Поиск эффективного набора взаимодействующих компонентов программных систем на основе роевого интеллекта[1]

А. А. Гусев

Кубанский государственный университет
350040, Россия, Краснодар, ул. Ставропольская, 149

e-mail: alexandrgsv@gmail.com

Аннотация. В статье рассмотрено применение роевого интеллекта для решения актуальной задачи выбора компонентов распределенной информационной системы на основе экспериментальных оценок критериев качества взаимодействия программных компонентов. Для реализации роевого интеллекта использован алгоритм искусственной пчелиной колонии. Произведена формализация задачи выбора программных компонентов Node.js, разработан общий критерий качества взаимодействия программных компонентов на основе линейной свертки 14 частных критериев качества взаимодействия. Произведена модификация алгоритма искусственной пчелиной колонии для работы с натуральными наборами программных компонентов с учетом введенного дополнительного условия останова алгоритма на основе заданного критерия сходимости. Полученное решение указывает на более высокую скорость сходимости алгоритма искусственной пчелиной колонии в сравнении с генетическим алгоритмом при решении задачи выбора программных компонентов. Полученное решение может использоваться при разработке распределенных информационных систем на основе Node.js с использованием рассмотренных в исследовании программных компонентов.
Ключевые слова: качество систем и программ, QoS, эффективность взаимодействия программ, эволюционные вычисления, роевой интеллект, алгоритм пчелиной колонии.

1. Введение

Роевой интеллект является разновидностью коллективного интеллекта, проявляемого в живой природе общественными насекомыми. Интерес к изучению поведения общественных насекомых обусловлен их высокой эффективностью в решении оптимизационных задач, таких как нахождение кратчайшего пути между муравейником и источником пищи или оптимальное распределение рабочих ролей в пчелином улье. Несмотря на то что эти насекомые сами по себе сравнительно просто

[1] Работа выполнена при финансировании Министерства науки и высшего образования Российской Федерации, проект 25.13253.2018/12.1 «Разработка технологической концепции Дата-центра междисциплинарных исследований в образовании».

устроены, взаимодействуя друг с другом и образуя рой, они способны демонстрировать достаточно сложное поведение, направленное на решение оптимизационных задач. В последние два десятилетия значительное число исследований посвящены разработкам и практическому применению [1–3] алгоритмов численной оптимизации, вдохновленных моделями поведения роя общественных насекомых. Наиболее распространенными методами данной группы являются метод искусственной муравьиной колонии [4, 5], метод роя частиц [6, 7], метод искусственной пчелиной колонии [8, 9], алгоритм светлячков [10, 11], алгоритм поиска кукушки [12, 13], алгоритм летучей мыши [14] и др. Среди указанных методов алгоритм искусственной пчелиной колонии (АПК) является достаточно новым и в то же время уже показавшим свою эффективность [15] методом, используемым при решении задач управления программной разработкой.

Целью данной работы является решение задачи выбора компонентов информационной системы на основе экспериментально вычисляемых критериев качества взаимодействия на примере выбора компонентов фреймворка Node.js с помощью АПК.

Актуальность решения данной задачи обусловлена тем, что по мере развития и повсеместного внедрения распределенных информационных систем в области цифровой экономики, оказания государственных услуг, телемедицины важной становится задача проектирования таких информационных систем с учетом требований к качеству обслуживания (QoS) в конкретных условиях функционирования.

Эффективным программно-математическим средством обеспечения QoS в современных программных системах, разрабатываемых с использованием компонентно-ориентированного подхода, является методика выбора программных компонентов на этапе разработки информационной системы с учетом экспериментально вычисляемого критерия качества взаимодействия программных компонентов. Базовым элементом такой методики является методика проведения воспроизводимых экспериментов [16, 17] по оценке качества взаимодействия компонентов.

Ранее было получено решение задачи выбора программных компонентов с использованием генетического алгоритма [18]. Однако было отмечено немонотонное убывание целевого функционала качества взаимодействия, что указывает на плохую сходимость генетического алгоритма в решении данной задачи и требует исследования применения иных методов эволюционных вычислений, таких как АПК, в решении данной задачи.

Статья состоит из 5 разделов, первый раздел — Введение, во втором разделе сформулирована задача, в третьем разделе приводится описание метода решения задачи, в четвертом разделе представлены результаты поиска эффективного выбора

компонентов информационной системы на основе экспериментальных оценок, пятый раздел завершает статью.

2. Постановка задачи

Пусть имеется набор $f_1, \ldots, f_n$ булевых параметров, характеризующих функциональные возможности разрабатываемой программной системы. Функциональные возможности требуется реализовать альтернативным набором программных компонентов. Сопоставим каждому f_i, $i = \overline{1,n}$ набор альтернативных программных компонентов $d_i = (q_1^i, \ldots, q_{j-1}^i, q_j^i, q_{j+1}^i, \ldots)$, $i = \overline{1,n}$. Обозначим все такие наборы, как $D = (d_1, \ldots, d_n)$.

Обозначим Ω — множество всех возможных уникальных конфигураций, соответствующих выбору по одной альтернативе q_j^i из каждого $d_i \in D$ для реализации каждого f_i, $i = \overline{1,n}$.

Обозначим $\Psi(\omega), \omega \in \Omega$ — экспериментально оцениваемый общий функционал, определяющий качество взаимодействия программных компонентов:

$$\Psi(\omega) = \sum_{j=1}^{N_R} w_j R_j, \tag{1}$$

где N_R — общее количество рассматриваемых частных критериев качества R_j; w_j — веса критериев $\sum_{j=1}^{N_R} w_j = 1$.

Используются следующие частные критерии качества R_j: R_1 — время работы микропроцессора, затраченное на инициализацию эксперимента (мкс); R_2 — время работы микропроцессора, затраченное на исполнение системных функций в ходе инициализации эксперимента (мкс); R_3 — прирост утилизируемого объема оперативной памяти, отмечаемый по завершении инициализации эксперимента (включая heap, code segment и stack) (байт); R_4 — прирост размера heap (кучи), отмечаемый по завершении инициализации эксперимента (байт); R_5 — прирост объема используемой heap (кучи), отмечаемый по завершении инициализации эксперимента (байт); R_6 — прирост объема памяти, используемой объектами C++, связанными с JavaScript объектами, отмечаемый по завершении инициализации эксперимента (байт); R_7 — реальное время, затраченное на инициализацию эксперимента (нс); R_8 — время работы микропроцессора, затраченное на эксперимент (мкс); R_9 — время работы микропроцессора, затраченное на исполнение системных функций в

ходе эксперимента (мкс); R_{10} — прирост утилизируемого объема оперативной памяти, отмечаемый по завершении эксперимента (включая heap, code segment и stack) (байт); R_{11} — прирост размера heap (кучи), отмечаемый по завершении эксперимента (байт); R_{12} — прирост объема используемой heap (кучи), отмечаемый по завершении эксперимента (байт); R_{13} — прирост объема памяти, используемой объектами C++, связанными с JavaScript объектами, отмечаемый по завершении эксперимента (байт); R_{14} — реальное время, затраченное на эксперимент (нс).

Используются следующие весовые коэффициенты w_j, задающие цели в области QoS: $w_1 = w_3 = \ldots = w_{10} = w_{12} \ldots = w_{14} = 0.07; w_2 = w_{11} = 0.08.$

Задача эффективного выбора программных компонентов на основе экспериментальной оценки качества взаимодействия формулируется следующим образом:

$$\Psi(\omega) \to \min_{\omega \in \Omega}. \qquad (2)$$

В табл. 1 приведен набор используемых функций и компонентов, реализующих эти функции в эксперименте.

Таблица 1. Перечень используемых функций и компонентов

Функция	*Компоненты, реализующие функцию*	*Описание*
Filter	Lodash Underscore	Последовательно проверяет все элементы массива на предмет соответствия условию и возвращает массив, состоящий из элементов, для которых проверка дала значение «Истина»
First	Lodash Underscore	Возвращает первый элемент массива
FsRead	Fs-extra Fs	Считывает данные из файла
FsReaddir	Fs–extra	Считывает содержимое каталога, возвращая массив имен файлов и директорий в каталоге
FsReaddirRecursive	Recursive-readdir	Рекурсивно считывает содержимое каталога, возвращая массив имен файлов и директорий в каталоге
HashMD5	Hasha md5 Ts-md5	Вычисляет MD5-хеш от заданного набора данных
Map	Lodash Underscore Средства языка JavaScript	Применяет заданную функцию ко всем элементам массива, возвращая тем самым новый массив, состоящий из преобразованных элементов
PathResolve	Path	Формирует полный пусть к файлу или директории на основе заданного массива элементов пути
StringReplace	Средства языка JavaScript	Находит и заменяет подстроку в переданной строке
ZipCompress	Adm-zip Jszip Zipit	Производит архивацию переданного массива файлов и возвращает сформированный Zip-архив

3. Метод

Алгоритм пчелиной колонии — это полиномиальный эвристический метод оптимизации, имитирующий поведение медоносных пчел при сборе нектара [6]. АПК относится к методам роевого интеллекта.

Тремя основными компонентами модели поведения роя медоносных пчел являются:

- источники питания: значение источника питания зависит от многих факторов, таких как близость к улью, питательность и простота извлечения нектара;
- рабочие фуражиры: агенты, связанные с конкретным источником питания, на котором они «трудоустроены». Агенты переносят информацию о своем источнике, расстоянии до него и его выгодности и делятся этой информацией с некоторой вероятностью;
- нерабочие фуражиры: агенты, постоянно находящиеся в поиске новых источников питания. Выделяется два типа нерабочих фуражиров: разведчики, исследующие среду вокруг улья в поисках новых источников питания и наблюдатели, ожидающие в улье и осваивающие новый источник питания с помощью информации, которой делятся рабочие фуражиры.

Блок-схема АПК приведена на рис. 1.

В математической форме потенциальные решения оптимизационной задачи представляются АПК в виде кортежа переменных (координат источника питания или решения):

$$X_i = \left(x_{i,1}, x_{i,2}, ..., x_{i,n}\right), \tag{3}$$

где X_i обозначает i-е решение в популяции; n — размерность решения.

Для улучшения положения источника питания рабочая пчела использует следующее уравнение:

$$v_{ij} = x_{ij} + \Theta_{ij}\left(x_{ij} - x_{kj}\right), \tag{4}$$

где $k \in \{1, 2, ..., BN\}$; BN — количество рабочих пчел; $j \in \{1, 2, ..., n\}$, k и j — случайно выбираемые индексы с условием $k \neq j$; Θ_{ij} — случайное число от –1 до 1.

Значение функции приспособленности v_{ij} проверяется и, если оно оказывается лучше, чем значение x_{ij}, происходит замена x_{ij} на v_{ij}, в противном случае x_{ij} сохраняется в следующей итерации.

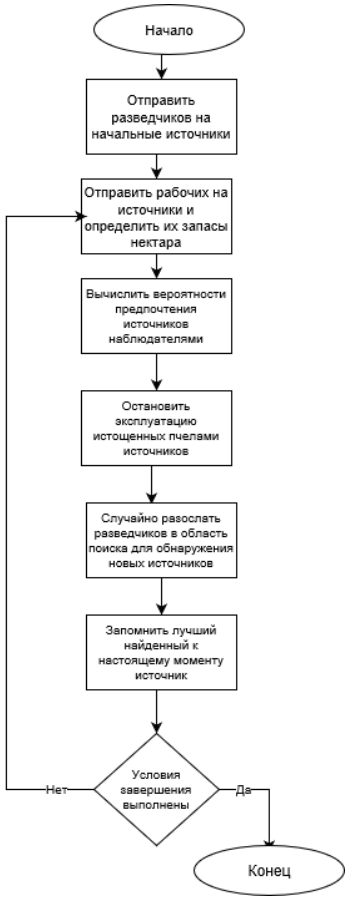

Рисунок 1. Блок-схема алгоритма пчелиной колонии

Как только все рабочие пчелы завершат оптимизационный процесс на своих источниках, они делятся информацией о своих улучшенных источниках с наблюдателями в процессе танца в улье. После этого наблюдающая пчела выбирает i-й источник с вероятностью, задаваемой следующим уравнением:

$$P_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j}, \tag{5}$$

где fit_i — значение функции приспособленности i-го решения или, другими словами, отношение количества рабочих пчел к количеству нектара в источнике, идентифицируемом позицией i; SN — количество источников питания, равное количеству рабочих пчел BN. Если i-е решение оказывается лучшим, вероятность выбора

i-го источника будет выше. Если положение источника не улучшается после предопределенного количества итераций, источник отвергается.

Затем пчела-разведчик ищет новый источник питания X_i^* для замены отвергнутого источника X_i в соответствии с уравнением

$$x_{i_k}^* = lb_j + rand(0,1) \times (ub_j - lb_j), \quad (6)$$

где ub и lb — верхняя граница и нижняя граница соответственно; $rand(0,1)$ — случайное число между 0 и 1.

Процесс работы алгоритма продолжается, пока не будет достигнуто предопределенное число итераций или выполнены иные условия остановки.

Для численного представления конфигураций выбора программных компонентов вводится отображение кодирования $G:\Omega \to \Lambda \subseteq \mathbb{N}^n$. Таким образом, каждой конфигурации $\omega \in \Omega$ соответствует натуральный набор $\zeta = G(\omega)$, $\omega \in \Omega$, $\zeta \in \Lambda \subseteq \mathbb{N}^n$, называемый позицией источника. В процессе работы АПК позиции источников представляются как $\zeta_p^k = (\alpha_{1_p}^k ... \alpha_{n_p}^k)$, $k = \overline{1, |\Theta_p|}$, где каждая $\alpha_{i_p}^k$ принимает значения от 1 до $|d_i|$, соответствующие порядковому номеру выбранной альтернативы из d_i; Θ_p — множество позиций источников (популяция решений), принадлежащих *p*-й итерации АПК. Вводится также обратное отображение $G^{-1}:\Lambda \to \Omega$, осуществляющее преобразование положения источника в соответствующий ему выбор программных компонентов. С учетом введенных обозначений $\Psi(\omega)$ можно рассматривать как ценовую функцию АПК, а исходная задача (2) сводится к задаче

$$\Psi(\omega) \to \min_{\omega \in \{G^{-1}(\zeta), \zeta \in \Theta_{term},\}}, \quad (7)$$

где Θ_{term} — последняя популяция решений перед остановкой АПК.

Поскольку исследуемые наборы компонентов при решении задачи (7) являются натуральными, в уравнениях (4) и (6) дробная часть второго слагаемого отбрасывается, случайное число принимает значение в интервале (0;1) в обоих случаях. Остановка алгоритма осуществляется как при достижении предопределенного количества итераций, так и при превышении заданного количества последовательных итераций, в которых модуль изменения лучшего решения между двумя последовательными итерациями не превосходит заранее заданный порог сходимости алгоритма.

Конфигурация экспериментального стенда:

– характеристика вычислительной системы — микропроцессор: Intel®Core™ i7—7700; количество ядер: 4; логических процессоров: 8, тактовая частота: 3,60 ГГц; оперативная память: 12,0 ГБ;

- программное обеспечение — операционная система (Хост): Ubuntu 16.04 LTS; версия MATLAB: R2018a; версия Vagrant: 2.2.4; версия Node.js: 10.15.3;
- параметры виртуальной машины: 2 ядра процессора; 2,0 ГБ ОЗУ; Ubuntu 16.04 LTS; используемое средство провизии: Ansible; используемые средства обмена файлами с виртуальной машиной: NFS-сервер + BindFS внутри виртуальной машины;
- установленное дополнительное системное ПО; git, make, htop, iotop, rsync, node-gyp;
- параметры роевого алгоритма — размер улья: 20; количество пчел-наблюдателей: 20; предел оставления источника: 120; предельное количество итераций: 100; предельное количество стагнирующих итераций: 10; порог сходимости алгоритма: 0.001.

4. Результаты

В результате осуществления 16 итераций эволюционного поиска было найдено решение задачи (3), соответствующее значению функционала качества взаимодействия программных компонентов 0.216. Эволюционный поиск занял 96 секунд и завершился после 10 стагнирующих итераций в соответствии с заданным порогом сходимости алгоритма. Экспериментальные замеры для терминальной итерации эволюционного поиска представлены в табл. 2, найденное решение идентифицировано в табл. 4.

Таблица 2. Взвешенные экспериментальные замеры и значение $\Psi(\omega)$ *в терминальной итерации*

№ источника, i	w_1R_1	w_2R_2	w_3R_3	w_4R_4	w_5R_5	w_6R_6	w_7R_7	w_8R_8
1	0.0168	0	0.0185	0.048	0.0192	0.0127	0.035	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	$\Psi(\omega)$	
	0	0	0	0.0129	0.0672	0.0374	0.2957	
2	0.0168	0	0.0222	0.048	0.0216	0.0137	0.0322	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	$\Psi(\omega)$	
	0	0	0	0.0121	0.0672	0.0335	0.2953	
3	0.0168	0	0.0207	0.0443	0.0207	0.0124	0.0282	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	$\Psi(\omega)$	
	0	0	0	0.013	0.0672	0.0357	0.287	
4	0.0196	0	0.0213	0.0443	0.0215	0.0143	0.0288	0
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	$\Psi(\omega)$	
	0	0.0146	0	0.0118	0.0672	0.0287	0.2723	
5	0.0168	0	0.0167	0.0517	0.0194	0.0126	0.0303	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	$\Psi(\omega)$	
	0	0	0	0.0121	0.0672	0.0339	0.2887	

Окончание таблицы 2.

№ источника, i	w_1R_1	w_2R_2	w_3R_3	w_4R_4	w_5R_5	w_6R_6	w_7R_7	w_8R_8
6	0.0112	0	0.0055	0.0367	0.0036	0.0047	0.0302	0
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0.0169	0	0.012	0.0672	0.028	0.216	
7	0.0168	0	0.0186	0.0443	0.0183	0.0116	0.0269	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0	0	0.0131	0.0672	0.0349	0.2797	
8	0.014	0	0.0168	0.0407	0.0181	0.0116	0.0348	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0	0	0.0129	0.0672	0.0382	0.2822	
9	0.014	0	0.0168	0.0443	0.0157	0.009	0.0275	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0	0	0.013	0.0672	0.0352	0.2707	
10	0.0168	0	0.0189	0.0443	0.0208	0.0133	0.0299	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$		
	0	0	0	0.0131	0.0672	0.0326		
11	0.014	0	0.0166	0.0553	0.0158	0.0091		
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$		
	0	0	0	0.013	0.0672	0.0315		
12	0.0196	0	0.0204	0.048	0.0216	0.0143		
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0	0	0.0121	0.0672	0.0309	0.2849	
13	0.0196	0	0.0187	0.048	0.0194	0.0126	0.0296	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0	0	0.0121	0.0672	0.0294	0.2801	
14	0.014	0	0.0208	0.0443	0.0207	0.0133	0.0347	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0	0	0.0118	0.0672	0.0294	0.2803	
15	0.0168	0	0.0186	0.048	0.0216	0.0143	0.0311	0
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0.0143	0	0.0118	0.0672	0.0278	0.2715	
16	0.014	0	0.016	0.0443	0.0155	0.009	0.0261	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0.0152	0	0.013	0.0672	0.0353	0.2836	
17	0.014	0	0.0167	0.0407	0.0148	0.0074	0.0259	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0	0	0.0128	0.0672	0.0347	0.2622	
18	0.0196	0	0.0186	0.0443	0.0214	0.0137	0.0337	0
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0.0206	0	0.0118	0.0672	0.0296	0.2807	
19	0.0168	0	0.0167	0.0553	0.0159	0.0091	0.0249	0.028
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0	0	0	0.013	0.0672	0.0366	0.2835	
20	0.0168	0	0.0205	0.048	0.0216	0.0143	0.0301	0
	w_9R_9	$w_{10}R_{10}$	$w_{11}R_{11}$	$w_{12}R_{12}$	$w_{13}R_{13}$	$w_{14}R_{14}$	Ψ(ω)	
	0.028	0	0	0.012	0.0672	0.0285	0.287	

Таблица 3. Найденное решение

Позиция источника	*Соответствующий выбор программных компонентов*	
	Функция	Компонент
[2 3 2 1 1 1 1 2 1 1]	Filter	Underscore
	Map	Underscore
	First	Underscore
	PathResolve	Средства языка JavaScript
	StringReplace	Средства языка JavaScript
	ZipCompress	Adm-zip
	HashMD5	Hasha
	FsRead	Fs
	FsReaddir	Fs-extra
	FsReaddirRecursive	Recursive-readdir

График эволюции решений представлен на рис. 2.

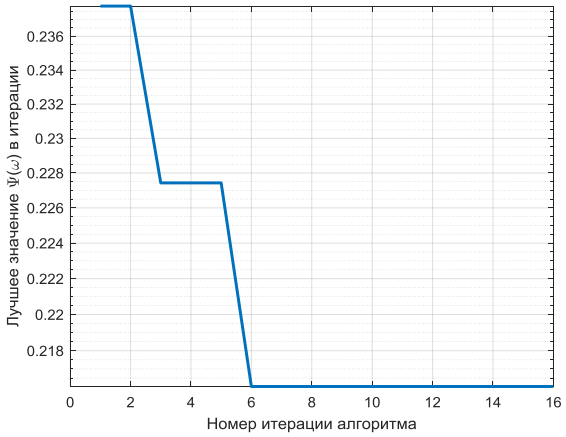

Рисунок 2. График эволюционного поиска выбора компонентов информационной системы на основе минимизации общего критерия качества взаимодействия программных компонентов Ψ(ω)

График эволюционного поиска демонстрирует монотонное убывание Ψ(ω), что указывает на то, что АПК сходится при решении данной задачи.

5. Обсуждение и выводы

В статье рассмотрена целочисленная модификация вещественного алгоритма искусственной пчелиной колонии для управления поиском эффективного набора программных компонентов, проведены экспериментальные исследования и представлены их результаты.

1. Полученное решение задачи выбора компонентов распределенной информационной системы с помощью модифицированного алгоритма искусственной пчелиной колонии отличается монотонностью графика эволюционного поиска, указывая на лучшую сходимость АПК в сравнении с генетическим алгоритмом, примененным для решения этой задачи ранее [18].

2. Данный результат согласуется с сравнительными исследованиями генетических алгоритмов и методов роевого интеллекта в управлении разработкой программ, обзор которых приведен в [15].

3. Решение, аналогичное полученному ранее с помощью генетического алгоритма [18], было найдено за 6 итераций эволюционного поиска с помощью АПК, что говорит о значительно более высокой скорости сходимости АПК в сравнении с генетическим алгоритмом при решении задачи выбора программных компонентов.

Дальнейшие исследования будут направлены на разработку модификаций АПК, обладающих более высокими показателями скорости сходимости и качества получающихся решений для использования в экспериментальном выборе эффективных наборов программных компонентов распределенных информационных систем.

Литература

[1] *Saurabh B., Kunwar A., Samaresh M., Madhabananda D.* A Swarm intelligence based chaotic morphological approach for software development cost estimation // *International Journal of Intelligent Systems and Applications.* 2018. Vol. 9. P. 13–22.

[2] *Jain D. K., Kumar A., Sangwan S. R., Nguyen G. N., Tiwari P.* A Particle Swarm Optimized Learning Model of Fault Classification in Web-Apps // *IEEE Access*. 2019. Vol. 7. P. 18480–18489.

[3] *Chiang H. S., Sangaiah A. K., Chen M. Y., Liu J. Y.* A Novel Artificial Bee Colony Optimization Algorithm with SVM for Bio-inspired Software-Defined Networking // *International Journal of Parallel Programming*. 2018. P. 1–19. (https://doi.org/10.1007/s10766-018-0594-6)

[4] *Gülcü Ş., Mahi M., Baykan Ö. K., Kodaz H.* A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem // *Soft Computing*. 2018. Vol. 22. No. 5. P. 1669–1685.

[5] *Wei W., Tian Z., Peng C., Liu A., Zhang Z.* Product family flexibility design method based on hybrid adaptive ant colony algorithm // *Soft Computing*. 2018. P. 1–12. (https://doi.org/10.1007/s00500-018-3622-y)

[6] *Ran C., Yaochu J.* A social learning particle swarm optimization algorithm for scalable optimization // *Information Sciences*. 2015. Vol. 291. P. 43–60.

[7] *Delgarm N., Sajadi B., Kowsary F., Delgarm S.* Multi-objective optimization of the building energy performance: A simulation-based approach by means of particle swarm optimization (PSO) // *Applied Energy*. 2016. Vol. 170. P. 293–303.

[8] *Karaboga D., Basturk B. J.* A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm // *Journal of Global Optimization*. 2007. Vol. 39. No. 3. P. 459–471.

[9] *Xue Y., Jiang J., Zhao B., Ma T.* A self-adaptive artificial bee colony algorithm based on global best for global optimization // *Soft Computing*. 2018. Vol. 22. No. 9. P. 2935–2952.

[10] *Marinaki M., Marinakis Y.* A Glowworm Swarm Optimization algorithm for the Vehicle Routing Problem with Stochastic Demands // *Expert Systems with Applications*. 2016. Vol. 46. P. 145–163.

[11] *Li Y., Ni Z., Jin F., Li J., Li F.* Research on Clustering Method of Improved Glowworm Algorithm Based on Good-Point Set // *Mathematical Problems in Engineering*. 2018. Vol. 2018. Article ID 8724084.

[12] *Wang G. G., Gandomi A. H., Zhao X., Chu H. C. E.* Hybridizing harmony search algorithm with cuckoo search for global numerical optimization // *Soft Computing*. 2016. Vol. 20. No. 1. P. 273–285.

[13] *Shehab M., Khader A. T., Al-Betar M. A.* A survey on applications and variants of the cuckoo search algorithm // *Applied Soft Computing*. 2017. Vol. 61. P. 1041–1059.

[14] *Adarsh B. R., Raghunathan T., Jayabarathi T., Yang X. S.* Economic dispatch using chaotic bat algorithm // Energy. 2016. Vol. 96. P. 666–675.

[15] *Brezočnik L., Fister I., Podgorelec V.* Solving Agile Software Development Problems with Swarm Intelligence Algorithms. // *New Technologies, Development and Application II* NT Lecture Notes in Networks and Systems. Vol 76. ed. I. Karabegović. — Springer, Cham, 2019. P. 298–309.

[16] *Ильин Д. Ю., Гусев А. А.* Проведение воспроизводимых экспериментов по оценке эффективности работы компонентов программного обеспечения // Прикладные исследования и технологии ART 2019: сб. тр. рег. конф. — М. : МТИ, 2019. С. 54–56.

[17] *Ильин Д. Ю., Никульчев Е. В., Колясников П. В.* Выбор технологических решений для разработки программного обеспечения распределенных информационных систем // *Современные информационные технологии и ИТ-образование*. 2018. Т. 14. № 2. С. 344–354.

[18] *Ильин Д. Ю., Гусев А. А., Никульчев Е. В.* Генетический алгоритм выбора компонентов информационных систем на основе экспериментальных оценок критериев качества // *Прикаспийский журнал: управление и высокие технологии*. 2019. № 2. С. 113–125.

***Автор*:**

Александр Алексеевич Гусев — преподаватель кафедры прикладной математики, Кубанский государственный университет; соискатель кафедры управления и моделирвоания систем, МИРЭА — Российский технологический университет

Swarm intelligence search for an effective set of interacting software components

A. A. Gusev

Kuban State University, 149 Stavropolskaya st., Krasnodar, Russia 350040
e-mail: alexandrgsv@gmail.com

Abstract. The paper deals with the application of swarm intelligence to solve the topical problem of selecting components of a distributed information system based on experimental evaluations of the software interaction quality criteria. The swarm intelligence is implemented using the artificial bee colony algorithm. The formalization of the problem of selecting Node.js software components is made in the paper, the general software components interaction quality criterion is presented as a linear convolution of 14 quality criteria. The algorithm of artificial bee colony was modified to work with natural sets of software components considering the additional stop condition of the algorithm based on a given criterion of convergence. The obtained solution indicates a higher rate of convergence of the artificial bee colony algorithm in comparison with the genetic algorithm in solving the problem of selecting software components. The resulting solution can be used in the development of similar distributed information systems.

Keywords: system and software quality, QoS, software interaction efficiency, evolutionary computation, swarm intelligence, artificial bee colony.

References

[1] *Saurabh B., Kunwar A., Samaresh M. & Madhabananda D.* (2018) *IJISA*, **9**:13–22.
[2] *Jain D. K., Kumar A., Sangwan S. R., Nguyen G. N. & Tiwari P.* (2019) *IEEE Access*, **7**:18480–18489.
[3] *Chiang H. S., Sangaiah A. K., Chen M. Y., & Liu J. Y.* (2018) *International Journal of Parallel Programming*, 1–19. (https://doi.org/10.1007/s10766-018-0594-6)
[4] *Gülcü Ş., Mahi M., Baykan Ö. K., & Kodaz H.* (2018) Soft Computing, **22**(5):1669–1685.
[5] *Wei W., Tian Z., ... & Zhang Z.* (2018) *Soft Computing*, 1–12. (https://doi.org/10.1007/s00500-018-3622-y)
[6] *Ran C., Yaochu J.* (2015) *Information Sciences*, **291**:43–60.
[7] *Delgarm N., Sajadi B., Kowsary F. & Delgarm S.* (2016) *Applied Energy*, **170**:293–303.
[8] *Karaboga D. & Basturk B. J. A* (2007) *Journal of Global Optimization*, **39**(3):459–471.
[9] *Xue Y., Jiang J., Zhao B. & Ma T.* (2018) *Soft Computing*, **22**(9):2935–2952.
[10] *Marinaki M. & Marinakis Y.* (2016) *Expert Systems with Applications*, **46**:145–163.
[11] *Li Y., Ni Z., Jin F., Li J. & Li F.* (2018) *Mathematical Problems in Engineering*, **2018**:8724084.
[12] *Wang G. G., Gandomi A. H., Zhao X. & Chu H. C. E.* (2016) *Soft Computing*, **20**(1):273–285.
[13] *Shehab M., Khader A. T. & Al-Betar M. A.* (2017) *Applied Soft Computing*, **61**:1041–1059.
[14] *Adarsh B. R., Raghunathan T., Jayabarathi T. & Yang X. S.* (2016) *Energy*, **96**:666–675.
[15] *Brezočnik L., Fister I. & Podgorelec V.* (2019) *Lecture Notes in Networks and Systems*, **76**: 298–309.
[16] *Ilin D. Y., Gusev A. A.* (2019) Provedenie vosproizvodimyh eksperimentov po ocenke effektivnosti raboty komponentov programmnogo obespecheniya. In *Proc. conf. ART 2019*. P. 54–56. [In Rus]
[17] *Ilin D. Y., Nikulchev E. V., Kolyasnikov P. V.* (2018) *Sovremennyye informatsionnyye tekhnologii i IT-obrazovaniye*, 14(2):344–354. [In Rus]
[18] *Ilin D. Y., Gusev A. A., Nikulchev E. V.* (2019) *Prikaspiyskiy zhurnal: upravleniye i vysokiye tekhnologii*, (2):113–125. [In Rus]