АЛГОРИТМЫ ПАРАЛЛЕЛЬНОЙ РЕАЛИЗАЦИИ АНАЛИЗА ФОРМАЛЬНЫХ ПОНЯТИЙ ДЛЯ ПРИБЛИЖЁННЫХ МНОЖЕСТВ В ОДНОРОДНЫХ СЕМАНТИЧЕСКИХ СЕТЯХ

В.П. Ивашенко
Старший преподаватель кафедры интеллектуальных информационных технологий БГУИР, кандидат технических наук

С.В. Синцов
Инженер-программист, компания Altoros, магистрант кафедры интеллектуальных информационных технологий БГУИР

Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь
E-mail: ivashenko@bsuir.by, ssivikt@gmail.com

Abstract. This paper considers a way of parallel implementation of an algorithm of formal concept analysis for rough sets in homogeneous semantic networks. The algorithm description consists of two parts. The first part describes two supply algorithms for set intersection and union operations respectively, which are adapted for parallel implementation on heterogeneous compute architectures with data stream parallelism. The second part describes steps for building graph of class-subclass ontology (taxonomy) from a rough formal context.

Наличие данных большого объёма в информационных системах требует эффективных механизмов их обработки, применение которых в сочетании с тенденциями автоматизации задач анализа данных, ориентированных на применение моделей и методов искусственного интеллекта, поддерживающих обработку семантически структурированных данных, позволяет сократить затраты на подготовку и интерпретацию результатов анализа пользователем. В работе рассматриваются подход и алгоритмы, способные обеспечить автоматизацию процесса построения онтологических структур и поддержку создания компонентов [4] при проектировании баз знаний, использующих однородные семантические сети [3] для представления информации. Актуальность работы обусловлена потребностью в интеллектуальной обработке больших объёмов данных и знаний, возрастающими требованиями ко временным затратам обработки данных и знаний, а также развитием семантических моделей представления знаний. Особенностью рассматриваемой задачи является поддержка работы в условиях неполноты информации, что достигается за счёт применения модели унифицированного семантического представления знаний [6] и сочетания средств анализа формальных понятий (АФП) [12, 13] и аппарата приближённых множеств (ПМ) [12]. Следует учитывать, что обработка больших данных происходит при ограниченных

временны́х и физических ресурсах и с привлечением приближённых, носящих вероятностный характер, методов обработки данных [1], что увеличивает неполноту и неопределённость информации. Поэтому возможность проведения анализа в условиях неполной информации в этом случае является необходимой. Методы АФП для обработки больших данных исследованы, например, в [9, 10], однако в них не учитывается неполнота и неопределённость информации. Результат решения задачи АФП не только позволяет выявлять порядок организации знаний (отношения между понятиями, визуализируя эти отношения в виде диаграмм Хассе), но и может служить основой для построения гипотез и формулировки теорем о предметной области [12]. Особенностью работы является также то, что алгоритмы в рамках рассматриваемого подхода предлагаются как элемент технологии, ориентированной на создание интеллектуальных систем, и выстраиваются в согласии с соответствующими реализационными и архитектурными принципами [8].

Целью работы является разработка программных средств автоматизации построения онтологий, представляемых в виде однородных семантических сетей с базовой теоретико-множественной интерпретацией [3]. Для этого 1) разрабатываются алгоритмы базовых операций обработки множеств, которые допускают параллельную реализацию на гетерогенной вычислительной архитектуре [[10]]; 2) описывается метод построения графа родовидовой онтологии (таксономии) по формальному контексту, заданному на приближённых множествах.

Система, основанная на знаниях [2], состоит из базы знаний (БЗ), машины обработки знаний (МОЗ) и пользовательского интерфейса (ПИ). Компоненты БЗ и МОЗ относятся ко внутренней части системы, основанной на знаниях, которая скрыта от пользователя; ПИ относится ко внешней части, доступной пользователю.

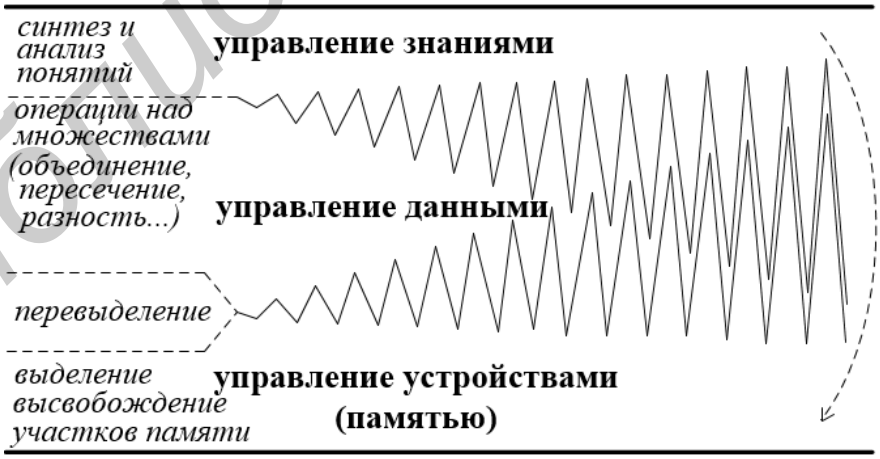

Рис. 1. Уровни внутренней части системы, основанной на знаниях

В системе, основанной на знаниях, можно выделить три уровня (рис. 1): 1) уровень управления устройствами; 2) уровень управления данными; 3) уровень

управления знаниями. Управление знаниями, представленными в виде однородных семантических сетей с базовой теоретико-множественной интерпретацией, требует решения задачи представления и обработки множеств, что включает в себя реализацию различных структур данных: динамические массивы, списки, деревья и т.д. [5], а это, в свою очередь, требует реализации базовых операций динамического управления памятью [5] на первом уровне: выделение, высвобождение и перевыделение участков памяти.

Решение задачи АФП для ПМ в однородных семантических сетях поддерживается на нижних уровнях и полностью обеспечивается на уровне три.

Производительность и масштабирование использующихся при АФП алгоритмов зависит, в частности, от вычислительной программно-аппаратной платформы, которая выбрана основой для их реализации. При обработке больших данных в качестве таких платформ используются высокопроизводительные параллельные и распределённые гетерогенные вычислительные системы классов ОКМД или МКМД. Вместе с тем, глобальные компьютерные сети являются одним из основных хранилищ и источников больших данных. В многомашинных вычислительных системах для получения высокой производительности при решении задач и обеспечения эффективной работы пользователя хотя бы с частью имеющегося объёма данных необходимым является эффективное использование возможностей каждого узла и технологий, способных сочетать возможности продолжающих получать в настоящее время развитие гетерогенных архитектур. Что касается веб-технологий, то для клиентской стороны известна по крайней мере одна веб-технология – Superconductor [14], которая позволяет производить тяжеловесные вычисления и визуализировать большие объёмы данных в браузерах. Фреймворк Superconductor основан на технологиях WebGL и WebCL, которые переносят в глобальные сети, соответственно, возможности технологий OpenGL и OpenCL и способность взаимодействовать друг с другом [10]. Однако на сегодняшний день доступны лишь экспериментальные реализации WebCL: плагин для Firefox от бывшей компании Nokia, WebKit-webcl браузер от компании Samsung, Chromium-webcl браузер от компании AMD [10]. Для серверной же стороны имеется широкий выбор инструментов, среди которых реализации стандартов OpenCL (node-opencl и node-webcl) под серверную платформу NodeJS, технологии CUDA и OpenMP.

В связи с вышеизложенным рассматриваемые алгоритмы ориентированы на реализацию на гетерогенных параллельных вычислительных системах с поддержкой параллелизма потока данных.

На первом уровне системы, основанной на знаниях, реализована система (t_{em}), динамического управления памятью, гарантирующая время $\ln^2(m)$ (m – размер памяти) для операций выделения и высвобождения участков памяти [17], а также константное среднее амортизационное время для операции перевыделения, в случае, если число добавленных в связный участок элементов данных превышает фактическое число произведённых перевыделений исходного связного участка памяти.

На втором уровне реализуются используемые при построении графа родовидовой онтологии операции пересечения и объединения (мульти)множеств.

Алгоритм 1. Операция *IntersectMultisets*(⟨*A*, *B*⟩) пересечения мультимножеств.

1. $C \leftarrow BinarySearch(\langle A, B\rangle)$.
2. Отметить (параллельно) в массиве C нулями найденные элементы массива A в массиве B, а ненайденные – единицами:

$$C[i] \leftarrow \begin{cases} i \,\middle|\, A\left[\max\left(\{0\} \cup \{i-1\}\right)\right] < A[i] \\ 0 \,\middle|\, A\left[\max\left(\{0\} \cup \{i-1\}\right)\right] = A[i] \end{cases}.$$

3. Вычислить максимумы $D = MaxScatter(C)$:

$k \leftarrow ((\sim 0) >> (clz(Length(A))+1))+1$

пока $(k > 0)$:

если $(i \geq k)$, *то* $D[i] \leftarrow \max(\{D[i]\} \cup \{D[i-k]\})$

$k >>= 1$

4. Вычислить

$$E[i] \leftarrow \begin{cases} 1 \,\middle|\, D[i]-i \leq U[i]-L[i] \\ 0 \,\middle|\, D[i]-i > U[i]-L[i] \end{cases}.$$

5. Вычислить (параллельно) массив F из префиксных сумм массива E:

$F[i] \leftarrow E[i]+PrefixSum(E)[i]$.

6. $R[F[i]-1] \leftarrow A[i]$.
7. Возвратить $\langle R, F[Length(A)-1]\rangle$.

Алгоритм 2. Операция *UniteMultisets*(⟨*A*, *B*⟩) объединения мультимножеств.

1. Выполнить поиск минимального и максимального индекса вхождения каждого элемента множеств A и B:

$\langle L, U\rangle \leftarrow RangeBinarySearch(\langle A, B\rangle)$.

2. Отметить (параллельно) в массиве C нулями найденные элементы массива A в массиве B, а ненайденные – единицами:

$$C[i] \leftarrow \begin{cases} i \,\middle|\, A\left[\max\left(\{0\} \cup \{i-1\}\right)\right] < A[i] \\ 0 \,\middle|\, A\left[\max\left(\{0\} \cup \{i-1\}\right)\right] = A[i] \end{cases}.$$

3. Вычислить максимумы $D = MaxScatter(C)$:

$k \leftarrow ((\sim 0) >> (clz(Length(A))+1))+1$

пока $(k > 0)$:

если ($i \geq k$), *то* $D[i] \leftarrow \max(\{D[i]\} \cup \{D[i-k]\})$

$k >>= 1$

4. $E[i] \leftarrow D[i] - i - U[i] + L[i]$.
5. Вычислить

$$G[i] \leftarrow \begin{cases} 0 \,|\, D[i] - i \leq U[i] - L[i] \\ 1 \,|\, D[i] - i > U[i] - L[i] \end{cases}.$$

6. Вычислить (параллельно) массив H префиксных сумм массива G:
7. $H \leftarrow PrefixSum(G)$.
8. Отметить (параллельно) в массиве R нулями найденные элементы массива A в массиве B, а ненайденные – единицами.

для i *от* 0 *до* $Length(A) + Length(B) - 1$:

$R[i] \leftarrow 1$

если ($A[i] = B[L[i] + E[i]]$), *то* $R[L[i] + E[i] + H[i]] \leftarrow 0$

9. Вычислить (параллельно) массив T префиксных сумм массива R.

$T \leftarrow PrefixSum(R)$

10. Выполнить (параллельно) для каждого элемента массива T:

если (($T[i] < Length(B)$) , ($T[\max(\{0\} \cup \{i-1\})] = T[i]$)), *то* $R[T[i]] \leftarrow B[i]$.

После этого массив R содержит все элементы массива B.

11. Выполнить (параллельно) для каждого элемента массива R:

если ($T[\max(\{0\} \cup \{i-1\})] = T[i]$), *то* $R[L[i] + E[i] + H[i]] \leftarrow A[i]$.

После этого массив R содержит и элементы массива A.

12. Возвратить $\langle R, T[Length(A) + Length(B) - 1] + 1 \rangle$.

Модель АФП для ПМ основана на модели многозначного анализа формальных понятий [[13]] и является её частным случаем. Пусть G – множество объектов предметной области, M – множество свойств, которыми могут обладать объекты G, а $V=\{0, ½, 1\}$ – множество из трёх рациональных чисел. Приближённый формальный контекст F_R определяется как тройка $\langle G, M, I_R \rangle$, где $I_R = V^{G \times M}$ – характеристический предикат. Таким образом, для любой пары $\langle x, y \rangle$, где $x \in G$ и $y \in M$, справедливо одно из следующих утверждений:

1. Если $I_R(\langle x, y \rangle)=0$, то объект x не обладает свойством y.
2. Если $I_R(\langle x, y \rangle)=1$, то объект x обладает свойством y.
3. Если $I_R(\langle x, y \rangle)=½$, то неизвестно обладает ли объект x свойством y.

Пусть $X \subseteq G$ ($Y \subseteq M$), тогда нижняя $L(X)$ ($\bar{L}(Y)$) и верхняя $U(X)$ ($\bar{U}(Y)$) аппроксимации множества атрибутов (объектов) для множества объектов (атрибутов) задаются как:

$L(X)=\{y|\forall x((x \in X) \rightarrow (I_R(\langle x,y \rangle)>0))\}$, $U(X)=\{y|\forall x((x \in X) \rightarrow (I_R(\langle x,y \rangle)=1))\}$,

$\bar{L}(Y)=\{x|\forall y((y \in Y) \rightarrow (I_R(\langle x,y \rangle)>0))\}$, $\bar{U}(Y)=\{x|\forall y((y \in Y) \rightarrow (I_R(\langle x,y \rangle)=1))\}$.

Построение графа родовидовой онтологии (таксономии) по заданному F_R сводится к трём следующим шагам:

1. Сформировать наборы экстенсионалов.
2. Сформировать наборы интенсионалов.
3. Построить граф родовидовой онтологии (таксономии).

Наборы экстенсионалов и интенсионалов рассчитываются с помощью операций объединения и пересечения множеств, исходя из того, что для каждого объекта (атрибута) известно множество атрибутов (объектов), которыми он обладает, и множество атрибутов (объектов), которыми он не обладает. Множество атрибутов (объектов), которыми обладает объединение экстенсионалов (интенсионалов) рассчитывается как пересечение множеств атрибутов (объектов), которыми обладают объекты (атрибуты) каждого экстенсионала (интенсионала), а множество объектов (атрибутов), которыми объединение экстенсионалов (интенсионалов) не обладает, рассчитывается как объединение множеств атрибутов (объектов), объекты (атрибуты) каждого экстенсионала (интенсионала) которыми не обладают.

Правило 1. Если для любого атрибута (объекта) известно, что все объекты (атрибуты) экстенсионала (интенсионала) совместно обладают им либо – нет, то среди равных по интенсионалу (экстенсионалу) экстенсионалов (интенсионалов) выбирается единственный наиболее мощный экстенсионал (интенсионал).

Правило 2. Если для каких-либо атрибутов (объектов) какого-либо экстенсионала (интенсионала) ровно для одного из его объектов (атрибутов) неизвестно обладает ли он ими, тогда как остальные объекты (атрибуты) совместно все в каждом из любых более мощных расширяющих исходный экстенсионалов (интенсионалов) обладают этими какими-либо атрибутами (объектами), то последние, более мощные, экстенсионалы (интенсионалы) оставляются, а первый исключается из рассмотрения.

Правило 3. Никаких других исключений экстенсионалов (интенсионалов), кроме как по первым двум правилам, не осуществляется и рассматриваются все остальные экстенсионалы (интенсионалы), которые можно получить путём объединения уже рассмотренных.

После того, как сформированы наборы экстенсионалов (интенсионалов), осуществляется построение онтологии, которая включает в себя онтологию экстенсионалов и интенсионалов (решётка). Граф онтологии формируется путём сопоставления каждому экстенсионалу (интенсионалу) из сформированных наборов вершины (SC-элемента) и указания связок отношений их непосредственного включения.

На третьем шаге устанавливаются связки отношения подмножества на экстенсионалах и интенсионалах и устанавливаются связки между $\langle L(X), X\rangle$, $\langle X, \hat{U}(X)\rangle$, $\langle \hat{L}(Y), Y\rangle$, $\langle Y, \hat{U}(Y)\rangle$ в соответствии с выражениями ниже.

$$\hat{L}(X) = P_L\left(L,\overline{U},\overline{L},X\right),\ \hat{U}(X) = P_U\left(U,\overline{L},\overline{U},X\right);\hat{\overline{L}}(Y) = P_L\left(\overline{L},U,L,Y\right),\ \hat{\overline{U}}(Y) = P_U\left(\overline{U},L,U,Y\right),$$

$$P_L(\alpha,\beta,\gamma,X)=\begin{cases}\alpha(X)\Big|\big(\beta(\alpha(X))=X\big)\vee\big(\gamma(\alpha(X))\supseteq\beta(\alpha(X))\big)\\ \beta(\alpha(X))\Big|\big(\beta(\alpha(X))\subset X\big)\wedge\big(\gamma(\alpha(X))\subset\beta(\alpha(X))\big)\end{cases},$$

$$P_U(\alpha,\beta,\gamma,Y)=\begin{cases}\alpha(Y)\Big|\big(\beta(\alpha(Y))=Y\big)\vee\big(\gamma(\alpha(Y))\subseteq\beta(\alpha(Y))\big)\\ \beta(\alpha(Y))\Big|\big(\beta(\alpha(Y))\supset Y\big)\wedge\big(\gamma(\alpha(Y))\supset\beta(\alpha(Y))\big)\end{cases}.$$

наборы	{a}	{b}	{c}	{d}	{a,b,d}	{a,d}	{b,c}	{b,d}	{c,d}	{a,c,d}	{b,c,d}	{a,b,c,d}
{1}	0	0	1	0	0	0	0	0	0	0	0	0
{2}	½	0	½	1	0	½	0	0	½	½	0	0
{3}	0	½	1	½	0	0	½	½	½	0	½	0
{4}	1	1	0	1	1	1	0	1	0	0	0	0
{1,2}	0	0	½	0								
{1,3}	0	0	1	0								
{1,2,3,4}	0	0	0	0								
{2,3}	0	0	½	½								
{2,4}	½	0	0	1								
{3,4}	0	½	0	½								
{1,2,3}	0	0	½	0								
{2,3,4}	0	0	0	½								

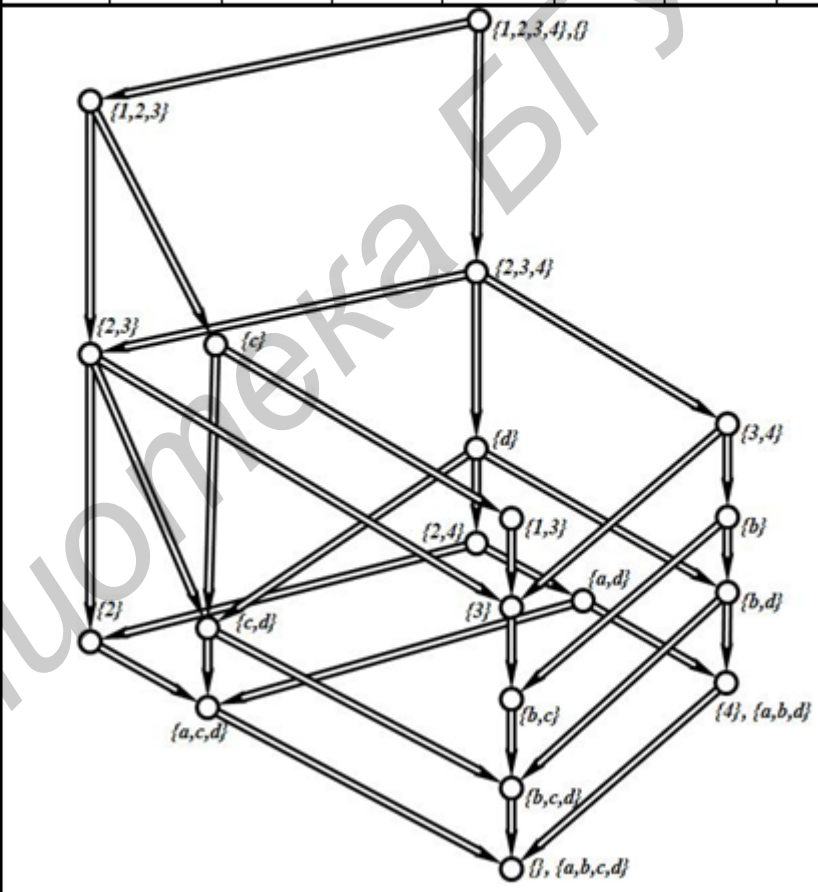

Рис. 2. Пример построения таксономии на множестве из 4-х объектов и 4-х признаков

Исключаются из онтологии все построенные связки отношения вида $\langle X, Z\rangle$ такие, что существует Y, что $\langle X, Y\rangle$ и $\langle Y, Z\rangle$ принадлежат отношению. Пример построенной онтологии изображён на рис. 2, где G={1, 2, 3, 4}, M={a, b, c, d}.

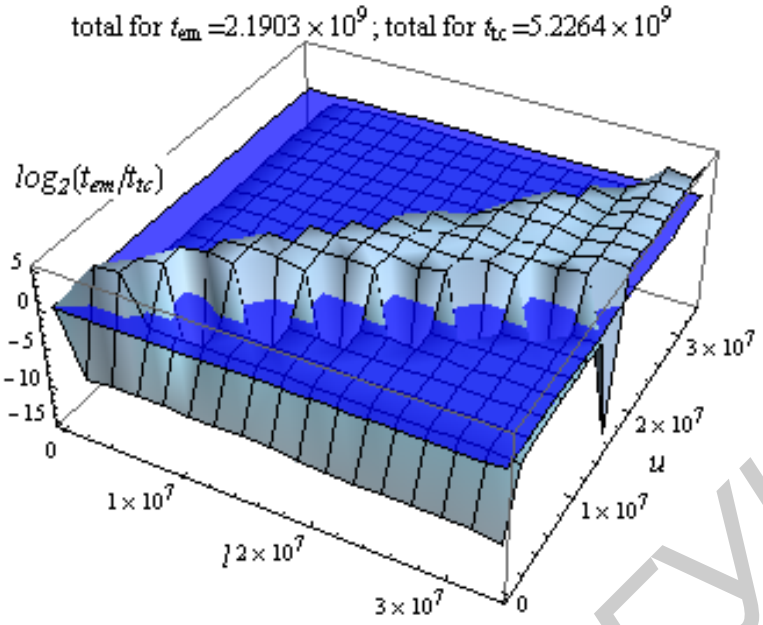

Рис. 3. График отношения времени исполнения операции перевыделения участка размером *l* ячеек на участок размером *u* ячеек, где размер ячейки равен 32 битам

На рис. 3 приведён график экспериментальных результатов сравнения работы операций нижнего уровня системы t_{em} с системой tcmalloc (t_{tc}) [15]. Для операций уровня управления данными по алгоритмам *1* и *2* временная сложность в соответствии с теоретической оценкой ожидается не более чем $O(\ln(n)*n/p+\ln^2(m))$ (n – суммарная мощность множеств, p – количество процессоров). Для операций третьего уровня – теоретическая оценка даёт выражение $O(n^2*\ln^2(n)*2^{2n}/p+\ln^2(m))$.

Литература

[1]. Божко, С.С. Некоторые алгоритмы потоковой обработки данных / Божко С.С., Пилецкий И.И., Слисенко К.Ю. // BIG DATA and Predictive Analytics. – Минск: БГУИР, 2015 С. 202–205.

[2]. Гаврилова, Т.А. Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф. Хорошевский – СПб. : Питер, 2000. – 384 с.

[3]. Голенков, В.В. Графодинамические модели параллельной обработки знаний: принципы построения, реализации и проектирования / В.В. Голенков, Н.А. Гулякина // OSTIS-2012. С. 23–52.

[4]. Голенков, В.В. Семантическая технология компонентного проектирования систем, управляемых знаниями / В.В. Голенков, Н.А. Гулякина // OSTIS. 2015. С. 57–78.

[5]. Кнут, Д. Искусство программирования: в 3х т. – М.: Вильямс, 2014. – 832 с.

[6]. Ивашенко, В.П. Модели и алгоритмы интеграции знаний на основе однородных семантических сетей / В.П. Ивашенко // OSTIS. 2015. С. 111–132.

[7]. Ивашенко, В.П. Алгоритмы полилогарифмической временной и логарифмической пространственной сложности для системы динам. распределения линейно адресуемой памяти с однородным доступом к данным / В.П. Ивашенко // Карповские научные чтения: сб. науч. ст. – Минск, 2012. – Вып. 6, ч. 1. – С. 196–201.

[8]. Ивашенко, В.П. Принципы платформенной независимости и платформенной реализации OSTIS / В.П. Ивашенко, М.М. Татур // OSTIS. 2016. С. 145–150.

[9]. Andrews, S. Analysis of Large Data Sets using Formal Concept Lattices / S. Andrews, C. Orphanides // Proc. of the 7th International Conf. on Concept Lattices and Their Applications 2010. – P. 104–115.

[10]. Kaeli, D. Heterogeneous Computing with OpenCL 2.0. / D. Kaeli, P. Mistry, et al. – MA USA, 2015.

[11]. Kuznetsov, S.O. Fitting Pattern Structures to Knowledge Discovery in Big Data / S.O. Kuznetsov // Proc. 11th International Conference on FCA, 2013. – Vol. 7880, P. 254–266.

[12]. Poelmans, J. / Fuzzy and rough formal concept analysis: a survey J. Poelmans, D. Ignatov, S. Kuznetsov, et al. // INT J GEN SYST, 2014. – Vol. 43, Issue 2, P. 105–134.

[13]. Stumme, G. FCA-MERGE: bottom-up merging of ontologies / G. Stumme, A. Maedche // Proc. of the 17th International joint conference on AI: Seattle, 4–10 Aug. 2001. – P. 225–230.

[14]. Superconductor. [Электронный ресурс]. – Режим доступа: http://superconductor.github.io/superconductor. – Дата доступа: 30.04.2016.

[15]. TCMalloc: Thread-Caching Malloc. [Электронный ресурс]. – Режим доступа: https://google-perftools.googlecode.com/svn/trunk/doc/tcmalloc.html. Дата доступа: 30.04.2016.