УДК 621.391; 004.031

ДОСТИЖЕНИЕ КОНСЕНСУСА В МУЛЬТИАГЕНТНОЙ СИСТЕМЕ НА ОСНОВЕ ПРОТОКОЛА ЛОКАЛЬНОГО ГОЛОСОВАНИЯ

Ю.А. ДУЙНОВА, Н.И. ЛУЩИК, С.И. ПОЛОВЕНЯ

УО «Белорусская государственная академия связи»
Ф. Скорины, 8/2, Минск, 220114, Беларусь

Поступила в редакцию 23 марта 2018

Рассматривается алгоритм достижения консенсуса в децентрализованной сети с помехами и специализированное программное обеспечение. Рассмотрены задачи достижения консенсуса группой взаимодействующих агентов.

Ключевые слова: достижение консенсуса, децентрализованные системы, балансировка загрузки.

.Введение

В настоящее время децентрализованные сети находят все более широкое применение. В таких сетях важным является эффективное распределение пакетов (заданий) среди большого числа агентов (узлов). С этим вопросом сталкиваются на производстве, в транспорте, а так же сенсорных сетях [1]. Задание может быть выполнено как одним агентом, так и может быть распределено между группой агентов. Задача балансировки загрузки состоит в поддержании равномерной загруженности всех агентов во времени. В большинстве публикаций, рассматривающих возможность решения этой задачи, посвященных вычислительным системам, не учитываются разрывы связи, помехи и задержки. В сетевых системах наоборот, помехи, задержки и разрывы связи учитываются. Задача балансировки загрузки тесно связана с задачей достижения консенсуса [2]. В работе [3] был предложен протокол локального голосования для задачи достижения консенсуса в сети агентов с нелинейной динамикой, в условиях изменяющейся топологии, помех и задержек. В работе [4] задача балансировки загрузки была преобразована в задачу достижения консенсуса в динамических сетях. А так же была рассмотрена возможность применения протокола локального голосования для децентрализованной балансировки загрузки в сети с изменяющейся топологией, с учетом помех и задержек при поступлении всех заданий в систему в определенный начальный момент времени.

В данной статье изучается возможность создания универсального программного обеспечения для определения параметров и условий достижения консенсуса на основе протокола локального голосования.

Динамическая сеть

В последнее время все чаще используются распределенные системы, для которых актуальна задача распределения загрузки. Обычно при распределении заданий по узлам приветствуется равномерная загрузка.

На практике используются два подхода: централизованная динамическая балансировка загрузки и распределенная (децентрализованная). При централизованном подходе определяется некий ресурс, который аккумулирует информацию о состоянии всей сети и принимает решения

о распределении заданий для каждого из узлов [5]. При распределенном подходе алгоритмы балансировки загрузки выполняются на всех узлах, обменивающихся информацией о состояниях с другими узлами сети. Перераспределение происходит только между соседними узлами.

Представим модель сети агентов (узлов), выполняющих параллельно однотипные задания, в которой допускается перераспределение заданий между агентами на основе обратных связей.

Рассмотрим $N = \{1, \dots, n\}$ – набор интеллектуальных агентов (узлов), каждый из которых выполняет поступающие задания по принципу очереди. Задания поступают в систему в различные дискретные моменты времени $t = 1,2, \dots, T$ на разные узлы. Связь между узлами определяется топологией динамической сети.

В момент времени t поведение каждого агента $i \in N$ описывается двумя характеристиками: q_t^i – загруженность или длина очереди из атомарных элементарных заданий узла i в момент времени t; r_t^i – производительность узла i в момент времени t.

При достаточно общих предположениях можно считать, что динамика изменений загруженности агентов описывается следующими уравнениями:

$$q_{t+1}^i = q_t^i - r_t^i + z_t^i + u_t^i; i \in N, t = 0,1, \dots, T,$$

где $u_t^i \in \mathbb{R}$ – управляющие воздействия, которые в момент времени t воздействуют на узел i, z_t^i – размер нового задания, поступившего на узел i в момент времени t.

Если $N_t^i \neq 0$, то в момент времени t узел i получает данные о производительности соседних узлов $(r_t^i, r_t^j, \forall i \in N, j \in N_t^i)$ и наблюдения об их загруженности с учетом зашумления:

$$y_t^{i,j} = q_{t-d_t^{i,j}}^j + \omega_t^{i,j}, j \in N_t^i,$$

где $\omega_t^{i,j}$ – помехи, а $0 \leq d_t^{i,j} \leq \bar{d}$ – целочисленная задержка, $\bar{d}$ – максимально возможная задержка. Кроме того, в каждый момент времени t узел i имеет зашумленные данные о своей загруженности

$$y_t^{i,j} = q_t^j + \omega_t^{i,j},$$

При обеспечении равномерной загрузки всех узлов сети рассматривается две постановки задачи: стационарная и нестационарная.

При стационарной постановке все задания поступают в систему на разные узлы в начальный момент времени, в то время как при нестационарной постановке задачи новые задания могут поступать в систему на любой из i узлов в различные моменты времени t

Для сокращения времени выполнения всех заданий необходимо использовать протокол перераспределения заданий с течением времени. Это позволит увеличить пропускную способность системы и уменьшить время выполнения заказов в системе. В стационарном случае, с одной стороны, если с момента времени t все задания выполняются только теми агентами, к которым они поступили, то время реализации всех заданий определяется как

$$T_t = \max_{i \in N} \frac{q_t^i}{r_t^i}$$

С другой стороны, желаемое время работы всей системы может составить

$$T_{min} = \frac{\sum_{i=1}^n q_0^i}{\sum_{i=1}^n r_0^i}$$

если все узлы работают одновременно над общим объемом заданий.

Метод достижения консенсуса

Широкое применение мультиагентных технологий привлекает все большее число исследователей к задачам управления и распределенного взаимодействия в сетях динамических систем [6]. Однако решение таких задач не всегда является удовлетворительным из-за сложности обмена информацией и ее полноты, а также наличия задержек и помех.

Для решения задачи достижения консенсуса группой взаимодействующих агентов, обменивающихся информацией, часто применяются алгоритмы типа стохастической аппроксимации с уменьшающимся размером шага [7]. При постоянно меняющихся внешних состояниях агентов с течением времени алгоритмы стохастической аппроксимации с уменьшающимся до нуля размером шага неработоспособны. Поэтому актуальной является задача исследования свойств алгоритма типа стохастической аппроксимации при малом постоянном или неуменьшающемся до нуля размере шага при нелинейной постановке задачи, в условиях случайно изменяющейся структуры связей в сети при действии помех.

Применительно к теории графов динамическая сеть состоит из набора агентов (узлов) $N = \{1, \dots, n\}$ Граф (N, E) определяется N и множеством ребер или дуг E. Множеством соседей узла i называется $N^i = \{j: (j, i) \in E\}$, т. е. множество узлов с ребрами, входящими в i Поставив в соответствие каждому ребру$(j, i) \in E$ вес $A^{i,j} > 0$, определяем матрицу смежности (или связности) $A = [a^{i,j}]$ графа, обозначаемого далее G_A (верхние индексы у переменных показывают соответствующие номера узлов). Взвешенная полустепень захода вершины i определяется как сумма i -й строки матрицы A:

$$d^i(A) = \sum_{j=1}^{n} d^{i,j}$$

Каждому агенту $i \in N$ в момент времени $t = 1,2, \dots, T$ ставится в соответствие изменяющееся во времени состояние $x_t^i \in \mathbb{R}$, динамика которых описывается разностными уравнениями

$$x_{t+1}^i = x_t^i + f^i(x_t^i, u_t^i)$$

с некоторыми функциями $f^i(\cdot,\cdot): \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, зависящими от состояний в предшествующий момент времени x_t^i и управляющих воздействий $u_t^i \in \mathbb{R}$, которые в момент времени t воздействуют на узел i

Пусть мультиагентная система состоит из динамических агентов с входами u_t^i, выходами $y_t^{i,t}$, и состояниями x_t^i.

Узлы i и j называются согласованными в сети в момент времени t тогда, когда $x_t^i = x_t^i$. Для достижения консенсуса требуется согласовать все узлы между собой, т.е. найти такой протокол управления, который переводит все состояния в одно и то же постоянное значение $x^a = x^i, \forall i \in N$.

Предположим, что структура связей динамической сети моделируется с помощью последовательности ориентированных графов $\{(N, E)\}_{t \geq 0}$, где $E_t \in E$ меняются во времени. Если $(j, i) \in E_t$, то узел i в момент времени t получает информацию от узла j для целей управления с обратной связью. Обозначим A_t − соответствующие E_t матрицы смежности;

$E_{max} = \{(j, i): \sup_{t \geq 0} a_t^{i,j} > 0\}$ − максимальное множество каналов связи. Для формирования правила управления каждый узел $i \in N$имеет информацию о своем собственном состоянии (может быть и зашумленную) $y_t^{i,i} = y_t^{-i} + w_t^{i,i}$ и, если$N_t^i \neq \emptyset$, зашумленные наблюдения о состояниях соседних узлов

$$y_t^{i,j} = x^j_{t - d_t^{i,j}} + w_t^{i,j}, j \in N_t^i$$

где $w_t^{i,i}, w_t^{i,j}$ − помехи, а $0 \le d_t^{i,j} \le \bar{d}$ − задержка, $\bar{d}$ − максимально возможная задержка. Так как система начинает работу при $t = 0$, неявное требование к множеству соседних узлов: $j \in N_t^i \Rightarrow t - d_t^{i,j} \ge 0$. При $w_t^{i,j} = 0$ для всех остальных пар i, j, определим $\overline{w_t} \in \mathbb{R}^{n^2}$ вектор, составленный из элементов $w_t^{i,j}, i, j \in N$.

Алгоритм управления, называемый протоколом локального голосования, задается формулами:

$$u_t^i = a_t \sum_{j \in \bar{N}_t^i} b_t^{i,j} (y_t^{i,j} - y_t^{i,i})$$

где $a_t > 0$ − размеры шагов протокола управления, $b_t^{i,j} > 0, \forall_j \in \bar{N}_t^i$. Принимая $b_t^{i,j} = 0$для всех остальных пар i, j, определим $B_t = \left[b_t^{i,j}\right]$ − матрицу протокола управления.

Пусть (Ω,F,P) − основное вероятностное пространство. Обозначим: E − математическое ожидание и E_x− условное математическое ожидание при условии x

Основные условия, при которых протокол локального голосования обеспечивает достижение консенсуса следующие:

$\forall_i \in N$ функции – $f^i(x, u)$ липшицевы по x и u:

$$\left| f^i(x, u) - f^i(x', u') \right| \le L_l (L_x |x - ||x'| + |u - u'|),$$

и при любом фиксированном x функции $f^i(x,\cdot)$ такие, что $E_x f^i(x, u) = f^i(x, E_x u)$;

2. $\forall_i \in N, j \in N^i \cup \{i\}$ помехи наблюдений $w_t^{i,j}$– центрированные, независимые, одинаково распределенные случайные величины с ограниченными дисперсиями: $E(w_t^{i,j})^2 \le \sigma_w^2$.

3. $\forall_i \in N, j \in N^i$ появление переменных ребер (j, i) в графе G_{A_t} − независимые, случайные события, вероятность которых $p_a^{i,j}$ (т.е. матрицы – независимые, одинаково распределенные случайные матрицы).

4. $\forall_i \in N, j \in N^i$ веса $b_t^{i,j}$ в протоколе управления – ограниченные, случайные величины: $\underline{b} \le b_t^{i,j} \le \bar{b}$ с вероятностью 1, и существуют пределы $b^{i,j} = \lim_{t \to \infty} E b_t^{i,j}$.

5. Существует конечная величина $\bar{d} \in \mathbb{N}$: $\forall_i \in N, j \in N^i\ d_t^{i,j} \le \bar{d}$ с вероятностью 1 и целочисленные задержки $d_t^{i,j}$– независимые, одинаково распределенные случайные величины, принимающие значения $k = 0, \dots, \lfloor \bar{d}$ с вероятностями $p_k^{i,j}$

Кроме того, все эти случайные величины и матрицы независимы между собой, и их элементы имеют ограниченные дисперсии.

Обозначим $\bar{n} = n(\bar{d} + 1)$ и определим матрицу A_{max} размерности $\bar{n} \times \bar{n}$ по правилу:

$$a_{max}^{i,j} = p_{j \operatorname{div} \bar{d}}^{i,((j-1) \bmod n)+1} b^{i,((j-1) \bmod n)+1}$$

Здесь операция mod – остаток от деления, a div − деление нацело.

Будем считать, что для матрицы структуры связей сети A_{max} выполняется следующее условие:

6. Граф (N, E_{max}) имеет остовное дерево, и для любого ребра $(j, i) \in E_{max}$ среди элементов $a_{max}^{i,j}, a_{max}^{i,j+n}, \dots, a_{max}^{i,j+\bar{d}n}$ матрицы A_{max} найдется хотя бы один ненулевой.

Обозначим $\bar{x}_t = [x_t^1; \dots; x_t^n]$ − соответствующий вектор-столбец, полученный вертикальным соединением n чисел. Положим $\overline{X}_t = 0$ для $-\bar{d} \le t \le 0$, и определим $\overline{X}_t \in \mathbb{R}^{n\bar{d}}$ – расширенный вектор состояний $\overline{X}_t = [\bar{x}, \tilde{x}_{t-1}, \dots, \tilde{x}_{t-\bar{d}}]$, где $\tilde{x}_{t-k}$ − вектор, состоящий из таких x_{t-k}^i, что, $\exists j \in N^i \exists k' \ge k: p_{k'}^{i,j} > 0$, т.е. это значение с положительной вероятностью участвует в формировании хотя бы одного из управляющих воздействий. В дальнейшем

для простоты будем считать, что так введенный расширенный вектор состояний равен $\overline{X}_t = [\overline{x}, \overline{x}_{t-1}, \dots, \overline{x}_{t-\overline{d}}]$, т.е. в него входят все компоненты со всевозможными задержками, не превосходящими $\overline{d}$

Из-за наличия помех, задержек, неопределенностей в протоколе управления и переменной структуры связей точного консенсуса достичь достаточно сложно. Поэтому мы будем рассматривать задачу о достижении приближенного консенсуса – ε-консенсуса.

Алгоритм моделирования достижения консенсуса

Рассмотрим пример системы, состоящей из вычислительных узлов. Зададим начальные значения длин очереди и производительностей узлов. Предположим, что производительности не меняются с течением времени.

Разработанный алгоритм можно охарактеризовать как:

– дискретный (так как алгоритм состоит из отдельных шагов – команд);

– однозначный (каждая из команд описывает лишь одно возможное действие пользователя);

– понятный (каждая из команды входят в систему команд пользователя);

– результативный (пользователь достигает решения задачи за конечное число шагов).

Алгоритм работы программного обеспечения включает четыре этапа:

– ввод исходных значений количества агентов N и максимального времени наблюдения T_{max} для создания матриц, предназначенных для хранения результатов расчета на каждый промежуток времени от 0 до T_{max} для каждого агента от 1 до N;

– ввод начальных значений производительности R и начального состояния X для каждого агента, а также запись их в массив;

– расчет текущего состояния X для каждого агента за все периоды;

– принятие решения о времени достижения консенсуса.

Алгоритм (рисунок 1) не предполагает объявление и инициализацию всех переменных, участвующих в решении задачи, до начала первого этапа. Это объясняется тем, что при моделировании процесса достижения консенсуса количество агентов, определяющих размер массивов для сохранения промежуточных решений, задается вручную пользователем, то есть является динамическим и заранее неизвестным. Поэтому по ходу решения задачи определяется (генерируется случайным образом или вводится пользователем) количество отдельных величин, участвующих в формуле нахождения текущего состояния агента.

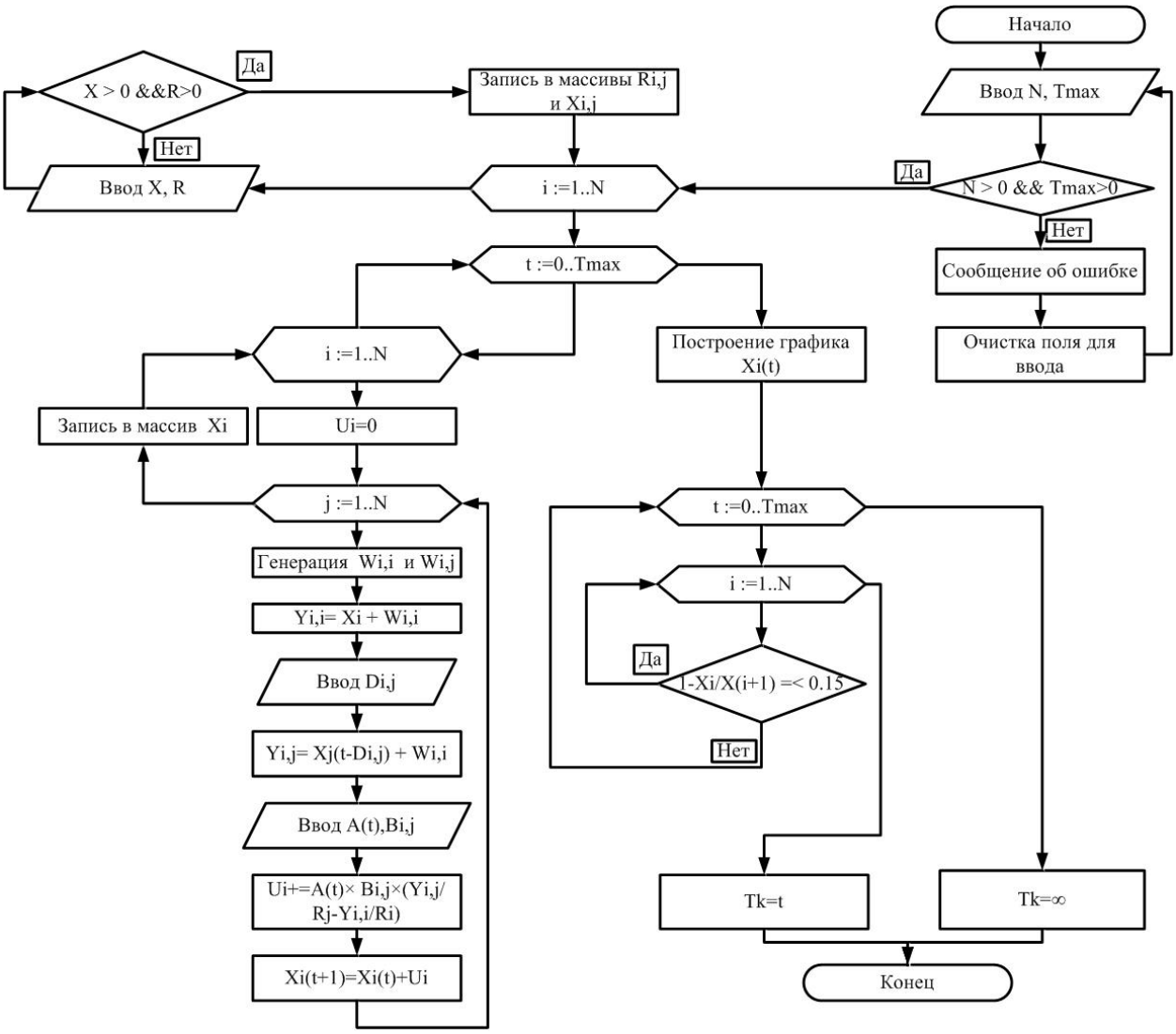

Рисунок 1 –Алгоритм достижения консенсуса

На первом этапе пользователь должен ввести исходные значения количества агентов N и максимального времени наблюдения T_{max}. Так как обе эти величины должны быть больше нуля, происходит проверка введённых значений на положительные числа (графические поля для ввода предусматривают невозможность ввода не числовых символов). Если числа не являются положительными, появляется сообщение об ошибке и повторном вводе. Соответствующее неверному значению поле ввода очищается. Данные величины определяют размер массивов для записи рассчитываемых значений в каждый момент времени для всех агентов.

На втором этапе пользователь вводит исходные значения производительности R и начального состояния X для каждого агента в начальный момент времени $(t = 0)$. Так как эти величины также должны быть больше нуля, происходит проверка их на положительные числа (графические поля для ввода также предусматривают невозможность ввода не числовых символов). При корректном введенном значении производительности R и начального состояния X эти величины записываются в соответствующий массив. Размерность массива определяется как X, нумерация массива начинается с 0 (период времени отсчитывается от начального до максимального).

На третьем этапе программа приступает к расчёту текущего состояния X для каждого агента. Создается переменная U_i для хранения значения управляющих воздействий. Для формирования стратегии управления $Y_{i,i}$ каждый агент имеет информацию о своём собственном

состоянии $X_{i,i}$, которое может быть зашумлено в результате помех $W_{i,i}$, генерируемых случайным образом для каждого агента в каждый момент времени. Далее необходимо определить значение целочисленной задержки $D_{i,j}$, с которой каждый агент должен получить информацию о зашумленном состоянии соседей $Y_{i,j}$. Затем определяется размеры шагов протокола управления $A(t)$ и коэффициента протокола $B_{i,j}$, и программа вычисляет значение управляющих воздействий U_i и состояния в следующий момент времени X_{i+1}, которое записывается в массив $X_i(t)$.

После вычисления всех значений текущего состояния X для всех агентов на четвертом этапе по полученным результатам стоится график, моделирующий поведение совокупности агентов в течение заданного времени T_{max}. Состояние каждого агента представляется отдельной кривой, значение которой вычисляется для каждого отсчета времени без аппроксимации. Программа, перебирая все отрезки времени, находит тот, в котором состояния агентов стремится к достижению консенсуса.

Условием схождения состояний агентов к консенсусу будем считать таким, при котором отношение текущего состояния любого агента ко всем текущим состояниям оставшихся агентов не превосходит 15%. Такое условие вытекает из предварительных математических расчетов и результатов имитационного моделирования. Первое определенное значение времени, удовлетворяющее заданному условию, становится временем схождения к консенсусу. В противном случае считается, что в заданных условиях агенты не смогут достичь консенсуса и необходимо изменить исходные данные для получения положительного ответа.

Программное обеспечение

Для визуализации результатов моделирования разработано программное обеспеченье на объектно-ориентированном языке программирования Java с помощью специальной среды разработки Eclipse.

Объектно-ориентированный язык программирования выбран благодаря своей парадигме представлять программы в виде совокупности объектов. Каждый из создаваемых объектов относится к определенному классу и является его конкретным представителем, то есть экземпляром. Сами же классы образуют иерархию наследования, передавая доступ к своим атрибутам и методам.

Атрибут класса – это набор характеристик или свойств, которыми обладают объекты. Так каждый агент имеет следующие свойства:

– производительность в момент времени *t*;

– максимальное время обработки заданий;

– длина очереди в момент времени *t*;

– начальный объём задания и другие.

Методы класса характеризуют поведение объектов, то есть говорят о том, какие функции они могут выполнять. Так каждый агент может реализовать следующие действия:

– проверить введенные данные на корректность;

– получить начальные параметры, введенные с помощью графического пользовательского интерфейса;

– обработать очередь на данной итерации;

– записать данные о процессе перераспределения задач в двумерный массив;

– отобразить на графике результаты своего поведения по заданным начальным параметрам и другие.

Среда разработки программных продуктов Eclipse облегчает программирование графического пользовательского интерфейса (graphical user interface, GUI), построенного на взаимодействие следующих классов:

– java.swing используется для создание функциональных элементов: контейнеров и компонентов, а также для управления размещение с помощью специальных классов менеджеров;

– java.awt предназначен для управления размещением элементов управления на форме с помощью специальных классов-менеджеров LayoutManager;

– java.awt.event содержит набор интерфейсов, обрабатывающих взаимодействие пользователя с элементами GUI на основании модели событийного программирования.

Основными элементами управления для реализации поставленной задачи являются:

– JLabel отображает фиксированный текст;

– JTextField создает однострочное поле для ввода текста;

– JButton представляет собой кнопку, вызывающую определенное действие.

При запуске программного обеспечения пользователь попадает на главную форму, которая показана на рисунке 2. На данной форме необходимо ввести количество агентов или узлов, которые будут участвовать в процессе обработки заданий. В зависимости от заданного значения числа агентов по щелчку на кнопку «Ввести начальные значения» отображается необходимое количество полей для ввода начальных параметров:

– длины очереди из атомарных элементарных заданий узла *i* в момент времени *t*;

– производительности узла *i* в момент времени *t*.

Так как входные параметры имитационного моделирования такие как количество агентов, длина очереди, производительность являются положительными числами, перед началом запуска алгоритма расчета скорости схождения к консенсусу происходит проверка введенных данных, получаемые через объекты JTextField. Также поле для ввода числа агентов, участвующих в распределении задач, по своему физическому смыслу может принимать только целые числа.

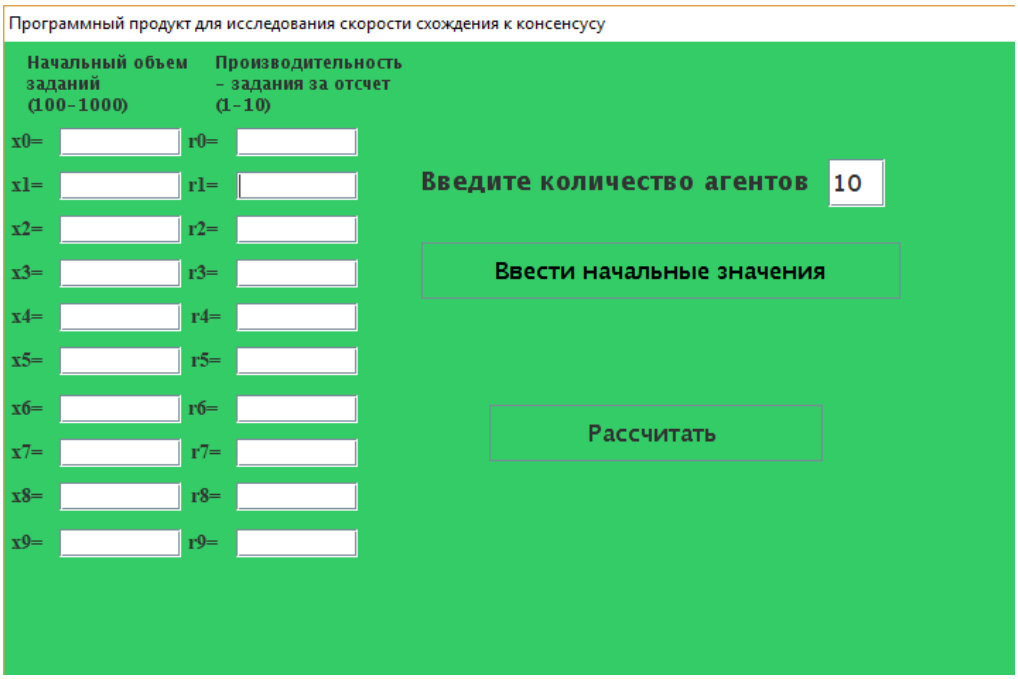

Рисунок 2 – Интерфейс главной формы программного продукта

Далее по щелчку на кнопку «Рассчитать» программное обеспечение согласно алгоритму, показанному на рисунке, моделирует поведения каждого агента при распределении заданного объёма задач и визуализирует их на вспомогательной форме в виде графика, как показано на рисунке 3.

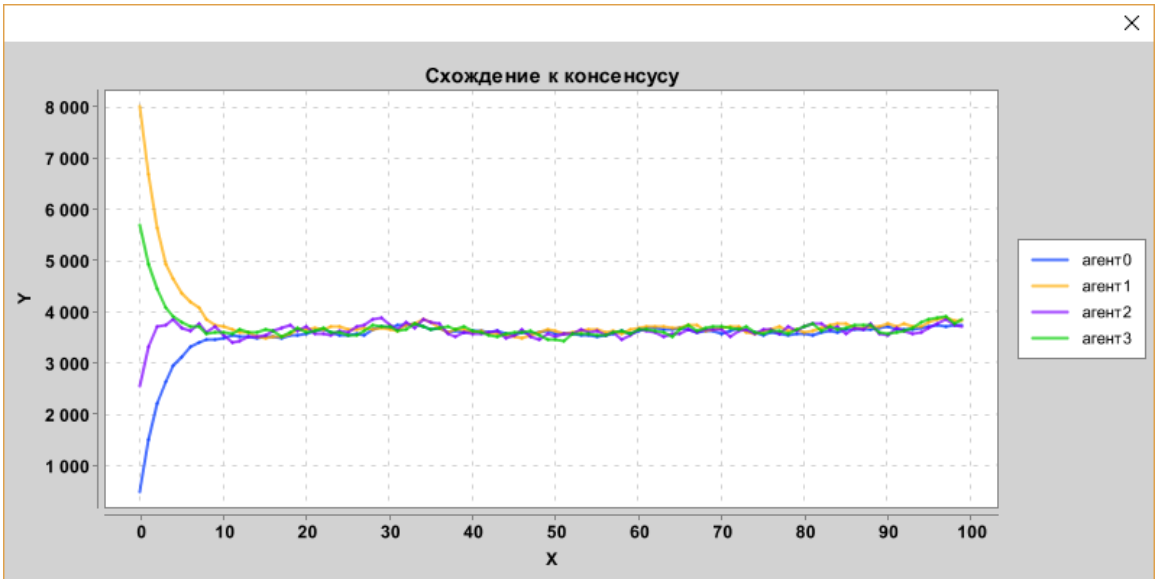

Рисунок 3 – Визуализация состояний агентов

Состоянию каждого агента в течении времени наблюдения соответствует линия определенного цвета в соответствии с легендой. ОсьY является динамической и подстраивается под задаваемый объем распределяемых заданий. Оси X соответствует время продолжительность моделирования поведения агентов.

Ниже приведен фрагмент программного кода формирования массивов значений состояния каждого агента за рассматриваемый промежуток времени. Сформированные одномерный xData[] и двумерный yData[] массивы являются одними из 6 входных параметров функции построения графика.

```
double[] xData = new double[t_max];
            for ( i = 0; i < t_max; i++) {
                        xData[i]=i;
             }
double[][] yData = new double[q[0].length][q.length];
      for ( i = 0; i < q[0].length; i++) {
         for ( j = 0; j < q.length; j++) {
           yData[i][j] = q[j][i];
         }
      }
```

Также определяется скорость схождения к консенсусу в рассматриваемом случае, значение которой отображается на главной форме, как показано на рисунке 4.

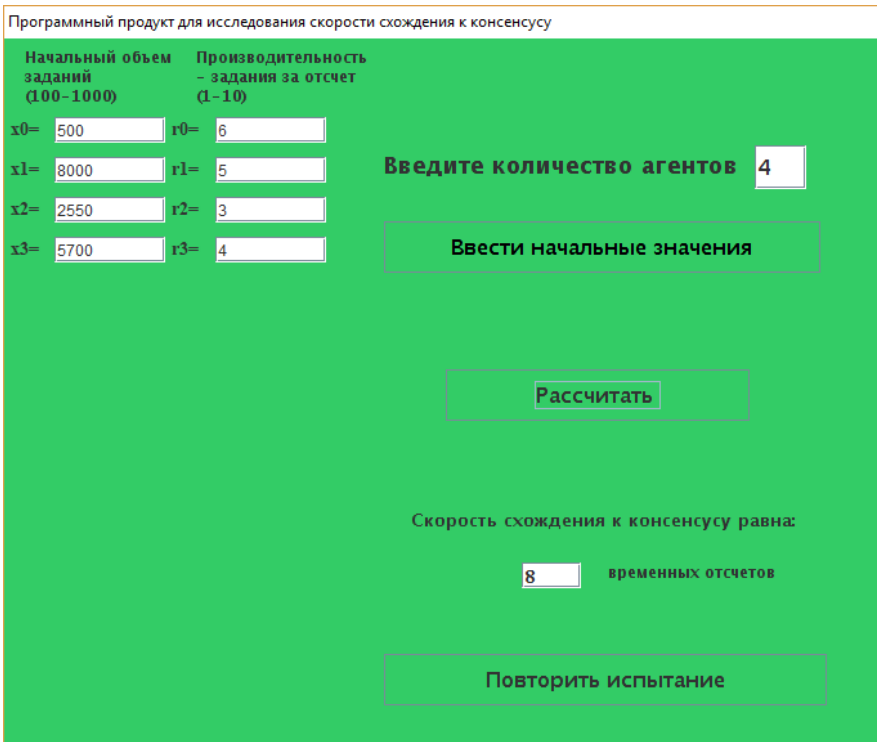

Рисунок 4 – Отображение результатов моделирования на главной форме

Алгоритм управления агентами предусматривает помехи при получении информации как о состоянии соседних агентов, так и о своем собственном. При повторном нажатии на кнопку «Рассчитать» программный продукт моделирует новую линию поведений всех агентов, следовательно, при новых условиях скорость схождения к консенсусу может увеличится или уменьшиться.

Повторное моделирование может проводится со следующими целями:

– ввод нового либо выход действующего агента;

– перераспределение объема заданий между агентами;

– наблюдение за выполнением новой задачи.

Для повторного проведения моделирования необходимо щелкнуть по кнопке «Повторить испытание».

Заключение

Рассмотрены задачи достижения консенсуса в децентрализованных системах при переменной структуре связей, помехах и задержках. В статье рассматривается применение протокола локального голосования для задачи достижения консенсуса. Предложен алгоритм и создано программное обеспечения, позволяющее исследовать предельные параметры агентов (узлов) при достижении консенсуса.

CONSENSUS ACHIEVEMENT IN THE MULTIAGENT SYSTEM ON THE BASIS OF THE LOCAL VOTING PROTOCOL

Y. A. DUINOVA, N.I. LUSCHIK, S.I. POLOVENYA

Abstract

An algorithm for achieving consensus in a decentralized network with interference and specialized software is considered. The tasks of achieving consensus by a group of interacting agents are considered.

Список литературы

1. Скобелев П.О. Мультиагентные технологии в промышленных применениях: к 20-летию основания Самарской научной школы мультиагентных систем // Мехатроника, управление, автоматизация. 2011. № 12. С. 33-46.
2. Амелина Н.О. Исследование консенсуса в мультиагентных системах в условиях стохастических неопределенностей // Вестник ННГУ. 2013. № 1 (3). С. 173-179.
3. Huang M. Stochastic approximation for consensus: a new approach via ergodic backward products// IEEE Transactions on Automatic Control. 2012. Vol. 57, No. 12. P. 2994-3008.
4. Амелина Н.О., Фрадков А.Л. Приближенный консенсус в стохастической динамической сети с неполной информацией и задержками в измерениях // Автоматика и телемеханика. 2012. № 11. С. 6-30.
5. Граничин О.П. Стохастическая оптимизация и системное программирование // Стохастическая оптимизация в информатике. Т. 6. – СПб: Изд-во СПбГУ, - 2010. - С. 3-44.
6. Амелина Н.О. Мультиагентные технологии, адаптация, самоорганизация, достижение консенсуса // Стохаст. оптимизация в информатике. 2011. Т. 7. Вып. 1. С. 149–185.
7. Kar S., Moura J.M.F. Distributed consensus algorithms in sensor networks with imperfect communication: link failures and channel noise // IEEE Trans. Sig. Process. 2009. Vol. 57, №. 1. P. 355–369.