Новый сетевой подход для задач, не имеющих однозначного решения[1]

В. И. Васильев, аспирант
Санкт-Петербургский государственный университет
gnome@bk.ru

В статье описывается новый перспективный подход организации параллельных вычислений, наследующий традиционное достоинство нейрокомпьютинга — эффективный параллелизм и гибкость мультиагентных технологий. Предлагаемая концепция позволяет получать решение с точностью, зависящей от количества элементарных вычислителей и структуры интеллектуальной сети. При этом, в связи с тем что каждый процессор по построению разделен на процесс вычисления (чистая математическая функция) и управления (любой алгоритм, чье влияние на вычисление и работу сети ограничено заданным набором элементарных управляющих воздействий), возможно гибким образом описывать различные недетерминированные системы.

Ключевые слова: интеллектуальные сети, нейрокомпьютеры, мультиагентные системы, коннективизм.

1. Введение

В полном согласии с широко известным законом Мура [1], на сегодняшний день мы продолжаем наблюдать в действии тенденции миниатюризации и экспоненциального повышения мощности процессоров. Несмотря на это, длительность выполнения некоторых сложных вычислительных задач на современной однопроцессорной системе все еще может достигать недель, месяцев и даже лет [2]. Примером подобных задач может выступать, например, рассчет модели ядерного взрыва (количество операций с плавающей точкой достигает величины 10^{14}), или экспериментов с моделью атмосферы (порядка 10^{16}-10^{17} операций). Очевидным способом повышения производительности в этом случае является увеличение числа одновременно работающих над задачей процессоров. Таким образом, мы переходим от приоритетов бесконечного наращивания мощности одного процессора, к приоритету построению многоядерных, многопроцессорных, и даже кластерных вычислительных систем.

Однако оказывается, что простое добавление вычислителей не дает ожидаемый линейный прирост скорости вычислений. Так, согласно гипотезе М. Минского, производительность параллельной

системы растет (примерно) пропорционально логарифму числа процессоров, то есть гораздо медленнее линейной функции [3]. Таким образом исследователи подходят к проблеме организации эффективных вычислений в сетях из нескольких вычислителей, удачно названной А.Н. Горбанем [4] “проблемой эффективного параллелизма”. В рамках различных технологий организации параллельных вычислений данную проблему решают по разному.

Традиционно, подходы к организации архитектур современных параллельных систем классифицируют по степени сложности процессорных элементов на крупноблочный, среднезернистый и мелкозернистый параллелизм [2].

Крупноблочный параллелизм подразумевает использование вычислительных систем, составленных из небольшого числа (десятки, сотни) мощных компьютеров, соединенных в вычислительные кластеры различных типов. Эффективные параллелизм в таких системах обычно базируется на использовании эффективных моделей организации вычислений, например, широко применимой модели MapReduce [5], в рамках которой вычисление разделяется на шаги анализа и дальнейшей свертки результата.

Принципиально иной подход можно обнаружить в мультиагентных системах, где результат вычислений получается в результате взаимодействия множества автономных целенаправленных программных модулей (агентов). Подобный способ позволяет получать решения задач не имеющих точного решения даже в условиях неопределенности [6].

В свою очередь, мелкозернистый параллелизм свойственен вычислительным системам, составленным из огромного числа (десятки, сотни тысяч) относительно простых (и даже аналоговых) процессорных элементов. Характерной чертой таких систем является регулярность связей и, в большинстве случаев, локальность взаимодействий между вычислителями. При этом, поскольку время выполнения отдельных элементов предельно мало, применяя закон Амдаля-Уэра [2], время решения задачи не ограничивается используемой элементарной базой, что позволяет использовать мелкозернистый параллелизм в системах реального времени [2].

Основным подходом, позволяющим организовывать высокоэффективный параллелизм в таких системах является коннективизм (от англ. “connectionism”; моделирование явлений процессами становления в сетях из связанных между собой простых элементов [7]).

В рамках коннективизма память и логика работы оказываются распределенными в связях между элементарными процессорами, что позволяет переложить основную нагрузку с вычислителей на архитектуру системы. Традиционно, главным направлением, развивающим парадигму коннективизма является нейроинформатика. Стоит заметить, что несмотря на то, что нейрокомпьютеры предоставляют универсальные мелкозернистые параллельные архитектуры для решения различных классов задач, подобные системы, как правило узко специализированы.

Возможно ли сохранив коннективизм в качестве основного движущего принципа выйти за рамки мелкозернистого параллелизма и принесет ли выгоду подобный подход?

В рамках ответа на данный вопрос представляется особенно интересным опробовать плоды скрещивания нейрокомпьютинговой и мультиагентной парадигм. Синтезированный подход, описанный в статье, наследует коннективизм в качестве направляющего принципа нейроинформатики, при этом вместо примитивных формальных нейронов или блоков нейронов используются сложные вычислительные элементы (“агенты”, или “умные нейроны”). Каждый нейрон имеет регулярную связь с набором других агентов и единственной информацией известной ему об “окружающем мире” являются весовые коэффициенты на соответствующих ему “синапсах”.

На структурную сложность подобного нейрона налагается следующее ограничение — вычислитель обязан делиться на управляющий и вычислительные элементы. При этом вычислитель представляется в виде чистой (без побочных эффектов) математической функцией, а элемент может оказывать лишь ограниченное управляющее воздействие на процесс вычислений и передачи их результатов по аксонам. Это обеспечивает выполнение трех основных требований к построению мультиагентных систем:

1) автономность;
2) ограниченность представлений;
3) децентрализованность.

В следующем разделе представлено подробное формальное описание концепции: даны соответствующие определения; описаны принципы работы и ограничения, накладываемые на управляющие и вычислительные процессы; в конце раздела приведено краткое резюме. В заключении рассматриваются перспективы подхода и приводятся дальнейшие планы исследовательской работы.

2. Интеллектуальные сети

Определение 1. *Агент*

Агентом $a \in \mathbb{A}$ *будем называть тройку* $< f, c, m >= \mathbb{F} \times \mathbb{C} \times \mathbb{M}$, *где* $m \in \mathbb{M}$ *— текущая память агента (гетерогенной природы);* $\mathbb{F}$ *— всевозможные чистые (без побочных эффектов) вычисления вида* $\mathbb{M} \times \inf\limits_{k=1} \bigcup V_{in}^k \to \mathbb{M} \times \inf\limits_{l=1} \bigcup \mathbb{V}_{out}^l$; $\mathbb{C}$ *— варианты управления и* V_{in}, V_{out} *— линейные пространства (над* $\mathbb{W}$*), содержашие возможные значения на связях нейрона.*

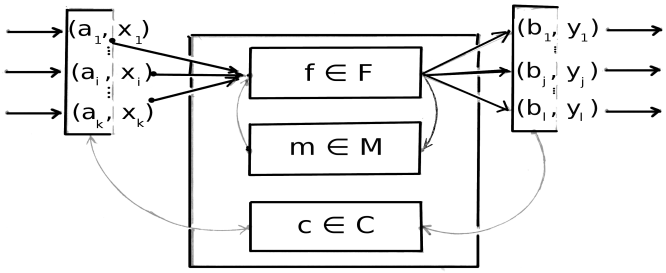

Рис. 1: Агент $a =< f, m, c >$ имеет k входящих и l исходящих связей.

Определение 2. *Эволюция агента*

Эволюцией агента *будем называть отображение вида* $a_e : \mathbb{A} \to \mathbb{A}$, *такое что для* $\forall f \in \mathbb{F}, c \in \mathbb{C}, m_1 \in \mathbb{M}$ *существует* $m_2 \in \mathbb{M}$: $a_e(f, c, m_1) =< f, c, m_2 >$. *То есть с течением времени, под воздействием* c *и* f, *в состоянии агента меняется только память* m.

Определение 3. *Интеллектуальная сеть*

Интеллектуальной сетью *будем называть взвешенный орграф* $g = (\mathbb{A}', \mathbb{E})$, *где* $\mathbb{A}' = [1..n] \times \mathbb{A}$ *— множество вершин;* $\mathbb{A}$ *— множество агентов (нейронов);* $[1..n]$ *— индексы;* $\mathbb{E}$ *— множество взвешенных дуг вида* $< (a_{in}, a_{out}), (w, v) >\in (\mathbb{R} = \mathbb{A}' \times \mathbb{A}') \times (\mathbb{W} \times (\mathbb{V}' \in 2^{\mathbb{V}}))$; $\mathbb{W}$ *— поле весовых коэффициентов;* $\mathbb{V}$ *— линейное пространство над* $\mathbb{W}$ *и выполняется условие: для* $\forall a, b \in \mathbb{A}'$, *таких что* $a = (i, \cdot)$ *и* $b = (j, \cdot) \Rightarrow i \neq j$.

Тогда $\mathbb{G} = 2^{\mathbb{A}'} \times 2^{(\mathbb{R}=\mathbb{A}'\times\mathbb{A}')\times(\mathbb{W}\times(\mathbb{V}'\in 2^{\mathbb{V}}))}$ — множество всех возможных сетей.

О п р е д е л е н и е 4. *Для удобства введем следующие обозначения:*

- *пусть* $n(g)$ *обозначает* мощность сети *(количество агентов), тогда если* $g = (\mathbb{A}', \mathbb{E})$, *то* $n(g) = |\mathbb{A}'|$;
- *пусть* $a_i(\mathbb{A}', \mathbb{E}) = x$, *такой что* $(i, x) \in \mathbb{A}'$, *то есть* $\{a_i\}_{i=1}^{|\mathbb{A}'|}$ *— множество проекций из* $\mathbb{A}$ *в* $\mathbb{A}'$;
- *пусть* $r_{i,in}(\mathbb{A}', \mathbb{E}) = \mathbb{X}$, *такое что для* $\forall x \in X$ *выполняется* $x = ((\cdot, a_i), \cdot) \in \mathbb{E}$. *То есть* $r_{i,in}$ *— все входящие ребра* i*-го агента;*
- *пусть* $r_{i,out}(\mathbb{A}', \mathbb{E}) = \mathbb{X}$, *такое что для* $\forall x \in X$ *выполняется* $x = ((a_i, \cdot), \cdot) \in \mathbb{E} \Rightarrow r_{i,out}$ *— все исходящие ребра* i*-го агента;*
- *пусть определены проекции* v, w, *такие что:* $v(\cdot, (\cdot, v')) = v'$ *и* $w(\cdot, (w', \cdot)) = w'$.

О п р е д е л е н и е 5. *Входящая/исходящая связь*

Входящей (исходящей) связью $e \in \mathbb{E}$ *агента* $a \in \mathbb{A}$ *называем такую дугу* $e =< (a_{in}, a_{out}), (w, v) >\in \mathbb{E}$, *что* $a_{out} = a$ *(или, соответственно,* $a_{in} = a$*). Значение* v *называем* текущим значением связи*; величину* w — текущим весом связи.[2]

О п р е д е л е н и е 6. *Эволюция интеллектуальной сети*

Отображение g_e *вида* $\mathbb{G} \to \mathbb{G}$ *будем называть эволюцией интеллектуальной сети.*

О п р е д е л е н и е 7. *Процесс эволюции*

Процессом эволюции интеллектуальной сети g *будем называть функцию вида* $g : (\mathbb{T} = \mathbb{R}) \to \mathbb{G}$, *где* $\mathbb{T}$ *— непрерывное время.*

О п р е д е л е н и е 8. *Вычислительный процесс*

Математическую функцию вида $f : \mathbb{M} \times \bigcup\limits_{k=1}^{\inf} \mathbb{V}_{in}^k \to \mathbb{M} \times \bigcup\limits_{l=1}^{\inf} \mathbb{V}_{out}^l$ *называем* вычислительным процессом *агента* $a =< f, c, m >$.

[2] Следуя нейрокомпьютерной аналогии можно считать, что агенты образуют аксо-дендритную связь.

Считаем, что новый вычисленный результат мгновенно оказывается в качестве значений на соответствующих исходящих связях. Так, если в момент j-го шага эволюции агент a имел l исходящих связей и, соответственно, на j-м шаге им был получен результат $f_j(m_j, v_{in}) = v_{out} = \{v_{out,i}\}_{i=1}^{l}$, то $v_{out,i}$ немедленно запишется как новое значение на i-й исходящей связи. Соответственно, если в момент j-го шага агент имел k входящих связей, то результат был получен на основании данных $v_{in} = \{v_{in,i}\}_{i=1}^{k}$, где $v_{in,i}$ — значение на i-м входящем ребре.

Определение 9. *Управляющий процесс, управляющее воздействие*

Управляющим процессом $c \in \mathbb{C}$ *называем любой алгоритм (в интуитивном понимании), который может оказывать лишь следующие* управляющие воздействия *на вычислительный процесс и структуру сети:*

1. Запуск (или остановка) вычислительного процесса

   Формально считаем, что если управляющий процесс c агента a *приостановил* вычислительный процесс на промежутке времени $[t_{pause}, t_{start}]$, то для $\forall t$ из этого промежутка $a(t) = const$.

2. Чтение (или изменение) памяти агента

   Формально считаем, что в любой момент времени t управляющему процессу c агента $a =< f, c, m >$ доступна память m (чтение). Кроме того, c может изменить m_{old} на m_{new} в любой момент времени t (запись). Тогда $\exists \Delta t : \forall t' \in (t, t + \Delta t],\ a(t) =< f, c, m_{new} >$ и $a(t + \Delta t) =< f, c, m_{newest} >$, где $< y, m_{newest} >= f(x, m_{new})$. В этом случае вычислительный процесс f обязан немедленно начать новое вычисление на основании изменившегося содержимого памяти.

3. Ветвление (или прекращение) работы

   (a) *Ветвление* означает, что агент делится на два идентичных оригиналу агента, причем “идентичность” включает в себя как сохранение содержимого памяти, так и воссоздание связей, которые были у оригинала.

   Формально: пусть в момент ветвления t процесс эволюции сети $g(t)$ имел значение $g_1 =< \mathbb{A}'_1, \mathbb{E}_1 >$, тогда сразу

после ветвления процесс будет иметь значение $g(t+0) = g_2 =< \mathbb{A}'_2, \mathbb{E}_2 >$, где $\exists x \in [1..n] : a_x(g_2) \equiv a_{n(g_2)+1}(g_2)$ и $r_{i,in} \equiv r_{n(g_2)+1,out}$..

(b) *Прекращение работы* означает, что агент исчезает вместе со всеми входящими/исходящими связями. То есть формально: пусть в момент уничтожения t процесс эволюции сети $g(t)$ имел значение $g_1 =< \mathbb{A}'_1, \mathbb{E}_1 >$, тогда сразу после уничтожения агента a_x, процесс будет иметь значение $g(t+0) = g_2 =< \mathbb{A}'_2, \mathbb{E}_2 >$, такой то $\neg\exists(x, a_x) \in g_2$.[3]

4. Чтение исходящих и входящих весов / изменение входящих весов

   Формально считаем, что управляющему процессу c агента a доступны для чтения значения исходящих и входящих весов, но изменять у него есть возможность только *входящие* весы. При этом управляющий процесс не может изменять *значения* связей — это может делать только вычислительный процесс.[4]

Заметим, что управляющий процесс может узнать о значении (не весе) только если вычислительный процесс запишет это значение в память (а для этого память должна иметь вид $\mathbb{M} = \mathbb{M}' \times \bigcup\limits_{k=1}^{\inf} \mathbb{V}^k_{in}$ или $\mathbb{M}' \times \bigcup\limits_{l=1}^{\inf} \mathbb{V}^l_{out}$). Это сделано специально, чтобы как можно сильнее разграничить роли выполняемые процессами.

Краткое резюме

1. каждый агент обладает гетерогенной памятью $m \in M$, которая изменяется в процессе работы, отражая качественную эволюцию агента;

2. каждый агент поделен на два процесса выполняющихся параллельно: вычислительный f и управляющий c;

3. вычислительный процесс итерирует строго математическую функцию вида $f : \mathbb{M} \times \bigcup\limits_{k=1}^{\inf} \mathbb{V}^k_{in} \to \mathbb{M} \times \bigcup\limits_{l=1}^{\inf} \mathbb{V}^l_{out}$;

[3] Не умаляя общности считаем, что сразу после разделения каждому из агентов известно кто есть оригинал, а кто копия.

[4] Не умаляя общности считаем, что управляющий процесс каждого агента имеет также внутреннюю память, а значит так как только ему дозволено изменять входные весы, то мы условимся что он запоминает (то есть читает значения входящих весов).

4. управляющий процесс оказывает управляющие воздействия на агента, корректируя его работу;

5. каждое управление строится на ограниченном числе доступных примитивов: запуск/остановка вычислений, ветвление (прекращение) работы агента, чтение/запись значения памяти, чтение исходящего/изменение входящего веса.

Заметим, что все накладываемые ограничения на структуру интеллектуальной сети соответствуют ключевым принципам мультиагентных систем, а именно выполняется:

1. автономность — агент производит вычисление изолированно, только на основании значений и весов на входящих/исходящих связях;

2. ограниченность представления — единственной доступной агенту информацией о внешнем мире является значения весовых коэффициентов на исходящих связях;

3. децентрализация — управляющее воздействие доступное агенту ограничивается изменением входящих весовых коэффициентов.

3. Заключение

В статье рассмотрен новый перспективный гибридный подход, наследующий традиционное достоинство нейрокомпьютерных вычислений — эффективный параллелизм и гибкость мультиагентных технологий. Ценность концепции состоит в том, что ее возможно использовать в системах реального времени (решение получается с любой степенью достоверности в зависимости от количества участвующих нейронов и времени работы).

Рассматриваемая концепция в определенном плане является обобщением нейросетевого подхода. При этом сама попытка подобного обобщения не является принципиально новой — наряду с мультиагентной концепцией и парадигмой коннективизма, подход имеет схожие черты с уже проводившимися ранее исследованиями [8–12].

Однако уже сейчас возможно говорить о перспективности использования описанной гибридной парадигмы в различных областях, где требуется решение задач оптимизации, задач не имеющих

однозначного ответа, или задач с неопределенными условиями. В связи с чем, описанный теоретический аппарат находит свое закономерное развитие в кандидатской работе, ставящей целью создание систем автоматизированного стилистического анализа классической музыки.

Список литературы

[1] *Moore G.E.* Cramming more Components onto Integrated Circuits // Electronics. 38(8). April 9. 1965.

[2] *Шпаковский Г.И.* Реализация параллельных вычислений: MPI, OpenMP, кластеры, грид, многоядерные процессоры, графические процессоры, квантовые компьютеры. — М.: Изд-во Белорусского ГУ. 2011. 176 с.

[3] *Minsky M.L.* Вычисления и автоматы. Монография — Изд. Мил Месива. 1971

[4] *Горбань А.Н.* Нейроинформатика: кто мы, куда мы идем, как путь наш измерить? // Вычислительные технологии. Ч– М.: Машиностроение. Ч– 2000. Ч– №4. С. 10–14.

[5] *Lin J., Dyer C.* Data-Intensive Text Processing with MapReduce. University of Maryland. April 2010.

[6] *Граничин О.Н., Амелин К.С.* Мультиагентное сетевое управление группой легких БПЛА // Нейрокомпьютеры: разработка, применение. 2011. № 6. С. 64–72.

[7] *Rumelhart D.E., McClelland J.L.* Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1,2: Foundations. Cambridge. MA: MIT Press. 1986.

[8] *Владимирович А.Г., Граничин О.Н., Макаров А.А.* Нестандартная машина Тьюринга // Стохастическая оптимизация в информатике. 2005. Т. 1. С. 29–47.

[9] *Граничин О.Н., Жувикина И.А.* Новая модель процесса вычислений: обобщение концепции машины Тьюринга // Нейрокомпьютеры: разработка, применение. 2006. №7. С. 24-31.

[10] *Граничин О.Н., Жувикина И.А.* Новая концепция процесса вычислений, основанная на эволюционных примитивах // Системное программирование. 2006. Т. 2. С. 68–83.

[11] *Граничин О.Н., Васильев В.И.* Гибридная модель процесса вычислений: обобщение концепции машины Тьюринга // Нейрокомпьютеры: разработка, применение. 2010. №6. С. 51–58.

[12] *Granichin O.N., Vasilev V.I.* Computational model based on evolutionary primitives. Turing machine generalization // International Journal of Nanotechnology and Molecular Computation. Vol. 2. 2010. No. 2. PP. 30–43.