OSTIS-2015
(Open Semantic Technologies for Intelligent Systems)

УДК 004.822:514+004.272:43+004.272:32

ПРЕДСТАВЛЕНИЕ СЕМАНТИЧЕСКИХ СЕТЕЙ И АЛГОРИТМЫ ИХ ОРГАНИЗАЦИИ И СЕМАНТИЧЕСКОЙ ОБРАБОТКИ НА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ С МАССОВЫМ ПАРАЛЛЕЛИЗМОМ

Ивашенко В.П.[*], Вереник Н.Л.[*], Гирель А.И.[*], Сейткулов Е.Н.[**], Татур М.М.[*]

[*] *Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь*

ivashenko@bsuir.by

nick.verenik@gmail.com

tatur@i-proc.com

[**] *Евразийский национальный университет имени Л.Н. Гумилева, г. Астана, Казахстан*

seitkulov_y@enu.kz

В работе приводится описание способов и некоторых алгоритмов организации хранения и представления семантических сетей в архитектуре проблемно-ориентированного семантического процессора. Рассмотрена возможность применения разработанных алгоритмов для решения прикладных задач.

Ключевые слова: массовый параллелизм; семантическая сеть; семантическая обработка информации; каноническая разметка графа; интеграция знаний.

Введение

К системам искусственного интеллекта, или интеллектуальным системам, обычно относят системы, способные решать задачи на семантическом уровне, связанные с анализом и обработкой знаний в некоторой заданной предметной области. Примерами могут служить различные системы поддержки принятия решений, экспертные и поисковые системы, направленные на широкий спектр прикладных задач, таких как: управление взаимоотношениями с клиентами (CRM-системы), планирование ресурсов предприятий (ERP-системы), аналитический мониторинг контента, массовое обслуживание (контроль информационных потоков) в сетях и др. В настоящее время при построении сложных информационных систем акцент проблемы все больше смещается от физического аспекта, связанного с хранением и передачей данных, СУБД, к семантическому. В основе любой «интеллектуальной» модели обработки информации можно выделить ряд актуальных базовых задач, таких как представление знаний (например, в виде семантической сети), семантический анализ информации, ассоциативный поиск информации по некоторому ключу, модификация семантической сети в соответствии с результатом обработки информации.

Под семантической сетью понимается графовая структура, вершины и дуги которой в соответствии с определенными правилами наделены некоторой смысловой нагрузкой. Аппарат семантических сетей обеспечивает не только визуализацию структуры информации, но, в первую очередь, позволяет разрабатывать формальные методы и алгоритмы анализа и модификации знаний. Характерной чертой решаемых задач является их высокая структурная и динамическая сложность, общее решение таких задач зачастую сводится к классу трансвычислительных: уже при относительно небольшом числе вершин графа (узлов семантической сети) время решения методом перебора превышает возможности любых теоретически мыслимых вычислительных систем. Однако с учетом специфики частных задач, а также в результате формулировки задачи на языке теории графов, позволяющих обеспечить высокую степень ассоциативности, можно получить более простые методы решения, часто практически реализуемые.

Информационный взрыв (в частности, переход к многоуровневой иерархической структуре знаний, а

также активное использование метаинформации) привел к резкому увеличению объёмов хранимых и обрабатываемых знаний, значительно усложнил их структуру, и, следовательно, объём и структуру семантических сетей. Полученные в результате семантические сети обладают сложной нерегулярной структурой, а используемые алгоритмы обработки информации различаются от системы к системе, используя большое количество эвристик, построенных на особенностях конкретной прикладной системы. Все это обусловило проблему создания эффективной аппаратной платформы с параллельной архитектурой, ориентированной на данный класс задач [Guo et al., 2009], [Allemang et al., 2008], [Kitano et al., 1992], [Chung et al., 1995], [Голенков и др., 2012].

Известны попытки реализовать алгоритмы семантической обработки с применением серийных аппаратных платформ с параллельной архитектурой, таких как графические ускорители (GPU) [Brodkorb et al., 2013], [Navarro et al., 2014], кластеры и суперЭВМ [Каляев и др., 2012], [Kitano et al., 1992], [Chung et al., 1995], [Jan et al., 2012]. В случаях, когда использование GPU не ведёт к кардинальному повышению производительности по сравнению с персональными компьютерами, а использование мощных кластеров или суперЭВМ на практике зачастую невозможно по экономическим, тактико-техническим, массогабаритным и прочим характеристикам, в качестве альтернативы могут быть разработаны и применены специализированые вычислительные системы, ориентированные на решение задач обработки знаний. Однако на сегодняшний день подобные комплексы представлены лишь в виде единичных лабораторных образцов, дорогих и сложных в применении, а общедоступных систем такого класса на рынке серийных устройств не существует.

Своеобразной «золотой серединой» между универсальными системами и спецпроцессорами, ориентированными на обработку знаний, являются проблемно-ориентированные процессоры [Tatur et al., 2010]. Основная идея в их использовании состоит в том, чтобы обеспечить унификацию процессора в «некоторых» рамках, сохранив при этом достаточный уровень производительности. В то же время достигаемые технические характеристики и издержки на обеспечение универсальности оригинальной архитектуры должны быть конкурентными по сравнению со спецпроцессорами и серийными параллельными процессорами (многоядерными CPU, GPU, DSP).

В статье приводится описание разработанной оригинальной архитектуры ассоциативного процессора (обозначим ее как SNP, Semantic Network Processor) и процесса ее интеграции в существующую интеллектуальную систему (ИС). Основной задачей, решаемой архитектурой, является та самая унификация процесса решения интеллектуальных задач, сохранив при этом некоторую независимость от аппаратной платформы. Так, на текущий момент, была разработана программная реализация процессора (на основе key-value БД RocksDB) и реализация на основе GPU (технология CUDA), с учетом выполнения как на одном графическом ускорителе, так и кластере GPU. В дальнейшем планируется реализовать архитектуру в виде проблемно-ориентированного процессора (на базе FPGA), что позволит сильно сократить стоимость одной ячейки памяти.

В результате, заказчик получает возможность построить интеллектуальную систему на основе SNP (в идеале используя банк готовых базовых решений) и выбрать подходящую под технические требования аппаратную платформу. При изменении технического задания (возросшем объеме базы знаний, изменении требований ко времени решений и т.д.) смена аппаратной платформы не должна повлиять на программную реализацию самой системы.

В системах, основанных на знаниях [Гаврилова и др., 2000], одной из важных задач является задача интеграции знаний, заключающаяся в выявлении фрагментов семантической сети, представляющих одни и те же знания, и их слиянии, что позволяет исключить избыточные элементы семантической сети, способные затормозить до предела работу системы, или сократить их количество, увеличить число ассоциаций для конкретных понятий, уточнить структуру модели предметной области. Задача интеграции необходима для обеспечения производительности и развития интеллектуальных систем в условиях неполноты знаний [Нариньяни, 2000], которая в свою очередь неизбежна в открытых, обучающихся интеллектуальных системах. Для решения задачи в условиях неполноты предложен ряд моделей и подходов [Ивашенко, 2007, 2009b, 2011b, 2012a, 2013b], [Кудрявцев, 2008] , [Кофман, 1982], [Тэрано, 1993], [Doan and Havely, 2005], [Giunchiglia et al., 2006], [Jean-Mary et al., 2007], [IDEF5, 1994], [Stumme, 2001]. Однако даже в этих условиях некоторые фрагменты знаний могут быть представлены полностью и так же важно выявить дублирование таких фрагментов с целью устранения избыточных знаний. Для выявления такого дублирования предлагается подход на основе установления сходства элементов дублирующихся фрагментов, использующий каноническую разметку графа семантической сети. Каноническая разметка графа семантической сети также может быть применена для решения частного случая задачи поиска по образцу, когда искомый граф изоморфен образцу.

1. Проблемно-ориентированный семантический процессор

1.1. Архитектура

Ассоциативная вычислительная система (ВС) есть не что иное, как одна из разновидностей параллельных вычислительных систем SIMD класса

(рисунок 1), в которых n процессорных элементов ПЭ представляют собой простые устройства, как правило, последовательной поразрядной обработки. При этом каждое слово (ячейка) ассоциативной памяти имеет свое собственное устройство обработки данных (сумматор). Операция осуществляется одновременно всеми n ПЭ. Все или часть элементарных последовательных ПЭ могут синхронно выполнять операции над всеми ячейками или над выбранным множеством слов ассоциативной памяти.

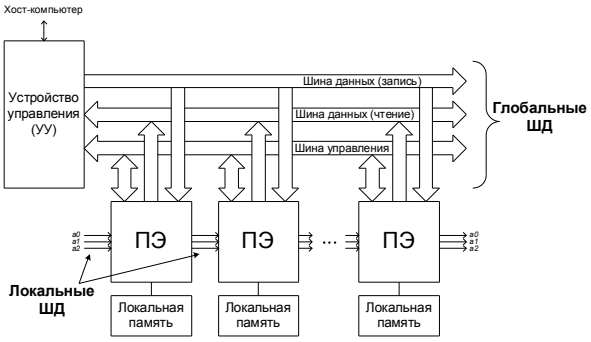

Рисунок 1 – Структурная ассоциативного процессора

Время обработки N m-разрядных слов в ассоциативной ВС определяется выражением [Цилькер и др., 2007]:

$$T = mt\left(\frac{N}{n} + K\right),$$

где t – время цикла ассоциативной памяти; n – число ячеек ассоциативной системы; K – коэффициент сложности выполнения элементарной операции (количество последовательных шагов, каждый из которых связан с доступом к памяти). Таким образом, время обработки T является константой и в большей мере зависит от величины $\frac{N}{n}$, т.е. количества ячеек памяти, приходящихся на один ПЭ. После фиксации величины $\frac{N}{n}$ (выбора конфигурации нитей GPU или настройка схемы ПЭ на FPGA [Verenik et al., 2014]) время обработки остается неизменным вне зависимости от общего объема обрабатываемой памяти.

SNP представляет собой классическую SIMD архитектуру, в которой каждый ПЭ содержит схему характерную для ассоциативной памяти (рисунок 2). На схеме SNP представлен одним устройством управления (УУ), обеспечивающим последовательное выполнение команд программы, которые транслируются множеству процессорных элементов, каждый из которых обрабатывает свои данные. Коммуникация с внешней средой осуществляется посредством глобальных шин данных (ШД):

- ШД записи – шина, по которой исполняемая команда параллельно транслируется на все ПЭ;
- ШД чтения – шина, по которой выполняется последовательное чтение данных с процессорных элементов (ШД с временным разделением);
- шина управления (ШУ) – множество управляющих сигналов.

Каждый ПЭ представляет собой ассоциативное запоминающее устройство (АЗУ) и включает в себя (см. рисунок 2):

- запоминающий массив для хранения N m-разрядных слов;
- регистр ассоциативного признака, куда помещается код искомой информации (признак поиска). Разрядность регистра k равна длине слова m;
- схемы совпадения, используемые для параллельного сравнения каждого бита всех хранимых слов с соответствующим битом признака поиска и выработки сигналов совпадения;
- регистр совпадений, где каждой ячейке запоминающего массива соответствует один разряд, в который заносится единица, если все разряды соответствующей ячейки совпали с одноименными разрядами признака поиска;
- комбинационную схему, которая на основании анализа содержимого регистра совпадений формирует сигналы, извещающие о том, что искомая информация:
  - *a0* – не найдена;
  - *a1* – содержится в одной ячейке;

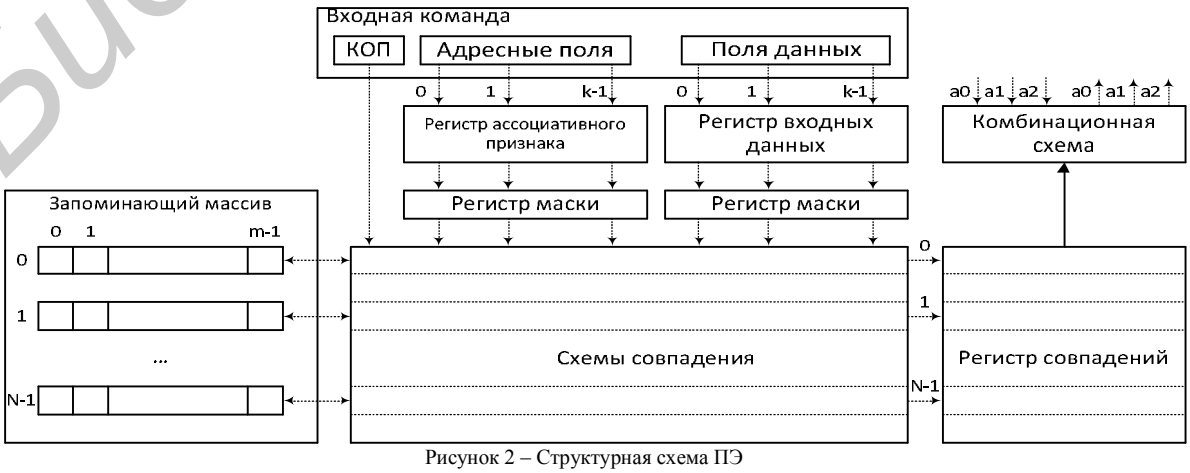

Рисунок 2 – Структурная схема ПЭ

- *a2* – содержится более чем в одной ячейке.

- регистр входных данных, куда помещаются данные для записи в память (запись производится в ячейки, подошедшие согласно признаку поиска). Разрядность регистра также равна *k*;
- регистры маски, позволяющие запретить сравнение определенных битов при поиске и изменения состояния определенных битов при записи;

Формируемые на выходе ПЭ сигналы *a0*, *a1*, *a2* поступают на вход последующего ПЭ, который, в свою очередь, учитывает их при формировании своих выходных сигналов. Тем самым, линейка ПЭ образует единую ассоциативную память, результирующие сигналы которой будут сформированы на выходе последнего ПЭ. Процессор является хорошо масштабируемым, позволяя наращивать объем данных системы простым подключением ПЭ к последнему в массиве. Сохраняется основное преимущество ассоциативной памяти – увеличение количества ПЭ приводит к увеличению общего объема памяти без увеличения результирующего времени обработки данных. А в случае использования процессора в качестве вычислительного ядра интеллектуальной системы из-за огромных размеров баз знаний эта особенность может оказаться решающей.

1.2. Представление команд и данных

Система команд SNP представлена всего одной командой (рисунки 2, 3).

Рисунок 3 – Формат команды SNP

Команды хранятся отдельно от данных, более подробное описание работы команды см. [Verenik et al., 2014].

Данными является классический двудольный граф регулярной структуры. Формат хранения данных определяется системный программистом, подробный пример см. в [Вереник и др., 2013]. Порядок преобразования семантической сети в «регулярный» граф схематично проиллюстрирован на рисунках 4 и 5 [Вереник и др., 2012].

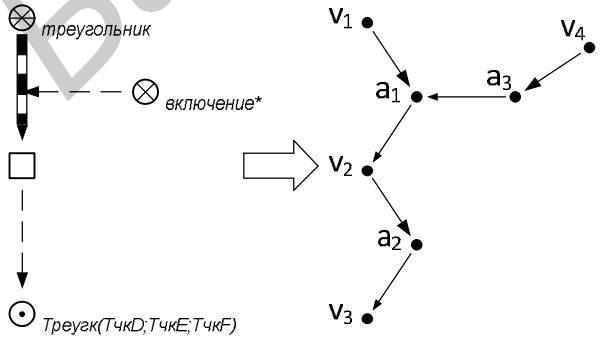

Рисунок 4 – Преобразование исходного графа семантической сети к орграфу

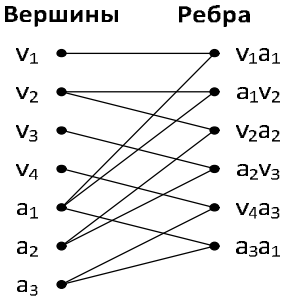

Рисунок 5 – «Регулярный» граф

2. Представление и каноническая разметка семантических сетей

Для решения задачи канонической разметки [Babai, 1980], [Hartke, 2009] графа семантической сети в работах [Ивашенко, 2007, 2013b] использовалось представление семантической сети в виде ориентированного псевдографа, основанное на преобразовании в соответствии со следующей схемой.

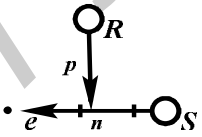

Рисунок 6 – Структура из пяти элементов, семантическая сеть

Фрагмент семантической сети, конструкция, структура из пяти элементов, изображённая на рисунке 6 в этом случае представляется в виде, изображённом на рисунке 7. В отличие от представления графа в SNP начало ребра семантической сети связывается с ним двумя встречными дугами, что может позволить обеспечить временную сложность поиска начала ребра меньшую, чем линейная.

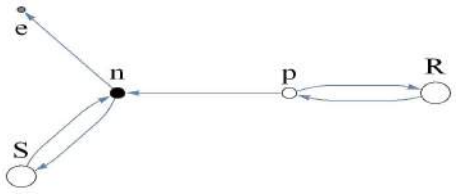

Рисунок 7 – Структура из пяти элементов, размеченный орграф

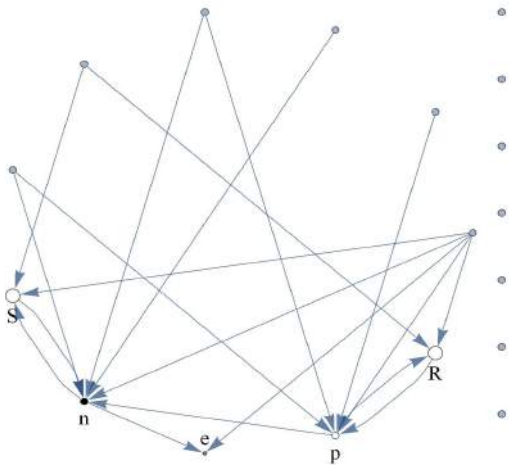

Рисунок 8 – Структура из пяти элементов и их типы, орграф

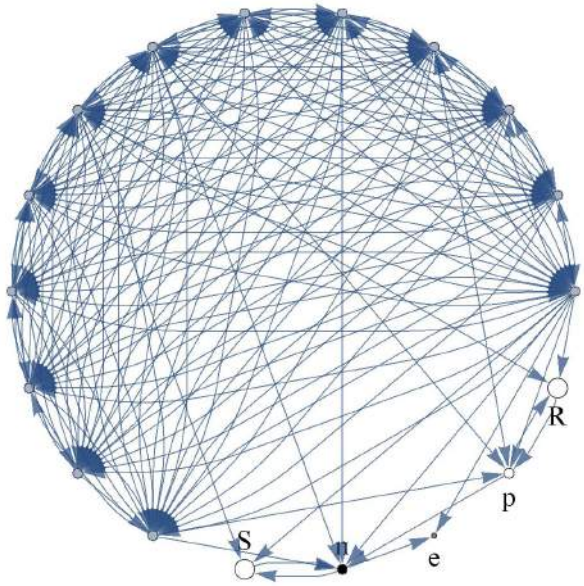

Рисунок 9 – Структура из пяти элементов, связный орграф

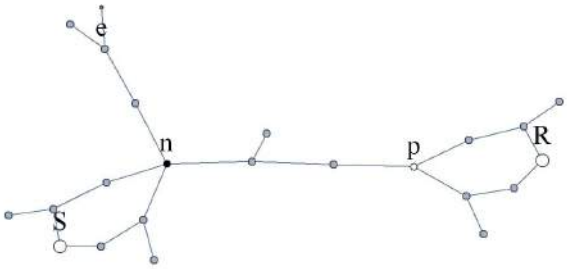

Рисунок 10 – Структура из пяти элементов, граф

Такое представление является структурно неоднородным, так как необходимо дополнительно хранить информацию о типах элементов семантической сети в неграфовом виде, например в виде меток. Для представления информации о типах элементов в графовом виде алгоритмы канонической разметки используют дополнительные вершины, кодирующие информацию о типах. Всего используется тринадцать вершин, позволяющих закодировать 2^{13} типов (рисунок 8). Для отделения и отличия этих вершин от вершин, представляющих элементы семантической сети, они связываются дугами в полный ориентированный граф (орграф), из которого вычтена упорядочивающая их цепочка, позволяя получить связный орграф (рисунок 9). При необходимости перейти от ориентированного псевдографа к неориентированному графу каждая дуга ориентированного псевдографа может быть заменена тремя новыми дугами и тремя новыми вершинами (рисунок 10).

При реализации алгоритмов канонической разметки была разработана библиотека алгоритмов преобразования графовых структур для разных представлений, включая матрицу смежности, форматы graph6, sparse6 и другие. В том числе поддерживается версия формата Transfer Graph Format (TGF), в котором хранится исходная семантическая сеть. Так же существуют конверторы в формат TGF и из него в формат SCs, который поддерживает представление конструкций языков модели унифицированного семантического представления знаний [Голенков и др., 2001], [Гулякина и др., 2004], [Ивашенко, 2003, 2004, 2009a, 2011a, 2013a, 2013b], [OSTIS, 2014]. Алгоритм преобразования семантической сети в графовое представление полагается на следующие условия:

- информация о типе и неграфовом содержимом элементов семантической сети таком, как системный идентификатор и содержимое, хранится локально, совместно с элементом семантической сети, то есть не требуется буферизировать эту информацию, чтобы обеспечить её связность;
- информация о содержимом элемента может быть указана независимо от команды генерации этого элемента;
- большинство алгоритмов имеют интерфейс генерации узловых элементов семантической сети и дуг с явным указанием инцидентных элементов;
- возможны нетривиальные структуры на семантической сети, например, когда каждая из двух дуг является концом другой.

$T \leftarrow$
$Z \leftarrow T$
$Y \leftarrow \varnothing$
пока $Y \subset Z$
  $X \leftarrow Z$
  $Z \leftarrow Y$
  $Y \leftarrow \varnothing$
  пока $\varnothing \subset X$
    для $\forall x \forall t \left(x \in X\right) \wedge \left(t = type\left(\left\langle T, x \right\rangle\right)\right)$
      если $x \in edges\left(T\right)$, то
        если $map\left(beg\left(\left\langle T, x \right\rangle\right)\right) \wedge map\left(end\left(\left\langle T, x \right\rangle\right)\right)$, то
          $b \leftarrow map\left(beg\left(\left\langle T, x \right\rangle\right)\right)$
          $e \leftarrow map\left(end\left(\left\langle T, x \right\rangle\right)\right)$
          $map\left(\left\langle x, gen\left(\left\langle x, t, b, e \right\rangle\right)\right\rangle\right)$
        иначе если $map\left(beg\left(\left\langle T, x \right\rangle\right)\right) \wedge \left(end\left(\left\langle T, x \right\rangle\right) = x\right)$, то
          $b \leftarrow map\left(beg\left(\left\langle T, x \right\rangle\right)\right)$
          $map\left(\left\langle x, gen\left(\left\langle x, t, b, x \right\rangle\right)\right\rangle\right)$
        иначе $Y \leftarrow Y \cup \left\{x\right\}$
      иначе $map\left(\left\langle x, gen\left(\left\langle x, t \right\rangle\right)\right\rangle\right)$
      если $map\left(x\right)$, то
        если $\exists i \left(i = identifier\left(x\right)\right)$, то
          $identifier\left(\left\langle map\left(x\right), i \right\rangle\right)$
        если $\exists c \left(c = content\left(x\right)\right)$, то
          $content\left(\left\langle map\left(x\right), c \right\rangle\right)$
$Y \leftarrow \varnothing$
пока $Y \subset Z$
  $X \leftarrow Z$
  $Z \leftarrow Y$
  $Y \leftarrow \varnothing$
  пока $\varnothing \subset X$
    для $\forall x \forall t \left(x \in X\right) \wedge \left(t = type\left(\left\langle T, x \right\rangle\right)\right)$

```
если map(beg(⟨T,x⟩)) ∧
   map(end(⟨T,x⟩)), то
   b ← map(beg(⟨T,x⟩))
   e ← map(end(⟨T,x⟩))
   если map(x), то
      set_end(⟨map(x),e⟩)
   иначе map(⟨x,gen(⟨x,t,b,e⟩)⟩)
   если ∃i(i = identifier(x)), то
      identifier(⟨map(x),i⟩)
   если ∃c(c = content(x)), то
      content(⟨map(x),c⟩)
иначе если map(beg(⟨T,x⟩)), то
   b ← map(beg(⟨T,x⟩))
   map(⟨x,gen(⟨x,t,b⟩)⟩)
   Y ← Y ∪ {x}
```

Здесь T – текст с элементами семантической сети, множество рёбер и вершин, x – элемент, вершина или ребро семантической сети, $type(\langle T,x\rangle)$ – тип элемента семантической сети, множество меток, в тексте T, $edges(T)$ – множество рёбер в тексте T, $map(\langle x,\alpha\rangle)$ – отобразить x на α, $map(x)$ – получить образ x, $beg(\langle T,x\rangle)$ – начало ребра в тексте T, $end(\langle T,x\rangle)$ – конец ребра в тексте T, $set\_end(\langle\alpha,e\rangle)$ – установить для элемента x конец e, $gen(\langle x,t\rangle)$ – сгенерировать элемент x с типом t, $gen(\langle x,t,b,e\rangle)$ – сгенерировать элемент x с типом t с началом b и концом e, $gen(\langle x,t,b\rangle)$ – сгенерировать элемент x с типом t с началом b, $identifier(x)$ – идентификатор элемента x, $identifier(\langle\alpha,i\rangle)$ – установить элементу α идентификатор i, $content(x)$ – содержимое элемента x, $content(\langle\alpha,c\rangle)$ – установить элементу α содержимое c.

Для решения задачи канонической разметки используется алгоритм Л. Бабаи [Babai, 1980].

```
⟨V,E⟩ ←
n ← |V|
r ← ⌈3 * ln n / ln 2⌉
f ← true
для (i ← 1; i < n; i++)
   p(i) ← i
   d1 ← 2 * Σ_{j=1}^{r} E(i,j)
   d2 ← 2 * (n − n²) * Σ_{j=1}^{r} (E(i,j) * E(j,i))
   d3 ← 2 * n² * Σ_{j=1}^{r} (E(i,j) + E(j,i))
   d4 ← (n² + (n+1)²) * E(i,i)
   d(i) ← d1 + d2 + d3 − d4
b ← sort(⟨d,1,n⟩)
для (i ← 1; i < r; i++)
   если d(b(i)) = d(b(i+1)), то f ← false
если f, то для (i ← r+1; i ≤ n; i++)
   k ← b(i)
   q(i) ←
   Σ_{j=1}^{r} (E(b(j),k) + (2 + E(b(j),k)) * E(k,b(j))) * 8^j
если f, то p ← sort(⟨q,r+1,n⟩)
если f, то для (i ← r+1; i < n; i++)
   если q(p(i)) = q(p(i+1)), то f ← false
← ⟨f, b ∘ p⟩
```

Здесь V – множество вершин ориентированного псевдографа, E – матрица (функция) смежности его вершин, $sort(\langle\alpha,\beta,\gamma\rangle)$ – возвращает отображение, сортирующее элементы α от индекса β до индекса γ, $\alpha\circ\beta$ – композиция отображений α и β.

Если с помощью него не удаётся получить разметку, то используется следующий рекурсивный алгоритм (*canonize*).

```
⟨V,E,m,p⟩ ←
n ← |V|
⟨m,p⟩ ← subclassify(⟨V,E,m,p⟩)
для (i ← 1; i < n; i++)
   если m(i) > i+1, то
      k ← m(i)
      s ← m
      s(i) ← i+1
      ⟨s,pi⟩ ← canonize(⟨V,E,s,p⟩)
      для (j ← i+1; j < k; j++)
         b ← p
         swap(⟨b,i,j⟩)
         s ← m
         s(i) ← i+1
         ⟨s,pj⟩ ← canonize(⟨V,E,s,b⟩)
         если less(⟨E,pi,pj⟩), то
            swap(⟨p,j−−,−−k⟩)
         иначе если less(⟨E,pj,pi⟩), то
            swap(⟨p,i,j⟩)
            j ← i+1
         иначе j++
← ⟨m, order(⟨V,E,m,p⟩)⟩
```

Здесь $canonize(\langle V,E,s,\alpha\rangle)$ – рекурсивный вызов, $less(\langle E,pi,pj\rangle)$ – сравнение матрицы E в перестановке pi с этой же матрицей в перестановке pj, $swap(\langle\alpha,\beta,\gamma\rangle)$ – обмен значений отображения α по индексам β и γ.

При реализации этого рекурсивного алгоритма на параллельной системе, возможен выигрыш, так как рекурсивные вызовы в алгоритме не зависят по данным, однако для реализации требуется наличие эффективных средств поддержки параллелизма независимых ветвей, разделяющих общие данные по чтению.

Рекурсивный алгоритм использует два алгоритма. Первый (*subclassify*) основан на подразбиении множества вершин исходного графа на основе анализа их степеней в подграфах,

образованных вершинами множеств разбиения. В некоторых случаях он позволяет получить полное решение за однократное применение.

$\langle V, E, m, p\rangle \leftarrow$
$n \leftarrow |V|$
для $(k \leftarrow 1; k \le n; k \leftarrow i)$
  $q \leftarrow k$
  для $(i \leftarrow 1; i < m(k); i{+}{+})$
    если $(E(p(i), p(i)) > 0)$, то
      $swap(\langle p, q{+}{+}, i\rangle)$
  пока $(k < q)$ $m(k{+}{+}) \leftarrow q$
  $s(i) \leftarrow 1$
пока $s \ne m$
  $s \leftarrow m$
  для $(ilb \leftarrow 1; ilb \le n; ilb \leftarrow u)$
    $u \leftarrow m(ilb)$
    для $(jlb \leftarrow ilb; jlb \le n; jlb \leftarrow m(jlb))$
      для $(i \leftarrow ilb; i < ulb; i{+}{+})$
        $q \leftarrow p(i)$
        $d1 \leftarrow \sum_{j=jlb}^{m(jlb)} (E(q, p(j)) + E(p(j), q))$
        $d2 \leftarrow \sum_{j=jlb}^{m(jlb)} (E(q, p(j)) * E(p(j), q))$
        $d3 \leftarrow \sum_{j=jlb}^{jlb} E(q, p(j))$
        $d(i) \leftarrow n^2 * d1 + (n - n^2) * d2 + d3$
      $b \leftarrow sort(\langle d, ilb, u\rangle)$
      $p \leftarrow p \circ b$
      для $(i \leftarrow u - 1; i \ge ilb; i{-}{-})$
        $m(i) \leftarrow u$
        если $d(b(i-1)) > d(b(i))$, то $u \leftarrow i$
$\leftarrow \langle m, p\rangle$

По теоретическим оценкам временная сложность этого алгоритма в случае параллельной реализации, если не учитывать временные затраты на доступ к памяти [Ивашенко, 2012b], может быть улучшена с $O(n^3)$ до $O((n^3 * \ln^3 p)/p^2)$, где n – количество вершин, а p – количество процессорных элементов.

Второй алгоритм (*order*) сортирует вершины во множествах разбиения в результате анализа матрицы смежности.

$\langle V, E, m, p\rangle \leftarrow$
$n \leftarrow |V|$
для $(i \leftarrow 1; i < n; i{+}{+})$
  если $m(i) > i + 1$, то
    для $(k \leftarrow i + 1; k < n; k \leftarrow u)$
      $u \leftarrow m(k)$
      $q \leftarrow k$
      для $(j \leftarrow k; j < u; j{+}{+})$
        если
        $(E(p(i), p(j)) * E(p(j), p(i)) > 0)$,
        то $swap(\langle p, q{+}{+}, j\rangle)$
      пока $(k < q)$ $m(k{+}{+}) \leftarrow q$
      для $(j \leftarrow k; j < u; j{+}{+})$
        если $(E(p(i), p(j)) > 0)$, то
        $swap(\langle p, q{+}{+}, j\rangle)$
      пока $(k < q)$ $m(k{+}{+}) \leftarrow q$
    $m(i) \leftarrow i + 1$
$\leftarrow p$

По теоретическим оценкам временная сложность второго алгоритма (*order*) в случае параллельной реализации, если не учитывать временные затраты на доступ к памяти, может быть улучшена с $O(n^2)$ до $O((n^2 * \ln p)/p)$. Однако, для достижения наиболее эффективных результатов в архитектуре вычислительной системы, которая будет исполнять этот алгоритм, должны быть предусмотрены эффективные средства исполнения операций редукции и параллельной (битонической) сортировки. В случае, когда такой поддержки нет, выигрыша может не быть или он будет незначительным.

Заключение

Рассмотрены основные принципы организации архитектуры проблемно-ориентированного процессора, предназначенного для решения задач на графах, рассмотрена задача канонической разметки семантической сети и алгоритмы её решения, приведена теоретическая оценка результатов параллельной реализации этой задачи.

Библиографический список

[Вереник и др., 2012] Разработка проблемно-ориентированных процессоров семантической обработки информации / Н. Л. Вереник, Е. Н. Сейткулов, М. М. Татур // Электроника-инфо. – 2012. – №8. – С. 95–98.

[Вереник и др., 2013] Имитационная модель векторного процессора на примере задачи поиска пути в графе / Н.Л. Вереник, Е. Н. Сейткулов, М. М. Татур // Искусственный интеллект. – 2013. – №4. – с.89–100.

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

[Голенков и др, 2001] Голенков, В.В. Представление и обработка знаний в графодинамических ассоциативных машинах /В. В. Голенков[и др.] – Мн. : БГУИР, 2001.

[Гулякина и др., 2004] Гулякина Н.А. Ивашенко В.П. Интеграция знаний в информационных системах. / Н.А. Гулякина, В.П. Ивашенко // Доклады БГУИР. – 2004. – №6. – С. 113-119.

[Голенков и др., 2012] Графодинамические модели параллельной обработки знаний: принципы построения, реализации и проектирования. / В. А. Голенков, Н. А. Гулякина // Открытые семантические технологии проектирования интеллектуальных систем: материалы II Междунар. науч.-техн. конф. (Минск, 16-18 февраля 2012 г.) / редкол. : В.В. Голенков (отв. ред.) [и др.] – Минск, БГУИР, 2012 – С. 23–52.

[Ивашенко, 2003] Представление нейронных сетей и систем продукций в однородных семантических сетях. / В.П. Ивашенко // Известия Белорусской инженерной академии. – 2003. – №1(15)/1. – с.184-188.

[Ивашенко, 2004] Применение однородных семантических сетей для представления знаний о нестационарных предметных областях. / В.П. Ивашенко // Известия Белорусской инженерной академии. – 2004. – №1(17)/3. – с.77-80.

[Ивашенко, 2007] Разработка баз знаний семантических интеллектуальных систем / В.П. Ивашенко // Дистанционное обучение – образовательная среда XXI века : материалы VI

Междунар. науч.-метод. конф., Минск, 22–23 нояб. 2007 г. / Белорус. гос. ун-т информатики и радиоэлектроники. – Минск, 2007. – С. 180–182

[Ивашенко, 2009a] Ивашенко В.П. Семантические модели баз знаний / В.П. Ивашенко Информационные системы и технологии (IST'2009): материалы V Междунар. конф.-форума в 2-х ч. Ч.2 – Минск: А.Н.Вараксин, 2009.- с.125-128.

[Ивашенко, 2009b] Ивашенко В.П. Алгоритмы верификации и интеграции баз знаний. Вестник Брестского государственного технического университета, БрГТУ, 2009, №5.

[Ивашенко, 2011a] Ивашенко В.П. Семантическая технология компонентного проектирования баз знаний. Материалы Международной научн.-техн. Конференции OSTIS,2011:Минск, Республика Беларусь, БГУИР 10-12 февраля 2011.

[Ивашенко, 2011b] Ивашенко В.П. Алгоритмы операций отладки и интеграции баз знаний. Дистанционное обучение – образовательная среда XXI века: материалы VII Международной научн.-метод. конференции (Минск, 1-2 декабря 2011г.). – Минск: БГУИР, 2011, сс.227-229.

[Ивашенко, 2012a] Ивашенко В.П. Семантическая модели интеграции и отладки баз знаний. Материалы Международной научн.-техн. Конференции OSTIS, 2012:Минск, Республика Беларусь, БГУИР 16-18 февраля 2012.

[Ивашенко, 2012b] Ивашенко, В.П. Алгоритмы полилогарифмической временной и логарифмической пространственной сложности для системы динамического распределения линейно адресуемой памяти с однородным доступом к данным / В.П. Ивашенко // Карповские научные чтения : сб. науч. ст. / Белорус. гос. ун-т; редкол. : А.И. Головня (отв. ред.) [и др.]. – Минск, 2012. – Вып. 6, ч. 1. – С. 196–201.

[Ивашенко, 2013a] Ивашенко В.П. Унифицированное представление и интеграция знаний. Материалы Международной научн.-техн. Конференции OSTIS, 2013:Минск, Республика Беларусь, БГУИР 20-23 февраля 2013.

[Ивашенко, 2013b] Ивашенко В.П. Интеграция на основе унифицированного представления знаний / В. П. Ивашенко // Электроника инфо. – 2013. – № 10. – С. 37–46.

[Каляев и др., 2012] Высокопроизводительные реконфигурируемые вычислительные системы нового поколения / И.А. Каляев, А. И. Дордопуло, И.И. Левин, Е.А. Семерников // Вычислительные методы и программирование новые вычислительные технологии. – 2011. – Т. 12. – №2. – С. 82–89.

[Кофман, 1982] Кофман А. Введение в теорию нечетких множеств. М.: Радио и связь, 1982. — 432 с.

[Кудрявцев, 2008] Кудрявцев Д. В. Практические методы отображения и интеграции онтологий. Семинар Знания и онтологии *Elsewhere*, КИИ-2008, Дубна, 2008.

[Нариньяни, 2000] Нариньяни А. С. НЕ-факторы: неточность и недоопределенность — различие и взаимосвязь // Изв. РАН. Сер. Теория и системы управления. 2000. № 5. С. 44-56.

[Тэрано, 1993] Прикладные нечеткие системы: Пер. с япон. / К. Асаи, Д. Ватада, С. Иваи и др.; Под ред. Т. Терано, К. Асаи, М. Сугэно. – М.: Мир, 1993.

[Цилькер и др., 2007] Цилькер Б.Я., Орлов С.А. Организация ЭВМ и систем: Учебник для вузов. – СПб.: Питер, 2007. – 668 с.

[Allemang et al., 2008] Dean Allemang, James Hendler. Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. *Elsevier*. – 2008. ISBN: 978-0-12-373556-0.

[Babai, 1980] Babai, L. Random graph isomorphism / L. Babai, P. Erdus, M. Stanley // SIAM J. on Computing. – 1980. – Vol. 9. – P. 628–635.

[Brodkorb et al., 2013] André R. Brodtkorb, Trond R. Hagen, Martin L. Sætra. Graphics processing unit (GPU) programming strategies and trends in GPU computing. *Journal of Parallel and Distributed Computing*, 73(1):4–13, January 2013.

[Chung et al., 1995] Minhwa Chung, Dan Moldovan. Parallel natural language processing on a semantic network array processor. *IEEE Transactions on Knowledge and Data Engineering*, 7(3):391–405, June 1995.

[Doan and Halevy, 2005] A. H. Doan, A. Y. Halevy. Semantic integration research in the database community: A brief survey. AI magazine, 26(1), 2005

[Giunchiglia et al., 2006] F. Giunchiglia, M. Marchese and I. Zaihrayeu (2006). Encoding classifications into lightweight ontologies. University of Trento Technical Report # DIT-06-016, March 2006.

[Guo et al., 2009] Qinglin Guo, Ming Zhang. Question answering based on pervasive agent ontology and Semantic Web. *Knowledge-Based Systems*. – August 2009. – Vol. 22. – Issue 6. – P. 443–448.

[**Hartke, 2009**] Hartke, S.G. McKay's canonical graph labeling algorithm / S.G. Hartke, A.J. Radclife // Communicating Mathematics. – 2009. – Vol. 479. – P. 99–111.

[IDEF5, 1994] Information Integration for Concurrent Engineering (IICE). IDEF5 Method Report. - Knowledge Based Systems, Inc. , 1408 University Drive East College Station, Texas, USA. September 21. 1994.

[Jan et al., 2012] Yahya Jan, Lech Jóźwiak. Scalable communication architectures for massively parallel hardware multi-processors *Journal of Parallel and Distributed Computing*. – November 2012. – Vol. 72. – Issue 11. – P. 1450–1463.

[Jean-Mary et al., 2007] Jean-Mary Y., Kabuka, M. ASMOV: Ontology Alignment with Semantic Validation. Joint SWDB-ODBIS Workshop, September 2007, Vienna, Austria, 15-20.

[Kitano et al., 1992] Semantic network array processor as a massively parallel computing platform for high performance and large-scale natural language processing. In *Proc. Int'l Conf. on Computational Linguistics (COLING '92)*, volume 2, pages 813–819, 1992.

[Navarro et al., 2014] Cristobal A. Navarro, Nancy Hitschfeld-Kahler, Luis Mateu. A survey on parallel computing and its applications in data-parallel problems using GPU architectures. *Communications in Computational Physics*, 15(2):285–329, February 2014.

[OSTIS, 2013] Открытая семантическая технология проектирования интеллектуальных систем [Электронный ресурс]. – 2010. - Режим доступа: http://ostis.net. – Дата доступа: 2.11.2014

[Stumme, 2001] Gerd Stumme and Alexander Maedche. FCA-merge: bottom-up merging of ontologies. In In Proceedings of 17th IJCAI, pages 225 {230, Seattle (WA), USA, 2001.

[Tatur et al., 2010] Tatur M., Adzinets D., Lukashevich M., Bairak S. Synthesis and Analysis of Classifiers Based on Generalized Model of Identification. *Advances in intelligent and soft computing*. – 2010. – Vol. 71. – P. 529–536.

[Verenik et al., 2014] Verenik N.L., Seitkulov Y.N., Girel A.I., Tatur M.M. Some regularities and objective limitations of implementing semantic processing algorithms on computing systems with massive parallelism. *Eurasian Journal of Mathematical and computer Applications*, 2(2):92–101, 2014.

SEMANTIC NETWORKS REPRESENTATION AND ALGORITHMS FOR THEIR ORGANIZATION AND SEMANTIC PROCESSING ON MASSIVELY PARALLEL COMPUTERS

Ivashenko V.P. [*], Verenik N.L. [*], Girel A.I.[*], Seitkulov Y.N. [**], Tatur M.M. [*]

[*]Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

ivashenko@bsuir.by

nick.verenik@gmail.com

tatur@i-proc.com

[**] L.N. Gumilyov Eurasian national university, Astana, Kazakhstan

seitkulov_y@enu.kz

In the article descriprion of semantic networks representation for processing on special parallel acrhitecture and semantic graph canonical labeling are considered. Some estimations for time complexity canonical labeling algorithms and potential benefits of parallel processing are given.