Сравнение подходов линейного программирования и генетического алгоритма к решению задачи оптимизации аппаратных ресурсов сервера для размещения виртуальных машин в ходе внедрения инфраструктуры виртуальных рабочих столов

К.А. Маковий[1], **Д.К. Проскурин**[1], **Ю.В. Хицкова**[1,2], **Я.В. Метелкин**[1]

[1]Воронежский Государственный Технический Университет, Московский проспект, 14, Воронеж, Россия, 394000
[2]Воронежский государственный университет, Университетская площадь, 1, Воронеж, Россия, 394018

Аннотация. В работе представлен компаративный анализ применения генетического алгоритма и метода ветвей и границ при решении задачи оптимизации. Сравнение производится на примере задачи оптимизации серверных ресурсов при размещении виртуальных машин при внедрении технологии виртуализации рабочих столов.

1. Введение

Облачные вычисления - это парадигма предоставления таких услуг, как программное обеспечение, платформа, инфраструктура, через Интернет [1]. Рабочий стол как услуга (Data as a Service - DaaS) - одна из новых технологий, предоставляемых облачными вычислениями, и основанная на технологии виртуализации настольных систем. Виртуализация - это изоляция вычислительных процессов и вычислительных ресурсов друг от друга. Данная технология позволяет организациям создавать прозрачные, масштабируемые и гибкие модели вычислений и приложений, что повышает эффективность использования вычислительных ресурсов. Технология виртуализации оказывает большое влияние на мир информационных технологий и становится одной из наиболее важных тем исследований и технологий в области ИТ.

Использование технологии виртуализации соответствует набирающей популярность концепции Green Computing. Эта концепция включает в себя вычислительные процедуры, политики и функции, связанные с использованием ИТ. Целью Green Computing является использование компьютеров и ИТ таким образом, который бы позволял экономить энергию, деньги и охранять окружающую среду [2].

Виртуализация рабочего стола − это технология предоставления пользователю доступа к удаленному рабочему столу, размещенному в облачном центре обработки данных [3]. Пользователи могут получать доступ к своим настольным приложениям и работать с ними, используя любые устройства (концепция BYOD). Технология виртуализации предоставляет такие преимущества как простота управления, мобильные вычисления, гибкость, безопасность

данных и снижение общих затрат для владельцев организации, например, тонкие клиентские устройства, предназначенные для доступа к виртуальному рабочему месту, используют от 1 до 5 Вт [2]. Кроме того, эти устройства не выделяют много тепла и, как правило, работают без вентилятора.

Пользователи VDI могут получить доступ к виртуальным машинам с помощью протоколов удаленного рабочего стола, например, протокола Microsoft Remote Desktop (MS RDP), протокола приложений Sun Application Link (ALP), протокола Citrix High Definition Experience Protocol (HDX), интернет-протокол (PCoIP) или протокола блокированной асинхронной передачи (BLAST) из VMware.

Особенностью внедрения технологии виртуализации являются довольно большие затраты на первом этапе [4], поэтому переход на данную инфраструктуру следует начинать с предварительной оценки потребности в серверных ресурсах [5].

В настоящий время в литературе существует два принципиально различных подхода к решению задачи оптимизации размещения виртуальных серверов на аппаратных платформах – статический и динамический. Статический подход заключается, как правило, в сведении к задаче многомерной упаковки в контейнеры или задаче о рюкзаке. Данный подход используется в случаях, когда потребность виртуальной машины в аппаратных ресурсах сервера, их количество заранее известны и постоянны в каждый момент времени.

Динамический подход к решению задачи оптимизации серверных ресурсов при внедрении технологии виртуализации основывается на поддержке наиболее популярными гипервизорами, например, VMWare ESX, Xen, Linux KVM, Hyper-V технологии динамической миграции, позволяющей перемещать виртуальные машины «на ходу» без прерывания их работы.

В данных моделях оптимизации авторы в качестве входных данных рассматривают набор одинаковых аппаратных серверов, необходимых для размещения виртуальных машин с различными требованиями к ресурсам. В данной работе предлагается модель для минимизации затрат на размещение некоторого количества виртуальных машин, таким образом, возникает необходимость рассмотрения аппаратных серверов с различными характеристиками, что позволит повысить коэффициент утилизации используемых серверов. Решаемая задача относится к классу недетерминированной полиномиальной сложности (NP) [6], что доказывается сведением её к задаче об упаковке в контейнеры. Данный класс задач практически неразрешим и не существует алгоритмов, способных решить их за разумное время.

2. Постановка задачи

В процессе внедрения технологии виртуальных рабочих столов возникает необходимость приобретения аппаратных серверов, на которых будут размещаться пользовательские виртуальные машины. Поскольку переход на данную технологию экономически обоснован в случае виртуализации большого числа рабочих мест [4], задача минимизации затрат на этапе закупки аппаратного обеспечения имеет большой экономический эффект.

Предполагается, что оборудование выбирается исходя из необходимого конечным пользователям числа виртуальных рабочих мест, с учетом требований виртуальных машин к ресурсам, которые можно определить следующими способами:

– рекомендации вендоров, которые включает в себя требования к ресурсам в зависимости от выполняемых приложений, изучением нагрузки на ИТ-инфраструктуру;

– использование утилиты VMware View Planner, предназначенной для имитации различных сценариев рабочей нагрузки в среде VMWare Horizon 7;

– использование консольной утилиты esxtop, которая позволяет отслеживать все аспекты производительности сервера виртуализации.

Переход на виртуальную инфраструктуру удобно начинать с пилотного проекта [7] для оценки целесообразности, требуемого времени на реализацию, затрат, наличия или отсутствия каких-либо побочных явлений и оценки размера эффекта.

После оценки требований необходимо выбрать для закупки набор серверов таким образом, чтобы обеспечить выполнение заданного количества виртуальных машин и минимизировать стоимость приобретаемого оборудования.

Сформулированная задача является задачей нелинейного целочисленного программирования. В общем случае такого рода задачи решаются перебором всех возможных значений, и существующие методы отыскания оптимального плана сводятся к тому, чтобы этот перебор сократить. В работах [5,8] используется эвристический подход, а именно, применена декомпозиция, результатом которой является две подзадачи уже линейного целочисленного программирования, решаемые методом ветвей и границ.

Другим способом решения задачи является использование генетического алгоритма для поиска оптимального решения.

3. Математическая модель

Мы ставим задачу обеспечения работы известного количества виртуальных машин с определенными требованиями к аппаратным ресурсам на некотором наборе серверных аппаратных платформ, оперативную память которых, предустановленную производителем, можно расширять, докупая дополнительные модули памяти. Объем докупаемой памяти ограничивается максимальным объемом памяти, поддерживаемой данной платформой, а также количеством свободных слотов для модулей памяти, размещенных на материнской плате этой платформы. Для увеличения объема оперативной памяти можно использовать дополнительные модули памяти разного объема и, соответственно, стоимости.

Предположим, что производительность виртуальной машины приемлема, если объем памяти сервера достаточен для работы виртуальной машины только в оперативной памяти без использования страничного файла подкачки..

Для предложенной модели введем следующие переменные:

S –множество моделей серверных платформ $s_i \in S$, $i=1..m$, которые могут использоваться как аппаратные серверы;

C – множество стоимостей моделей серверных платформ, где $c_i \in C$, $i=1..m$ – стоимость платформы s_i;

N – число серверов модели s_i которое будет использоваться в конечном наборе $n_i \in N$, $i = 1..m$;

M– максимальный объем ОЗУ, который может быть добавлен к модели s_i, $m_i \in M$, $i = 1..m$;

R – объем памяти j-го модуля, $r_j \in R$, $j=1..k$, где k – это количество типов модулей памяти;

Cv– стоимость j-го модуля памяти, $c_{v_j} \in Cv$, $j=1..k$;

P– число слотов памяти сервера $s_i, p_i \in P$, $i = 1..m$.

Для определения конфигурации, оптимальной с точки зрения стоимости, мы должны найти целевую функцию, отражающую полную стоимость набора аппаратных серверов. Полная стоимость каждого сервера соответствующей модели складывается из стоимости базовой серверной платформы (c_i) и стоимости дополнительных модулей памяти ($\sum_{j=1}^{k} c_{v_j} a_{ji}$), где a_{ji} - число модулей памяти типа j на сервере i.

Каждая из платформ может встречаться в конечном наборе не один раз, и наряду с повторяющимися конфигурациями памяти, в оптимальном результирующем наборе могут быть одинаковые платформы с различными вариантами заполнения планками памяти.

Для определения количества вариантов заполнения планками памяти слотов серверных платформ представим данную задачу как комбинаторную задачу определения количества сочетаний с повторениями. Общей формулой сочетаний с повторениями из n по k является следующая формула:

$$\tilde{C}_n^k = \frac{(n+k-1)!}{k!(n-1)!} = C_{n+k-1}^k \quad (1)$$

где k – число выбираемых элементов, n – число типов элементов из которых осуществляется выбор.

Для нашей задачи в качестве выбираемых элементов рассматриваются слоты памяти на серверной платформе, а число типов элементов – это число вариантов устанавливаемых модулей памяти плюс вариант отсутствия модуля памяти в слоте. Согласно формуле (1) для серверной платформы c_i количество вариантов заполнения слотов планками памяти q_i будет равно:

$$q_i = \tilde{C}_{k+1}^{p_i} = \frac{(k+p_i)!}{p_i!k!} \quad (2)$$

Общее количество вариантов заполнения слотов планками памяти всех серверов является суммой числа вариантов для каждого сервера q_i:

$$q = \sum_{i=1}^{m} q_i \quad (3)$$

Таким образом, целевая функция будет выглядеть следующим образом:

$$F = \sum_{l=1}^{q} \sum_{i=1}^{m} (c_i + \sum_{j=1}^{k} c_{v_j} a_{ji}) n_{il} \quad (4)$$

Для решения оптимизационной задачи необходимо учесть следующие ограничения.

1. Общий объем добавляемой к серверной платформе памяти не должен превышать максимальный объем, поддерживаемый серверной платформой:

$$\sum_{j=1}^{k} r_j a_{ji} \leq m_i, \; i=1..n, \quad (5)$$

2. Общий объем модулей памяти не может превышать числа слотов памяти на сервере:

$$\sum_{j=1}^{k} a_{ji} \leq p_i, \; i=1..n \quad (6)$$

3. Общий объем памяти на всех серверах из конечного набора должен обеспечивать достаточный объем памяти для работы необходимого числа виртуальных машин:

$$\sum_{i=1}^{m} ([(\sum_{j=1}^{k} r_j a_{ji})/V]) \geq N_V \quad (7)$$

где N_V – это минимально необходимое количество виртуальных машин, V – объем памяти, необходимы для одной виртуальной машины.

4. Чтобы получить решение, которое имеет смысл, добавим ограничение, требующее, чтобы количество серверов и модулей памяти было целочисленным:

$$n_{il}, a_{ji} \geq 0, \; n_{il}, a_{ji} - \text{целые} \quad (8)$$

Задача минимизации целевой функции (4) с учетом ограничений (5-8) представляет собой оптимизационную задачу, которую предлагается решить с использованием генетического алгоритма.

4. Подход к решению задачи с помощью методов линейного программирования

На первом этапе нашего численного решения мы заполняем оптимальным образом слоты памяти аппаратных серверов планками памяти, чтобы достичь определенной степени заполнения памяти от максимально возможной. В качестве шага мы взяли шаг в 25%, то есть мы рассматриваем варианты с заполнением памяти на 25%, 50%, 75% и 100%. При этом для каждой аппаратной платформы мы минимизируем стоимость закупки необходимого объема памяти подбором необходимых планок памяти. Такую задачу можно сформулировать следующим образом:

$$C_{v_i}^{q} = min \sum_{j=1}^{k} c_{v_j} a_{ji} \quad (9)$$

с учетом следующих ограничений:

$$\begin{cases} \sum_{j=1}^{k} r_j \, a_{ji} = m_i q \\ \sum_{i=1}^{k} a_{ji} \leq p_i \end{cases}, \; i=1..m \quad (10)$$

где q – степень заполнения, которая может быть 0,25, 0,5, 0,75 и 1, что соответствует заполненности оперативной памяти сервера на, соответственно, 25%, 50%, 75% и 100%.

Совокупность полученных решений мы будем рассматривать как множество аппаратных платформ, то есть на базе одной платформы мы получаем четыре сервера во множестве аппаратных платформ, из которых осуществляется выбор. Соответствующую задачу можно сформулировать следующим образом:

$$min \sum_{i=1}^{m} \sum_{j=1}^{4} (c_i + c_{v_i}^{q_j}) \, a_{ij}, \quad (11)$$

где $c_{v_i}^{q_j}$ – это результат (9) с учетом ограничения (7).

Данная задача представляет собой задачу целочисленного линейного программирования. Численное решение было получено с помощью пакета Matlab.

5. Генетический алгоритм

Процесс решения оптимизационной задачи генетическим алгоритмом включает четыре подготовительных этапа:

1) разработка структуры кодирования решения, позволяющая производить оценку полученного решения, называемая схемой представления;
2) создание начальной популяции;
3) выбор оператора для генерации потомков;
4) определение правила отбора решений для участия в дальнейшей селекции.

Для кодировки хромосом используется список комплектаций серверов, элементами которого являются экземпляры класса Serv. Хромосома представляет собой набор генов, каждый из которых содержит число серверов (комплектаций), порядковый номер локуса гена соответствует порядковому номеру комплектации в списке комплектаций серверов. Для класса Serv определены следующие свойства:

- наименование базы;
- порядковый номер базы;
- стоимость базы;
- количество слотов для модулей RAM;
- количество модулей RAM 4Gb;
- количество модулей RAM 8Gb;
- количество модулей RAM 16Gb;
- количество модулей RAM 32Gb;
- общая стоимость;
- максимальный объем памяти, поддерживаемый данной платформой.

Первая популяция, состоящая из множества хромосом, генерируется случайным образом. После получения хромосом первой популяции они сортируются по возрастанию общей стоимости набора, и, затем, скрещиваются попарно (четная с нечетной). При этом позиция точки разбиения выбирается случайным образом. При скрещивании двух хромосом получаются две новые хромосомы, которые, с заданной вероятностью, подвергаются случайной мутации, или направленной мутации, и, после этого, становятся членами нового поколения популяции. Кроме того, в новое поколение добавляется хромосома из предыдущего поколения, имеющая наименьшую стоимость среди своего поколения хромосом, чтобы не потерять лучший результат, достигнутый на предыдущем этапе.

Под случайной мутацией мы подразумеваем добавление единицы в локус со случайно выбранной позицией. Под направленной мутацией подразумевается следующая процедура. Среди непустых локусов хромосомы определяется локус с минимальной стоимостью гигабайта памяти и локус с максимальной стоимостью гигабайта памяти. Локус с максимальной стоимостью гигабайта уменьшается на один, а локус с минимальной стоимостью гигабайта увеличивается на один до тех пор, пока общий объем памяти хромосомы не достигнет требуемого значения. Процесс генетического отбора ограничен заданным количеством поколений. Необходимое количество поколений определялось экспериментально путем нескольких вычислений для одинаковых параметров задачи, увеличение количества серверов сопровождалось необходимостью увеличения размера популяции и количества поколений.

Для реализации генетического алгоритма было разработано приложение на языке Java. В приложении предусмотрена возможность импорта исходных данных о серверных платформах и их характеристиках из файла Excel, а также задание размера требуемой оперативной памяти, размера популяции, количества поколений, % случайных и направленных мутаций.

6. Исходные данные

Обычно в организациях используется однотипное оборудование определенного поставщика, что упрощает обслуживание, сопровождение, поэтому разумно выбирать аппаратные платформы, предлагаемые одним производителем. В рамках данной работы мы рассматриваем серверные платформы для малого и среднего бизнеса линейки ML. Аппаратные характеристики и стоимость решений взяты с сайта одного из поставщиков серверов HP Proliant [9] (таблица 1).

Таблица 1. Характеристики аппаратных платформ.

№	Наименование	Объем ОЗУ, Гб	Кол-во модулей ОЗУ, шт.	Макси-мальный объем ОЗУ, Гб	Слотов ОЗУ, шт.	Цена, USD	Процессор
1	ML150 Gen9 NHP	4	1	512	16	1580	E5-2603v3 - 1.60
2	ML150 Gen9 Hot Plug	8	1	512	16	1700	E5-2609v3 - 1.90
3	ML150 Gen9 NHP	8	1	512	16	1960	E5-2609v3 - 1.90
4	ML350p Gen8	8	2	384	24	3300	E5-2620 - 2.00
5	ML350p Gen8	8	2	384	24	4300	E5-2630 – 2.30
6	ML350p Gen8	32	4	384	24	4440	E5-2620 – 2.00
7	ML350e Gen8 Hot plug	8	2	192	12	1874	E5-2420 – 1.90
8	ML350p Gen8 E5-2620 Hot Plug	16	2	384	24	3556	E5-2620 – 2.00
9	ML350p Gen8 E5-2620	8	2	384	24	3169	E5-2620 – 2.00
10	ML350e Gen8 Hot plug	2	1	96	12	1624	E5-2407 – 2.20
11	ML350p Gen8 HPM	16	2	384	24	7100	E5-2640v2 – 2.00
12	ML350e Gen8v2 Hot Plug	8	1	192	12	1660	E5-2407v2 – 2.40
13	ML350 Gen9	8	1	384	24	2510	E5-2609v3 – 1.90
14	ML350 Gen9	16	1	384	24	3590	E5-2620v3 – 2.40
15	ML350 HPM Gen9	32	2	768	24	6518	E5-2630v3 – 2.40

Параметры серверов, которые используются для выбора, представлены в таблице 1. Для расчета использовалось значение требуемой оперативной памяти для машины 4Гб. Для каждого из выбранных серверов доступны планки памяти для расширения ОЗУ объема 2Гб, 4Гб, 8Гб, 16Гб, 32Гб и стоимостью, соответственно, 26, 136, 215, 315, 840 USD.

7. Результаты

Далее приведены результаты, полученные с помощью генетического алгоритма и целочисленного линейного программирования. Для сравнения результатов были использованы три различных набора исходных данных:

1) необходимо разместить 1500 виртуальных машин, выбор осуществляется из первых 10 серверов таблицы 1;

2) необходимо разместить 1000 виртуальных машин, выбор осуществляется из всех 15 серверов таблицы 1;

3) необходимо разместить 700 виртуальных машин, выбор осуществляется из первых 5 серверов таблицы 1.

В таблицах 2,3 приведены результаты решения оптимизационной задачи размещения 1500 виртуальных машин, требующих по 4 Гб оперативной памяти каждая, при условии выбора первых 10 серверов из списка серверов, приведенного в таблице 1. В первом столбце указан порядковый номер серверной базы из таблицы 1.

В таблице 2 дополнительно указан процент заполнения слотов оперативной памяти планками памяти, соответствующий значению q в (9) и (10).

Таблица 2. 1500 ВМ, выбор из 10 серверов, линейное программирование.

№	Наименование	Запол-нение	Процессор	Кол-во	Кол-во планок 2 Гб	Кол-во планок 4 Гб	Кол-во планок 8 Гб	Кол-во планок 16 Гб	Кол-во планок 32 Гб	Стои-мость
1	ML150 Gen9 NHP	25%	E5-2603v3 - 1.60	1	6	0	0	7	0	3941
2	ML150 Gen9 Hot Plug	50%	E5-2609v3 - 1.90	23	0	0	1	13	1	6850
Общая стоимость					161491 USD					
Общий объем ОЗУ					6020 Гб					

Таблица 3. Для 1500 ВМ 10 серверов, генетический алгоритм.

№	Наименование	Процессор	Кол-во	Кол-во планок 2 Гб	Кол-во планок 4 Гб	Кол-во планок 8 Гб	Кол-во планок 16 Гб	Кол-во планок 32 Гб	Стои-мость
1	ML150 Gen9 NHP	E5-2603v3 - 1.60	17	0	0	0	10	5	8930
3	ML150 Gen9 NHP	E5-2609v3 - 1.90	1	0	0	0	13	1	6895
1	ML150 Gen9 NHP	E5-2603v3 - 1.60	1	0	2	1	10	2	6897
Общая стоимость				165602 USD					
Общий объем ОЗУ				6000 Гб					

Основные параметры расчета с помощью генетического алгоритма были следующие: размер популяции – 3000, 150 поколений, процент случайных и направленных мутаций – 15, результаты приведены в таблице 3.

В таблицах 4,5 приведены результаты решения оптимизационной задачи размещения 1000 виртуальных машин, требующих по 4 Гб оперативной памяти каждая, при условии выбора первых 15 серверов из списка серверов, приведенного в таблице 1. Основные параметры расчета с помощью генетического алгоритма были следующие: размер популяции – 4000, 300 поколений, процент случайных мутаций – 10% и направленных мутаций – 15%, результаты приведены в таблице 5.

В таблицах 6,7 приведены результаты решения оптимизационной задачи размещения 700 виртуальных машин, требующих по 4 Гб оперативной памяти каждая, при условии выбора первых 5 серверов из списка серверов, приведенного в таблице 1. Основные параметры расчета с помощью генетического алгоритма были следующие: размер популяции – 4000, 300 поколений, процент случайных мутаций – 10% и направленных мутаций – 12%, результаты приведены в таблице 7.

Таблица 4. Для 1000ВМ 15 серверов, линейное программирование.

№	Наименование	Запол-нение	Процессор	Кол-во	Кол-во планок 2 Гб	Кол-во планок 4 Гб	Кол-во планок 8 Гб	Кол-во планок 16 Гб	Кол-во планок 32 Гб	Стои-мость
2	ML150 Gen9 Hot Plug	50%	E5-2609v3 - 1.90	1	0	0	1	13	1	6850
13	ML350 Gen9	75%	E5-2609v3 – 1.90	1	4	0	0	17	0	7969
13	ML350 Gen9	100%	E5-2609v3 – 1.90	9	0	0	1	21	1	10180
	Общая стоимость					106439 USD				
	Общий объем ОЗУ					3870 Гб				

Таблица 5. Для 1000ВМ 15 серверов, генетический алгоритм.

№	Наименование	Процессор	Кол-во	Кол-во планок 2 Гб	Кол-во планок 4 Гб	Кол-во планок 8 Гб	Кол-во планок 16 Гб	Кол-во планок 32 Гб	Стои-мость
1	ML150 Gen9 NHP	E5-2603v3 - 1.60	1	0	0	0	6	6	8510
1	ML150 Gen9 NHP	E5-2603v3 - 1.60	5	0	0	0	6	8	10190
1	ML150 Gen9 NHP	E5-2603v3 - 1.60	7	0	0	0	10	2	6410
13	ML350 Gen9	E5-2609v3 – 1.90	1	0	1	3	17	1	9486
	Общая стоимость				113816 USD				
	Общий объем ОЗУ				4008 Гб				

Таблица 6. Для 700 машин 5 серверов, линейное программирование.

№	Наименование	Запол-нение	Процессор	Кол-во	Кол-во планок 2 Гб	Кол-во планок 4 Гб	Кол-во планок 8 Гб	Кол-во планок 16 Гб	Кол-во планок 32 Гб	Стои-мость
2	ML150 Gen9 Hot Plug	50%	E5-2609v3 - 1.90	11	0	0	1	13	1	6850
	Общая стоимость					75350 USD				
	Общий объем ОЗУ					2816 Гб				

Таблица 7. Для 700 машин 5 серверов, генетический алгоритм.

№	Наименование	Процессор	Кол-во	Кол-во планок 2 Гб	Кол-во планок 4 Гб	Кол-во планок 8 Гб	Кол-во планок 16 Гб	Кол-во планок 32 Гб	Стои-мость
1	ML150 Gen9 NHP	E5-2603v3 - 1.60	8	0	0	0	11	4	8405
1	ML150 Gen9 NHP	E5-2603v3 - 1.60	1	0	0	0	12	0	5360
2	ML150 Gen9 Hot Plug	E5-2609v3 - 1.90	1	0	1	0	8	0	4356
	Общая стоимость				76956 USD				
	Общий объем ОЗУ				2800 Гб				

Полученные результаты показывают принципиальную применимость обоих предложенных методов к решению задачи оптимизации серверных аппаратных ресурсов в процессе

развертывания инфраструктуры виртуальных рабочих столов. По всем протестированным комбинациям подход, основанный на методах линейного программирования, показывает лучшие результаты, хотя порядок значений целевой функции, полученной обоими методами, совпадает. Одной из проблем генетического алгоритма на функциях, имеющих помимо глобального минимума большое количество локальных минимумов, является тенденция скатывания в локальный минимум и «застревания» там. Для решения этой проблемы необходимо применение дополнительных алгоритмических решений.

8. Литература

[1] Ali, M. Cloud Computing Applications / M. Ali, M.H. Miraz // Proceedings of the International Conference on Cloud Computing and eGovernance. – 2013. – P. 1.

[2] Agrawal, S. Virtual Desktop Infrastructure in Higher Education Institution: Energy Efficiency as an Application of Green Computing / S. Agrawal, R. Biswas, A. Nath // Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on IEEE. – 2014. – P. 601-605.

[3] Kochut, A. Power and performance modeling of virtualized desktop systems / A. Kochut // Modeling, Analysis & Simulation of Computer and Telecommunication Systems, 2009. MASCOTS'09. IEEE International Symposium on IEEE. – 2009. – P. 1-10.

[4] Маковий, К.А. Экономическое обоснование внедрения технологии виртуализации рабочих столов (Virtual Desktop Infrastructure) в ИТ- инфраструктуру высшего учебного заведения / К.А. Маковий, Ю.В. Хицкова // Современная экономика: проблемы и решения. – 2015. – Т. 2, № 62. – С. 75-81.

[5] Проскурин, Д.К. Задача выбора серверных ресурсов для внедрения инфраструктуры виртуальных рабочих столов / Д.К. Проскурин, К.А. Маковий // Вестник Воронежского государственного технического университета. – 2017. – Т. 13, № 4. – С. 26-32.

[6] Speitkamp, B. A mathematical programming approach for server consolidation problems in virtualized data centers / B. Speitkamp, M. Bichler // IEEE Transactions on services computing. – 2010. – Т. 3, № 4. – С. 266-278.

[7] Маковий, К.А. Пилотный проект виртуализации рабочих мест в компьютерном классе Воронежского ГАСУ / К.А. Маковий, Н.В. Шипилов // Научный вестник Воронежского государственного архитектурно-строительного университета. Серия: Студент и наука. – 2016. – № 10. – С. 113-117.

[8] Makoviy, K.A. Server hardware resources optimization for virtual desktop infrastructure implementation / K.A. Makoviy, et al. // CEUR Workshop Proceedings. – 2017. – Vol. 1904. – P. 178-183.

[9] Серверы и комплектующие Hewlett-Packard. − Электрон. дан. − Режим доступа: http://www.proliant.ru/ files/File/HP_proliant_price_09_15.xls.

A comparison of linear programming and the genetic algorithm approaches to the problem of optimizing the server hardware resources for hosting virtual desktops

К.А. Makoviy[1], **D.К. Proskurin**[1], **Yu.V. Khitskova**[1,2], **Ya.V. Metelkin**[1]

[1]Voronezh State Technical University, Moskovsky prospect, 14, Voronezh, Russia, 394000
[2]Voronezh State University, Universitetskaya pl. 1, Voronezh, Russia, 394018

Abstract. The article presents a comparison of genetic algorithm and the linear programming method approaches for solving the optimization problem. The comparison is performed on the example of the task of server hardware recourses optimization when placing virtual machines during virtual desktop infrastructure implementation.

Keywords: Virtual Desktop Infrastructure, optimization, linear programming, genetic algorithm.