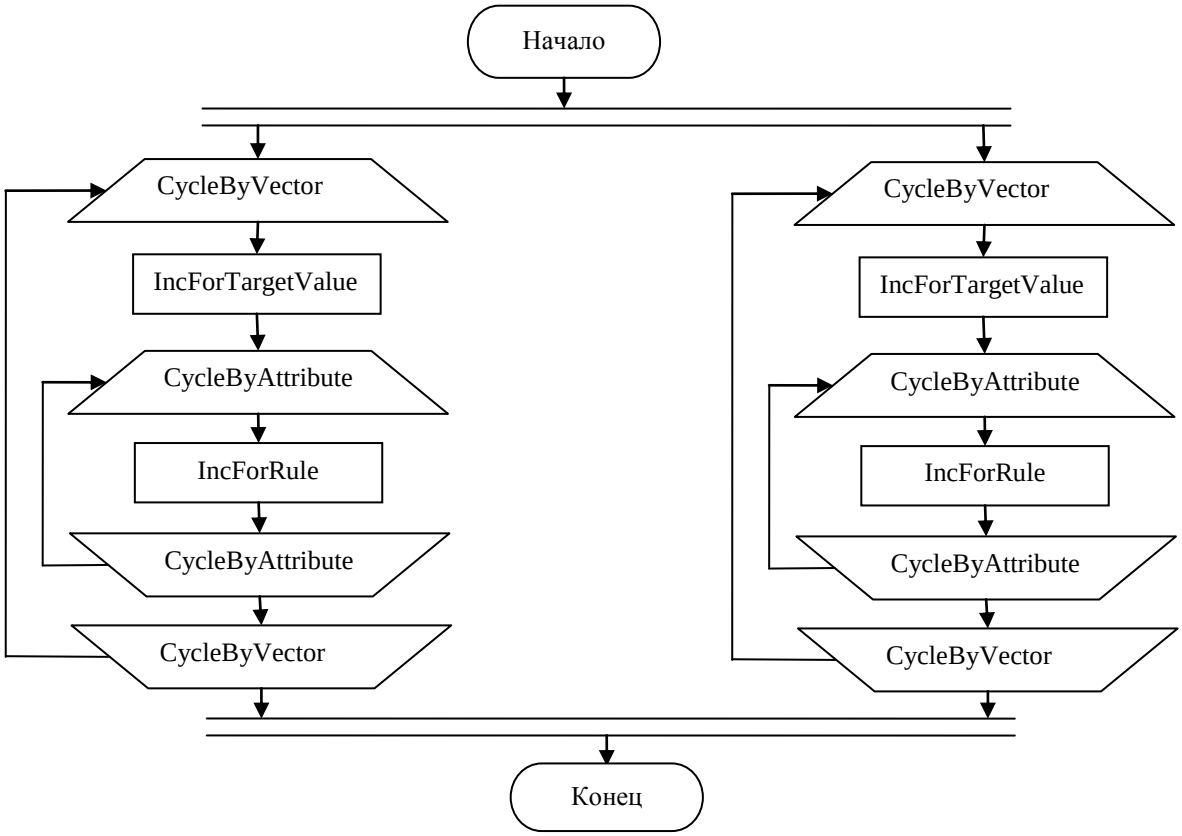

Рис. 2

На следующем шаге необходимо измерить время работы последовательного и параллельного алгоритмов и по полученным данным вычислить ускорение и эффективность работы построенного параллельного алгоритма ИАД. Если получены неудовлетворительные оценки, можно вернуться на предыдущий шаг и выбрать другой тип выполнения параллельного алгоритма. Если изменение типа выполнения параллельного алгоритма не позволяет повысить эффективность алгоритма до необходимого уровня, можно вернуться на предпоследний шаг первого этапа и реструктурировать алгоритм из реализованных функциональных блоков (не изменяя сами блоки).

Описанную методику можно представить в виде блок-схемы (рис. 3). На ней штриховой линией обозначена граница этапов методики, затрагивающих структуру алгоритма и не затрагивающих ее (изменяющих характеристики параллельного выполнения за счет настройки).

В данной методике наиболее трудоемкими являются первые 3 шага, связанные с формированием и реализацией функциональных блоков. Достоинством методики является то, что эти шаги не повторяются при отладке алгоритма и повышении его эффективности. Все изменения алгоритма в данном направлении связаны или с реструктуризацией алгоритма без изменения функциональных блоков, или с его перенастройкой. Необходимо заметить, что второй этап может быть автоматизирован при программной реализации метода оценки эффективности.

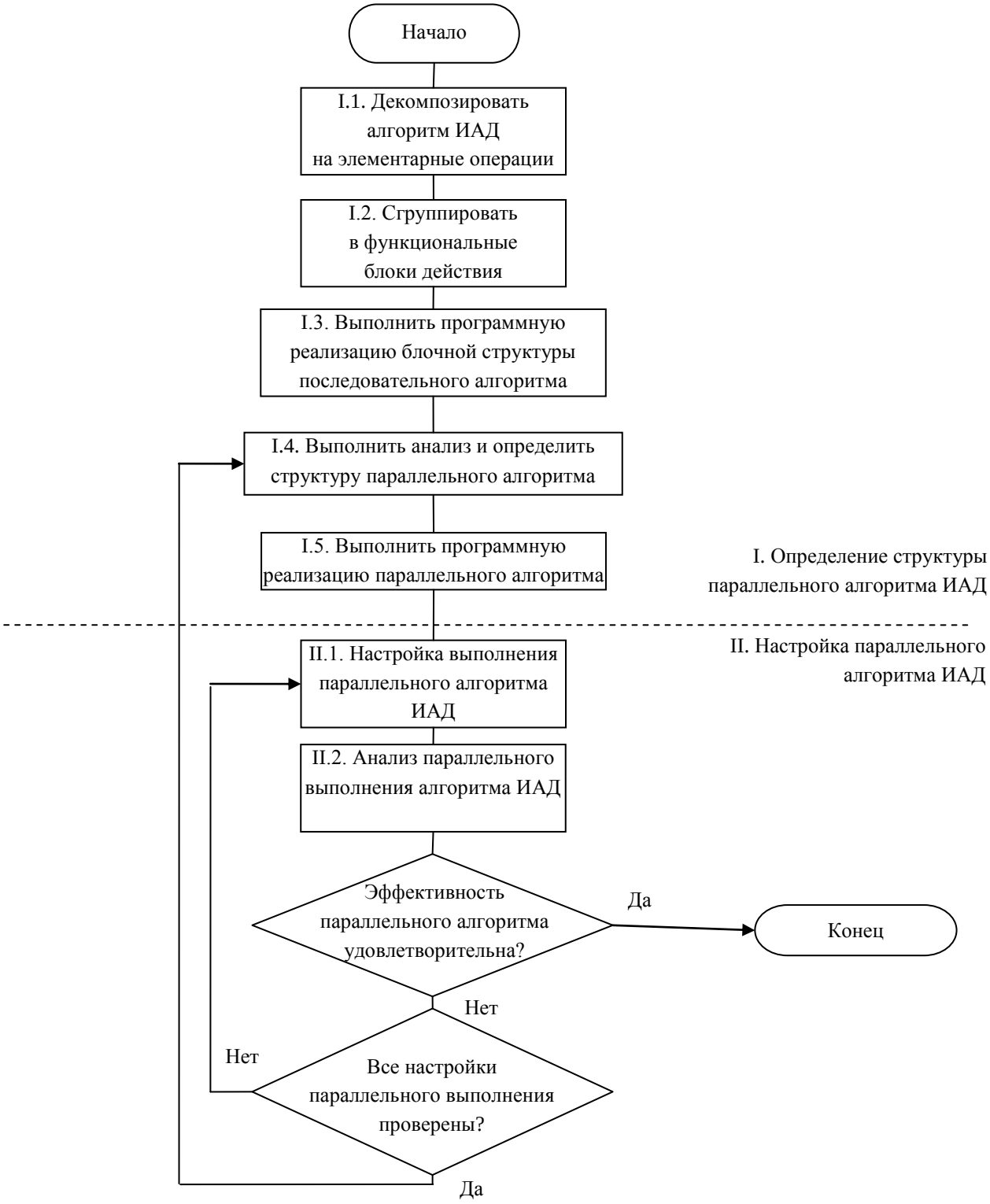

Рис. 3

Данный метод может быть основан на вычислении ускорения как отношения временных характеристик последовательного алгоритма (реализованного на шаге I.3) и параллельного алгоритма (реализованного на шаге I.5). Можно также расширить параметры анализа, добавив методы оценки балансировки параллельного выполнения алгоритма. Указанные направления исследований позволят еще более снизить трудоемкость данной методики и повысить эффективность алгоритмов, распараллеливаемых с ее помощью.

СПИСОК ЛИТЕРАТУРЫ

1. Каршиев З. А., Холод И. И.. Анализ распределенных данных: обзор методов объединения моделей // Сб. докл. XIV Междунар. конф. по мягким вычислениям и измерениям SCM`2011, СПб., 23–25 июня, 2011. Т. 1. С. 163–166.

2. Интеллектуальный анализ распределенных данных на базе облачных вычислений / М. С. Куприянов, И. И. Холод, З. А. Каршиев и др. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2011.

3. Интеллектуальный анализ данных в распределенных системах / М. С. Куприянов, И. И. Холод, З. А. Каршиев, И. А. Голубев. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2012.

4. Холод И. И. Способы параллелизации алгоритмов интеллектуального анализа // Изв. СПбГЭТУ «ЛЭТИ». 2012. Вып. 9. С. 39–47.

5. Куприянов М. С., Каршиев З. А. Формальная модель выполнения алгоритмов интеллектуального анализа // Изв. СПбГЭТУ «ЛЭТИ». 2012. Вып. 9. С. 60–68.

6. Холод И. И. Унифицированная модель Data Mining // Сб. докл. XIV Междунар. конф. по мягким вычислениям и измерениям SCM`2012, СПб., 25–27 июня, 2012. Т. 1. С. 237–240.

I. I. Kholod, Z. A. Karshiyev

METHODS PARALLELIZING DATA MINING ALGORITHMS

This article describes how to parallel data mining algorithms based on the block structure. This technique takes into account the characteristics of the environment of parallel algorithms, such as the distribution of the data being analyzed, as well as features of the algorithm is parallelized.

Mining, parallel algorithms, distributed data mining

УДК 004.93.12

И. В. Сорокин, А. В. Копыльцов

ИСПОЛЬЗОВАНИЕ НЕОКОГНИТРОНА ДЛЯ РАСПОЗНАВАНИЯ ВРЕДОНОСНЫХ ФАЙЛОВ

Построен алгоритм использования неокогнитрона для сравнения бинарных данных. Рассмотрены этапы обучения и тестирования нейросети с целью выявления в анализируемых файлах характерных участков, позволяющие судить о принадлежности файла к определенному классу вредоносных программ.

Распознавание образов, нейронные сети, неокогнитрон, вредоносные файлы

Оценивание похожести бинарных файлов является одной из проблем в системах обработки и хранения данных. Например, в теории информационной безопасности большую роль играют алгоритмы, позволяющие сравнивать исполняемые файлы между собой. В системах поиска одной из проблем является выделение схожих документов. Альтернативным способом для решения подобных проблем является применение методов обработки изображений и распознавание образов.

Нейронные сети нашли широкое применение во многих областях, где требуется автоматическая обработка данных, изображений и т. п. Например, использование нейросетевого подхода в медицине позволяет сегментировать медицинские снимки на участки для выявления отклонений и постановки точного диагноза [1]. Анализ геофизических данных, основанный на нейросетевых методах, используется при выявлении сейсмической активности [2], поиске залежей нефти и газа [3]. Теория нейронных сетей находит применение и во многих других областях, таких, как машиностроение, космическая отрасль, экономика и социология.