УДК 004.94+547.022

РЕАЛИЗАЦИЯ МУРАВЬИНОГО АЛГОРИТМА ФОЛДИНГА БЕЛКОВ МЕТОДАМИ ПРОГРАММНЫХ АГЕНТОВ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

Бекмуратов Т.Ф.
академик АН РУз, д.т.н., главный научный сотрудник
Центра разработки программных продуктов и аппаратно-программных комплексов
при Ташкентском университете информационных технологий,
тел.: +(998971) 234-07-59, e-mail: bek.tulkun@yandex.com

Базаров Р.К.
младший научный сотрудник Центра разработки программных продуктов и аппаратно-программных комплексов при Ташкентском университете информационных технологий,
тел.: +(99897) 737-48-05, e-mail: rustam.bazarov@gmail.com

Базаров Д.К.
младший научный сотрудник Центра разработки программных продуктов и аппаратно-программных комплексов при Ташкентском университете информационных технологий,
тел.: +(99890) 187-48-02, e-mail: marvelday@mail.ru

Проблема фолдинга белков, связанная с поиском третичной структуры белка по его первичной аминокислотной последовательности, является важнейшей в структурной биологии. К сожалению, даже такая грубая модель, как HP-PFP-2, учитывающая только гидрофобные взаимодействия аминокислотных остатков на двумерной решетке, описывает NP-полную задачу. Она успешно решается только эвристическими методами глобальной оптимизации, например, муравьиным алгоритмом. В статье исследуются способы модификации и распараллеливания муравьиного алгоритма для задачи фолдинга белков. Подробно описана программная реализация параллельного муравьиного алгоритма с использованием программных агентов, на платформе JADE в грид-системах. Обсуждаются результаты вычислительного эксперимента.

Ключевые слова: алгоритм оптимизации муравьиной колонии, фолдинг белков, агентские системы.

THE IMPLEMENTATION OF ANT COLONY OPTIMIZATION ALGORITHM FOR THE STUDY OF PROTEIN FOLDING PROBLEM IN DISTRIBUTED SYSTEMS BY THE SOFTWARE AGENTS

Bekmuratov T.F., Bazarov R.K., Bazarov D.K.

The protein folding problem associated with the search of the tertiary structure of the protein at the primary amino acid sequence, is important in structural biology. Unfortunately, even such a crude model as HP-PFP-2, taking into account only hydrophobic interaction of amino acid residues on a two-dimensional lattice is NP-hard and can be successfully solved only by heuristic methods of global optimization, for example, the ant colony optimization algorithm.The article examines the methods of modification and parallelization of ant colony optimization algorithm for the problem of protein folding. The software implementation of parallel ant colony optimization algorithm on the JADE platform in grid systems described in detail The article discusses the results of computational experiment.

Keywords: the ant colony optimization algorithm, protein folding problem, agent systems.

TAQSIMLANGAN TIZIMLARDA AGENT DASTURLASH USULIDA OQSIL FOLDINGI UCHUN CHUMOLI ALGORITMINI TADBIQ ETISH

Bekmuratov T.F., Bazarov R.K., Bazarov D.K.

Dastlabki aminokislotali ketma-ketligi bo'yicha oqsil foldingining uchlamchi tuzilishini izlash strukturali bioligiya sohasining asosiy muammosi hisoblanadi. Afsuski, HP-PFP-2 tipdagi qo'pol modellar ham faqat ikki o'lchovli panjarada aminokislota qoldiqlarining gidrofobli ta'siri inobatga olingan, NP murakkab hisoblanadi va faqat evristik usulda global optimallashtirish orqali echimi olinadi. Masalan, chumoli algoritmi yordamida. Maqolada oqsil foldingi masalasi uchun chumoli algoritmini parallel hisoblash va takomillashtirish usullari tadqiq etilgan. Grid tizimi Jade platformasida chumoli algoritmini parallelashtirish dasturiy tatbiqi to'liq tavsiflangan. Hisoblash tajribalardan olingan natijalari muhokama etilgan.

Tayanch iboralar: oqsil foldingi, chumolilar koloniyasini optimallashtirish algoritmi, oqsil foldingi, agent tizimlari.

1. Введение

1.1. Проблема фолдинга белков

Задача поиска третичной структуры белка по его первичной аминокислотной последовательности (protein folding problem, PFP) является важнейшей задачей структурной биологии. К сожалению, даже такая грубая модель структуры белка на двумерной решетке, как HP-PFP-2D [1], в которой учитываются только гидрофобные взаимодействия аминокислотных остатков, описывает NP-полную задачу [2]. Эта задача успешно решается только эвристическими методами глобальной оптимизации, например, муравьиным алгоритмом [3] на ресурсах распределенных вычислительных систем.

1.2. Структура белковой последовательности

Белок представляет собой полипептидную цепь, состоящую из аминокислотных остатков. Остаток — это то, что остается от свободной аминокислоты после ее встраивания в полипептидную цепь.

На рис. 1 представлен фрагмент белковой цепи, содержащей два последовательных остатка и их боковые группы R_l и R_{l+1}. Атом углерода при R-группе обозначен через C^{α}, а следующий за ним, имеющий полуторную связь с кислородом O, - через C'. Также на рисунке показаны углы вокруг ковалентных связей: φ, χ, ψ, ω.

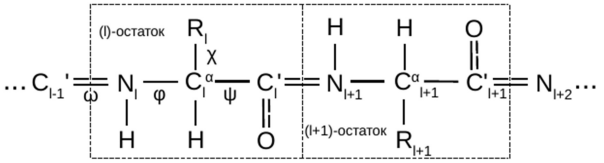

Рис. 1. Строение белковой цепи

Остатки в цепи связаны друг с другом полуторными, жесткими и плоскими (ω=0º) пептидными связями между атомами углерода C' и азота N.

Вращение вокруг одинарных валентных связей по углам φ и ψ обеспечивает гибкость белковой цепи и ее способность сворачиваться в глобулу.

Области запрещенных и разрешенных значений углов φ и ψ называют картами Раманчандрана. Эти области определяются размером R-группы каждой аминокислоты. Если предположить, что каждый остаток имеет только три конформационных состояния, соответствующих двум степеням свободы по углам φ и ψ соответственно:

а) (φ, ψ) = (-140º, -100º)..(0º,0º);

б) (φ, ψ) = (-180º..80º)..(0º, 180º);

в) остальные [4],

то для белка длиной L=101 число конформаций составит $3^{100} = 5x10^{47}$. Если, как предположил Левинталь [5], на каждую конформацию белок будет тратить 10^{-13}с, то потребуется 10^{27} лет на перебор всех возможных вариантов. Это дольше времени существования вселенной. Однако, как показывает практика, небольшие глобулярные белки сворачиваются в течение нескольких секунд.

Следовательно, самоорганизующийся белок следует по какому-то специальному пути сворачивания, и та структура, где этот путь заканчивается, и является его нативной структурой, не зависимо от того, есть ли еще более стабильная укладка, или нет.

Таким образом, решение задачи фолдинга разбивается на две задачи:

1. Поиск самой стабильной конформации белковой молекулы путем минимизации энергии всех видов внутримолекулярных взаимодействий, (термодинамика сворачивания).

2. Поиск способа получения самой быстро достижимой метастабильной структуры (кинетика сворачивания).

Сосредоточим внимание на первом вопросе, поскольку в решении второй задачи видимых успехов нет [6].

Сворачивание белка в организме (in-vivo) происходит при биосинтезе или сразу после него. В «пробирке» (in-vitro) способны сворачиваться только небольшие белки (200-300 остатков). На рис. 2 показана упрощенная схема процесса сворачивания белковой последовательности (in-vitro) [6].

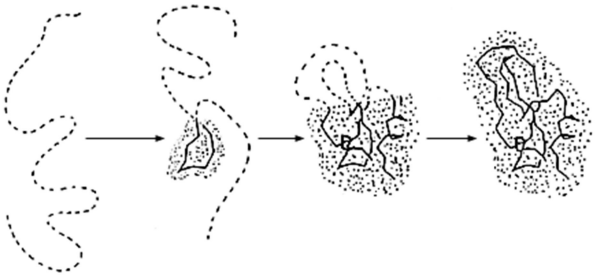

Рис. 2. Процесс сворачивания белковой цепи (in-vitro)

Фолдинг белка in-vitro начинается в позиции, именуемой в дальнейшем точкой инициации сворачивания. Выделенная на рисунке область соответствует части глобулы, уже получившей финальную конформацию. Для простоты боковые группы цепи на рисунке не показаны.

Существенное влияние на процесс свертывания оказывает взаимодействие между R-группами. Различают пять типов таких взаимодействий [7]: *ковалентные* поперечные связи между остатками цистеина (рис. 3*а*), *электростатические* взаимодействия — между противоположно заряженными R-группами (рис. 3б), *гидрофобные*, обусловленные водобоязнью некоторых остатков (на рисунке не показаны, поскольку это пример кооперативных, а не парных взаимодействий), *водородные* — между остатками с гидроксильными группами (рис. 3*в*) и *вандервальсовы* взаимодействия: любые два атома или молекулы на расстоянии свыше 2-3Å слабо притягиваются (рис. 3г) [6, 7].

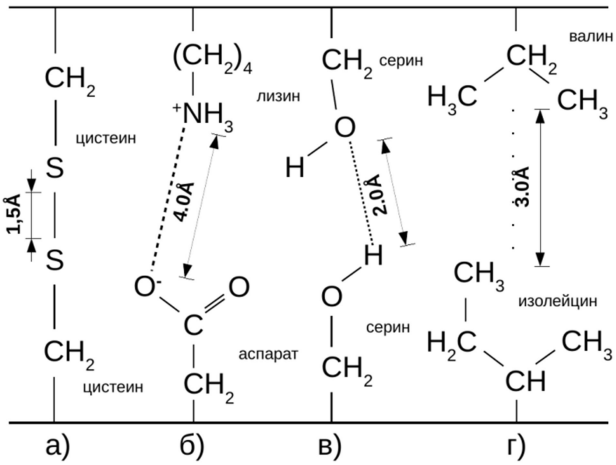

Рис. 3. Примеры типов взаимодействий между боковыми группами аминокислотных остатков в белковой цепи

На рис. 3 показаны два фрагмента цепи, соседние в глобуле, которые содержат по четыре аминокислотных остатка. Между их боковыми группами возникли вышеперечисленные типы взаимодействий. Также показаны приблизительные расстояния между взаимодействующими атомами и молекулами внутри белка, характерные для данных типов связей и взаимодействий. На рис. 1 боковые группы остатков главной цепи обозначены символами R_l.

Белковые структуры относятся к *сложным системам*, таким, которые состоят из множества элементов с динамически изменяющимся характером их взаимодействия. Систему определяют [8] как сложную, если ей присущи свойства:

1) *многообразие элементов и форм связей между ними*: 20 остатков, имеющих парные и кооперативные взаимодействия электрической, квантовой и химической природы;

2) *многомерность:* существует $10^{390}=20^{300}$ вариантов построения для типичного водорастворимого белка из 300 остатков, причем каждые 10^9 в среднем только один имеет стабильную структуру [9]. Цепи со стабильной структурой получены в результате естественного отбора.

3) *многокритериальность*: существует, по меньшей мере, два критерия: стабильность и время сворачивания белковой молекулы.

4) многовариантность: неясно, сколько существует способов достижения оптимальных значений по этим двум критериям стабильности и времени сворачивания.

5) многократные изменения состава и/или структуры системы: полярные боковые группы остатков могут завязывать энергетически выгодные водородные связи с молекулами воды, т.е. вовлекать в систему новые элементы. Скорость сворачивания и стабильность структуры также определяются внешними условиями: температурой, уровнем ph и типом растворителя. Самоорганизация крупных белков невозможна без вспомогательных белков - шаперонов.

б) многоплановость: существует несколько *моделей* и *методов* определения пространственной структуры белка. С помощью *полуэмпирических методов квантовой химии* осуществляется оптимизация геометрии и зарядового состояния остатков реакционного центра белка. Методами *эмпирических силовых полей* (молекулярная динамика) оцениваются парные взаимодействия молекул, рассматриваемых в качестве классических упругих частиц.

К сожалению, первая группа — квантово-химических методов — не может оперировать большими группами молекул, а вторая — молекулярная динамика — имеет низкую точность. Не учитываются, в частности, электронные эффекты (поляризуемость атомов, перенос электрона, образование и разрыв химических связей), а кооперативные гидрофобные взаимодействия и вовсе не могут быть смоделированы. Кроме того, время сворачивания небольших белков занимает *миллисекунды*, а максимально достижимое время моделирования при существующем уровне развития вычислительной техники: *микросекунды.*

Математическая сложность компьютерного фолдинга состоит в необходимости решать как геометрическую задачу поиска *плотной упаковки шаров* (непрерывная математика), так и комбинаторную задачу (дискретная математика). Плюс ко всему, дискретный компонент задачи использует сложную идею случайного блуждания без самопересечений (self-avoiding walk), которая по силе превышает даже NP-полноту (#P-полнота). Оптимизационные задачи с таким множеством непрерывно-дискретных признаков не имеют на сегодняшний день основательной математической теории [10]. Поэтому рассматривают различные упрощающие модели структуры белка. Аминокислоты в глобулярных белках можно представить в виде квадратов (решетчатая модель) или окружностей со стороной/диаметром от 4.0 до 6.2Å (в среднем 5.3Å), в зависимости от размера R-группы [11].

Согласно решетчатой HP-модели, предложенной Кеном Диллом в 1985 г. [1], учитываются только гидрофобные взаимодействия в малых глобулярных белках (до 200 остатков), доминирующих [12] в водном окружении. Белок, как сложная система, при таких упрощениях содержит два типа элементов: гидрофобные и гидрофильные остатки (рис. 4). Две гидрофобные аминокислоты образуют контакт, если они соседние в свертке, но не в цепи. Задача состоит в нахождении такой свертки белковой цепи, в которой достигается максимум контактов между гидрофобными остатками. Под энергией свертки понимается число этих контактов со знаком минус.

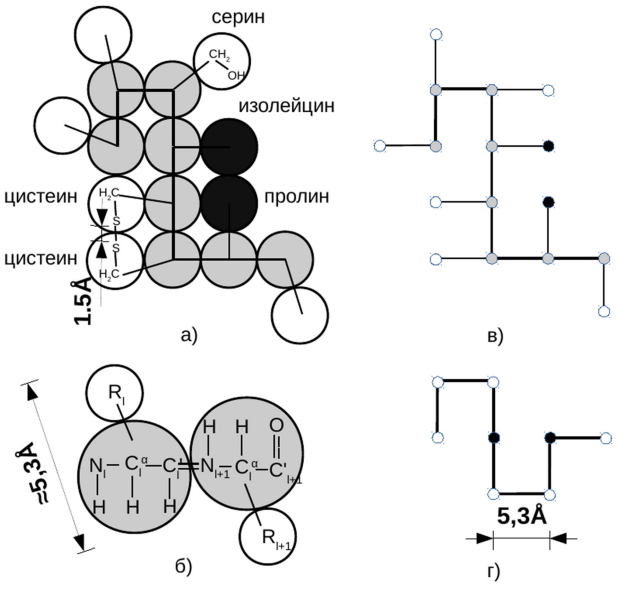

Рис. 4. Нерешетчатая (а) и решетчатые (в, г) HP-модели структур белка на плоскости

Белки в процессе сворачивания формируют ядро, образуемое гидрофобными остатками (на рис. 4 обозначены черным цветом). Боковые группы гидрофильных остатков (обозначены белым) «болтаются» в воде. Но даже при таких упрощениях поиск оптимальной свертки является NP-полной задачей [2] и может эффективно решаться только вероятностными алгоритмами глобальной оптимизации на ресурсах распределенной вычислительной системы (РВС).

Целью настоящей работы является создание РВС, и распределенного алгоритма решения задачи фолдинга белков, реализуемого с помощью программных агентов.

2. Характеристика типов РВС, муравьиного алгоритма задачи фолдинга белков и агентских систем

2.1. Распределенные вычислительные системы представляют собой множество соединенных каналами связи независимых компьютеров, которые, с точки зрения пользователя прикладного программного обеспечения (ПО), выглядят единым целым [13]. При запуске приложения в РВС пользователю следует учитывать: объединяемые ресурсы обладают различной вычислительной мощностью; на одном и том же ресурсе сосуществуют несколько заданий от различных пользователей; ресурсы в РВС соединены сетью разной производительности; в процессе реализации приложения осуществляется непрерывный обмен данными между ресурсными центрами (РЦ) системы, сокращающими общее время решения задач. Поэтому пользователю следует минимизировать обмен данными между несколькими РЦ, а с учетом сильной загруженности последних отдельные части приложения целесообразно сделать самостоятельными. Учет указанных обстоятельств и их реализация обеспечивают организацию распределенных, а не параллельных приложений.

2.2. Алгоритм оптимизации муравьиной колонией (Ant Colony Optimization, ACO) является одним из немногих алгоритмов, эффективно реализуемых в РВС. Задача его реализации состоит в поиске оптимального способа распределения алгоритма на вычислительных ресурсах РВС. Для этого существуют три варианта:

1) распределение колоний муравьев между рабочими узлами ресурса [14] одного РЦ;

2) распределение муравьев одной колонии, позволив им «блуждать» в поисках наименее загруженного узла как между рабочими узлами одного ресурса, так и между территориально разнесенными РЦ;

3) распределение операций, выполняемых муравьями [15]. Последнее имеет смысл делать не на кластерах, а на суперкомпьютерах (векторных, матричных или на графических процессорах) с минимальным временем передачи данных между вычислительными элементами процессора.

Подробно методы распараллеливания популяционных алгоритмов глобальной оптимизации освещены в [16].

В данной статье рассматривается второй способ распределения, основанный на агентском подходе к реализации муравьиного алгоритма.

2.3. Агентские системы и мобильные агенты

Агент - вычислительная сущность, действующая от имени другой сущности для выполнения какого-либо задания или достижения заданной цели.

Агентская система - замкнутый набор программных средств, включающий в себя знания определенной предметной области и обладающий способностью вести себя независимо, выполняя при этом действия, необходимые для достижения определенных целей.

Агенты создаются в динамически изменяющейся среде и обладают следующими свойствами:

- *автономность:* способность действовать от имени пользователя или других программ путем изменения способа выполнения своих задач;
- *проактивность:* способность решать свои задачи, а также принимать решения на основе своих внутренних решений;
- *реактивность:* способность реагировать на внешние воздействия, события, стимулы и адаптировать свое поведение и механизм принятия решений;
- *коммуникация и кооперация:* способность взаимодействовать и выходить на связь с другими агентами в мультиагентной системе для обмена информацией, получения инструкций и выдачи ответов;
- *переговорная функция:* способность к переговорам с другими агентами;

- *обучение:* способность принимать решения и улучшать производительность в результате взаимодействия с внешней средой;

- *мобильность:* способность агента менять свое положение в окружении, не меняя самого окружения.

В нашем случае - это способность агента перемещаться с одного ресурса РВС на другой.

Мобильные Агенты [17] являются последней ступенью эволюции систем, основанных на мобильности кода. Мобильный агент – это программа, способная мигрировать по сети с одного узла на другой, обеспечивая переносимость не только кода, но и контекста выполнения.

Идея самоуправляемого выполнения, т.е. способности к миграции к источникам данных, предложена в [18, 19] как альтернатива клиент-серверной технологии, обладающая более эффективным и гибким режимом коммуникации.

Обзор средств интеграции агентских и распределенных вычислительных грид-систем представлен в [20], где рассматриваются программные средства JADE Extensions, Bond, ASF, MAGDA. В [21] описывается агентская система в облачной РВС. Пример интеграции агентской платформы Theatre с программным обеспечением промежуточного уровня (ППО) globus toolkit-4.0.8 приводится в работе [22].

На РВС, построенных с использованием агентского ППО Hermes [23], решались задачи в области управления производством, биоинформатики и системной биологии.

Программная реализация агентского муравьиного алгоритма для задачи коммивояжера описана в работе [24]. Эта РВС создана на основе ПО JADE.

И, наконец, агентский подход к решению аналогичной рассматриваемой в статье задаче с помощью генетического алгоритма рассмотрен в [25].

Применение технологии мобильных агентов в РВС, как правило, обусловлено необходимостью решения таких задач, как распределение загрузки, обработка ошибок и обнаружение служб [26].

В [27] программные агенты используются для назначения ресурсов и распределения приложений и данных. В [28] описана созданная нами агентская система для задач телемедицины.

Разработка агентских распределенных систем предусматривает решение следующих задач: создание системы поиска ресурсов и оценки степени их занятости с целью оптимального распределения агентов в РВС; обеспечение мобильности агентов; разработка системы передачи сообщений между агентами, находящимися в различных РЦ; обеспечение целостности и безопасности работы распределенного приложения.

Поэтому важным является обеспечение безопасности, т.е. предоставление возможности запуска агентских приложений в проблемно-ориентированной РВС только членам определенной виртуальной организации (ВО).

Для этого использован MAGDA – плагин [29], интегрирующий агентское ПО JADE [30] с ППО gt-4.0.7 [31].

2.4. Проблемно-ориентированная агентская РВС

Агентское ПО JADE включает: окружение (runtime environment), запускаемое на одном компьютере (host), где агенты могут существовать; библиотеку классов (library of classes) - для разработки агентов; графические средства (graphical tools) - для администрирования и мониторинга активности агентов. Каждый экземпляр окружения называется контейнером (container), поскольку может содержать несколько агентов. Множество активных контейнеров называется платформой (platform). Главный контейнер (main container) всегда находится в активном состоянии, остальные регистрируются при запуске. При этом обычный контейнер сообщает главному свой адрес: (host, port), по которому тот сможет его найти. Главный контейнер всегда содержит два служебных агента: систему управления агентами (agent management system, AMS) и службу каталогов (directory facilitator, DF), благодаря которой агенты находят друг друга.

Свои задачи агент выполняет посредством различных «поведений» (behaviours). В ПО JADE поведение добавляется при запуске агента или внутри других поведений с помощью метода *addBehaviour()*, а реализуется в методах *action()* и *done()*. Различают общие, простые, одноразовые и циклические поведения.

Агентская РВС, один из ресурсов которой приведен на рис. 5 [21], представляет собой множество территориально разнесенных кластерных систем. Каждая из них содержит один управляющий – *инфраструктурный* узел (front-end, FE) и набор *рабочих* узлов (worker node, WN), обменивающихся сообщениями по локальной сети.

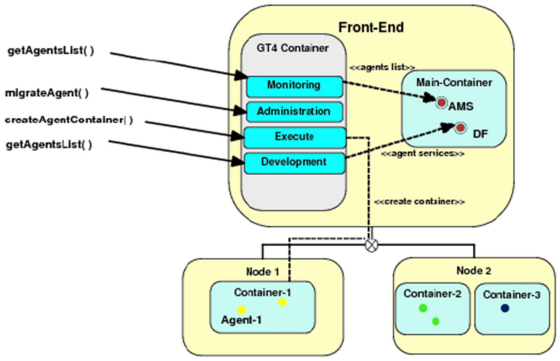

Рис.5. Ресурс агентской РВС

Каждый ресурс агентской РВС состоит из платформы мобильных агентов (Mobile Agents Platform) и дополнительного программного компонента, обеспечивающего доступность мобильных агентов для аутентификации и авторизации с помощью специально разработанных грид-служб [29].

Агентская платформа, в свою очередь, состоит из агентских контейнеров, которые могут выполняться как на одном, так и на нескольких рабочих узлах. Рабочий узел, содержащий хотя бы один контейнер, назовем *агентским узлом.*

Ресурсы и узлы, а в облачных РВС и сеть между ними, могут быть как физическими, так и виртуальными. Клиентское приложение остается на пользовательском компьютере, имеющем доступ к РВС, и взаимодействует с ней через вызов вышеупомянутых грид-служб.

Программный компонент агентской РВС представлен четырьмя грид-службами, развернутыми на инфраструктурном узле с заранее установленным ППО gt-4.0.7:

1. **Administration** предоставляет набор операций, позволяющих пользователю управлять агентской платформой и конфигурацией агентов. Эта служба необходима для балансировки нагрузки между ресурсами РВС. Перемещение агентов осуществляет администратор ВО.

2. **Monitoring** позволяет администратору следить за агентской платформой и ресурсами РВС. Задача службы - помочь в принятии решений о переносе агентов между ресурсами.

3. **Execute** позволяет аутентифицированному пользователю запускать мобильные агенты в РВС и передавать им необходимые сообщения.

4. **Development** позволяет устанавливать готовые и разрабатывать новые мобильные агенты. Служба полезна для обнаружения имеющихся в РВС грид-служб.

Обмен сообщениями между агентами, запущенными на разных агентских ресурсах, осуществляется по протоколу MTP (Message Transport Protocol). В плагин MAGDA входит своя версия MTP, обеспечивающая возможность перемещения агентов между ресурсами в РВС.

Исходные коды, скомпилированные gar- и jar-файлы и соответствующие руководства по установке и использованию этих служб можно найти в [29].

3. Программная реализация алгоритма оптимизации муравьиной колонией на ресурсах агентской РВС

Математическая постановка, муравьиный алгоритм решения рассматриваемой задачи и его реализация на РВС с видеокартами (**A**nt **C**olony **O**ptimization for HP-model protein folding problem on graphic processing units, GPU-ACO-HP-PFP-2) описаны в работе [15]. В данной статье представлена реализация ACO-HP-PFP-2 на ресурсах агентской РВС.

3.1. Структура и параметры программы

Входными данными для программы являются: число муравьев в колонии – M; число направлений сворачивания – N; коэффициенты α и β, учитывающие влияние феромонов и следов соответственно; коэффициент испарения следов — ρ и начальные значения матрицы феромонов — τ_0. Последовательность Q, состоящая из L полярных (гидрофильных) и неполярных (гидрофобных) аминокислот, эвристическая информация – η, матрица феромонов – τ, вектор вероятностей случайных величин – P; тур муравья, представленный последовательностями узлов – T и направлений – Y, вектор приращений в названиях узлов – ΔT = (1, 2L, -1, -2L); энергия – E, представляющая собой сумму гидрофобных контактов с отрицательным знаком;

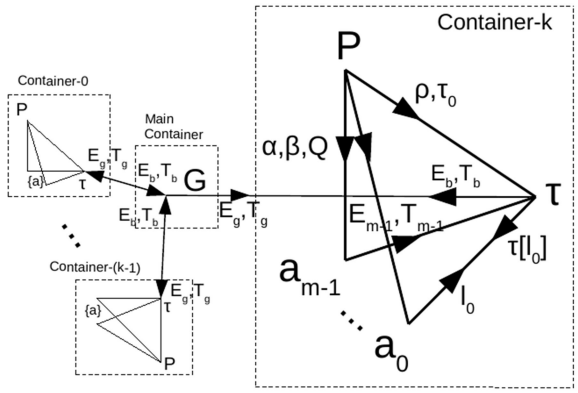

Рис. 6. Распределение агентов муравьиного алгоритма фолдинга белков в агентской РВС

3.2. Этапы выполнения распределенного муравьиного алгоритма в ресурсах РВС

1) **Создание контейнеров на рабочих узлах ресурса**

В главном контейнере создается k обычных контейнеров (по числу рабочих узлов в данном ресурсе проблемно-ориентированной РВС, и G-агент, сохраняющий в выходном файле свертки белковой последовательности T_g с наименьшим значением энергии E_g среди всех рабочих узлов данной РВС.

2) **Создание агентов.** В каждом контейнере создается агент параметров P-агент, который в свою очередь создает M агентов муравьев, назовем их A-агенты, передавая им параметры α, β и сворачиваемую последовательность Q, а также *T-агента* - агента феромонных следов с параметрами ρ и τ_0 (рис. 6).

3) **Создание белковых сверток.** Каждый A-агент каждого контейнера вызывает поведение stepBhv, осуществляющее сворачивание последовательности от точки инициации i до конца последовательности L, затем от точки инициации до ее начала. Точку инициации каждый муравей выбирает наудачу, как случайное число в диапазоне от 0 до L-1 (рис. 7).

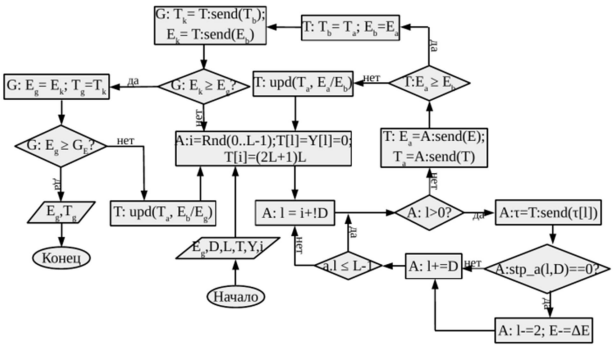

Рис.7. Блок-схема распределенного алгоритма фолдинга белков

В процессе выполнения этого поведения Т-агенту посылается текущее положение аминокислоты *l* в свертке, а в ответ принимается вектор *τ*[l] — строка матрицы феромонов *τ*[l][d], l=0..L-1, d=0..N-1 агента феромонов.

При (l<0), когда белковая последовательность полностью свернута, поведение stpBhv приостанавливается и запускается поведение sendTE, в котором А-агент передает полный тур Ta и энергию свертки Ea агенту феромонов для локального обновления таблицы τ[l][d]. По завершении поведения sendTE текущий муравей обнуляет параметры T, E, I, после чего процесс поиска свертки с лучшим значением энергии повторяется снова. Агент феромонов, имея лучшую энергию свертки от других муравьев данного контейнера, сравнивает ее с энергией, полученной от очередного муравья и, если она больше, обновляет значение лучшей энергии Eb и тура Tb. Затем осуществляет локальное обновление матрицы феромонов лучшим туром Tb . Лучший тур и его энергия отправляются в главный контейнер G-агенту.

4) Выбор наилучшей свертки. G-агент собирает $E^{(k)}_b$, $T^{(k)}_b$, k=0..K-1, со всех контейнеров, анализирует $E^{(k)}_b$ и выбирает тур T_g с наименьшим значением энергии $E_g = \min\{E^k_b, k=0..K-1\}$ и широковещательно рассылает Т-агентам всех контейнеров для глобального обновления их матриц феромонов.

Если E_g станет равным некоторому значению оценки энергии G_E, то процесс поиска новых сверток прекращается. Оценка G_E выбирается как число, равное половине гидрофобных контактов аминокислотной последовательности [10].

Каждый А-агент колонии сворачивает белок в прямом направлении с помощью процедуры формирования белковой свертки stp_a(l,D), описываемой в следующем разделе.

При этом, если процесс фолдинга зашел в тупик, т.е. функция stp_a(l,D) (рис. 7) вернула нулевое значение, то происходит откат процесса построения свертки на шаг назад: l-=2.

Для операторов блок-схемы распределенного алгоритма использованы следующие обозначения:

<имя_агента>: <список операторов>

Например, запись T: E_a = A:send(E) означает: присвоить локальной переменной E_a агента T переданное агентом A функцией send() локальное значение переменной E.

3.3. Процедура формирования белковой свертки

Расчет вероятности перемещения, само перемещение муравья, а вместе с тем и включение в частичную свертку еще одной аминокислоты выполняет процедура *stp_a(l,D),* вызываемая каждым муравьем колонии в циклическом поведении *stpBhv* (рис. 8а).
В качестве аргументов процедура принимает позицию текущей аминокислоты l и флаг сворачивания D. При D=0 фолдинг происходит слева от точки инициации, при D=1 - справа.

Сначала определяется текущий узел cc (**c**urrent **c**ity) по формуле: cc = (D)? t[l-1] : t[l+1].

Затем в цикле по всем направлениям d=0..N-1 выбираются следующие узлы {nc} (**n**ext **c**ities). Если среди них есть посещенные, то вероятности по соответствующим направлениям приравниваются к нулю. Для остальных рассчитывается эвристическая информация (рис. 8б). Далее определяются и суммируются вероятности выбора направления по известным значениям вектора феромонов и рассчитанной функцией mkη(nc,l,D) эвристической информации. В диапазоне от 0 до Σ выбирается случайное число r. Затем инкрементируется направление сворачивания d и увеличивается переменная P_0 на значение вероятности P[d] в соответствующем направлении. Если сумма $P_0 > r$, то, значит, выбран соответствующий узел T[l] на шаге l. Сам узел, направление сворачивания и приращение энергии запоминаются. Функция stp_a(l,d) возвращает 1 при удачном выполнении и 0 при неудачном, когда не удалось выбрать направление сворачивания аминокислотной последовательности (рис. 9).

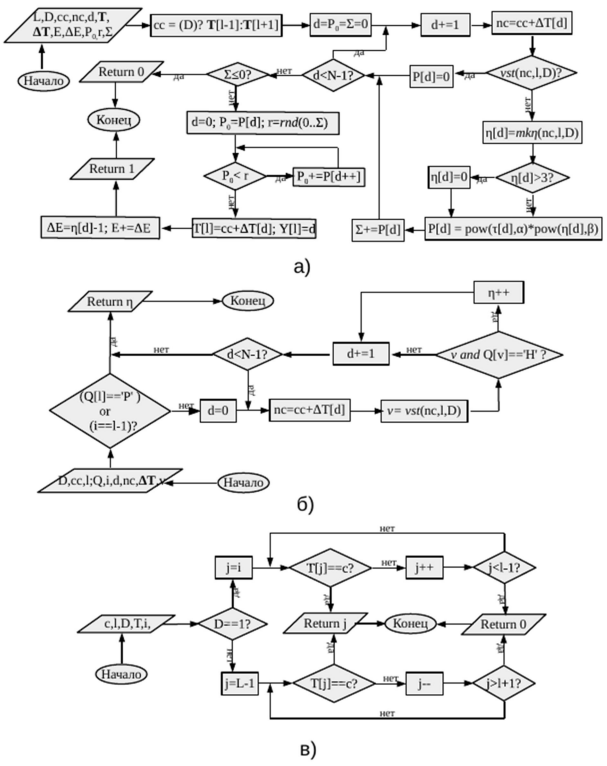

Рис. 8. Блок-схема процедуры stp_a(l,d) и вызываемых в ней функций: vst(nc,l,D) и mkη(nc,l,D).

Функция vst(nc,l,D) (рис. 8в) проверяет, был ли посещен узел nc, следующий за текущим cc, соответствующим узлу T[l-1] или T[l+1] в зависимости от флага D.

Функция mkη(nc,l,D) (рис. 8б) находит эвристику η[d] = ΔE+1, где ΔE — значение приращения энергии частичной свертки при выборе следующего узла. Например, выбор узла nc по направлению d = 1 уменьшит энергию свертки на ΔE=2 (рис. 9), поскольку текущая гидрофобная

аминокислота становится соседом для двух других таких же гидрофобных аминокислот.

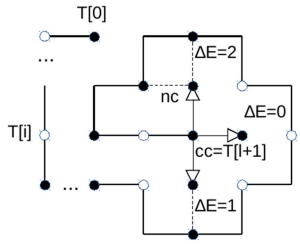

Рис. 9. Расчет эвристики для узла nc

При d=3 ΔE=1, а выбор направления d=0 не уменьшит энергию текущей свертки. Соответственно η[0] = 1, и вероятность выбора данного направления будет определяться только феромонами, полученными от *T-агента*.

Исходный код распределенного алгоритма фолдинга белков находится в файлах ParamsAgent.java TauAgent.java AntAgent.java. Этот код компилируется и запускается с помощью следующих команд:

```
[globus@gpu lib]$ javac -classpath $CLASSPATH ParamsAgent.java
TauAgent.java AntAgent.java && java jade.Boot –gui
"Vasiliy:ParamsAgent"
```

где
CLASSPATH=.:jade.jar:jadeTools.jar:http.jar:commons-codec-1.3.jar
Vasiliy-имя собственое для агента параметров.

Переменная CLASSPATH обеспечивает доступ к необходимым библиотекам ПО JADE:

- jade.jar: содержит все JADE пакеты, кроме add-ons, MTPs и графических средств;
- jadeTools.jar: графические средства;
- http.jar: основанный на HTTP протокол MTP. По умолчанию инициируется при запуске агентской платформы;
- commons-codec-1.3.jar содержит кодеки Base64, используемые JADE [30].

В результате этой команды запустится главный контейнер (рис. 10), содержащий агента параметров, десять агентов-муравьев, агента феромонов и два служебных агента (AMS и DF).

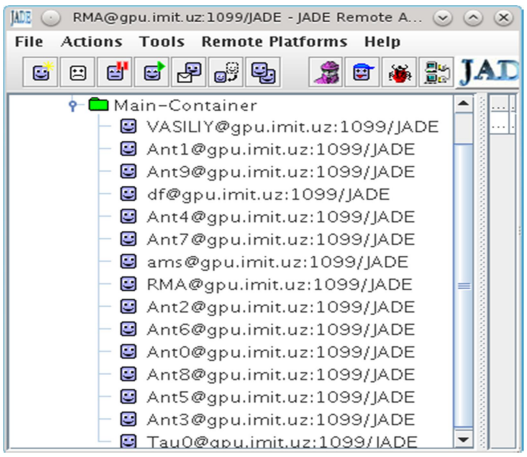

Рис. 10. Панель управления агентской системой

С панели управления агентской системой есть возможность уничтожать агентов, приостанавливать их работу или запускать новых агентов в данном контейнере для оценки эффективности всей системы в целом.

Поскольку перед нами стоит задача запуска этой программы на ресурсах РВС и только зарегистрированным в ВО пользователям, то нужно сначала создать эту РВС, затем интегрировать агентское и промежуточное ПО, установленное на инфраструктурном узле каждого ресурсного центра в РВС, написать клиентское приложение для вызова соответствующих грид-служб на инфраструктурном узле.

Аппаратной платформой агентской РВС являются виртуальный инфраструктурный узел: fe.imit.uz и физический рабочий узел: wn011.imit.uz, выполняющий также роль клиентского ПК.

4. Процедуры создания ресурсного центра проблемно-ориентированной агентской РВС

1. Установка ППО. Скаченный с сайта [31] архивный файл ППО gt-4.0.7 следует распаковать в домашние каталоги пользователя globus узлов каждого ресурса РВС и запустить установочный скрипт:

```
[globus@fe gt4]$ ./configure --prefix=$GLOBUS_LOCATION &&
make && make install
```

Перед выполнением команды конфигурации следует записать значения переменных окружения: JAVA_HOME, ANT_HOME, GLOBUS_LOCATION в переменную PATH.

2. Создание центра сертификации на инфраструктурном узле РВС осуществляется командой

```
[globus@fe ~]$ $GLOBUS_LOCATION/setup/globus/setup-simple-ca
```

После запуска этого скрипта следует указать данные для администратора ВО, который будет подписывать сертификаты новых пользователей и ресурсов.

3. Генерация и подпись сертификатов в РВС. Запрос сертификатов ресурсов осуществляется на регистрируемом узле командой

```
[root@fe ~]# grid-cert-request -host `hostname`,
```

В результате ее выполнения будут сгенерированы файлы: hostkey.pem, hostcert_request.pem. Последний из них, подписанный командой:

```
[globus@fe ~]$ grid-ca-sign -in hostcert_request.pem \
    -out hostcert.pem,
```

следует отправить на подпись администратору ВО.

Выходной файл hostcert.pem возвращается администратору узла, сгенерировавшему запрос.

Аналогично создаются пользовательские сертификаты. Только команда grid-cert-request

вызывается регистрируемым пользователем без аргументов. Перед запуском контейнера грид-служб на инфраструктурном узле следует подписанный сертификат узла и ключ к нему поместить в каталог /etc/grid-security/ под именами containercert.pem и containerkey.pem соответственно.
Сам контейнер грид-служб запускается командой:

```
[globus@fe ~]$  globus-start-container -p 8443
```

4. Интеграция агентского ПО с промежуточным. Скомпилированные файлы для грид-служб, обеспечивающих интеграцию JADE с gt-4.0.7, хранятся в архивах с gar-расширением на сайте [29]. Скачиваем их и разворачиваем с помощью команды globus-deploy-gar. Например, служба мониторинга разворачивается так:

```
[globus@fe ~]$ globus-deploy-gar \
    serviziGRID_monitoring_GridMonitoring.gar
```

Аналогично устанавливаются остальные три грид-службы. Чтобы изменения вступили в силу, контейнер грид-служб следует перезапустить. В результате в списке грид-служб контейнера появятся записи:

```
[globus@fe ~]$  globus-start-container -p 8443
...
https://127.0.0.1:8443/wsrf/services/GridAdministratorService
https://127.0.0.1:8443/wsrf/services/GridDeveloperService
https://127.0.0.1:8443/wsrf/services/GridMonitoringService
 https://127.0.0.1:8443/wsrf/services/GridUserService
...
```

Это адреса, по которым следует обратиться при вызове этих грид-служб в клиентской части распределенного агентского приложения.
5. Проблемная ориентация РВС выполняется созданием клиентского приложения и клиентской грид-службы, осуществляющей запуск этого приложения на узлы созданной РВС.

Ниже приводим исходный код скрипта запуска приложения:

```
[globus@fe ~]$ cat ./JADE-bin-3.5/scriptAVVIO/myscripts.sh
#!/bin/sh
ORIG=`pwd`
cd ~/JADE-bin-3.5/lib
CLASSPATH=.:jade.jar:jadeTools.jar:http.jar:commons-codec-1.3.jar
export CLASSPATH ORIG
java ~/300813/jade.Boot $1 >> /tmp/out.txt
cd $ORIG
```

5. Запуск программы и визуализация результатов вычислительного эксперимента

Далее, осуществляется запуск самой программы, находящейся в каталоге ~/300813 на инфраструктурном узле, которой в качестве аргумента передается имя агента.

Перед запуском следует получить временный прокси-сертификат на использование ресурсов РВС командой:

```
[globus@fe ~]$ grid-proxy-init
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until: Fri Aug  1 00:08:27 2014
```

После того как пользователь ВО введет только ему известный пароль (GRID pass phrase), подписанный администратором ВО, будет сгенерирован сертификат на использование ресурсов в течение 12 часов. В данном конкретном случае сертификат годен до пятницы 1 августа 2014 г. Далее, следует запустить контейнер на узле fe.imit.uz:

```
[globus@fe ~]$ globus-start-container -p 8443
```

и вызвать службу: GridAdministratorClient с клиентской машины wn011.imit.uz, которая в свою очередь вызывает скрипт запуска задания: myscripts.sh, расположенный на fe.imit.uz.

Значения свертки и соответствующие им значения энергий поступают в выходной файл: /tmp/out.txt. Вот фрагмент содержимого этого файла:

```
[globus@wn011 ~]$  cat /tmp/out.txt
x      y      T[]    Q   E
0      0      820    1   8
0      -1     780    0   8
1      -1     781    1   8
...
```

Первые два столбца — декартовы координаты узла, третий — имя узла, четвертый — аминокислотная последовательность, пятый — энергия свертки.

Ниже приведен скрипт визуализации этого файла:

```
#!/bin/bash
STEP=20
ANTS=10
i=0
/usr/bin/split -l $STEP -d /tmp/out.txt cities
while [ "$i" -le $ANTS ]; do
/usr/bin/gnuplot -persist  -e "
set label 'E=`tail -1 cities0$i | gawk '{print $6}'`' font 'Times, 14' at 3,3;
plot [-5:5] [-5:5] 'cities0$i' with steps linewidth 2, 'cities0$i' using 1:2:4
with labels font 'Times, 14'; " ;
i=$(( i + 1 ))
done
```

Этот скрипт рисует свертки последовательности «HPHPPHHPHPPHPHHPPHPH», созданные колонией из десяти муравьев (рис. 11).

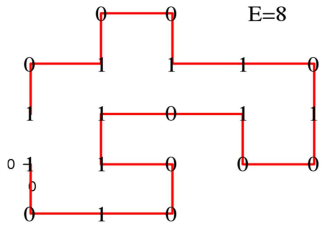

Рис.11. Визуализация результатов фолдинга белковой последовательности

На (рис. 11) нуль соответствует полярной аминокислоте, единица — гидрофобной. На графике также отображается число гидрофобных контактов в свертке.

Таким образом, в результате выполнения изложенных выше процедур была создана РВС из двух узлов и развернуты грид-службы MAGDA для интеграции агентского ПО JADE-3.5 с ППО gt-4.0.7. С рабочего узла запущена платформа из 14 агентов:

служебные AMS- DF- агенты, P-, T-агенты и десять A-агентов. Созданы скрипты запуска программы и визуализации результатов ее работы.

6. Заключение

По результатам описанного выше вычислительного эксперимента заключаем следующее:

Такое свойство агентской системы, как автономность, выражается в том, что для работы программы достаточно запустить одного агента-параметров и агента-муравья.

Агент-муравей может не обращаться к агенту феромонов для построения свертки (что отображается свойством проактивности). Вместе с тем, свойство коммуникации и кооперации проявляется в том, что качество получаемых решений при этом оказывается сильно зависимым от объема дополнительной информации, оставляемой другими агентами муравьями в виде следов, передаваемых T-агенту.

Агент параметров следит за временем получения качественного решения и динамически обновляет параметры алгоритма (пока лишь число обращений к матрице феромонов перед обнулением муравьев) с целью минимизации времени получения наилучшей свертки. В этом выражается способность агентской системы к обучению.

Разработанная программа агентской системы масштабируема, не требовательна к ресурсам, может запускаться фоновым процессом на обычных персональных компьютерах. Основной недостаток этой программы, по сравнению с mpi- и gpu-версиями [9,10], состоит в том, что она слишком медленная. Тем не менее, программа дает лучшее решение: строит свертки с числом топологических контактов 8 для последовательности из 20 символов (рис.6). Она хорошо подходит для сворачивания коротких последовательностей (до 50 остатков).

На программный продукт получено свидетельство [33].

Литература

[1] *DillK.A.* Theory for the folding and stability of globular Proteins / Biochemistry № 24. - 1985. - Pp. 1501-1509.

[2] *Crescenzi P., Goldman D., Papadimitriou C., Piccolboni A., M. Yannakakis.* On the complexity of protein folding. J Comp Bio, 5, 1998.

[3] *Shmygelska A. and H.H. Hoos.* An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem // Bmc Bioinformatics. - 2005. - № 6(1). - 30 p.

[4] *A.S. Kolaskar and Sangeeta Sawant*, Prediction of conformational states of amino acids using Ramachandran plot / International Journal of Peptide and Protein Research. 1996. - № 47. - Pp. 110-116.

[5] *C. Levinthal.* How to fold graciously // Mössbaun Spectroscopy in Biological Systems Proceedings. Univ. of Illinois Bulletin. Vol.67. -1969. - № 41. - Pp. 22-24

[6] *Финкельштейн А.В., Птицын О.Б.* Физика белка: Курс лекций с цветными и стереоскопическими иллюстрациями и задачами. - М.: КДУ, 2012. - 456 с.

[7] *Макеев А.В.* Основы биологии. - М.: Мир, 1997. - 235 с.

[8] *Малышенко А.М.* Математические основы теории систем: учебник для вузов. – Томск: Изд-во Томского политехнического университета, 2008. – 364 с.

[9] *Alberts B., Johnson A., Lewis J.* Molecular Biology of the Cell. New York Garland Science, 2002. – Режим доступа: https://www.ncbi.nlm.nih.gov/books/NBK26830

[10] *Istrail S, Lam F.* Combinatorial algorithms for protein folding in lattice models: a survey of mathematical results. *Commun. Inf. Syst*., 2009. - № *9*(4). - Pp. 303–346.

[11] *Richards F.M.* Areas, volumes, packing and protein structure / Annual Review of Biophysics and Bioengineering. - 1977. - Vol.6, № 1. - Pp. 151-176.

[12] *Dill K.A.* Dominant forces in protein folding. Biochemistry, 29:7133-7155, 1990.

[13] *Таненбаум Э., М. ван Стеен.* Распределенные системы. Принципы и парадигмы. – СПб.: Питер, 2003. – 877с. (Серия «Классика computer science»).

[14] *Бекмуратов Т.Ф., Мухамедиева Д.Т., Базаров Р., Ахмедов Д.Д.* Параллельный муравьиный алгоритм оптимизации // Узб. журнал «Проблемы информатики и энергетики». – Ташкент, 2014. - № 1-2. - С.11-15.

[15] *Базаров Р.К.* Реализация муравьиного алгоритма фолдинга белков на графических процессорах // Проблемы вычислительной и прикладной математики. - Ташкент, 2017. - № 1. - С. 86-91.

[16] *Карпенко А.П.* Параллельные популяционные алгоритмы одно- и многоцелевой оптимизации. Режим доступа: http://agora.guru.ru/abrau2014/pdf/240.pdf

[17] *Bellifemine, Fabio; Poggi, Agostino; Rimassa, Giovanni.* JADE: A FIPA-compliant Agent Framework ; In: Proceedings of PAAM'99, London, April 1999. - Pp. 97-108.

[18] *Boese, Joos-Hendrik; Feuerstack, Sebastian.* Adaptive User Interfaces for Ubiquitous Access To Agent-based Services; In: Workshop on Human- Agent Interaction, Agentcities ID3; Barcelona, Spain, 2003.

[19] *Cao, J.; Kerbyson, D.J.; Nudd, G.R.* Performance Evaluation of an Agent-Based Resource Management Infrastructure for Grid Computing. // Proceedings of 1st IEEE/ACM International Symposium on Cluster Computing and the Grid. -Brisbane, Australia, May 15-18 2001.- Pp. 311-318.

[20] *Senobari M., Drozdowicz M. , Paprzycki M., Kuranowski W., Ganzha M., Olejnik R., Lirkov I.* Combining an JADE-agent-based Grid infrastructure with the Globus middleware—Initial Solution. // Proceedings of the CIMCA-IAWITC 2008 Conference, IEEE CS Press, Los Alamitos, CA, 2008. - Pp. 890-895.
[21] *Aversa R., Martino B., Rak M., Venticinque S.* Cloud agency: A mobile agent based cloud system // Proc. Int Conf. Complex, Intelligent and Software Intensive Systems, 2010. - Pp. 132-137.
[22] *Cicirelli, F, Furfaro, A, Nigro, L, Pupo, F.* Agents Over The Grid: An Experience Using The Globus Toolkit 4. // Proceedings of the 26th European Conference on Modelling and Simulation (ECMS'2012), 2012. - Pp. 78-85.
[23] *Corradini F., Merelli E.* Hermes: agent-based middleware for mobile computing. / Lecture Notes in Computer Science, Vol 3465. Springer-Verlag, Berlin/Heidelberg, 2005. - Pp. 234-270.
[24] *S. Ilie and C, Badica*, "Effectiveness of Solving Traveling Salesman Problem Using Ant Colony Optimization on Distributed Multi-Agent Middleware," // Proceedings of international Multiconference on Computer Science and Information Technology (IMCSIT), 2010. - Vol.5. - Pp.197-203.
[25] *González P.P.G., Beltrén H.I., Rojo-Domínguez A., Eduardo M., Gutiérrez S.* Multi-Agent Systems Applied in the Modeling and Simulation of Biological Problems: A Case Study in Protein Folding. / World Academy of Science, Engineering & Technology. Issue 34. - 2009. - Pp. 128-137.
[26] *Cao, J.; Spooner, D.P.; Jarvis, S.A.; Nudd, G.R.* Grid Load Balancing Using Intelligent Agents // Future Generation Computer Systems special issue on Intelligent Grid Environments: Principles and Applications. 21:1. 2005. - Pp. 135-149.
[27] *Cao J., Spooner D.P., Turner J.D., Jarvis S.A., Kerbyson D.J., Saini S., Nudd G.R.* Agent-based Resource Management for Grid Computing // 2nd IEEE International Symposium on Cluster Computing and the Grid. - Berlin, Germany, May, 2002. - 350 p.
[28] *Adilova F.T., Ibragimov R.Sh., Bazarov R.K.* Agent-based modeling and simulation: application in telemedicine // ICEIC-2008. Proceedings of 9th International Conference on Electronics, Information and Communication. - Tashkent, June 24-27 2008. - PS2-17. 2008. - P. 31.
[29] Magda: A mobile Agent based Grid Infrastructure [Электронный ресурс] / University of Neaples. Italy. - Режим доступа: http://parsec.unina2.it/~magda.
[30] *Bellifemine F., Poggi A., Rimassa G.* JADE: A FIPA-compliant Agent Framework // Proceedings of PAAM'99, London, April 1999. - Pp. 97-108.
[31] Globus Toolkit 4.0.7 Download [Электронный ресурс] / University of Chicago. - Режим доступа: http://toolkit.globus.org/toolkit/downloads/4.0.7
[32] *Bellifemine F., Caire G., Greenwood D.* Developing Multi-Agent Systems with JADE. Willey, 2007. - 286 p.
[33] *Базаров Р.К.* Реализация муравьиного алгоритма для изучения фолдинга белков на плоскости методами программных агентов (Agent-ACOHPPFP-2) // Агентство по интеллектуальной собственности РУз. Свидетельство № DGU-04104. 09.12.2016 г.