Инженерный вестник Дона, №4 (2017)
ivdon.ru/ru/magazine/archive/n4y2017/4543

Индивидуализация управляющих устройств в условиях окружающего интеллекта на основе эволюционных алгоритмов

Л.Х. Манучарян, А.Н. Пачев

Ростовский государственный университет путей сообщения, Ростов-на-Дону

Аннотация: Поскольку пользователи могут иметь весьма разные предпочтения, то персонализация окружающих устройств имеет первостепенное значение. Предложено несколько подходов для установки такой персонализации в форме компьютерного обучения или более специализированных и основанных на научных знаниях и инновациях подходов к обучению. Несмотря на большие успехи в оптимизации, эволюционные алгоритмы в этом контексте мало изучены, главным образом потому, что они известны как элементы, медленно поддающиеся изучению. Так или иначе, в настоящее время существуют довольно быстрые оптимизаторы на основе эволюционных алгоритмов. В данной статье проведен анализ исследования пригодности эволюционных алгоритмов для «окружающего интеллекта».
Ключевые слова: окружающий интеллект, эволюционные алгоритмы, персонализация, оптимизатор, CMA-ES, пользователь, контроллер, компьютерное обучение, датчик

Рост «окружающего интеллекта» становится все более и более очевидным в нашей повседневной жизни: нас окружает все большее число устройств, которые выполняют все виды измерений и пытаются использовать эту информацию разумным способом, например, контролируя некоторые исполнительные механизмы или обеспечивая некоторую форму обратной связи. Для того, чтобы внешние устройства или устройства достаточно разумно функционировали, им необходимо уметь обучаться на поведении пользователя [1,2]. Пользователи могут иметь совершенно отличные параметры не похожие друг на друга, а, следовательно, и система никогда не будет эффективной, и пользовательский интерфейс будет не соответствовать ожиданиям, если только одна стратегия была бы в рабочем состоянии. В дополнение к этому, физическим устройствам необходимо научиться взаимодействовать друг с другом, а с учетом огромного количества различных устройств на рынке, мы не можем заранее определить то, какими они будут.

Изучение параметров и способов создания эффективной кооперации между устройствами было предметом исследования в области окружающего интеллекта (или в тесно связанных с этим областях таких, как всепроникающая компьютеризация и сплошная компьютеризация) [3-5]. Есть три ступени адаптации: начальная фаза, во время которой собираются данные; изучение формы поведения, основанной на собранных данных, и преодоление динамической внешней среды. Главным образом, на первом этапе малое количество методик компьютерного обучения подходят в силу того, что у них едва ли есть какие-либо данные/опыт для усвоения, тогда как это – ключевой этап. На этом этапе алгоритм обучения должен учиться «на лету». Один из решателей проблем, которые, как известно, хорошо работают в природе, эволюционные алгоритмы, не получили большого внимания в этой области и, в частности, не только для описанной подзадачи. Хотя эволюционные алгоритмы в основном рассматриваются как медленные оптимизаторы, их работа показала, что они подходят для ряда оптимизационных проблем, смотри [6]. Более того, такие подходы, как генетическое программирование, очень подходят для формирования сложных контроллеров [7].

В данной статье мы исследуем пригодность эволюционных алгоритмов для задач, связанных с окружающим интеллектом, тем самым предполагая, что данные не доступны нам с самого начала. Точнее, мы изучаем сценарий, где несколько (возможно, гетерогенных) устройств необходимо контролировать простым путем, и при этом беря в расчет предпочтения нескольких пользователей. Логическое обоснование для старта с простым сценарием это то, что мы хотим исследовать вопрос, связанный с возможностью эволюционных алгоритмов решить относительно простую проблему подходящим способом до того, как мы перейдем к более сложным проблемам. Мы используем современный развивающийся оптимизатор, а

именно: CMA-ES [8]. Учитывая природу устройств в окружающем интеллекте, мы используем разные варианты алгоритма: централизованный в противовес отображенному центральному контроллеру и ряду распределенных контроллеров, подавляющие отдельные устройства с помощью их собственного контроллера. В качестве критериев оценки мы оцениваем качество решений, найденных на основе процента от оптимального решения, а также время, необходимое для поиска разумного решения. Мы сравниваем результат с алгоритмическим сравнительным тестом, таким как поиск максимума и симуляция восстановления.

В нашем подходе мы предполагаем среду, в которой несколько внешних устройств будут оснащены датчиками и исполнительными механизмами и будут иметь контроллеры, которые выражают их поведение, то есть распределение сенсорной совокупности значений для действий. Распределение между устройствами и контроллерами остаются открытыми: сперва крайний член пропорций каждого устройства может иметь свой собственный контроллер, тогда как на другой стороне спектра может быть один контроллер для всех устройств в совокупности. В среде присутствует один или несколько пользователей, каждый из которых имеет свои собственные предпочтения в конкретных ситуациях. Здесь ситуация представляет собой уникальную комбинацию сенсорных совокупностей значений или набор таких комбинаций, в которые все карты элементов сопоставляются с похожими ситуациями. Изучение подобного маппинга может являться другим стремлением получить информацию, но в данном предварительном исследовании эволюционных алгоритмов, для окружающего интеллекта это выходит за рамки наших возможностей. Главной целью нашего исследования является создание контроллеров для окружающих устройств, которые лучше удовлетворяли бы предпочтения

пользователя, проблема, которую мы формулируем как задачу максимизации следующей функции:

$$F = \sum_{\forall S:SIT} \sum_{\forall U:USER} user\_satisfaction(U, S, actions_{controllers}(S, U)) \quad , \qquad (1)$$

Другими словами, контроллеры должны во всех ситуациях находить набор действий, который больше будет удовлетворять пользователей. Поскольку удовлетворенность пользователей может обеспечиваться лишь ими самостоятельно, то это связано с тем, что с каждым пользователем все время требуется консультация, а новый контроллер служит источником. Таким образом вторичная цель состоит в том, чтобы свести к минимуму количество соответствующих сравнений алгоритмами для нахождения решения, что поможет избежать чрезмерного вмешательства в сторону пользователя, и пользователь должен выявить все неудовлетворяющие решения.

Мы предполагаем, что контроллер оптимизирован для каждой ситуации отдельно. Для каждой из подобных ситуаций контроллер представлен с помощью числовых значений на каждое действие, которое оно может выполнить. Для двоичного воздействия возможные значения явно ограничены 0 и 1, тогда как для непрерывных воздействий (например, интенсивность света, объем звука) команда может принимать любое значение, которое соответствует данной команде. В таблице 1 показан пример подобного вида.

Таблица № 1

Пример представления для одного контроллера для одной ситуации

a_1	a_2	a_3	a_4
1	0.5	0	0.25

Чтобы решить задачу, которую мы создали, мы используем вариацию различных походов: современные эволюционные алгоритмы, включающие CMA-ES и Cooperative Co-Evolution наряду с алгоритмическим сравнительным тестом, включающим в себя поиск максимума и симуляция восстановления. Альтернативой также будет использование обучения методом проб и ошибок, но с учетом различных ситуаций, исследованных в данной статье это неподходящий вариант и поэтому в деталях здесь он не будет разобран. Каждый из этих алгоритмов будет показан подробнее ниже.

CMA-ES - это эволюционная стратегия. В целом эволюционные алгоритмы работают с совокупностью элементов (в нашем случае выражая ценность команд под определенные ситуации) из которых элементы делятся для изменения данных (на один единственный элемент) и пересекается (комбинирование двух или более элементов), результатом являются новые элементы. Из общей суммы данными библиотеки элементов новые элементы делятся вновь, и этот процесс продолжается пока не будет достигнут какого-либо критерия завершения. Эволюционная стратегия - это вариант в котором команды содержат выделенное поле, которое определяет вероятность изменений, также это поле подвержено модификации, и, следовательно, вероятность изменений самостоятельно адаптируется. В CMA-ES используется ковариационная матрица для повышения эффективности генерации новых элементов. Объяснение всех деталей алгоритма выходит за рамки данной статьи. Мы используем CMA-ES в двух вариантах: один развиваемый оператор цикла, в котором центральный контроллер просто определяет действие для исполнительного звена всех устройств для данной конкретной ситуации и децентрализованное средство решения, при котором каждое устройство имеет свой собственный контроллер и совокупность CMA-ES для выполнения доработки контроллера. В последнем случае мы используем кооперативный коэволюционный подход, предложенный Potter и

De Jong [9]. Здесь одно устройство выбирается при фиксации контроллеров других устройств до тех пор, пока не будет найден лучший вариант. Каждый из элементов совокупности выбранного устройства потом является сравнением в объединении с этими лучшими контроллерами, возникшие в результате оценки пригодности. После этого, выбрано следующее устройство, и так далее. Устройства выбираются циклическим алгоритмическим подходом. Для непрерывных воздействий действительное значение округляется до ближайшего значения (то есть 0/1) во время этапа оценивания.

Рядом с CMA-ES мы также попробуем более простой вариант эволюционных алгоритмов, а именно так называемые «StandardGA», которые в противоречии с высокотехнологичными CMA-ES, выстроены из элементов, которые состоят из битов и используют менее высокотехнологических операторов. Комбинации битов могут представлять собой длительное воздействие для нашего случая. Изменения происходят через бит-флипы, тогда как пересечения осуществляются путем выбора точки пересечения и выделение первой части одного из прямых предков и второй части из другой. Отбор осуществляется с помощью возможности пропорционального соответствия элемента.

Для поиска максимума мы просто пробуем случайные шаги в любом направлении значения для команды и использования сравнения, лучшим решением (исходное состояние с добавлением случайного шага или вычитания оного) будет выборка в качестве следующего контроллера. Процесс завершается, как только остановившееся заданные условия встретятся.

В методе симуляции восстановления, который работает с концепцией показания температурных процессов, шаг выполняется в поисковом

пространстве (то есть для команды), которое приравнена к температуре [10]. Используемая температурная функция:

$$T = \frac{T_0}{\log(k)} \quad , \qquad (2)$$

где T_0 - начальная температура, k - число шагов.

Новое решение допускается, когда они улучшаются, если нет, то они выбираются с вероятностью $e^{\frac{\Delta C}{T}}$, которая зависит от температурного процесса и вырабатывает прирост, ΔC. Эта схема позволяет провести больше исследований на начальном этапе и больше разработок в конце данного процесса.

Далее мы опишем модель, которую мы использовали для оценки выше указанных алгоритмов. Сначала объясняется тематическое исследование, сопровождаемое экспериментальной моделью.

По мере того как проходит практический пример, мы фокусируемся на настройке офиса со светом, который необходимо контролировать. Как уже было сказано, основное внимание уделяется не сложным различным ситуациям, а изучению потенциала эволюционных алгоритмов для относительно простого случая. На рис.1 данная ситуация показана более подробно.

По сути, световых индикаторов больше, чем пользователей и у каждого пользователя есть свои особые предпочтения в выборе интенсивности света. Параметр n определяет сложность ситуации. Здесь n описывает количество пользователей n^2 и количество световых индикаторов $(n-1)^2$. Мы устанавливаем тот уровень сложности, который мы хотим изучить, как различные подходы с равномерным увеличением сложности. Мы выбрали несколько световых индикаторов, которые по количеству меньше, чем количество пользователей, для создания более интересной ситуации.

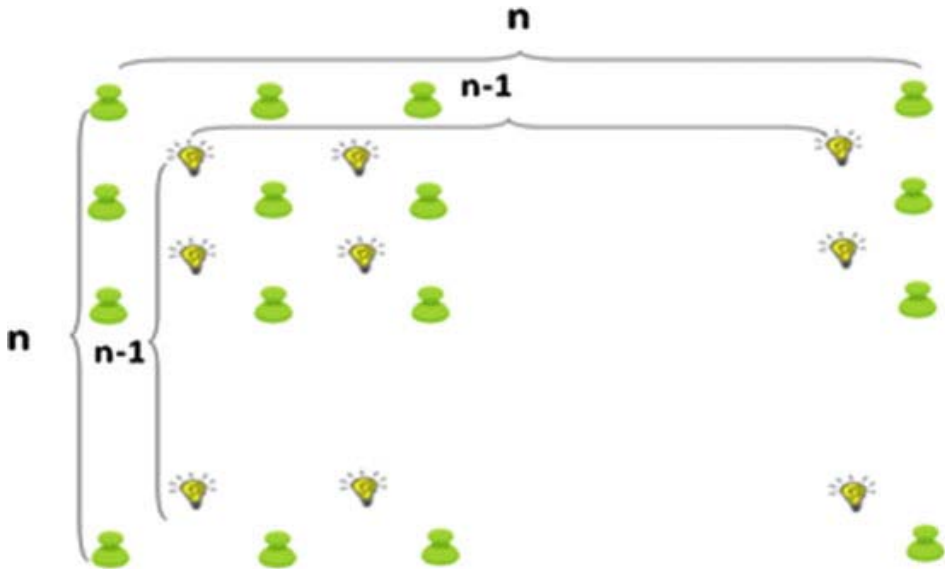

Рис. 1. – Схема сценария

Интенсивность света, используемая пользователем *U*, определяется совокупностью всех световых индикаторов, посредством чего доля каждого светового элемента определяется с помощью следующего уравнения:

$$I(U) = \sum_{\forall L:LIGHTS} \frac{P(L)}{4\pi D(U,L)^2} \quad , \qquad (3)$$

где *P(L)* - мощность света, *D (U, L)* - расстояние от света *L* к пользователю *U*.

Пока мы принимаем одну ситуацию, в которой все пользователи присутствуют в офисе. Каждый пользователь имеет предпочтительную интенсивность света, тем самым определяя функцию (1) пользовательской удовлетворённости (*U*, *S*, *action controllers* (*S*, *U*)):

$$user\_satisfaction(U,S,actions_{controllers}(S,U)) = \left|I(U) - pref\_\text{int}(U)\right| \quad , \qquad (4)$$

Мы осуществили целостную систему в *Matlab*, за исключением *CMA-ES*. В наших экспериментах, мы запускали ряд различных алгоритмических установок, в соответствии с изложенным ранее подходом, который показан в таблице 2. *CMA-ES* запускается и с централизованной, и с распределенной настройкой контроллера, *standard GA* - только с распределенной, и другие сравнительные тесты запускаются только централизованно. Обратим

внимание, что случай не содержит каких-либо входных состояний в момент создания таких подходов, как нештатное обучение методом проб и ошибок. Точные параметры алгоритма представлены в таблице.

Мы пробовали разные уровни сложности, а именно n = 3,4,5, ..., 12, что составило в сумме до 10 ситуаций. Для каждой ситуации мы генерировали 10 частных случаев с разными пользовательскими потребностями, и для каждого случая мы выполнили 30 запусков на выполнение алгоритмов с данными их вероятностного характера. В дополнение к указанным выше показателям мы также запускаем *LP solver* для нахождения решения проблемы. Предположим, что расчет в пределах 20% от оптимального решения являются удовлетворительными. Как критерий остановки, *CMA-ES* использует приспосабливаемое улучшение, как метрику для других алгоритмов, алгоритм останавливается, если он находится в пределах 20% от оптимального решения или превышает на 100,000 оценку пригодности.

Таблица № 2

Экспериментальные установки и настройки

Аббревиатура	Алгоритм	Тип контроллера	Конкретные параметры
C-ES	CMA-ES	Централизованный	Число людей: $4 + (3 \cdot \log((n - 1)\ 2))$
D-ES	CMA-ES	Распределенный	Число людей = 4 для каждого контроллера
D-GA	Standard GA	Распределенный	Количество бит: 16 Число людей: 100 Скоростькроссовера: 0,6 Скорость мутации: 1/16
C-SA	Simulated annealing	Централизованный	$T_0 = 100$
C-HC	Hill climbing	Централизованный	Случайное число из диапазон [0, 10]

Далее описаны результаты эксперимента. Во-первых, мы рассмотрим некоторое количество сравнений, необходимых для принятия разумного

решения (то есть в пределах 20% от оптимального значения). На рис. 2 показаны результаты для различных алгоритмов. По графику можно заметить, что централизованный контроллер, генерируемый посредством алгоритма *CMA-ES*, намного превосходит альтернативные алгоритмы, хотя вариант объема работ децентрализованного *CMA-ES* до сих пор относительно недалек. Вычисление алгоритма кажется выглядит подходящим, учитывая показательное распределение количества световых индикаторов, которые необходимо контролировать, как функцию от n на оси x. Точнее, для простой ситуации, включающей 9 пользователей и 4 световых индикатора, система может найти хорошее решение в пределах 200 сравнений. Однако, ему требуется более 1500 сравнений в более сложной ситуации, включающей в себя 121 световой индикатор и 144 пользователя. Итак, хоть с научной точки зрения ускоритель хорош, то со стороны пользователя это все довольно громоздко. Вариант объема работ между разными запусками является медленным процессом для *CMA-ES*. Когда мы смотрим на поиск максимума и симуляция восстановления, то можем заметить, что объем работ поиска максимума намного хуже, с большими изменениями. Симуляция восстановления действует лучше, но не может достичь скорости видов *CMA-ES. Standard GA* в распределительной системе работает хуже, скорее всего, из-за самой распределительной системы, которая находится в сочетании с элементарностью эволюционных алгоритмов. Таблица 3 показывает полный обзор средней продолжительности для поиска решения с 20% от оптимального.

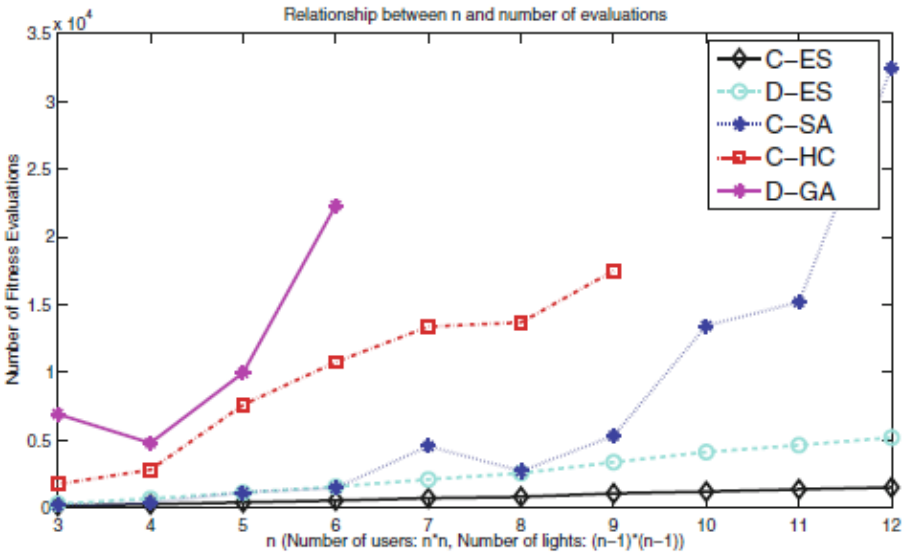

Рис. 2. – Среднее время до оптимального + 20% при изменении n

Более того, на рисунке 3 изображен порог вхождения централизованного *CMA-ES* для $n = 3$, где можно заметить, что в начале алгоритм относительно быстро усваивается, поэтому пользователи не подвергаются воздействию очень низкого качества решения задачи в длительный период времени.

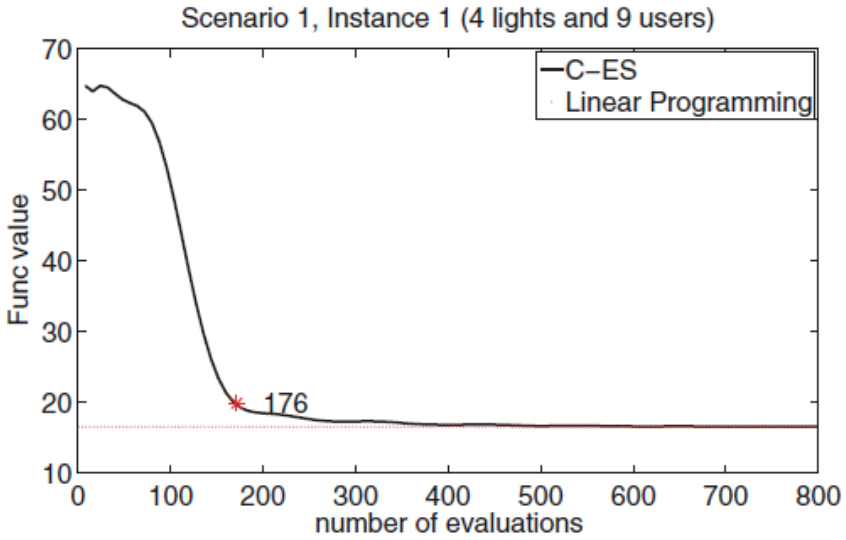

Рис. 3. – Количество оценок по сравнению с удовлетворенностью пользователей для C-ES

Таблица № 2

Обзор среднего времени для поиска решение

Аббриве атура	C-ES		D-ES		C-SA		C-HC		D-GA	
n	Ave	SD	Ave	SD	Ave	SD	Ave	SD	Ave	SD
3	179.2	28.9	333.6	54.3	206.8	107	1775.6	874.7	6941.8	10175.1
4	291	71.8	701.6	229.8	423	357.2	2828.3	2501.1	4806	3253.8
5	429.6	134.1	1148.8	443.7	1151	1789	7623.5	6169.5	9986.7	12361
6	569.4	87.9	1583.6	280.6	1484.3	919.2	10775.6	9118.9	22328.2	7644.7
7	743.4	206.8	2123.2	376.6	4580.3	7096.4	13388.7	12512.9	-	-
8	834	128.5	2584.4	352.7	2736.1	1572.1	13705.6	11093.5	-	-
9	1086.4	139.4	3396.8	350.3	5340.2	2325.6	17512.2	11957.3	-	-
10	1225.7	190.9	4140	282.8	13449.2	21352.6	-	-	-	-
11	1394	184.7	4648	401.6	15202.1	13940.4	-	-	-	-
12	1535.4	157.8	5206.8	552.9	32472.1	55847.1	-	-	-	-

В данной статье мы разобрали возможность использования эволюционных алгоритмов для персонализации в Окружающем интеллекте. К тому же мы попытались сформулировать функцию пригодности, необходимую для эволюционных алгоритмов, а также подбирали первый набор соответствующих вариантов эволюционных алгоритмов. В условиях проведения эксперимента мы заметили, что эволюционные алгоритмы имеют возможность найти нужные качественные решения, но поскольку задачи становятся все более сложными, то объем качества ухудшается. До сих пор пользовательская обратная связь находится только на базовом уровне удовлетворения. Конечно, более подробная обратная связь или первоначальная фаза исследования могут помочь в развитии скорости принятии решения и его качества. В избытке подходы использовались в первичном наблюдении за пользователями для получения первичных наборов приемлемых контроллеров. Однако, это не было целью данной статьи, мы просто хотели узнать может ли функционировать эволюционный алгоритмический обучающий подход с одной единственной обратной

связью, и ответом является, что для простых сред это вполне возможно, но по мере того как вещи будут усложняться, это станет слишком большой нагрузкой для пользователей, не говоря уже о том, что многофазные ситуации должны быть также взяты в расчет. Конечно, подход может еще продолжать функционировать, но наша интуиция является одной из тех вещей, которой необходимы решения альтернативных алгоритмов.

На данный момент явно определенная функция качества в форме пользовательской обратной связи была достигнута. Мы также могли заменить это на альтернативную функцию качества, которая была бы менее ориентированной (например, измерение производительности труда), это не изменило бы установки обучающей системы, которая показывает насколько обобщенными являются подходы. Насколько хорошо подход будет изучать оптимальное освещение, а будет ли необходимость в данном изучении, - это будет интересной темой для будущей работы. В дополнение, мы предполагаем исследовать более комплексно сложные ситуации, где события играли бы более заметную для всех роль и сравнивались скорее обучающими алгоритмами для более основанных на знаниях подходах.

Работа выполнена при финансовой поддержке РФФИ, проект 17-07-00620-а.

Литература

1. Шалова С.Х. Обзор и анализ исследований в области систем обволакивающего интеллекта // Инженерный вестник Дона, 2016, №4 URL: ivdon.ru/ru/magazine/archive/n4y2016/3926.

2. Истомин В.В. Прогнозирование поведения групп автономных интеллектуальных агентов на основе теории многоагентных систем // Инженерный вестник Дона, 2011, №4 URL: ivdon.ru/ru/magazine/archive/n4y2011/535.

3. A. Aztiria A., Izaguirre A., Augusto J.C. Learning patterns in ambient intelligence environments// Artificial. Intelligence, Vol. 34, №1, pp. 35–51, 2010.

4. Mozer M.C. Lessons from an Adaptive Home // Wiley, New York, pp. 271–294, 2005.

5. Weiser M., Ubiquitous computing // Computer, Vol.26 №10, pp. 71–72, 1993.

6. Eiben A.E., Smith J. Introduction to Evolutionary Computing, Springer, London, pp. 11–38, 2003.

7. Banzhaf W., Nordin P., Keller R., Francone F. Genetic Programming: An Introduction (Morgan Kaufmann, San Francisco, 1998), pp. 481

8. Hansen N., Ostermeier A. Adapting arbitrary normalmutation distributions in evolution strategies: the covariance matrix adaptation, in Proceedings of IEEE International Conference on Evolutionary Computation, May 1996, pp. 312–317

9. Potter M.A., Jong K.A.D. A cooperative coevolutionary approach to function optimization, in Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, PPSN III (Springer, London, UK, 1994), pp. 249–257

10. Van Laarhoven P., Aarts E. Simulated annealing, in Simulated Annealing: Theory and Applications, Mathematics and Its Applications, Vol. 37 (Springer, Netherlands, 1987), pp. 7–15

References

1. Shalova S.Kh. Inženernyj vestnik Dona (Rus), 2016. №4. URL: ivdon.ru/ru/magazine/archive/n4y2016/3926.

2. Istomin V.V. Inženernyj vestnik Dona (Rus), 2011. №4. URL: ivdon.ru/ru/magazine/archive/n4y2011/535.

3. A. Aztiria A., Izaguirre A., Augusto J.C. Artificial. Intelligence, Vol. 34, №1, pp. 35–51, 2010.

4. Mozer M.C. Lessons from an Adaptive Home: Wiley, New York, pp. 271–294, 2005.

5. Weiser M., Ubiquitous computing: Computer, Vol.26 №10, pp. 71–72, 1993.

6. Eiben A.E., Smith J. Introduction to Evolutionary Computing, Springer, London, pp. 11–38, 2003.

7. Banzhaf W., Nordin P., Keller R., Francone F. Genetic Programming: An Introduction (Morgan Kaufmann, San Francisco, 1998), pp. 481

8. Hansen N., Ostermeier A. Adapting arbitrary normalmutation distributions in evolution strategies: the covariance matrix adaptation, in Proceedings of IEEE International Conference on Evolutionary Computation, May 1996, pp. 312–317

9. Potter M.A., Jong K.A.D. A cooperative coevolutionary approach to function optimization, in Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, PPSN III (Springer, London, UK, 1994), pp. 249–257

10. Van Laarhoven P., Aarts E. Simulated annealing, in Simulated Annealing: Theory and Applications, Mathematics and Its Applications, Vol. 37 (Springer, Netherlands, 1987), pp. 7–15