УДК 004.051+004.421.4

ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ДОСЛІДЖЕННЯ ЧАСОВОЇ ТА ФУНКЦІОНАЛЬНОЇ ЕФЕКТИВНОСТІ БІОНІЧНИХ АЛГОРИТМІВ РОЗВ'ЯЗКУ ЕКСТРЕМАЛЬНИХ ЗАДАЧ

В. І. Шинкаренко, П. В. Ільченко, Г. В. Забула

Розроблене інструментальне середовище для визначення часової та функціональної ефективності алгоритмів. Передбачені можливості дослідження ефективності алгоритмів на множині особливих «незручних» функцій, яку можливо змінювати та доповнювати. Виконані комп'ютерні експерименти з визначенням теоретичних засад, підготовчих заходів, реалізацією та аналізом отриманих результатів. Отримані залежності часової та функціональної ефективності зграйного алгоритму від кількості параметрів функцій, глобальний екстремум яких визначається, та параметрів зграйного алгоритму: розміру популяцій та кількості епох. У розробленому середовищі передбачена можливість дослідження інших біонічних алгоритмів.

Ключові слова: часова ефективність, функціональна ефективність, біонічний алгоритм, комп'ютерний експеримент.

Разработана инструментальная среда для определения временной и функциональной эффективности алгоритмов. Предусмотрены возможности исследования эффективности алгоритмов на множестве особых «неудобных» функций, которое можно изменять и дополнять. Выполнены компьютерные эксперименты, включая определение теоретических основ, подготовку, реализацию и анализ результатов. Получены зависимости временной и функциональной эффективности роевого алгоритма от количества параметров функций, глобальный экстремум которых определяется, и параметров роевого алгоритма: размера популяций и количества эпох. В разработанной среде предусмотрена возможность исследования других бионических алгоритмов.

Ключевые слова: временная эффективность, функциональная эффективность, бионический алгоритм, компьютерный эксперимент.

An instrumental environment for determining the time and functional efficiency of algorithms has been developed. There are abilities of studying the effectiveness of algorithms on a set of special "uncomfortable" functions, which can be changed and implemented. Computer experiments were carried out, including the definition of theoretical foundations, preparation, implementation and analysis of the results. The dependency of the time and functional efficiency of the rouge algorithm on the number of parameters of functions whose global extremum is determined, and the parameters of the roaming algorithm: population size and number of epochs are obtained. In the developed environment, it is possible to study other bionic algorithms.

Keywords: time efficiency, functional efficiency, bionic algorithm, computer experiment.

Вступ

Існує велика кількість стохастичних алгоритмів, які використовують соціальні та біологічні ідеї. Одним з найбільш вивчених є зграйний алгоритм або Particle Swarm Optimization (PSO), розроблений Кеннеді и Еберхартом в 1995 році [1]. Ідея даного методу взята з соціальної поведінки деяких видів тварин, наприклад, зграй птахів, косяків риб або стада копитних.

Дослідження показали ефективність алгоритму та доцільність його застосування при рішенні задач як безумовної, так і умовної оптимізації функцій дійсних змінних. Постійно пропонуються нові варіанти алгоритму для покращення продуктивності методу або для розширення кола вирішуваних задач (наприклад, одна з перших модифікацій була пов'язана з ідеєю рішення однокритеріальних задач безумовної оптимізації з бінарними змінними за допомогою алгоритму PSO).

Крім PSO існують й інші алгоритми, які імітують поведінку окремих видів тварин. Найбільший інтерес з останніх розробок представляють наступні біонічні алгоритми: алгоритм пошуку зозуль (Cuckoo Search Algorithm, CSA) [2], алгоритм кажанів (Bat Algorithm, BA) [3], алгоритм світлячків (Firefly Algorithm, FFA) [4] та алгоритм зграї вовків (Wolf Pack Search, WPS) [5]. Кожен з зазначених алгоритмів імітує деяку характеристику певного виду тварин: CSA – спосіб відкладання яєць зозулями; BA – ехолокацію кажанів, FFA – випромінювання від світлячків, WPS – процес полювання зграї вовків.

Зграйний алгоритм – метод чисельної оптимізації, який базується на моделюванні поведінки популяції часток в просторі оптимізації, для використання якого не потрібно знати точного градієнту функції оптимізації [1]. Даний метод приваблює простотою реалізації, він може використовуватися для рішення багатьох задач, включаючи навчання нейронних мереж.

Зараз існують різноманітні модифікації зграєвого алгоритму оптимізації, розроблені для рішення різноманітних завдань оптимізації. Одна з перших модифікацій була запропонована Кеннеді та Еберхартом в 1997 році для рішення задач однокритеріальної безумовної оптимізації з бінарними змінними [6]. В 1998 році Ши та Еберхарт опублікували роботу, в якій описувалась методика автоматизованого налаштування параметрів алгоритму [7]. В 1999 році в статті Кеннеді [8] оглядався вплив обраної топології на результат роботи алгоритму при вирішенні тої чи іншої задачі оптимізації. В тому ж році була вперше запропонована модифікація алгоритму для рішення задач багатокритеріальної оптимізації [9]. В наступні роки були розроблені різноманітні методи на основі зграйного алгоритму, зокрема для навчання нейронних мереж.

У представленій роботі для оцінки якості біонічних алгоритмів були запропоновані показники функціональної [10] та часової ефективності [11].

Функціональні можливості розробленого інструментального середовища

Представлений у даній роботі інструментальний засіб призначений для дослідження будь-якого стохастичного алгоритму пошуку екстремуму функції. Такий алгоритм додається до вже існуючих у системі алгоритмів у призначеному для цього вікні. Алгоритм має бути представленій на мові Java та попередньо відлагоджений. Для опрацювання інструментальних засобів у систему доданий алгоритм PSO.

Для визначення часової та функціональної ефективності має бути сформована представницька множина функцій, подібних до тих, пошук екстремумів яких потребується. Наразі у системі маємо функції, запропоновані в [12]: сфери, гіпер-еліпсоїду, Растригіна, Розенброка, Аклі та Гріванка. Надана можливість додавання нових функцій. На рис. 1 наведена вкладка, яка дозволяє додавати, редагувати та видаляти функції для подальшого аналізу. Для цього слід додати реалізацію функції на мові Java. Саме вона і буде використовуватися у процесі аналізу з оптимізації. Додатково можна задати математичне представлення, що дозволить інструменту знайти глобальний екстремум та побудувати графіки функції. При цьому функціональна ефективність буде визначатись у порівнянні знайденого екстремуму з відомим точним значенням.

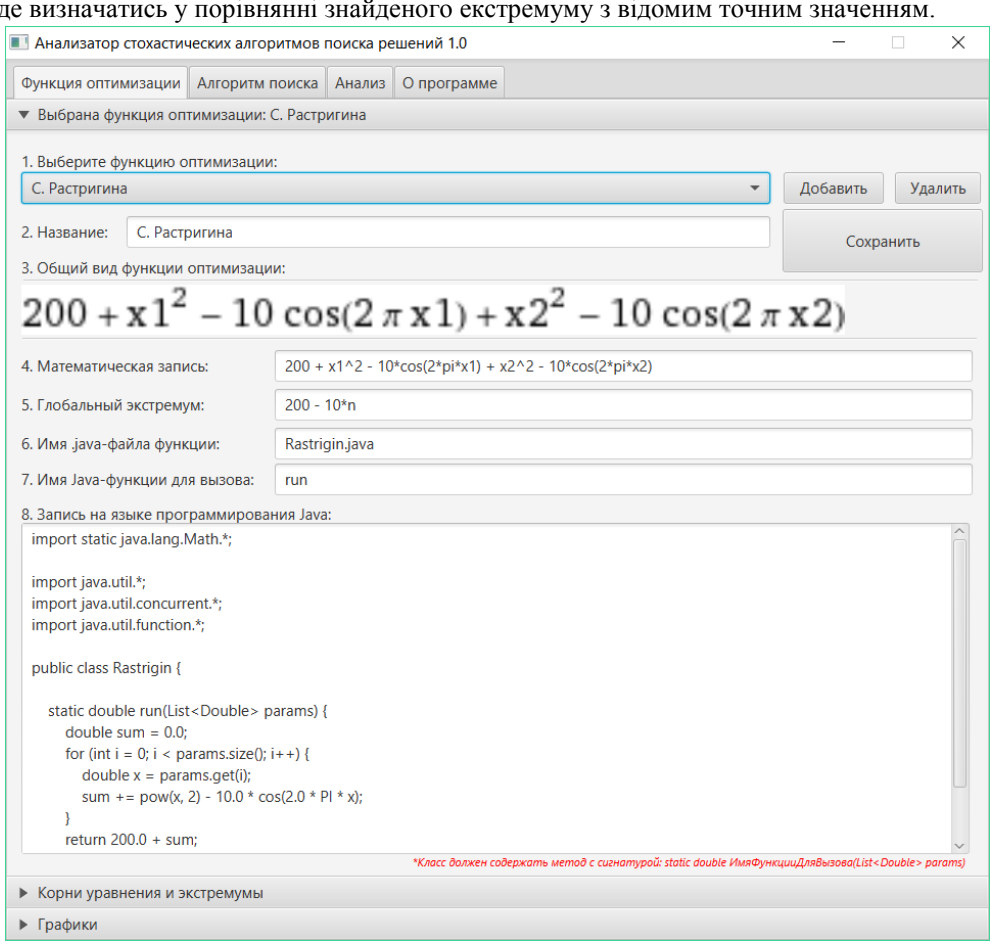

Рис. 1. Вкладка для завдання функції, екстремум якої визначається

Далі (рис. 2) наведений приклад побудови графіку функції Растригіна.

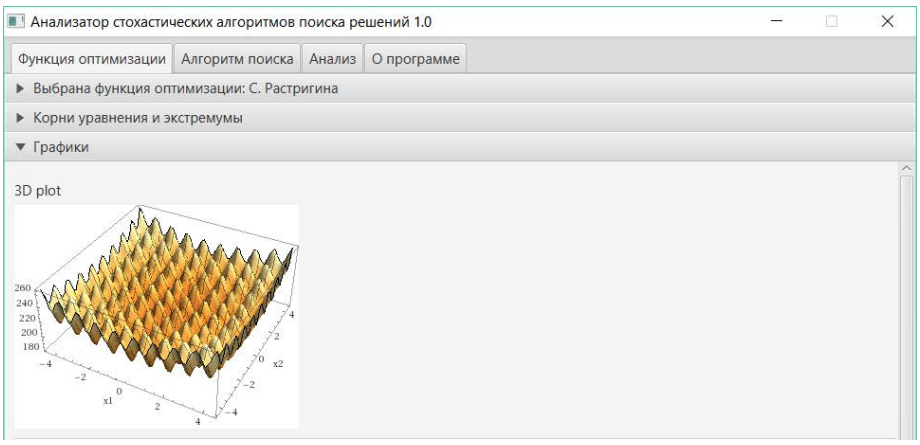

Рис. 2. Функції Растригіна

Користувач має можливість налаштування експерименту (верхня частина вкладники на рис. 3): обрати біонічний алгоритм, що вже є у системі; алгоритми, екстремуми яких знаходяться цим біонічним алгоритмом та показники ефективності.

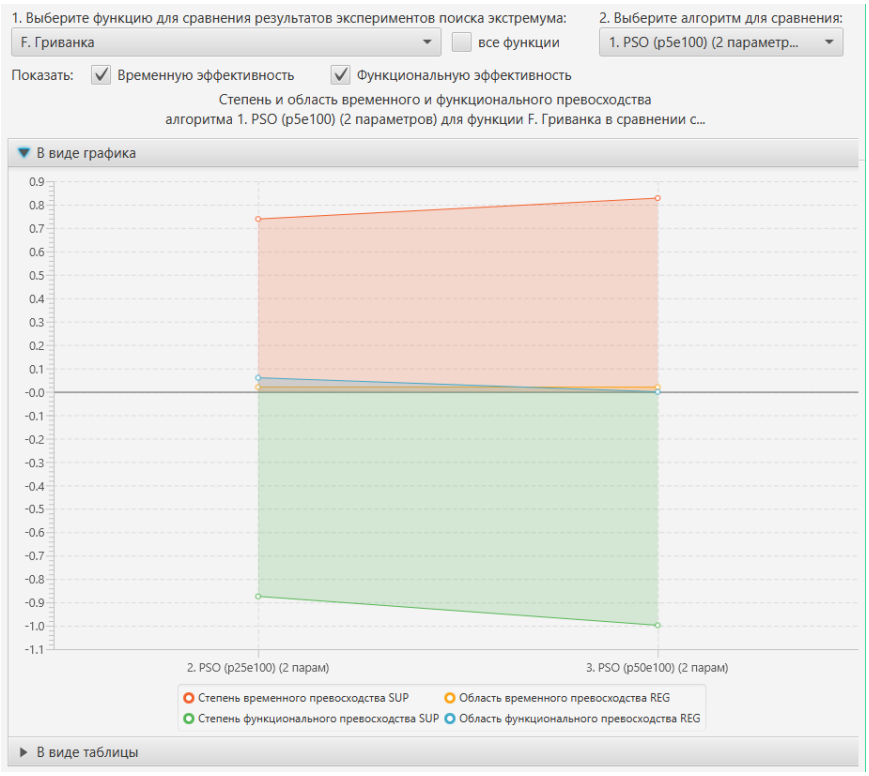

Рис.3. Настройка експерименту та порівняльні графіки

У результаті маємо порівняльні графіки та таблиці (приклад останніх на рис. 4)

Алгоритм	SUP time		REG time		SUP func		REG func	
1. PSO (p5e100) (2 парам)	-0.739	±0.078	0.980	±0.019	0.874	±0.027	0.940	±0.023
3. PSO (p50e100) (2 пар...	0.317	±0.211	0.150	±0.175	-0.529	±0.205	0.230	±0.133

Рис. 4. Приклад табличного представлення результатів

Часова ефективність біонічних алгоритмів

Одна із суттєвих особливостей стохастичних біонічних алгоритмів – потреба в значних часових ресурсах. Виконавцем природних біонічних алгоритмів є значна кількість особин. У комп'ютерній реалізації маємо один обчислювальний механізм та значно складнішу задачу пошуку. Тому й виникає задача оцінки можливостей алгоритмів за часовими показниками.

При наявності одного алгоритму вирішення деякої екстремальної задачі можна обмежитись оцінкою обчислювальної складності алгоритмів. Однак, знаходження таких оцінок є досить нетривіальною задачею і вони не враховують значного впливу програмно-апаратного середовища виконання алгоритмів.

Якщо існує декілька функціонально еквівалентних алгоритмів [11], то їх часову ефективність необхідно визначати за конкретними реалізаціями.

Час виконання алгоритму визначається функцією [11]:

$$E = f(v,t,x,\psi,r), \tag{1}$$

де v, t та x – відповідно об'єм, тип та значення вхідних даних; ψ – програмне середовище функціонування та створення виконуваного файлу; r – архітектура ЕОМ.

При наявності декількох функціонально еквівалентних алгоритмів слід порівнювати функції:

$$E_1 = f_1(v,t,x,\psi,r), E_2 = f_2(v,t,x,\psi,r), \ldots, E_n = f_n(v,t,x,\psi,r) \tag{2}$$

Для порівняння часової ефективності двох алгоритмів необхідно визначити[11]:

- наскільки в середньому один з алгоритмів перевершує інший;
- чи в усій досліджуваній області спостерігається перевершення одного алгоритму, якщо ні – то як співвідносяться розміри областей переваги кожного алгоритму.

Нехай V^* – множина можливих значень об'єму даних, T^* – множина можливих типів даних, X^* – множина можливих значень даних, Ψ^* – множина можливих програмних середовищ, R^* – множина архітектур ЕВМ, на яких можлива реалізація алгоритмів. Тоді для порівняння часової ефективності альтернативних обчислювальних

алгоритмів необхідно визначити ступінь переваги одного (*i*-го) алгоритму над іншим (*j*-им) на обмежених множинах, які є підмножинами вказаних вище множин $\overline{V} \in V^*$, $\overline{T} \in T^*$, $\overline{X} \in X^*$, $\overline{\Psi} \in \Psi^*$, $\overline{R} \in R^*$ визначається як:

$$SUP_{ij} \mid \overline{V},\overline{T},\overline{X},\overline{\Psi},\overline{R} = \frac{1}{N}\sum_{v\in\overline{V}}\sum_{t\in\overline{T}}\sum_{x\in\overline{X}}\sum_{\psi\in\overline{\Psi}}\sum_{r\in\overline{R}} g_{ij}(v,t,x,\psi,r),\ g_{ij}(v,t,x,\psi,r) = \frac{f_j(v,t,x,\psi,r) - f_i(v,t,x,\psi,r)}{\max(f_i(v,t,x,\psi,r),\ f_j(v,t,x,\psi,r))}, \quad (3)$$

де *N* – загальна кількість доданків, g_{ij} – ступінь переваги *i*-го алгоритму над *j*-им при деякому виконанні, f_i – час виконання *i*-го алгоритму.

Визначається також [11] область переваги одного (*i*-го) алгоритму над іншим (*j*-им):

$$REG_{ij} \mid \overline{V},\overline{T},\overline{X},\overline{\Psi},\overline{R} = \frac{1}{N}\sum_{v\in\overline{V}}\sum_{t\in\overline{T}}\sum_{x\in\overline{X}}\sum_{\psi\in\overline{\Psi}}\sum_{r\in\overline{R}} h_{ij}(v,t,x,\psi,r) \quad (4)$$

де h_{ij} – ознака переваги *i*-го алгоритму над *j*-им в точці

$$h_{ij}(v,t,x,\psi,r) = sign(f_i(v,t,x,\psi,r) - f_j(v,t,x,\psi,r)),\ sign(x) = \begin{cases} 1, \text{якщо } x > 0 \\ 0, \text{якщо } x \le 0 \end{cases}. \quad (5)$$

Слід зазначити, що множини $\overline{V},\overline{T},\overline{X},\overline{\Psi},\overline{R}$ – дискретні. Виходячи з необмеженої кількості можливих вхідних даних X^*, значення N є настільки великим, що практичне обчислення запропонованих показників не є можливим.

Оцінка S-показника. S-показник є статистичною оцінкою $SUP_{ij} \mid \overline{V},\overline{T},\overline{X},\overline{\Psi},\overline{R}$. За [11] S-оцінку ступені переваги *i*-го алгоритму над *j*-им визначимо як (%):

$$S_{ij} \mid \overline{V},\overline{T},\overline{X},\overline{\Psi},\overline{T},\overline{R} = \Theta_S \times 100 = \frac{1}{N}\sum_{k=1}^{N} g_{ijk}(v,t,x,\psi,r) \times 100, \quad (6)$$

де g_{ijk} – ступінь переваги *i*-го алгоритму над *j*-им (3) при *k*-му виконанні алгоритмів з випадковими значеннями v,t,x,ψ,r із області $\Omega = \overline{V} \times \overline{T} \times \overline{X} \times \overline{\Psi} \times \overline{R}$.

Там же [11] надане визначення довірчого інтервалу S-показника при коефіцієнті довіри β:

$$|\Theta_s - I| < t_{2,\beta}\sqrt{\frac{1}{6}\sum_{k=1}^{3}(\zeta_k - \Theta_S)^2}, \quad (7)$$

де I – точне значення S-показника, Θ_s – значення визначене за (6), ζ_k – значення S-показника обчислене для кожної з трьох частин $N/3$ вимірювань, $t_{2,\beta}$ – табличне значення розподілу Стьюдента при коефіцієнті довіри β.

Оцінка R-показника. Оскільки вирази для обчислення SUP_{ij} та REG_{ij} в (3) та (4) відрізняються тільки функцією під знаком суми, R-показник обчислюється аналогічно S-показнику. Випадковим чином обирається $\overline{N}$ точок в області $\Omega = \overline{V} \times \overline{T} \times \overline{X} \times \overline{\Psi} \times \overline{R}$. Доцільно взяти ті самі точки, що і для обчислення S-показника.

Використовуючи описану вище методику й позначивши, обчислюємо R-показник (%):

$$R_{ij} \mid \overline{V},\overline{T},\overline{X},\overline{\Psi},\overline{T},\overline{R} = \Theta_R \times 100 = \frac{1}{N}\sum_{k=1}^{\overline{N}} h_{ijk}(v,t,x,\psi,r) \times 100 \quad (8)$$

з довірчим інтервалом при коефіцієнті довіри β:

$$|\Theta_R - I| < t_{2,\beta}\sqrt{\frac{1}{6}\sum_{k=1}^{3}(\xi_k - \Theta_R)^2} \quad (9)$$

де ξ_k – значення R-показника обчислене для кожної з трьох частин $N/3$ вимірювань.

Таким чином, для визначення S- та R-показників необхідно:

- виконати послідовність експериментів з фіксацією часу виконання алгоритмів та знайденого значення екстремуму;
- відповідно до (6) обчислити S-показник та його (7) довірчий інтервал;
- відповідно до (8) обчислити R-показник та його (9) довірчий інтервал.

Функціональна ефективність біонічних алгоритмів

Однією з основних властивостей алгоритму є його функціональність. Ця властивість полягає в тому, що алгоритм реалізує деяку, в математичному розумінні функцію

$$y = f(x) \quad (10)$$

де $x \in X$, $y \in Y$ – вхідні та вихідні дані алгоритмів, елементи деяких визначених множин.

Дана властивість є однією з фундаментальних властивостей алгоритму разом з дискретністю, результативністю, доступністю, масовістю, послідовністю та однозначністю [13].

У [10] звертається увага та цю властивість у зв'язку з її особливістю стосовно нечітко специфікованих алгоритмів. Ця особливість в тому, що розроблений відповідно з нечіткою специфікацією алгоритм може в деяких випадках і в деякій ступені задовольняти чи не задовольняти вимогам користувача.

Під функціональною ефективністю будемо розуміти експлуатаційну характеристику алгоритму, яка відображає ступінь відповідності алгоритму потребам користувача [10] за своєю функціональністю.

Будь-який чітко специфікований алгоритм реалізує функцію, результати якої завжди повинні задовольняти потребам користувача. Можна вважати, що такий алгоритм має максимальну функціональну ефективність.

Нехай на множині Y задана функція

$$\rho = \varphi(y) \tag{11}$$

де ρ – оцінка якості отриманих алгоритмом результатів.

Для алгоритмів розв'язку екстремальних задач $\rho = y$, тобто в задачах мінімізації , чим менше значення функції $y = f(x)$ буде знайдене деяким реалізацією деякого алгоритму, тим він кращий за функціональною ефективністю.

Аналогічно з попереднім пунктом, пропонується два показники функціональної ефективності алгоритмів:

- ступень переваги одного (*i*-го) алгоритму над іншим (*j*-им) на обмеженій множині $\overline{X} \subseteq X$ визначається наступним чином:

$$S_{ij} \mid \overline{X} = \frac{1}{N} \sum_{x \in \overline{X}} \frac{\rho(f_i(x)) - \rho(f_j(x))}{\max(\rho(f_i(x)), \rho(f_j(x)))} \tag{12}$$

де N – загальна кількість реалізацій;

- область переваги:

$$R_{ij} \mid \overline{X} = \frac{1}{N} \sum_{x \in \overline{X}} sign(\rho(f_i(x)) - \rho(f_j(x))) \tag{13}$$

S- та R-показники є статистичною оцінкою показників функціональної ефективності SUP_{ij} та REG_{ij}, які визначаються аналогічно S- та R-показникам часової ефективності.

Підготовка до експериментів

Вибір та підготовка біонічного алгоритму. В даній роботі досліджується зграйний алгоритм. Він є реалізацією методу чисельної оптимізації, який базується на моделюванні поведінки популяції часток в просторі оптимізації.

Нехай $f(x)$ функція цілі, яку необхідно мінімізувати, причому $x = (x_1, ..., x_d, ..., x_D)$. Робота алгоритму починається зі створення популяції часток (тобто множини потенційних рішень x_i, $i = [1, N]$, де N – розмір популяції) випадковим чином. Частки являють собою вектор координат точки в просторі оптимізації (дійсних чисел) $x_i = (x_{i1}, \ldots, x_{id}, \ldots, x_{iD})$, $i = [1, N]$. Кожна частка рухається по поверхні графіку функції з якоюсь швидкістю $v_i = (v_{i1}, \ldots, v_{id}, \ldots, v_{iD})$, $i = [1, N]$. Частки змінюють свою швидкість та координати, ґрунтуючись на власному досвіді та досвіді інших часток. Таким чином, моделюється багатоагентна система, де агенти-частки рухаються до оптимальних рішень, обмінюючись при цьому інформацією з сусідами. Швидкість та координати часток оновлюються за такими формулами:

$$v_{id}^{t+1} = \omega \cdot v_{id}^{t} + c_1 \cdot rand() \cdot (p_{id} - x_{id}^{t}) + c_2 \cdot rand() \cdot (p_{gd} - x_{id}^{t}),\ x_{id}^{t+1} = v_{id}^{t+1} + x_{id}^{t} \tag{14}$$

де v_{id}^{t+1}, v_{id}^{t} – швидкості частки на *(t+1)* та *t*-ітераціях відповідно; x_{id}^{t+1}, x_{id}^{t} – координати часток на $t+1$ та t-ітераціях відповідно; p_{id} – краща позиція, знайдена i-ою часткою за t попередніх поколінь; p_{gd} – краща позиція, знайдена всім роєм за весь час роботи алгоритму; $rand()$ – випадкові числа з проміжку $[0,1]$; c_1, c_2 – коефіцієнти навчання на проміжку $[0, 2]$; ω – інерційна вага.

Константу c_1 називають когнітивним (пізнавальним) параметром, який дозволяє враховувати «власний досвід» (історію) частинки. Константу c_2 називають соціальним параметром. Вона дозволяє частинці враховувати досвід усього рою. Таким чином, c_2 керує впливом глобального кращого положення, а c_1 керує дією власного кращого положення на швидкість кожної частинки.

Роботу алгоритму також визначає топологія сусідства частинок, а саме яким чином вибирається кращий індивід для кожної частинки. Найбільш відомими є наступні топології [14]: кільце, клики. двовимірний тор, кластер.

Отже, ідея алгоритму полягає в тому, що частки, які спочатку рівномірно розподілені по поверхні функції, з плином часу (від покоління до покоління) починають групуватися («збиватися в зграї») поряд з локальними мінімумами, при чому найбільша зграя збирається поряд з глобальним оптимумом. При чому майже завжди існують частки, які знаходяться у стороні від таких зграй, а також частки, які виходять за межі допустимої області.

Аналіз та опис використаного програмно-апаратного середовища. Вимірювати продуктивність на віртуальній машині Java (втім, як і в будь-якому іншому керованому середовищі часу виконання) за своєю природою більш складно, ніж при роботі з кодом, що працює в некерованому вигляді. Причина в тому, що програмісти, які пишуть на C / C ++, практично все роблять самі. Операційна система надає лише мінімальний набір служб, наприклад, рудиментарні планування потоків.

Вся суть керованої системи полягає в тому, що навколишнє середовище часу виконання отримує можливість придбати певний контроль над усією екосистемою, так що розробнику не доводиться самому дбати про всі деталі. Завдяки цьому підвищується загальна продуктивність праці програміста, але від контролю над системою в якійсь мірі доводиться відмовитися. Єдина альтернатива – відмовитись замість цього від усіх переваг, які існують в керованому середовищі часу виконання, що практично завжди обертається серйознішими незручностями, ніж необхідність додаткової оптимізації продуктивності.

Ось деякі важливі функції платформи, через які ускладняється процес вимірювання оптимізація продуктивності: диспетчеризація потоків; прибирання сміття; динамічна компіляція.

Між цими функціями можуть виникати досить тонкі взаємодії. Наприклад, система компіляції використовує таймери, щоб вирішувати, які методи компілювати. Це означає, що на множину методів-кандидатів для компіляції можуть впливати такі чинники, як диспетчеризація потоків і прибирання сміття. Набір компільованих методів може відрізнятися від запуску до запуску.

Для проведення експериментів використовувалося апаратне середовище наступної конфігурації: процесор Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz (тактова частота 2.6 – 3.0 ГГц); 16 Гб оперативної пам'яті типу DDR4 та програмне середовище: Microsoft Windows 10 Версія 1709; Java SE Runtime Environment build 9.0.1+11; Java HotSpot 64-Bit Server VM build 9.0.1+11, mixed mode.

Опис підходу для визначення часу роботи алгоритму. У Java є два основні методи для доступу до часу: System.currentTimeMillis() і System.nanoTime(). Другий використовується для вимірювання часу з точністю вище мілісекунди. Значення System.nanoTime() відносне відповідно певного моменту часу, в більшості сучасних операційних системи цей метод отримає інформацію від лічильника міток реального часу (Time Stamp Counter, TSC). Це означає, що цей метод можна використовувати для запису тривалості виконання.

Для забезпечення максимальної точності розрахунків, в розробленому інструментальному програмному забезпеченні для аналізу ефективності алгоритмів, при вимірюванні тривалості виконання методу, використовувався системний метод System.nanoTime().

Вплив «розігріву» JIT компілятору на результати вимірювань. Платформу Java краще всього представляти як платформу з динамічною компіляцією. Це означає, що класи додатків піддаються подальшій компіляції під час виконання, в результаті чого перетворюються в машинний код.

Такий процес називається динамічною (just-in-time) компіляцією або джейнтингом. Зазвичай такій компіляції піддається тільки один метод в одиницю часу. Під «розігрівом» компілятору будемо розуміти попередній виклик вимірюваного методу 10000 разів. За результатами досліджень встановлено, що швидкість виклику скомпільованого методу може бути у 12 разів більшою, ніж інтерпретованого методу.

В подальшому, перед проведенням вимірювань, будемо використовувати «розігрів» JIT компілятору.

Для задання кількості раз виклику методу перед тим як він буде розглянутий як кандидат для динамічної компіляції, можна використовувати перемикач JVM "-XX:CompileThreshold=10000", де 10000 – кількість викликів методу.

Валідація інструментального середовища для проведення експериментальних досліджень

Випробування інструментальних засобів виконане на основі декількох експериментальних досліджень. Так, виконані дослідження впливу кількості часток, епох, параметрів функції на показники часової та функціональної ефективності алгоритму рою часток (Particle Swarm Optimization, PSO) для пошуку глобального мінімуму функцій. Позначимо PSOp*i*e*j* – алгоритм рою часток, де *i* – кількість часток, *j* – кількість епох. Наприклад алгоритм PSOp500e10 використовує кількість часток 500, кількість епох – 10.

Для експериментів будемо використовувати «важкі» функції оптимізації, запропоновані в [12]: сфери, гіпер-еліпсоїду, Растригіна, Розенброка, Аклі та Гріванка.

На рис. 5 и 6 наведені часової ефективності зі зростанням кількості епох та параметрів, відповідно.

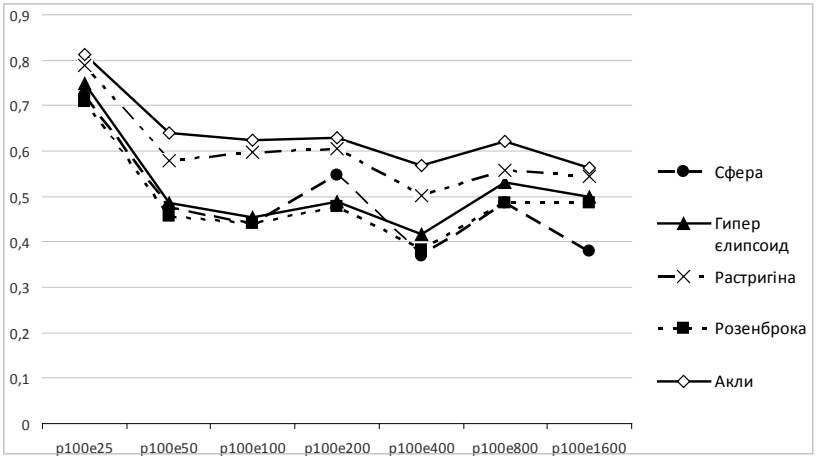

Рис. 5. S-оцінка часової ефективності алгоритмів зі зростанням кількості епох

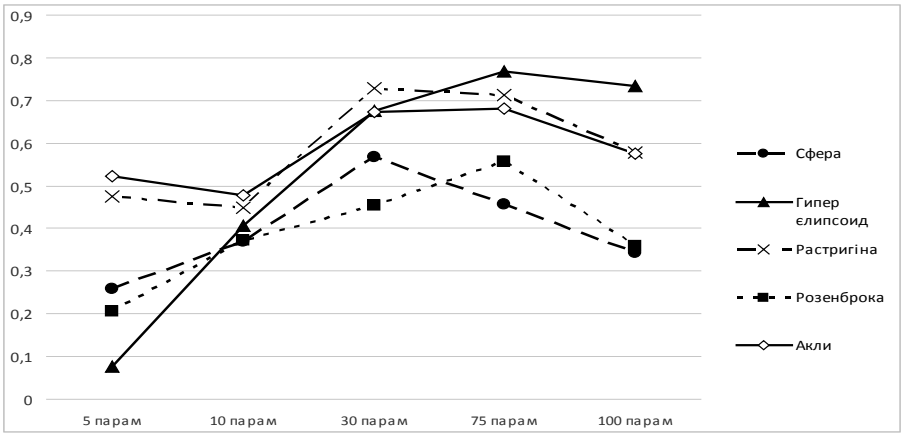

Рис. 6. S-оцінка часової ефективності алгоритмів зі зростанням кількості параметрів

Отримані значення часової та функціональної ефективності за пошуком екстремумів вище названих функцій усіх разом. На рис.7 представлені значення часової ефективності алгоритмів PSOp*i*e100 по відношенню до PSOp(*i*-1)e100, починаючи з PSOp5e100. Кількість епох фіксована, змінюється кількість часток у рої. Перше значення 0,74 отримане при збільшенні часток у 10 разів, при цьому час виконання збільшився приблизно у 4 рази. При подальшому збільшенні кількості часток у два рази приблизно у два рази збільшується час пошуку екстремумів (у середньому для всіх функцій). На рис. 7 відображені також довірчі інтервали.

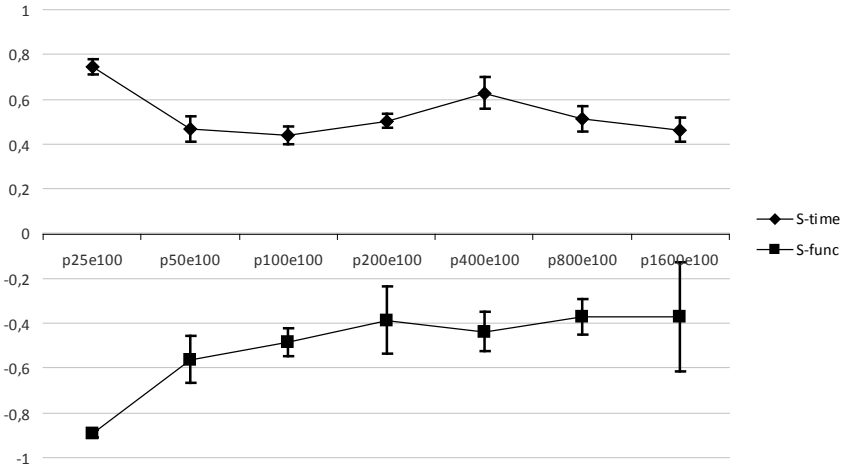

Рис. 7. Часова та функціональна ефективність алгоритмів PSOp*i*e100 з довірчими інтервалами

Функціональна ефективність алгоритмів PSOp*i*e100 виконана в порівнянні з алгоритмом, який знаходить точний мінімум. Така можливість є при дослідженнях, якщо точний мінімум відомий. Ефективність алгоритмів PSOp*i*e100 суттєво зростає при збільшенні кількості часток рою.

Всі отримані результати відповідають очікуваним, та відомим тенденціям обчислювальної складності. За проведеними експериментами та аналізом отриманих результатів щодо достатньо складних функцій можна стверджувати, що інструментальний засіб забезпечує адекватні оцінки часової та функціональної ефективності при розв'язку задач пошуку екстремумів функцій.

Висновки

Наразі спостерігається бурхливий розвиток перенесення алгоритмів природних систем у програмовані системи. Запропонований інструментальний засіб може бути корисним для розробників біонічних алгоритмів для виявлення ефективних складових алгоритмічних та параметричних рішень.

Прикладне значення має використання цього інструменту для розробки програмних засобів, зокрема систем штучного інтелекту, оптимізаційного напрямку. Тут корисна можливість налаштування параметрів біонічних алгоритмів за показниками функціональної та часової ефективності.

Література

1. Kennedy J. Particle Swarm Optimization. In proceedings of IEEE International Conference on Neural Networks / J. Kennedy, R. Eberhart. – 1995. – pp.1942-1948.
2. Yang X. S. Cuckoo search via Levy flights. In proceedings of World Congress of Nature & Biologically Inspired Computing / X. S. Yang, S. Deb. – 2009. – pp. 210-214.
3. Yang X. S. A new metaheuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization / X. S. Yang. – Springer. – 2010. – pp. 65-74
4. Yang X. S. Firefly algorithms for multimodal optimization. In proceedings of the 5th Symposium on Stochastic Algorithms, Foundations and Applications / X. S. Yang. – 2009. – pp. 169-178.

5. Yang X. S. Algorithm of Marriage in Honey Bees Optimization Based on the Wolf Pack Search. In proceedings of the International Conference of Intelligent Pervasive Computing / X. S. Yang, J. Chen. – 2007. – pp. 462-467
6. Kennedy J. A discrete binary version of the particle swarm algorithm. In proceedings of the International Conference on Computational Cybernetics and Simulation / J. Kennedy, R. C. Eberhart. – 1997. – 4104-4108 pp.
7. Shi Y. Parameter selection in particle swarm optimization. In proceedings of Evolutionary Programming VII (EP98) / Y.R. Shi, C. Eberhart. – 1998. – pp. 591-600.
8. Kennedy J. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In proceedings of IEEE Congress on Evolutionary Computation (CEC 1999) / J. Kennedy. – 1999. – pp. 1931-1938.
9. Moore J. Application of particle swarm to multiobjective optimization. Department of Computer Science and Software Engineering / J. Moore, R. Chapman – Auburn University. – 1999. – 840 p.
10. Шинкаренко В. И. Функциональная эффективность нечетко специфицированных алгоритмов / В. И. Шинкаренко // Проблеми програмування. – 2006. – № 1. – С. 24-33.
11. Шинкаренко В. И. Особенности оценки эффективности вычислительных алгоритмов. Проблемы программирования. № 1-2 / В. И. Шинкаренко. – 2001. – 23-29 с.
12. Molga M. Test functions for optimization need / M. Molga and C. Smutnicki. – 3 kwietnia. – 2005. – 43 p.
13. Цейтлин Г. Е. Введение в алгоритмику / Г. Е. Цейтлин. – Киев: Сфера. – 1998. – 310 с.
14. Madnes R. The Fully Informed Particle Swarm: Simpler, Maybe Better. IEEE Transactions of Evelutionary Computation / R. Madnes, J. Kennedy, J. Neves. – 2014. – pp. 204-210.

References

1. Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization, proceedings of IEEE International Conference on neural networks (ICNN'95) in.
2. Yang, X. S., & Deb, S. (2009, December). Cuckoo search via Lévy flights. In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on (pp. 210-214). IEEE.
3. Yang, X. S. A new metaheuristic bat-inspired algorithm Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) 2010 284 Berlin. Germany Springer, 65-74.
4. Yang, X. S. (2009, October). Firefly algorithms for multimodal optimization. In International symposium on stochastic algorithms (pp. 169-178). Springer, Berlin, Heidelberg.
5. Yang, C., Tu, X., & Chen, J. (2007, October). Algorithm of marriage in honey bees optimization based on the wolf pack search. In Intelligent Pervasive Computing, 2007. IPC. The 2007 International Conference on (pp. 462-467). IEEE.
6. Kennedy, J., & Eberhart, R. C. (1997, October). A discrete binary version of the particle swarm algorithm. In Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on (Vol. 5, pp. 4104-4108). IEEE.
7. Shi, Y., & Eberhart, R. C. (1998, March). Parameter selection in particle swarm optimization. In International conference on evolutionary programming (pp. 591-600). Springer, Berlin, Heidelberg.
8. Kennedy, J. (1999). Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on (Vol. 3, pp. 1931-1938). IEEE.
9. Moore, J., & Chapman, R. (1999). Application of particle swarm to multiobjective optimization. Department of Computer Science and Software Engineering, Auburn University, 32.
10. Shynkarenko, V. I. (2006). Functional effectiveness of algorithms with fuzzy specification. Problems in programming, (1), 24-33
11. Shynkarenko, V. I. (2001). Features of an estimation of efficiency of computational algorithms. Problems in programming, (1-2), 23-29.
12. Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs. Test functions for optimization needs, 101.
13. Zeitlin, H. E. (1998). Introduction to algorithmics. K.: Sphera, 310.
14. Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. IEEE transactions on evolutionary computation, 8(3), 204-210.

Про авторів:

Шинкаренко Віктор Іванович, д.т.н., професор, зав. кафедри «Комп'ютерні інформаційні технології»,