УДК 519.687.1

ТЕЛЕНИК С.Ф.,
РОЛИК А.И.,
САВЧЕНКО П.С.

АДАПТИВНЫЙ ГЕНЕТИЧЕСКИЙ АЛГОРИТМ ДЛЯ РЕШЕНИЯ КЛАССА ЗАДАЧ РАСПЕРЕДЕЛЕНИЯ РЕСУРСОВ ЦОД

Предложен адаптивный вариант генетического алгоритма, основанный на циклическом применении двух генетических алгоритмов, решающих задачи параметрической и алгоритмической адаптации, что позволяет определить стратегию выбора параметров генетических алгоритмов и вероятность их применения. Приведена последовательность этапов работы алгоритма, предложены модифицированные операторы мутации и кроссовера, введены понятия наград и производительности, которые позволили регламентировать последовательности применения генетических операторов и корректировать вероятность их применения. Эффективность предлагаемого алгоритма продемонстрирована на примере решения задачи распределения виртуальных машин в центре обработки данных.

An adaptive version of the genetic algorithm is proposed. Algorithm is based on the simultaneous use of two genetic algorithms that solve the problem of parametric and algorithmic adaptation. This allows you quickly select and adjust the strategy selection parameters for genetic algorithms and the likelihood of their use. Sequential steps of the algorithm are given, introduced the modified parameters of mutation and crossover, introduced the concepts of rewards and performance parameters that are allowed to regulate the sequence of genetic operators, and to adjust the probability of their use.

Введение

Для эффективного использования дорогостоящих вычислительных ресурсов центров обработки данных (ЦОД), хранилищ данных и других информационных и телекоммуникационных ресурсов необходимо, чтобы соответствующие системы управления ИТ-инфраструктурой обеспечивали рациональное распределение виртуальных машин (ВМ), осуществляли управление нагрузкой и решали множество других подобных задач. В [1] показаны преимущества применения генетических алгоритмов (ГА) по сравнению с эвристическими методами для решения задач такого рода. В тоже время требует прояснения ряд фундаментальных проблем, обусловленных спецификой этих алгоритмов, поскольку ГА одновременно используют несколько типов генетических операторов: унарных, бинарных и множественных. Трудно подобрать стратегию генерации значений вероятностей использования отдельных операторов таким образом, чтобы их применение давало положительный результат на протяжении всего времени работы ГА. Кроме того, каждый из операторов имеет множество параметров, оказывающих влияние на результаты работы алгоритма, и найти оптимальные значения этих параметров достаточно сложная задача. Решение этих проблем ГА в общем виде не представляется возможным, поэтому необходимо разработать эффективную стратегию подбора параметров операторов и определения вероятностей применения этих операторов в процессе эволюции ГА. Решению такой задачи и посвящена данная статья.

Анализ исследований и публикаций

В работах [2, 3] показано, что применение ГА для решения задач распределения вычислительных ресурсов позволяет получать достаточно производительные решения. В этих исследованиях сформулирован список проблем, которые необходимо выяснить при использовании ГА для решения задач различной размерности и ограничений.

В [4, 11] предложена идея адаптации генетических операторов и подстройки вероятностей их использования. Недостатком этих работ является адаптация только одного оператора — кроссовера и использование возможных подходов к адаптации независимо друг от друга.

В [1] предложен управляемый генетический алгоритм (УГА), позволяющий корректировать параметры работы алгоритма на всех этапах

решения задачи. Помимо этого УГА решает проблемы классического ГА: вырождение популяции, попадание в локальные экстремумы и т. д. Главными недостатками УГА являются необходимость участия администратора и привязка к узкому классу задач.

Для решения этих проблем предлагается использовать разработанный в данной работе адаптивный генетический алгоритм (АГА). АГА является обобщенным ГА, способным динамически изменять вероятности использования и значения параметров каждого из операторов.

Целью работы является разработка разновидности генетического алгоритма, способного самостоятельно изменять вероятности использования операторов ГА и значения параметров этих операторов в зависимости от результатов, получаемых в ходе решения задачи, и характера решаемых задач, а также проверка эффективности работы предлагаемого АГА на примере задачи распределения ресурсов ЦОД.

Основные положения

В задачах распределения ресурсов часто необходимо использовать целевые функции, являющиеся нелинейными и не унимодальными, ограничения – нелинейные и невыпуклые, при этом переменные могут быть булевыми, целыми, непрерывными либо смешанными. Во многих случаях реализация традиционных стратегий требует огромных объемов вычислений, а «время жизни» полученного результата зачастую невелико, поскольку ситуация безостановочно изменяется и постоянно требует новых решений. В таких случаях особый интерес представляет применение генетических алгоритмов, имеющих ряд привлекательных свойств, обусловленных их природой, которые могут дать хорошие результаты при решении задач с такими свойствами.

Идея, реализуемая в этой работе, довольно проста: можно ли придать генетическим алгоритмам свойства классических алгоритмов поиска, которые были популярными при решении инженерных задач в 70-80-х годах прошлого столетия? Генетический алгоритм, «стреляя» по области решений и отбирая лучшие популяции, постоянно улучшает решение. То же самое делают и алгоритмы поиска. Однако последние используют некоторую дополнительную информацию для увеличения скорости сходимости. Нельзя ли наделить алгоритмы поиска свойствами генетических алгоритмов? В [1] был предложен УГА, использование которого позволяет получать хорошие результаты. Скорость сходимости увеличивается за счет подбора вероятностей основных операций ГА на основе результатов, полученных на предыдущих шагах работы. В УГА специальные правила, используя параметры давления отбора, скорость сходимости и коэффициенты прироста популяции позволяют изменять вероятности применения генетических операторов и тем самым увеличить скорость сходимости.

Следующим естественным шагом должна быть адаптация ГА путем отбора тех же вероятностей, но на основе некоторого критерия оценки популяции. Таким критерием может служить вероятность получения оптимального решения на основе популяции либо скорости сходимости.

Поиск эффективной стратегии выбора частоты применения генетических операторов и параметров этих операторов для решения задач, возникающих при управлении ИТ-инфраструктурой, является прямой задачей адаптации генетического алгоритма.

Выделяют три типа адаптации: структурную, параметрическую и адаптацию алгоритма [13]. Структурная адаптация предполагает изменение структуры объекта. Для класса задач, решаемых в данной работе, такой возможности не существует, поэтому будут использованы только два оставшихся типа адаптации. Параметрическая адаптация позволяет корректировать параметры работы в процессе поиска решения, в то время как адаптация алгоритма подразумевает внесение изменений в ход алгоритма.

Общая постановка задачи адаптации ГА сводится к минимизации некоторой функции F, которая служит критерием адаптации для ГА и зависит от таких параметров как: типы операторов, которые следует использовать в эволюционном процессе, частота применения этих операторов и значения параметров, с которыми применяются операторы. Определим функцию адаптации следующим образом

$$F(M, C, \alpha_M, \alpha_C, \beta_M, \beta_C),$$

где M, C – параметры, регламентирующие применение операторов мутации и кроссовера, и принимающие значения из множества {0, 1}, причем равенство параметра нулю означает, что данный оператор не участвует в эволюционном процессе, а единица означает, что опера-

тор принимает участие в процессе эволюции; α_M, α_C – частоты использования операторов, принимающие значения из интервала [0;1]; β_M, β_C – множества возможных значений операторов мутации и кроссовера.

Нахождение явного вида функции F даже для простых случаев требует огромного количества вычислений, что сведет на нет все преимущества ГА. Использование структурной адаптации в рамках решения задач управления ИТ-инфраструктурой невозможно, поэтому при осуществлении адаптации предлагается использовать некоторую информацию касательно свойств функции F . Такой информацией будут типы операторов и параметры этих операторов, с которыми они принимают участие в эволюционном процессе.

В данной работе решение задачи создания АГА осуществляется последовательным циклическим использование двух ГА. Первый применяется с целью параметрической адаптации и используется для настройки параметров генетических операторов, увеличивающих скорость сходимости. Второй служит для алгоритмической адаптации и осуществляет подстройку ГА в зависимости от результатов, получаемых в ходе решения задачи. Таким образом, в работе используются два механизма адаптации операторов АГА, один из которых обеспечивает динамическое изменение значений вероятностей применения операторов, а второй подстраивает значения его параметров. В первом случае операторы, которые в процессе работы позволили улучшить решение, чаще применяются при генерации следующих поколений. Во втором — с момента запуска АГА контролируются и изменяются значения параметров операторов.

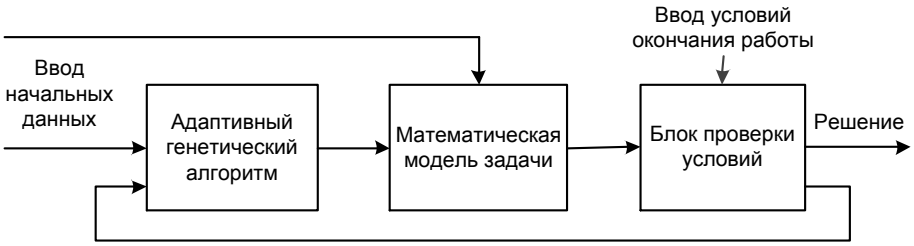


Рис. 1. Схема работы адаптивного генетического алгоритма

Схема работы предлагаемого АГА представлена на рис. 1. Математической моделью задачи являются задачи линейного программирования. К этому классу задач относится большинство задач, возникающих при управлении ИТ-инфраструктурой, например, задача распределения ресурсов, задача эффективного размещения файлов на серверах хранилища данных и т. п. Блок проверки условий служит для задания критериев остановки работы АГА. Это могут быть ограничения по времени работы либо по количеству эпох эволюционирования. Блок АГА – программная реализация предлагаемого алгоритма.

Как правило, для адаптации ГА [8] под решаемые задачи используется некоторый параметр управления [9]. Эффективность детерминистического управления была доказана для некоторых задач [4], однако его обобщение является проблематичным. Адаптивное управление [5] использует обратную связь, чтобы определить, как должны измениться параметры. Под адаптивным управлением [11] понимается введение дополнительной информации, позволяющей корректировать поведение операторов.

На сегодняшний день основные усилия исследователей ГА направлены на одновременную адаптацию только одного параметра. Наиболее полное сочетание форм управления [9] было представлено в работе [7], адаптация производилась одновременно по трем параметрам: вероятности мутации, кроссовера и размера популяции. Остальные формы адаптивных алгоритмов представлены в работах [6, 10, 12].

В данной работе предлагается обобщенная схема ГА, способного по ходу своей работы решать задачи параметрической и алгоритмической адаптации – динамически адаптировать как вероятность использования, так и значения параметров каждого из операторов. Вероятности применения операторов адаптированы детерминистическим способом. В этом случае все операторы применяются поочередно, а оператор, который в процессе работы позволил улучшить решение, автоматически будет использован в следующих эпохах. Таким образом, операторы, способствующие сходимости задачи, при генерации следующих поколений применяются чаще.

Приведем базовые определения и введем некоторые обобщения для классических операторов.

Кодирование данных. Каждая хромосома представляет собой последовательность бит. Все хромосомы $\text{chrom} = (b_1, ..., b_i, ..., b_n)$, $n = 2^k$, $k > 0$, $b_i \in \{0,1\}$, для всех $i = \overline{1, n}$, имеют длину 2^k, где k константа.

Мутация. Стандартный оператор мутации изменяет гены, путем инвертирования. Этот тип оператора имеет параметр *alter_rate*, который определяет количество генов в хромосоме, подлежащих инвертированию.

Предлагается традиционный алгоритм мутации усовершенствовать таким образом, чтобы вместо решения задачи для каждого гена о необходимости его инвертирования, производить инвертирование значения гена, равного 1, в 0 с вероятностью p_{10}, а 0 замещать 1 с вероятностью p_{01}. Такой подход позволяет применять ГА для подбора наилучших значений параметров генетических операторов. Например, если в родительской хромосоме некоторые значения вероятностей p_{10} и p_{01} не дали производительного решения, то в хромосоме потомка эти параметры обмениваются значениями, что может привести к улучшению результата.

Кроссовер. В работе предлагается использовать модифицированный оператор кроссовера, который генерирует c, $1 \le c < n$, различных точек кроссовера, делящих хромосомы на c+1 участков. При этом одно потомство наследует четные участки хромосомы первого родителя и нечетные участки второго родителя, второе потомство получается противоположным образом.

Суть предлагаемого адаптивного генетического алгоритма

Основным отличием предлагаемого алгоритма является использование двух стратегий адаптации, в связи с чем процесс получения решения происходит путем циклического повторения двух этапов. На первом этапе с помощью параметрической адаптации выбираются наиболее производительные значения параметров для каждого из используемых типов операторов. На втором этапе оценивается параметр производительности и с учетом результатов алгоритмической адаптации выбирается наиболее производительный тип оператора. Если полученное в результате решение не удовлетворяет заданным критериям, процесс повторяется с первого этапа.

Генетические алгоритмы используют несколько типов операторов, такие как унарные (мутация), бинарные (кроссовер), множественные (многоточечный кроссовер). Каждый тип оператора имеет набор параметров, влияющих на его поведение, а работа ГА начинается с определенными значениями параметров.

Пусть задано множество типов операторов $T = \{t^1, ..., t^j, ..., t^J\}$, где $j = \overline{1, J}$ — количество типов операторов, применяемых в ГА. Например, модифицированный оператор кроссовера или модифицированный оператор мутации. Каждый тип оператора t^j, $j = \overline{1, J}$, имеет набор параметров $P(t^j) = (p_{j,1}, ..., p_{j,k}, ..., p_{j,K})$, $k = \overline{1, K}$. Например, оператор модифицированной мутации имеет два параметра, которым соответствуют вероятности p_{10} и p_{01}, а кроссовер только один параметр – количество точек кроссовера.

Для генерации новых поколений ГА использует генетические операторы с параметрами, соответствующими его типу, причем параметры принимают значения из некоторого множества или интервала. Так, для модифицированного оператора мутации значения его параметров будут выбираться из интервала (0, 1), тогда как параметр оператора кроссовера может принимать любое положительное целочисленное значение, меньшее $(n-1)$.

Для увеличения шансов на выживание и размножение операторов со значениями параметров, показавших лучшие результаты, применяется параметрическая адаптация. Рассмотрим АГА, который использует типы операторов из множества T. Для каждого оператора типа $t^j \in T$, $j = \overline{1, J}$ строится популяция значений его параметров. Например, если для АГА определены два оператора: модифицированный оператор мутации и модифицированный оператор кроссовера, то фиксированным набором экземпляров операторов могут быть – для мутации $(alter\_rate, p_{10}, p_{01}) \in \{(0,1; 0,1; 0,9), (0,7; 0,8; 0), (0,4; 0,1; 0,1)\}$ для кроссовера – $(c) \in \{(1), (2), (n-1)\}$.

Обозначим набор используемых операторов АГА как $O = \{op_1, ..., op_h, ..., op_H\}$, $h = \overline{1, H}$, где H – количество операторов АГА.

Поведение каждого оператора корректируется изменением значений его параметров с использованием для каждого оператора эволюционной процедуры, функционирующей в пространстве возможных значений оператора. Для каждого оператора с момента запуска АГА производятся изменения значений его параметров. Поскольку оптимизируется относительно небольшое пространство поиска, то для поиска набора значений операторов, приводящих к улучшению результатов работы АГА, для каждого отдельного оператора также используется ГА. ГА, осуществляющий поиск лучших значений операторов, обозначим ОГА.

Популяция для каждого оператора будет состоять из хромосом, которые представляют собой набор возможных значений параметров этого оператора. Для генерации новых популяций используется одноточечный кроссовер. ОГА, применяемый для поиска в пространстве значений отдельного типа оператора наилучших значений, запускается как процедура ГА, обозначаемого далее ГГА и работающего над непосредственным поиском решения задачи. Полученное в результате использования ОГА решение служит исходными данными для ГГА. При этом АГА рассматривается как совокупность циклов ОГА и ГГА.

На этапе работы ГГА решается задача алгоритмической адаптации – увеличение вероятностей использования операторов, которые дают лучшие решения. Для оценки работы операторов введем следующие понятия.

Событие – применение конкретного генетического оператора. *Абсолютное улучшение* – событие, когда значение целевой функции нового поколения больше, чем значение целевой функции предшествующих поколений. *Улучшение* – новое поколение имеет значение целевой функции лучшее, чем у родителей. *Стабилизация решения* – значение целевой функции нового поколения незначительно отличается от родительского на протяжении ряда эпох. *Вырождение* – новое поколение имеет худшее значение целевой функции, чем родители, и ни одно из поколений не лучше, чем родительское.

Введем параметр производительности, показывающий эффективность использования оператора в текущем поколении и используемый в качестве обратной связи для эволюционного процесса. Основываясь на значениях показателей производительности, АГА корректирует значения вероятностей использования операторов. Параметр производительности для оператора op_h, $h = \overline{1, H}$ определим как функцию четырех переменных $\pi_{op_h}(ae, e, pw, w)$, где ae – количество абсолютных улучшений, e – количество улучшений, pw – количество стабилизированных решений, w – количество вырождений. Общее количество событий на шаге работы АГА определяется как $N = ae + e + pw + w$.

Параметр производительности введен с целью определения, какой из операторов дает лучшее решение на данном шаге работы АГА. Частота использования операторов учитывается относительными частотами появления событий определенного типа. При $N > 0$ относительные частоты рассчитываются так: для абсолютного улучшения – $\varphi_{ae} = ae / N$; улучшения – $\varphi_e = e / N$; стабилизированного решения – $\varphi_{pw} = pw / N$; вырождения – $\varphi_w = w / N$.

В данной работе принят следующий порядок сравнения параметров производительности, который на примере двух операторов будет формулироваться следующим образом. Более производительным будет оператор, параметр производительности которого характеризируется большим числом абсолютных улучшений. Если количество абсолютных улучшений одинаково, сравнение производится по количеству улучшений. Если количество улучшений также одинаково, сравниваются количества плоских событий. Если последнее не позволяет выделить лучшего оператора, то более производительным будет тот оператор, у которого меньше количество вырождений. При равном количестве вырождений используется мутация.

Для определения порядка использования операторов при работе АГА введено понятие награда. Каждому оператору $op_h \in O$, $h = \overline{1, H}$ приписывается награда $\rho(op_h)$, которая увеличивается в результате накопления положительного опыта. В первую очередь используется оператор, который имеет наибольшую награду. Значение параметра $\rho(op_h)$, $h = \overline{1, H}$ постоянно обновляется, а опыт, приобретенный при последних испытаниях, рассматривается как более актуальный.

Пусть $\chi(op_h)$, $h = \overline{1, H}$ ранг оператора op_h, присваиваемый в зависимости от производительности таким образом, что наивысший ранг получает наиболее производительный тип оператора. Причем присвоение ранга производится после завершения работы ОГА. Награды обновляются в конце каждой эпохи АГА согласно формуле

$$\rho(op_h) = \delta\rho(op_h) + \beta + \gamma\chi(op_h), h = \overline{1, H}$$

где δ – коэффициент ослабления, являющийся константой из множества {0, 1}. Причем $\delta = 0$ для оператора, только вступившего в работу, а $\delta = 1$ означает, что весь предыдущий опыт оператора полностью учтен. Коэффициент усиления γ служит для награждения лучших операторов. Коэффициент β, обычно имеющий значение меньше γ, используется для гарантии того, что оператор будет обязательно использован на этапе АГА.

Ниже приведен пример предлагаемого АГА для двух операторов – кроссовера и мутации, а также ограничений на продолжительность выполнения в виде количества эпох работы.

Шаг 1. Задать условия остановки алгоритма, которыми могут быть количество эпох эволюционирования генетического алгоритма или время работы алгоритма. Шаг 2. Инициализировать начальную популяцию случайным образом с учетом ограничений (1)–(3) и следующего правила: каждый раз при обращении к популяции для решения задачи ее особи отсортировываются в порядке убывания значения целевой функции. Наилучший представитель популяции – хранимое решение задачи. Лучшая из популяций запоминается как хранимая популяция. На начальном этапе хранимой является первичная популяция.

Шаг 3. Случайным образом инициализировать популяцию значений операторов с учетом ограничений, накладываемых для каждого типа операторов. Например, для параметров мутации и кроссовера могут использоваться следующие значения $(alter\_rate, p_{10}, p_{01}) \in \{(0,1; 0,1; 0,1), (0,2; 0,2; 0,2), (0,3; 0,3; 0,3)\}$ и $(c) \in \{(1), (2), (3)\}$ соответственно.

Шаг 4. Использовать каждое из значений соответствующих типов операторов для генерации промежуточных популяций. С помощью целевой функции выбрать значения, позволившие достичь наибольших показателей производительности для каждого из типов операторов. Например, для мутации это может быть $(alter\_rate, p_{10}, p_{01}) = (0,1; 0,1; 0,1)$, для кроссовера – $(c) = (1)$. В случае улучшения полученных результатов изменить хранимую популяцию, используя результаты, полученные посредством самого производительного их двух операторов, и перезаписать хранимое решение задачи. При равенстве операторов по производительности выбрать оператор мутации. Если не удалось улучшить начальное решение, перейти к шагу 3 и использовать ОГА для корректировки значений параметров применением оператора кроссовера. Если не удалось улучшить параметры операторов путем кроссовера, применить мутацию во избежание возможного попадания в локальные экстремумы. Применение параметра мутации при корректировке значений параметров операторов производится по циклу с помощью следующих правил: для кроссовера – увеличить на единицу количество точек кроссовера, а если количество точек равно $(n-1)$, принять следующее количество точек кроссовера равным 1; для мутации – увеличить каждое из значений параметров мутации на 0,1, а если значение любого из параметров равно 0,9, принять следующее значение равным 0,1. Если решение не улучшается, закончить работу и считать полученное решение наиболее производительным.

Шаг 5. Присвоить ранги операторам, рассчитать награды, применить операторы M раз (общее количество этапов работы ГГА будет равно $2 \times M$ на первой эпохе и $3 \times M$ на всех последующих) в порядке уменьшения величины награды. Если применение оператора улучшило решение, то использовать решение в качестве нового хранимого решения задачи, после чего продолжить работу с улучшенной популяцией.

Шаг 6. По истечении заданного числа этапов ГГА обновить значение параметров производительности и выбрать наиболее производительного оператора. Например, если $(alter\_rate, p_{10}, p_{01}) = (0,1; 0,1; 0,1)$ оказался наиболее производительным, то необходимо сохранить его и использовать в следующих эпохах. Перезаписать хранимое решение задачи.

Шаг 7. Если условия остановки АГА не выполнены, то перейти на шаг 3 и начать новую эпоху. На этапе работы ОГА выбрать наиболее производительные значения для всех типов операторов. Например, если

$(alter\_rate, p_{10}, p_{01}) = (0,2; 0,2; 0,2)$ и $(c) = (1)$ для мутации и кроссовера соответственно, то на этапе ГГА работа над популяцией будет проводиться для трех операторов (оператор мутации $(alter\_rate, p_{10}, p_{01}) = (0,1; 0,1; 0,1)$ переходит из предыдущей эпохи). По окончании работы ГГА пересчитать и сравнить параметры производительности, выбрать лучшего из трех операторов и перейти к следующей эпохе (шаг 3).

Шаг 8. Если условия остановки АГА выполняются, то завершить работу, использовать текущее хранимое решение в качестве решения задачи, иначе перейти на шаг 3 и продолжать работу до выполнения условия остановки.

Пример применения предлагаемого АГА

Пример применения предлагаемого алгоритма рассмотрим на задаче распределения ВМ в ЦОД.

Математическую модель задачи распределения ВМ по серверам представим следующим образом.

ЦОД содержит упорядоченное множество серверов $N = \{N_1, \ldots, N_n\}$, где n – количество физических серверов; подлежит распределению упорядоченное множество виртуальных машин $K = \{K_1, \ldots, K_m\}$, где m – количество ВМ; каждый сервер N_i, $i = 1, \ldots, n$, характеризуется двумя параметрами, определяющими его вычислительную мощность: Ω_i – мощность процессора сервера N_i; Γ_i – емкость ОЗУ сервера N_i; каждая ВМ K_j, j=1, …, m, имеет потребности в вычислительных ресурсах: ω_j – в процессорном времени; γ_j – в ОЗУ; матрица R распределения ВМ по серверам, $R = \|r_{ji}\|$ размера $m \times n$, где

$$r_{ji} = \begin{cases} 1, \text{если ВМ } K_j \text{ располагается на сервере } N_i, \\ 0, \text{в противном случае.} \end{cases}$$

Матрица R является решением задачи и определяет распределение множества K ВМ на множестве N физических серверов.

Считаем, что все серверы множества N имеют идентичные технические характеристики и, следовательно, одинаковые вычислительные ресурсы, поэтому полагаем, что $\Omega_i = 1$ и $\Gamma_i = 1$ для всех $i = 1, \ldots, n$, т. е.

$$\{\Omega_i, \Gamma_i\}|_{Ni} = \{1, 1\}, \text{ для всех } i = 1, \ldots, n. \quad (1)$$

Таким способом делаем переход от измерения вычислительных ресурсов серверов в абсолютных единицах, когда память измеряется в ГБ, а частота процессора в ГГц, к относительным.

Тогда потребности ВМ определяются как части от ресурсов сервера, нормированные относительно максимально возможной величины, равной единице.

Считаем, что потребности каждой ВМ в ресурсах не превышают возможностей сервера

$$\omega_j \le 1 \text{ и } \gamma_j \le 1, \text{ для всех } j = 1, \ldots, m \quad (2)$$

При решении задачи распределения ВМ для всех серверов множества N должно выполняться следующие ресурсное ограничение

$$\sum_{j=1}^{m} r_{ji}\omega_j \le 1 \text{ и } \sum_{j=1}^{m} r_{ji}\gamma_j \le 1, \text{ для } i = 1, \ldots, n. \quad (3)$$

Введем вектор $\vec{y} = \langle y_i \rangle$, $i = 1, \ldots, n$, где

$$y_i = \begin{cases} 1, \text{если на сервере } N_i \text{ располагается} \\ \quad \text{хотя бы одна ВМ}, \\ 0, \text{в противном случае.} \end{cases}$$

Тогда критерием оптимальности при решении задачи размещения ВМ на физических серверах будет

$$min \sum_{i=1}^{n} y_i, \quad (4)$$

т. е. сервера должны заполняться так, чтобы было задействовано минимальное их количество.

При выполнении критерия (4) будут минимизированы суммарные затраты S на обслуживание и энергоснабжение парка серверов ЦОД.

Целевую функцию можно представить следующим образом

$$S = \sum_{i=1}^{n} s_i y_i \quad (5)$$

где s_i – затраты на обслуживание и энергоснабжение i-го сервера.

В случае, когда сервера в ЦОД имеют идентичные технические характеристики, выражение (5) примет вид

$$S = s\sum_{i=1}^{n} y_i \quad (6)$$

где s – стоимость поддержания и затраты на энергоснабжение для одного сервера.

С учетом вышеизложенного задачу распределения множества K ВМ можно сформулировать следующим образом: необходимо размещать ВМ на серверах ЦОД таким образом, чтобы выражение (5) или (6) достигало минимального значения.

Результаты экспериментальных исследований

В работе произведены экспериментальные исследования на примере решения задачи распределения ресурсов ЦОД с помощью трех алгоритмов: классического ГА, управляемого генетического алгоритма [1], предлагаемого адаптивного генетического алгоритма.

По аналогии с [1–3] в данной работе исследования проводятся для различных вариантов соотношения ВМ и ресурсов серверов, на которых размещаются ВМ. Для исследований рассматриваются три варианта соотношений ресурсов, наиболее приближенные к реальным ситуациям:

— случай непропорциональных потребностей в ресурсах, когда в относительных единицах запрашиваемая величина процессорного времени (ЦП) значительно превышает запрашиваемый объем ОЗУ, т. е. $\omega_j >> \gamma_j$, для всех $j = 1, \ldots, m$. Подобный тип задачи возникает при наличии большого количества приложений, требующих сложный вычислений;

— также случай непропорциональных потребностей в ресурсах, когда запрашиваемая величина ЦП значительно меньше запрашиваемого объема ОЗУ, т. е. $\omega_j << \gamma_j$, для всех $j = 1, \ldots, m$. Такая задача возникает, когда на серверах ЦОД размещаются ВМ с приложениями, требующими обработки больших объемов данных;

— наиболее распространенный на практике случай, когда количество запрашиваемых ЦП и ОЗУ для всех ВМ распределено случайным образом в диапазоне [0,05; 0,6].

Пределы, в которых изменяются запрашиваемые ресурсы ВМ для различных экспериментов, представлены в таблице 1.

Табл. 1. Характеристики ВМ

№ эксперимента	Ограничения, ОЗУ	Ограничения, ЦП
1	0,3—0,45	0,05—0,15
2	0,45—0,6	0,05—0,15
3	0,05—0,15	0,3—0,45
4	0,05—0,15	0,45—0,6
5	0,05—0,6	0,05—0,6

В [1] доказано, что при количестве ВМ меньше пятидесяти эвристические и генетические алгоритмы дают приблизительно одинаковые результаты, а с увеличением количества ВМ выигрыш ГА становится все более ощутимым.

Отображение результатов работы КГА, УГА и АГА приводится с помощью количества высвобожденных серверов. Предполагается, что изначально для развертывания каждой ВМ выделяется отдельный сервер. Далее с помощью исследуемых алгоритмов производится оптимизация размещения ВМ на серверах с оценкой максимального количества высвобожденных серверов для каждого из алгоритмов. На рис. 2 представлен график зависимости количества высвобожденных серверов от размерности задачи (количества ВМ) для случая, когда количество запрашиваемых ЦП и ОЗУ для всех ВМ распределено случайным образом в диапазоне [0,05; 0,6]. По оси абсцисс откладывается количество ВМ, по оси ординат – количество высвобожденных серверов.

Для сравнения результатов работы УГА и АГА предлагается ввести понятие дополнительно высвобожденных серверов N_B. Значение N_B определяется как разность между количеством серверов $N_{КГА}$, высвобожденных в результате работы классического ГА, и количеством серверов $N_{УГА}$ и $N_{АГА}$, высвобожденных в результате применения УГА и АГА соответственно. Таким образом, на графиках рис. 3 показан выигрыш УГА и АГА относительно ГА в виде зависимости количества дополнительно высвобожденных серверов N_B от размерности задачи для различных вариантов соотношений запрашиваемых ресурсов. По оси абсцисс откладывается количество ВМ, которые необходимо расположить на серверах ЦОД. По оси ординат – количество дополнительно высвобожденных серверов для каждого из алгоритмов.

Данные для экспериментов генерировались случайным образом с равномерным законом распределения. Результаты экспериментальных исследований представлены на рис. 3.

Анализ графиков на рис. 3 позволяет сделать следующие выводы:

— использование УГА и АГА эффективнее, чем использование классического ГА;

— предлагаемый АГА является однозначным лидером независимо от условий эксперимента;

— при наличии ВМ с большими требованиями к ОЗУ либо ЦП (графики рис. 3,*б*, и рис. 3,*г*) эффективность использования УГА и АГА выравнивается, однако преимущество остается за АГА.

— для случая разброса требований в широком диапазоне [0,05; 0,6] (график на рис. 3,*д*), использование АГА наиболее эффективно.

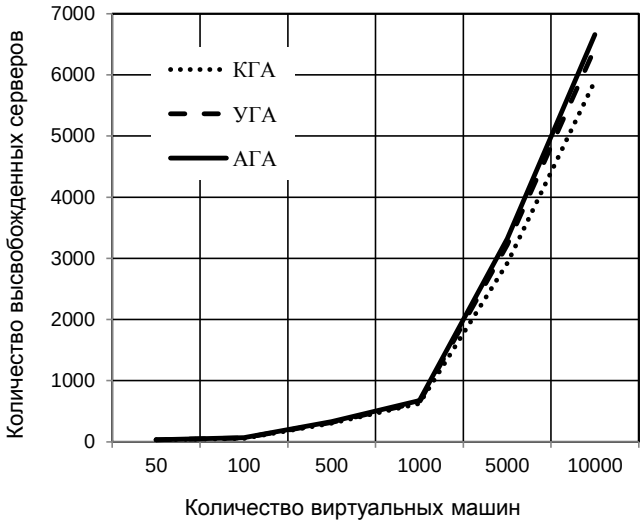
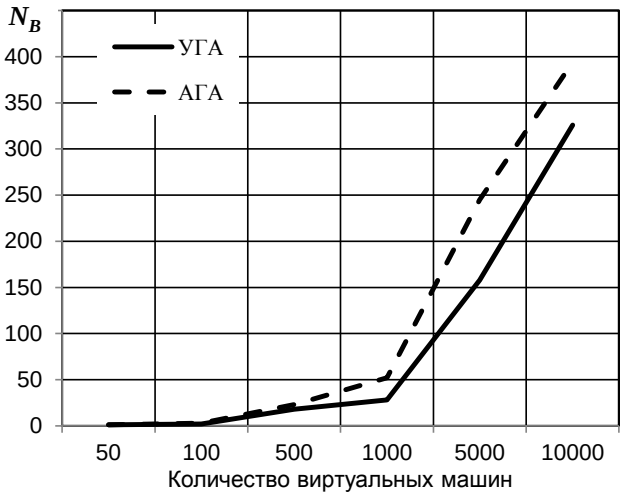
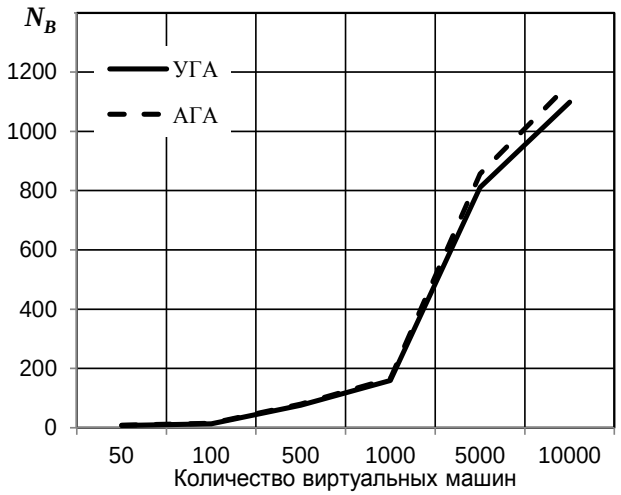


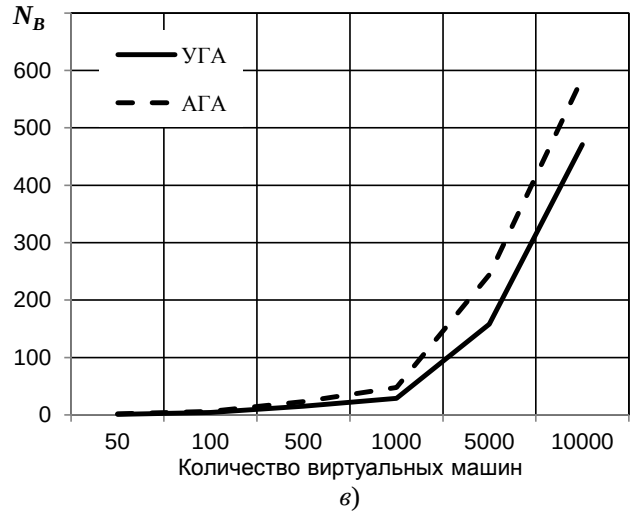
Рис. 2. Зависимость количества высвобожденных серверов от размерности задачи



а)



б)



в)

г)

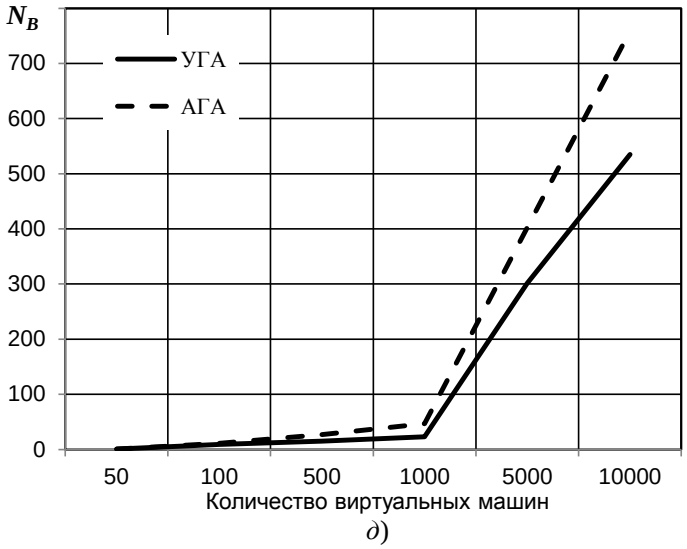


Рис. 3. Зависимость количества дополнительно высвобожденных серверов от размерности задачи, когда требования к запрашиваемым ресурсам лежат в диапазонах: а) к ЦП – [0,05; 0,15], к ОЗУ – [0,3; 0,45]; б) к ЦП – [0,05; 0,15], к ОЗУ – [0,45; 0,6]; в) к ЦП – [0,05; 0,15], к ОЗУ – [0,3; 0.45]; г) к ЦП – [0,05; 0,15], к ОЗУ – [0,45; 0,6]; д) к ЦП – [0,05; 0,6], к ОЗУ – [0,05; 0,6]

Выводы

Разработан адаптивный генетический алгоритм, позволивший решить проблему подбора значений для параметров генетических операторов и вероятностей применения этих операторов в процессе эволюции генетического алгоритма. Фундаментальной особенностью предложенного адаптивного генетического алгоритма является одновременное применение параметрической и алгоритмической адаптации. При проектировании адаптивного генетического алгоритма введены следующие понятия: модифицированные операторы кроссовера и мутации, параметрическая и алгоритмическая адаптация генетического алгоритма, сформированы правила сравнения операторов и понятия параметра производительность генетического оператора. На примере решения задачи распределения ресурсов доказана работоспособность и эффективность предлагаемого адаптивного генетического алгоритма. Доказано, что предлагаемый алгоритм позволяет за равное количество эпох получать лучшие результаты по сравнению с классическим и управляемым вариантом генетического алгоритма.

В следующих публикациях планируется развитие положений адаптивного генетического алгоритма, составление рекомендаций по настройке параметров алгоритма для конкретных задач, проведение экспериментальных исследований для других задач ИТ-сферы.

Литература

1. Теленик С.Ф. Управляемый генетический алгоритм в задачах распределения виртуальных машин в ЦОД / С.Ф. Теленик, А.И. Ролик, П.С. Савченко, М.Е. Боданюк // Вісник ЧДТУ. — 2011. — № 2. — С. 104—113.
2. Теленик С. Управління навантаженням і ресурсами центрів оброблення даних при віртуальному хостингу / С. Теленик, О. Ролік, М. Букасов, С. Андросов, Р. Римар // Вісн. Тернопільського держ. техн. ун-ту. — 2009. — Том 14. — № 4. — С. 198–210.
3. Теленик С.Ф. Генетичні алгоритми вирішення задач управління ресурсами і навантаженням центрів оброблення даних / С.Ф. Теленик, О.І. Ролік, М.М. Букасов, С.А. Андросов // Автоматика. Автоматизація. Електротехнічні комплекси та системи. — 2010. — №1 (25). — С. 106–120.
4. Back T. Optimal Mutation Rates in Genetic Search / T. Back // Fifth International Conference on Genetic Algorithms: University of Illinois at Urbana-Champaign, July 17–21, 1993. – P. 2–8.
5. Julstrom B. A. What Have You Done for me Lately? Adapting Operator Probabilities in a Steady-State Genetic Algorithm / B. A. Julstrom // Proc. of the Sixth International Conference on Genetic Algorithms: University of Pittsburgh, July 15–19, 1995. – P. 81–87.

6. Kosorukoff A. Using incremental evaluation and adaptive choice of operators in a genetic algorithm / A. Kosorukoff // Proc. of the Genetic and Evolutionary Computation Conference: (GECCO-2002), New York, USA, July 9–13, 2002. – P. 688.
7. Lis J. Self-adapting Parallel Genetic Algorithms with the Dynamic Mutation Probability, Crossover Rate and Population Size / J. Lis, M. Lis // Proc. of the 1st Polish National Conference on Evolutionary Computation. Oficina Wydawnica Politechniki, Warszawskiej. – 1996. – P. 324–329.
8. Michalewich Z. Genetic Algorithms + Data Structures = Evolution Programs / Z. Michalewich. – Berlin: Springer, 1996. – 387 p.
9. Michalewich Z. How to Solve It: Modern Heuristics / Z. Michalewich, D. Fogel. – Berlin: Springer, 2002. – 483 p.
10. Pelikan M. Combining the strengths of Bayesian optimization algorithm and adaptive evolution strategies / M. Pelikan, D.E. Goldberg, S. Tsutsui // Genetic and Evolutionary Computation Conference: (GECCO-2002), New York, USA, July 9–13, 2002. – P. 512–519.
11. Spears W.M. Adapting Crossover in Evolutionary Algorithms / W.M. Spears // 4th Annual Conference on Evolutionary Programming: San Diego, California, March 1–3, 1995. – P. 367–384.
12. Thierens D. Adaptive mutation rate control schemes in genetic algorithms / D. Thierens // Congress on Evolutionary Computation: CEC'02, Honolulu, Hawaii, May 12–17, 2002. – P. 980–985.
13. Растригин Л. А. Адаптация сложных систем / Л. А. Растригин // Изв. АН ЛатвССР. – 1978. – №5 (370). – С. 38—45.