электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

 # 11, ноябрь 2015

УДК 004.023

Применение генетических алгоритмов для динамической балансировки нагрузки между процессорами в распределенных вычислительных системах

Моисеев М.А., *студент*
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана,
кафедра «Системы обработки информации и управления»

Научный руководитель: Терехов В.И., к.т.н., доцент
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
кафедра «Системы обработки информации и управления»
chernen@bmstu.ru

Введение

При решении различных ресурсоемких задач появляется необходимость в быстрых вычислениях, для которых в настоящее время используют распределенные вычислительные системы (РВС) [1]. Последовательные вычисления в распределенных системах выполняются с учетом одновременного решения многих задач, при этом главным их достоинством является возможность горизонтального масштабирования — увеличения количества узлов, параллельно выполняющих одну и ту же функцию. Основной целью балансировки нагрузки является распределение задач между узлами (процессорами) в распределенной вычислительной системе для оптимизации использования ресурсов, сокращения времени обслуживания запросов, обеспечения отказоустойчивости. Балансировка нагрузки в распределенной системе является NP-полной задачей [10], в которой применение стандартных методов ветвей и границ, динамического или линейного программирования крайне затруднено. Анализ литературы [11], показывает, что генетические алгоритмы (ГА) часто применяются для решения задач подобного типа, однако при исследовании эффективности применения ГА в задаче динамической балансировки нагрузки между процессорами в РВС многим вопросам не уделялось должного внимания. Исследуем эффективность применения ГА в задаче динамической балансировки нагрузки между процессорами в распределенных вычислительных системах.

Балансировка нагрузки

Балансировка нагрузки применяется при необходимости оптимизации выполнения распределенных вычислений в РВС. Задачей балансировки нагрузки является организация равномерной нагрузки на вычислительные узлы распределенной вычислительной системы. Распределенное приложение — это совокупность логических процессов, взаимодействие между которыми происходит с помощью отправки сообщений. Логические процессы требуется равномерно распределять по вычислительным узлам (рис. 1), но возникает конфликт между сбалансированным распределением процессов по процессорам и скоростью обменов сообщениями между процессорами. Разрешение такого конфликта должно приводить к равномерной загрузке вычислительных узлов и коммуникационной среды.

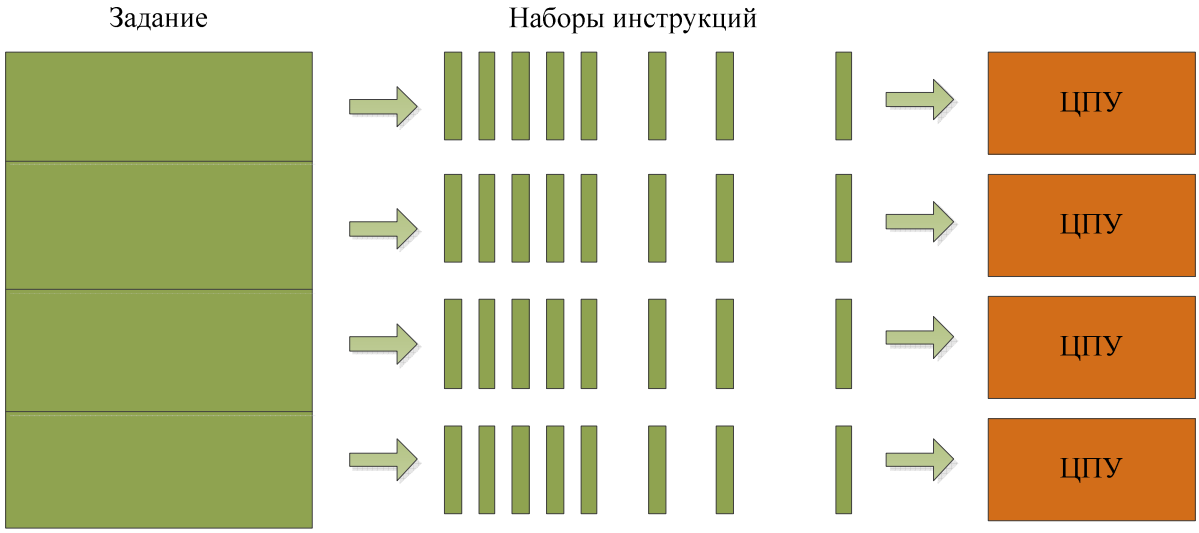

Рис. 1. Схема распределения задания по процессорам

Параллельные и распределенные вычислительные системы хорошо подходят для решения задач, которые можно разбить на более мелкие, чтобы выполнять их параллельно. Данные задачи разбиваются, а затем распределяются между несколькими узлами РВС для параллельных вычислений. Однако возникают ситуации, когда один узел полностью загружен и имеет большую очередь задач, а ресурсы другого узла практически не используются. В таком случае производительность РВС может быть улучшена путем перераспределения задач между узлами. Для реализации данной задачи используют алгоритм балансировки нагрузки.

Балансировка нагрузки может быть статической или динамической. Статическая балансировка применяется до начала работы РВС, но предварительное распределение

процессов дает плохой результат. Объясняется это тем, что состояние РВС постоянно меняется, узел может выйти из строя или попросту быть занят. Смысл динамической балансировки состоит в том, что она выполняется непосредственно во время работы РВС. При этом собирается информации о загрузке узлов, очереди задач и на основании собранных данных происходит перераспределение процессов между узлами. Рассмотрим приведенные способы балансировки нагрузки подробнее.

Статическая балансировка нагрузки

В статических алгоритмах используется информация, которую можно получить перед началом работы РВС. Используются такие метрики, как среднее время выполнения задания, стартовое время обработки задания, необходимые ресурсы. Но данная информация является неточной, так как во время работы РВС эти параметры могут изменяться. Наихудший результат будет получен в гетерогенной системе, где время выполнения задания сильно зависит от количества задач. Статическая балансировка нагрузки может быть оптимальной и неоптимальной. Основная проблема алгоритма статической балансировки в том, что он предполагает большое количество информации о системе, которую нельзя получить заранее. Схема статической балансировки нагрузки представлена на рис. 2.

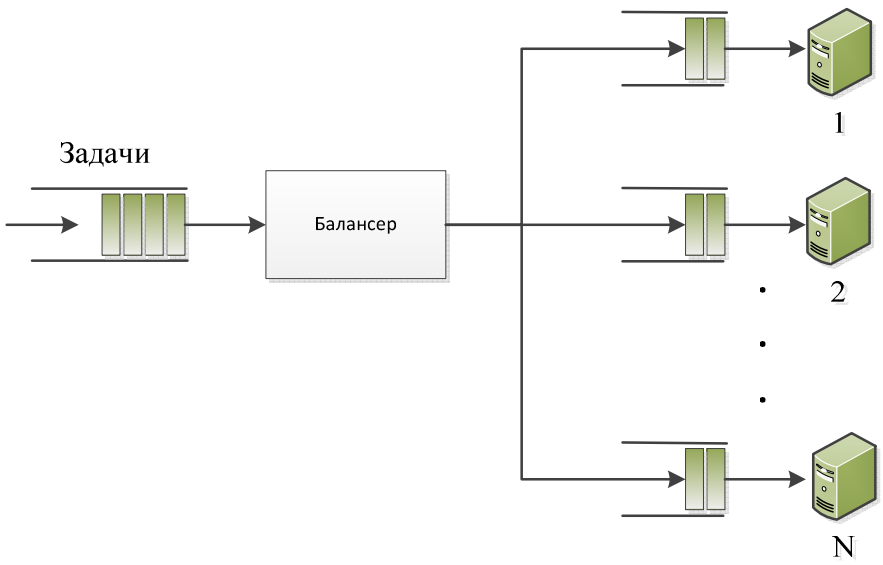

Рис. 2. Схема статической балансировки нагрузки

Динамическая балансировка нагрузки

Из-за низкой эффективности алгоритма статической балансировки, все больше внимания уделяется динамическому алгоритму балансировки, который при принятии

решения о распределении задач опирается на текущее состояние системы. При возрастании нагрузки на определенный узел РВС алгоритм перераспределяет задачи между другими менее загруженными узлами. Однако при мониторинге ресурсов узлов тратится процессорное время системы. Поэтому нужно определить, как часто будет проходить проверка состояния системы. Чем чаще будут сниматься метрики, тем больше лишней нагрузки будет выполнять РВС. Схема динамической балансировки нагрузки представлена на рис. 3.

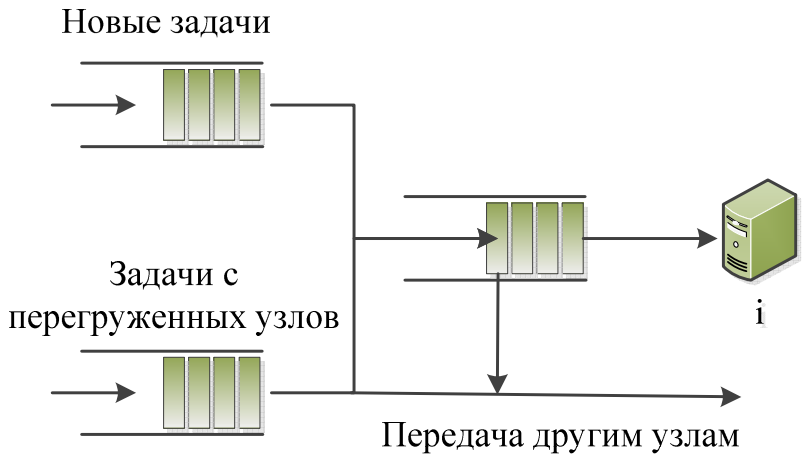

Рис. 3. Схема динамической балансировки нагрузки

Методы динамической балансировки нагрузки

В основе всех методов динамической балансировки нагрузки лежит сбор информации о текущем состоянии системы и принятие решение о том, с какого узла и на какой следует перенести задачи во время вычислительного процесса. Однако такой подход влечет за собой дополнительные расходы на сбор статистических данных о состоянии вычислительной среды, анализ данных и принятие решений. Поэтому необходимо найти баланс между временем поиска решения и его оптимальностью. Рассмотрим существующие методы динамической балансировки нагрузки.

Метод диффузии (diffusion method) является одним из наиболее популярных методов для решения задачи динамической балансировки нагрузки [5]. Работа метода напоминает тепловую диффузию. Начальная неоднородность температуры в среде устраняется через некоторое время благодаря диффузии. Критерием балансировки нагрузки является время счета процессора.

Для определения количества передаваемых процессоров и направления их передачи решается уравнение диффузии, которое сводится к линейному

дифференциальному уравнению, также называемому уравнением теплопроводности. В данном случае диффузией будет являться процесс передачи задач между процессорами.

Уравнение теплопроводности для времени счета:

$$\frac{\partial l}{\partial t} = c\nabla^2 l,$$

где l – нагрузка процессора, t - время, c – коэффициент диффузии, ∇^2 - оператор Лапласа[7].

Недостатком данного метода является достаточно медленная сходимость. Применение метода диффузии является оправданным при незначительном дисбалансе нагрузки, так как иначе потребуется огромное количество перемещений данных.

Метод многоуровневой диффузии (multi-level diffusion method) используется для ускорения сходимости метода диффузии [8]. Строится процессорный граф и к нему применяется алгоритм деления пополам. Далее происходит устранение несбалансированности нагрузки между двумя подграфами. Процесс повторяется до тех пор, пока возможно деление подграфов пополам. Метод гарантирует сходимость за $\log N$ операций деления пополам, где N – число процессоров. Недостатком является сложность разделения связного графа на два связных подграфа в случае слабой связанности.

Метод потенциала (method of potential) среди всех найденных решений выбирает то, при котором число перемещений данных между процессорами будет наименьшим [9]. Для нахождения оптимальной нагрузки процессоров решается система линейных уравнений (СЛУ). СЛУ можно решить с помощью метода сопряженных градиентов. Метод потенциала в несколько раза быстрее метода диффузии, но по-прежнему не является оптимальным по времени для решения задачи балансировки нагрузки.

Применение генетического алгоритма

Для поиска вычислительного узла, способного принять новые задачи, используют однонаправленные или широковещательные сообщения [6]. Оба метода имеют существенные недостатки. Метод, использующий однонаправленные сообщения, создает много повторных запросов, если все процессоры сильно загружены. Широковещательные сообщения требуют огромных затрат на коммуникацию при большом количестве вычислительных узлов в системе. Поэтому автор статьи предлагает использовать многоадресный подход, при котором сообщения отправляются ограниченному числу процессоров одновременно (рис. 4). Данный подход позволяет с высокой вероятностью найти нужный вычислительный узел с небольшими затратами на коммуникацию.

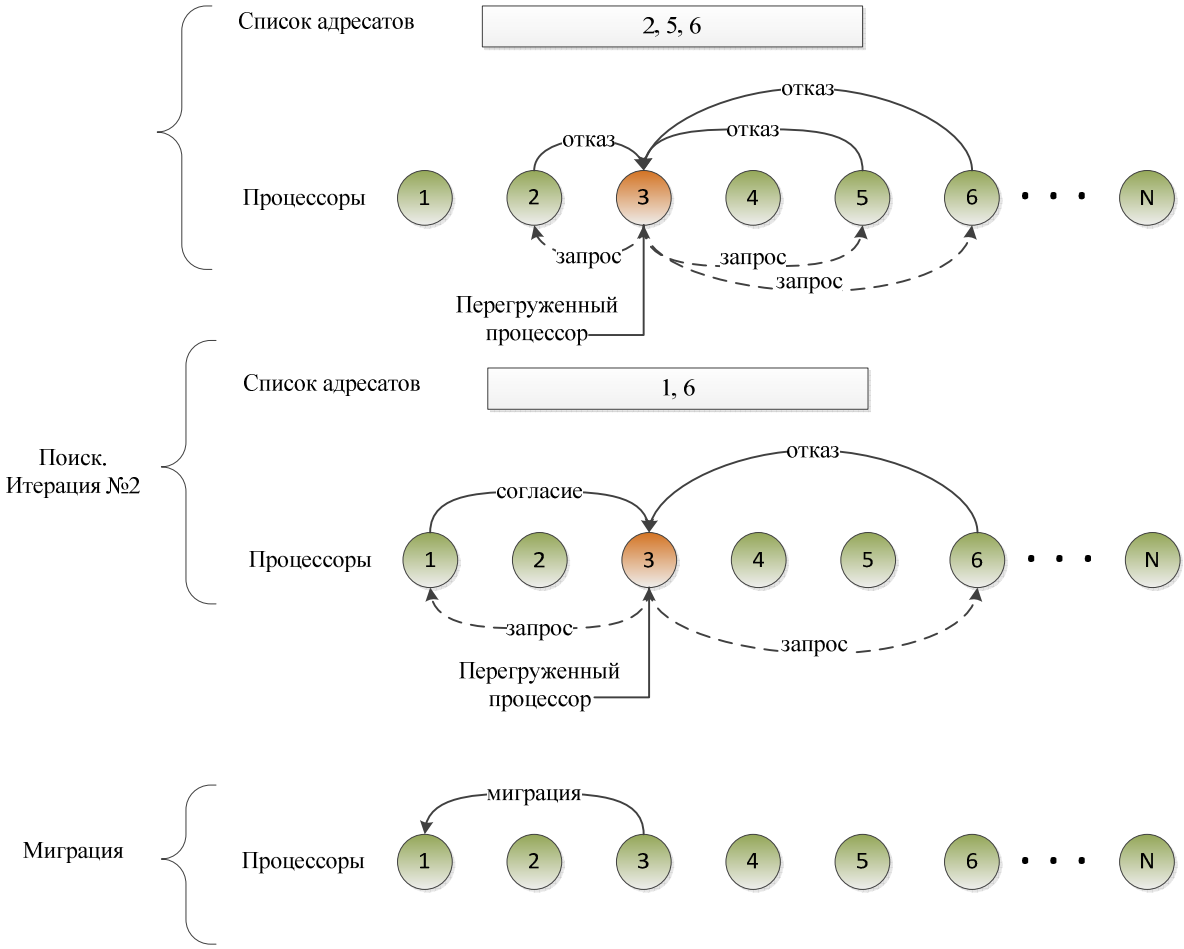

Рис. 4. Схема поиска процессора для миграции задач

При многоадресной отправке сообщения, сначала формируется список адресатов, состоящий из процессоров, которым будут отправлены запросы на передачу задач. Перегруженный процессор отправляет сообщения всем процессорам из списка адресатов. Если процессор, которому пришло сообщение с запросом на миграцию задач, не перегружен, то он отвечает согласием, иначе — отказом. Если все выбранные процессоры ответили отказом, то формируется новый список адресатов и алгоритм повторяется, пока не придет сообщение с согласием на миграцию. Когда перегруженный процессор получает сообщение с согласием на миграцию, задача с перегруженного процессора переносится на процессор, который отправил сообщение.

Для формирования списка адресатов автор статьи предлагает использовать генетические алгоритмы. Генетический алгоритм (ГА) является алгоритмом, основанным на принципах эволюции и естественного отбора. Сначала генерируется случайная последовательность решений. Данная последовательность называется популяцией. Каждый элемент популяции является решением задачи и называется хромосомой. Хромосома может быть представлена в виде строки, состоящей из генов. Геном будет

являться идентификатор процессора. Причем часто используется бинарное представление генов и хромосом в целом. На протяжении большого количества итераций происходит изменение (эволюция) хромосом в популяции. На каждом шаге итерации часть хромосом отсекается в результате селекции. Селекция представляет собой процесс, при котором каждая хромосома исследуется на приспособленность (fitness) с помощью специально построенной функции (fitness function). Наиболее приспособленные хромосомы с большей вероятностью смогут воспроизвести потомков, чем слабо приспособленные. Таким образом, хромосомы, содержащие неоптимальное решение задачи, будут постепенно исчезать. Новое потомство хромосом в популяции создается с помощью операции скрещивания. При скрещивании хромосомы группируются в пары и случайным образом обмениваются «генетическим материалом». На сегодняшний день существует множество алгоритмов скрещивания: частично отображаемое, упорядоченное, циклическое. Их выбор во много зависит от решаемой задачи. При скрещивании образуются новые хромосомы, ранее не встречавшиеся в популяции. В то же время их приспособленность может быть ниже, чем у родителей. Последним этапом эволюции является мутация, которая обеспечивает разнообразие «генетического материала» хромосомы. Другими словами, мутация увеличивает область поиска наилучшего решения. Схема работы ГА представлена на рис. 5.

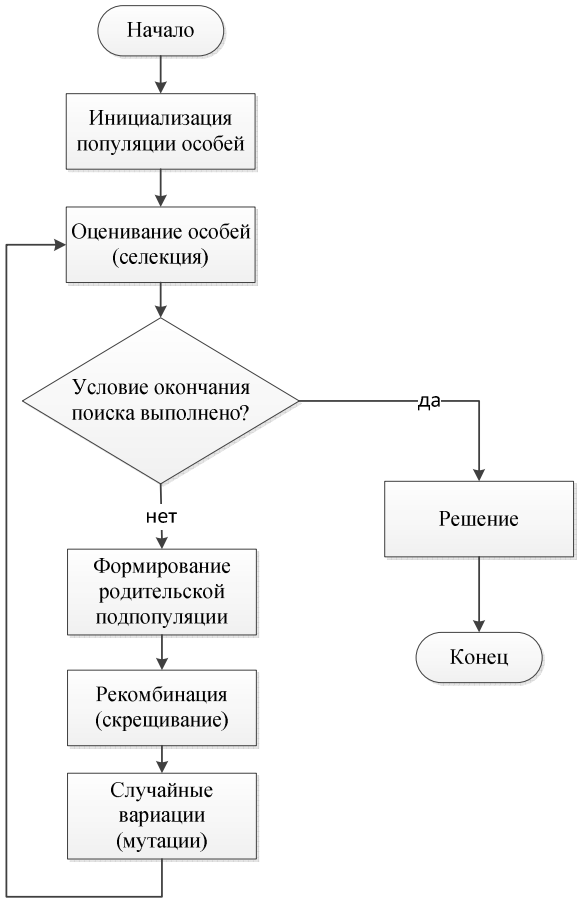

Рис. 5. Схема работы генетического алгоритма

При решении задачи динамической балансировки нагрузки хромосомой будет выступать список процессоров, которым будет отправлен запрос на миграцию, а фитнесс функция может вычисляться в следующем виде:

$$\Phi = \frac{1}{\alpha * \mathrm{T}_{\text{коммуникации}} + \beta * \mathrm{T}_{\text{проц.}}}, \qquad (1)$$

где α и β - весовые коэффициенты для параметров $\mathrm{T}_{\text{коммуникации}}$ (суммарное время передачи всех сообщений) и $T_{\text{проц.}}$ (суммарное время работы процессора).

Представленная фитнесс функция (1) позволяет отбирать процессоры, которые менее загружены и для которых затраты на коммуникацию будут небольшими, что будет способствовать равномерной загрузке процессоров и коммуникационной среды. Значения весовых коэффициентов вычисляются опытным путем, так как зависят от количества процессоров и их производительности.

Суммарное время передачи всех сообщений – $\mathrm{T}_{\text{коммуникации}}$ можно рассчитать по следующей формуле:

$$\mathrm{T}_{\text{коммуникации}} = \sum_{i=1}^{n} \mathrm{T}_i \ ,$$

где T_i – время, затрачиваемое на коммуникацию между процессорами

Ключевым параметром для динамической балансировки нагрузки в РВС является суммарное время работы процессора – $T_{\text{проц.}}$ [4]. Данный параметр можно представить в виде следующей формулы:

$$\mathrm{T}_{\text{проц.}} = \sum_{i=1}^{n} \text{ДОП}_i \ ,$$

где $ДОП_i$ – длина очереди процессора (Processor Queue Length). Данный параметр содержит количество запросов, которые стоят в очереди на обработку центральным процессором.

Согласно формуле фитнесс функции (1), чем больше значения параметров $T_{коммуникации}$ и $T_{проц.}$, тем меньше будет значение фитнесс функции. Отсюда следует вывод, что у наиболее приспособленных хромосом выведенные параметры будут меньше, чем у остальных. Таким образом, результатом работы генетического алгоритма будет оптимальный список процессоров, которым мы можем отправить запросы на миграцию задач. В результате получается довольно эффективное по балансировке и экономное по времени решение задачи балансировки нагрузки между процессорами в распределенных вычислительных системах.

Заключение

Были рассмотрены основные методы динамической балансировки нагрузки и сделаны выводы об их не оптимальности из-за быстрой сходимости или длительного времени работы. Было предложено и описано использование генетического алгоритма для динамической балансировки нагрузки, который показывает свою эффективность при решении подобных задач по сравнению с другими методами в различных экспериментах [10], так как позволяет найти оптимальное решение за более короткое время.

Список литературы

1. Радченко Г.И. Распределенные вычислительные системы: учебное пособие. Челябинск: Фотохудожник, 2012. 184 с.
2. Воеводин В.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
3. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы: учебное пособие. 2-е изд. М.: Физматлит, 2006. 320 с.

4. Lee S., Hwang C. A dynamic load balancing approach using genetic algorithm in distributed systems // Proceedings of IEEE International Conference on Evolutionary Computation. 1998. P. 639-644.
5. Cybenko G. Dynamic load balancing for distributed memory multiprocessors // J. Parallel and distributed computing. 1989. Vol. 7. P. 279–301. DOI: 10.1016/0743-7315(89)90021-X.
6. Munetomo M., Takai Y., Sato Y. A stochastic genetic algorithm for dynamic load balancing in distributed systems // Proceedings of IEEE International Conference on Systems, Man and Cybernetics. 1995. Vol. 4. P. DOI: 3795-3799. 10.1109/icsmc.1995.538379.
7. Hendrickson B., Devine K. Dynamic load balancing in computational mechanics // Computer Methods in Applied Mechanics and Engineering. 2000. Vol. 184. P. 485–500. DOI: 10.1016/S0045-7825(99)00241-8.
8. Horton G. A multi-level diffusion method for dynamic load balancing // Parallel Computing. 1993. Vol. 19. P. 209–229. DOI: 10.1016/0167-8191(93)90050-U.
9. Hu Y.F., Blake R.J., Emerson D.R. An optimal dynamic load balancing algorithm // Concurrency - Practice and Experience. 1998. Vol. 10. P. 467–483. DOI: 10.1002/(SICI)1096-9128(199805)10:6<467::AID-CPE325>3.0.CO;2-A.
10. Sandip K. G., Manpreet S. Enhanced Genetic Algorithm Based Load Balancing in Grid // IJCSI. 2012. Vol. 9, issue 3. No 2.
11. Goldberg D. E. Genetic algorithms in search, optimization, and machine learning // Reading, MA: Addison-Wesley. 1989. P. 6-56.