УДК 519.7, 519.8
DOI: 10.20535/SRIT.2308-8893.2018.2.07

МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ОПТИМАЛЬНОГО ОБРОБЛЕННЯ ДАНИХ У РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

Г.Г. ЦЕГЕЛИК, Р.П. КРАСНЮК

Анотація. Розглянуто задачі оптимізації використання обчислювальних ресурсів розподіленої інформаційної системи. Виконано математичні постановки оптимізаційних задач та запропоновано ефективні обчислювальні алгоритми побудови розв'язку задач, які ґрунтуються на стратегії «жадібного» вибору та використанні генетичних алгоритмів. Для генетичних алгоритмів побудови розв'язків, близьких до оптимальних, у задачах бінарного та дійсного кодування запропоновано та досліджено обчислювальну ефективність уведення параметрів самонавчання алгоритму, що забезпечує коригування популяцій у напрямі найкращої пристосованості.

Ключові слова: математичне моделювання, оптимізація, розподілені інформаційні системи, «жадібний» алгоритм, генетичний алгоритм.

ВСТУП

Ефективне використання інформаційних систем потребує розв'язання сукупності задач керування ресурсами системи та розподілу вхідних задач між вузлами. Метою керування є визначення ефективності розподілу задач і ресурсів з використанням різних критеріїв. Процес керування здійснюється динамічно (у процесі надходження задач та зміни конфігурації розподіленої інформаційної системи) спеціальною програмою — балансиром навантаження системи. Балансир навантаження керує потоком задач, що надходять у систему, розбиває їх на підзадачі та розподіляє між вузлами системи. Крім того, оброблення задач у вузлах інформаційної системи може вимагати додаткових ресурсів (зміни ємності запам'ятовувальних пристроїв, реплікації баз даних, використання спеціалізованих програмних комплексів тощо). Балансир навантаження має оптимізувати загальний час виконання завдань, а також мінімізувати вартість використання обчислювальних ресурсів.

Зазвичай керування ресурсами інформаційної системи та потоком задач зводиться до багатокритеріальних задач оптимізації, розв'язки яких доволі складно отримати із застосуванням точних методів, і для отримання результатів з достатньою для практики точністю застосовують низку евристичних підходів, що враховують специфіку предметної галузі, масштабованість

окремих задач у завданнях і таких, що використовують можливості паралельного виконання незалежних задач.

Оскільки натепер не існує універсального за ефективністю методу, який може забезпечити оптимальний розподіл для довільного класу задач та сукупності доступних ресурсів, то для проектування розподіленої інформаційної системи необхідно використовувати у сукупності якомога більше математичних моделей та обчислювальних алгоритмів, що мають бути закріплені за окремими функціональними модулями балансира навантаження. Вибір відповідної моделі або обчислювального алгоритму має ґрунтуватися на динамічній природі та забезпечувати ефективне керування у режимі реального часу.

Мета роботи — дослідити проблему оптимізації використання обчислювальних ресурсів розподіленої інформаційної системи.

Для досягнення поставленої мети потрібно:

- сформулювати та дослідити математичні моделі оптимізації використання обчислювальних ресурсів розподіленої інформаційної системи;
- навести ефективні обчислювальні алгоритми пошуку наближеного розв'язку однокритеріальних та двокритеріальних оптимізаційних задач, що використовують стратегію «жадібного» вибору та генетичні алгоритми;
- для генетичних алгоритмів побудови розв'язків запропонувати та дослідити обчислювальну ефективність уведення параметрів самонавчання алгоритму, що забезпечує коригування популяцій у напрямі найкращої пристосованості.

ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

Для збереження оптимального стану обчислювальної системи використовується балансування навантаження для усіх її ресурсів. Перевага динамічного балансування ресурсами перед статичним полягає в тому, що інформаційна система не повинна мати інформацію про поведінку завдання під час його виконання до його запуску на оброблення. Крім того, за динамічного планування бажаним є резервування ресурсів для отримання деякої впевненості в продуктивності ресурсів. Як наслідок, оптимальною стратегією оброблення завдань за такого сценарію є мінімізація часу виконання вхідних завдань, що складаються з наборів задач.

За динамічних сценаріїв керування за прийняття глобальних рішень щодо планування ресурсів може відповідати як один центральний планувальник, так і декілька розподілених планувальників. Вибір центрального планувальника має перевагу у простоті реалізації, але цей варіант погано масштабується, не є відмовостійким і зазвичай стає вузьким місцем для продуктивності системи. Наприклад, у праці [1] розглядається центральний метапланувальник, який використовує алгоритм зворотного заповнення (*Backfill*) для планування паралельного виконання завдань. У праці [2] досліджено повністю децентралізований, динамічний та ініційований відправником алгоритм планування та балансування навантаження для розподілених обчислювальних середовищ, головною особливістю якого є використання інтелектуальної стратегії пошуку вузлів-партнерів, на які можуть бути перенесені задачі. Якщо вся інформація про стан ресурсів і стан

завдань відома, оптимально поєднати завдання відповідно до ресурсів можна з використанням деякої цільової функції, що забезпечує мінімізацію часу виконання завдань.

Оскільки керування процесами у розподіленій інформаційній системі у загальному випадку є *NP*-повною задачею [3], то довести оптимальність алгоритмів чи зробити певні раціональні припущення щодо оптимальності не завжди можливо. Тому у прикладних дослідженнях розглядаються субоптимальні розв'язки, з яких виокремлюють такі категорії:

- наближені алгоритми, що використовують формальні обчислювальні моделі замість отримання усього простору розв'язків та вибору з нього оптимального варіанта. Ці алгоритми виконують пошук прийнятного результату, близького до оптимального за метрикою, що дозволяє оцінити похибку обчислень;
- евристичні методи, що є класом алгоритмів, які надають найбільш реалістичні припущення про апріорні знання щодо характеристик навантаження системи та процесів виконання завдань. Оцінки результатів, знайдених за цими методами, зазвичай ґрунтуються на числових експериментах над тестовими даними або на моделюванні. Найбільш популярними у дослідженнях є економічні підходи [4–6] та евристики, що ґрунтуються на природних процесах: генетичний алгоритм (*GA – Genetic Algorithm*) [7–9], імітаційний відпал (*SA — Simulated Annealing*) [10, 11], заборонений пошук (*TS — Taboo Search*) [10] та комбінована евристика [12].

ОПТИМІЗАЦІЯ ВИКОРИСТАННЯ ОБЧИСЛЮВАЛЬНИХ РЕСУРСІВ РОЗПОДІЛЕНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Нехай необхідно розподілити m задач серед n комп'ютерів розподіленої інформаційної системи (комп'ютери можуть мати неоднакову потужність). Задається план за часом та продуктивністю кожного комп'ютера системи:

- T_i, $i=\overline{1,m}$ − час активності кожного вузла системи, що є важливою характеристикою розподілених обчислень, які працюють за схемою «оплата за використання»;
- на кожному *j*-му комп'ютері системи має бути виконано не менше C_j задач.

Необхідно скласти такий план роботи вузлів системи, щоб забезпечити мінімальні витрати на оброблення пакета задач, якщо відомі час виконання кожної *j*-ї задачі на *i*-му комп'ютері системи — продуктивності τ_{ij} та вартість одиниці машинного часу p_{ij}, що витрачається на оброблення та транспортування *j*-ї задачі до *i*-го вузла розподіленої системи.

Інакше кажучи, потрібно визначити час роботи *i*-го комп'ютера з оброблення *j*-ї задачі в *i*-му вузлі мережі x_{ij}, щоб забезпечити мінімальні витрати на оброблення пакета задач за дотримання обмежень на час роботи вузлів мережі та задану мінімальну кількість задач C_j, які обробляються у вузлах.

За умовою пакет задач обробляється за заданий час $\max\limits_{i=\overline{1,m}}\{T_i\}$, тому це обмеження можна подати так:

$$\sum_{j=1}^{n} x_{ij} \le T_i , \quad i = \overline{1,m} . \tag{1}$$

Обмеження за заданою мінімальною кількістю задач, які можуть бути оброблені у вузлі, виглядає таким чином:

$$\sum_{i=1}^{m} \tau_{ij} x_{ij} \ge C_j , \quad j = \overline{1,n} . \tag{2}$$

Ураховуючи невід'ємність змінних $x_{ij} \ge 0$, можна сформувати як однокритеріальну оптимізаційну задачу — пошук мінімуму витрат на оброблення пакета підзадач для цільової функції

$$F(\{x_{ij}\}) = \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \to \min , \tag{3}$$

так і двокритеріальну — пошук мінімуму цільової функції (3) з умовою мінімізації загального часу оброблення пакета задач:

$$\max_{i=\overline{1,m}} \{T_i\} . \tag{4}$$

Задачу поставлено так, щоб витратити ввесь відведений час (за потреби мінімізувавши його) роботи комп'ютерів на оброблення пакета задач, але кількість розв'язаних задач на кожному комп'ютері системи не повинна бути меншою за C_j.

У деяких випадках можна послабити умову (2), тобто накласти обмеження на максимально допустиму кількість задач, які можуть бути оброблені у вузлі:

$$\sum_{i=1}^{m} \tau_{ij} x_{ij} \le C_j , \quad j = \overline{1,n} . \tag{5}$$

Оптимізаційні моделі (1)–(5) стосувалися випадку, коли розподілена система розв'язує пакет незалежних задач. Розглянемо ситуацію, коли у розподіленій інформаційній системі має бути розв'язана задача, що складається з *m* різних підзадач, які допускають паралельне виконання у *n* вузлах системи. Через неоднорідність комп'ютерів системи продуктивність виконання *j*-ї підзадачі неоднакова і дорівнює p_{ij}. Кожен *i*-й комп'ютер має максимальний сумарний ресурс часу для оброблення *m* задач, що дорівнює T_i. Потрібно максимізувати розв'язок задач, що за сутністю еквівалентно забезпеченню мінімізації дисбалансу, який виникає через затримання оброблення задач на комп'ютерах системи.

Необхідно визначити витрати часу на оброблення *j*-ї задачі на *i*-му вузлі системи, які не перевищують за сумою часові ресурси *i*-го вузла і забезпечують максимальну кількість оброблених задач.

Нехай x_{ij} — час, необхідний для розв'язання задачі *j* на комп'ютері *i*. Тоді загальна кількість задач, які можуть бути розв'язані на *i*-му комп'ютері системи, становить

$$\sum_{i=1}^{n} p_{ij} x_{ij} , \quad j = \overline{1,m} .$$

Оскільки кожна задача складається з різних підзадач, наявних в одному екземплярі, то кількість підзадач, які може обробити інформаційна система, повинна дорівнювати кількості задач, якщо загальна їх кількість є мінімальною:

$$\sum_{i=1}^{n} p_{ij} x_{ij} \to \min,\ j = \overline{1,m}. \tag{6}$$

Умова задачі (6) установлює обмеження на час, який використовує вузол i. Таким чином, математичну модель можна подати в такому вигляді:

$$\sum_{j=1}^{m} x_{ij} \le \mathrm{T}_i,\ i = \overline{1,n}; \tag{7}$$

$$x_{ij} \ge 0,\ i = \overline{1,n},\ j = \overline{1,m}. \tag{8}$$

Модель (6)–(8) є нелінійною, тому її можна звести до лінійної форми за допомогою перетворення, увівши в розгляд кількість розв'язаних задач

$$C\{(x_{ij})\} = \min\left(\sum_{i=1}^{n} p_{ij} x_{ij},\ j = \overline{1,m}\right), \tag{9}$$

Виразу (9) з математичної точки зору еквівалентне таке формулювання: максимізувати $F\{(x_{ij})\} = C\{(x_{ij})\}$ за обмежень

$$\sum_{i=1}^{n} p_{ij} x_{ij} - C\{(x_{ij})\} \ge 0,\ j = \overline{1,m} \tag{10}$$

та задач (7), (8).

ОПТИМАЛЬНЕ ВИКОРИСТАННЯ КОМП'ЮТЕРІВ РОЗПОДІЛЕНОЇ МЕРЕЖІ ЗА ПАРАЛЕЛЬНОГО РОЗВ'ЯЗАННЯ ЗАДАЧ

Нехай n — кількість доступних вузлів розподіленої інформаційної системи; m — загальна кількість задач, що допускають паралельне виконання у вузлах інформаційної системи; τ_{ij} — час, потрібний для розв'язування *i*-ї задачі у вузлі *j*; T_j — час, протягом якого можна використовувати обчислювальні потужності *j*-го вузла; c_{ij} — затрати на розв'язання *i*-ї задачі в *j*-му вузлі інформаційної системи; $x_{ij} \in \{0,1\}$, де 1 відповідає випадку розв'язання *i*-ї задачі в *j*-му вузлі, а нуль — у противному випадку. Тоді задача оптимального використання обчислювальних потужностей розподіленої інформаційної системи полягає у такому розподілі задач між вузлами системи, щоб забезпечити одночасно мінімальний час розв'язання усіх обчислювальних задач та мінімальні затрати на їх розв'язування.

Математична модель задачі є такою:

$$F\{(x_{ji})\} = \sum_{i=1}^{m}\sum_{j=1}^{n} \tau_{ij} x_{ij} \to \min;\ E\{(x_{jik})\} = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij} \to \min \tag{11}$$

за умов:

$$\sum_{i=1}^{m} \tau_{ij} x_{ij} \le \mathrm{T}_j,\ j = \overline{1,n};\ \sum_{j=1}^{n} x_{ij} = 1,\ i = \overline{1,m};\ x_{ij} \in \{0,1\},\ i = \overline{1,m},\ j = \overline{1,n}. \tag{12}$$

Оскільки розглянуті вище оптимізаційні задачі характеризуються великою розмірністю, то застосовувати точні методи до побудови їх розв'язків недоцільно через значні часові втрати. Задовільні для практичного використання результати можна отримати, застосувавши евристичні алгоритми, зокрема такі, як «жадібні» та генетичні, що дають наближений розв'язок поставленої задачі.

СТРАТЕГІЯ «ЖАДІБНОГО» ВИБОРУ ЗА ОПТИМІЗАЦІЇ ОБЧИСЛЕНЬ У РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ

«Жадібні» алгоритми [13–16] є інтуїтивними евристиками, у яких на кожному кроці приймається найбільш вигідне для цього кроку рішення, без урахування того, що відбувається на наступних кроках пошуку.

Опишемо загальну схему алгоритму для сформульованих вище математичних моделей (1)–(5) і (7)–(10), яка використовує ідею «жадібного» вибору.

Позначимо: $X(x_{ij}^*)$ — множина допустимих планів вихідної задачі, для яких $x_{ij} = x_{ij}^*$; $I = \{i = \overline{1,n},\ j = \overline{1,m}\}$ — множина індексів змінних; I_{00} — множина індексів змінних, яким процедурою «жадібного» вибору присвоєні нові значення.

Алгоритм.

Крок 0. Покладемо $X_{00} = X$, $I_{00} = \varnothing$ і виберемо початковий допустимий план $x^0 = (x_{11}^0, x_{12}^0, ..., x_{nm}^0) \in X_{00}$. Для кожної з $n \times m$ змінних знаходимо межі $x_{ij}^L \le x_{ij} \le x_{ij}^R$ такі, що $x^0 = (x_{11}^0, x_{12}^0, ..., x_{ij}, ..., x_{nm}^0) \in X_{00}$ для всіх x_{ij}, $x_{ij}^L \le x_{ij} \le x_{ij}^R$. Межі нерівностей можуть задаватись наближено.

Нехай сформовано множини I_{kl} та $X_{kl} \in X$.

Крок 1. Якщо $I_{kl} = I$, то всі змінні набули нових значення. Тобто побудовано допустимий вектор x, який береться за наближений розв'язок. В іншому випадку переходимо до *кроку 2*.

Крок 2. Для $(i, j) \in I \setminus I_{kl}$ знаходимо

$$(i_0, j_0) = \max_{(i,j) \in I \setminus I_{kl}} \left\{ \min_{x_{ij}^L \le x_{ij} \le x_{ij}^R} F(\{x_{ij}\}) \right\},$$

де $$x_{kl}^{L,R} = \begin{cases} x_{kl}^0, & (k,l) \in I \setminus I_{kl},\ k \ne i,\ l \ne j; \\ x_{kl}^*, & (k,l) \in I_{kl}. \end{cases}$$

Покладемо $x_{i_0 j_0}^* = \min\limits_{x_{i_0 j_0}^L \le x_{i_0 j_0} \le x_{i_0 j_0}^R} F$, $X_{k+1,l+1} = X_{kl}(x_{i_0 j_0}^*)$, $I_{k+1,l+1} = I_{kl} \cup (\{i_0, j_0\})$ і переходимо до *кроку 1*.

Для зменшення похибки наближеного розв'язку, отриманої з використанням цього алгоритму, пропонується процедура покращення результату. Зокрема, припускатимемо, що змінні вектора x пронумеровані в послідовності отримання значень у «жадібному» алгоритмі.

Алгоритм покращення розв'язку.

Як початковий план беремо розв'язок, отриманий за «жадібним» алгоритмом. Покладемо $M = \{(1,1),(1,2),...,(n.m)\}$, $(p,q) = (1,1)$.

Крок 1. Вибираємо ще не опрацьовану змінну $x_{pq} > 0$ з мінімальними індексами $(p,q) \in M$ і підбираємо значення $\delta_{pq} > 0$, на яке необхідно зменшити величину x_{pq}.

Крок 2. Знаходимо змінну x_{rs}, $r > p$, $s > q$ і величину $\delta_{rs} > 0$, на яку можна зменшити значення x_{rs}, не порушуючи допустимості плану та збільшивши при цьому цільову функцію. Для цього δ_{pq} та δ_{rs} повинні задовольняти умову

$$F(...,x_{pq}-\delta_{pq},...,x_{rs},...) < F(...,x_{pq}-\delta_{pq},...,x_{rs}-\delta_{rs},...).$$

Після закінчення *кроку 2* незалежно від того, знайдена чи ні змінна x_{rs}, покладемо $M = M \setminus \{(p,q)\}$ і переходимо до наступного значення із множини M і повертаємося до *кроку 1*. Процес припиняється, коли всі змінні у початковому плані будуть переглянуті ($M = \varnothing$).

Як показали результати числових експериментів (табл. 1), середня похибка наближеного розв'язку порівняно з точним, отриманим методом повного перебирання, знижується від 20% до 5% із застосуванням алгоритму покращення розв'язку.

Таблиця 1. Оцінка точності використання «жадібного» алгоритму та алгоритму покращення розв'язку в задачах (1–5) і (7)–(10) у відсотках

Розмірність задачі	**Величина середньої похибки «жадібного алгоритму» задачі (1)–(5)**	**Величина середньої похибки алгоритму покращення розв'язку задачі (1)–(5)**	**Величина середньої похибки «жадібного алгоритму» задачі (7)–(10)**	**Величина середньої похибки алгоритму покращення розв'язку задачі (7)–(10)**
$m = 10,\ n = 5$	15,2	3,9	16,7	4,2
$m = 18.0,\ n = 5$	17,4	4,8	18,9	5,4
$m = 28.0,\ n = 6$	21,9	5,1	22,3	5,8

СТРАТЕГІЯ ВИБОРУ РОЗПОДІЛУ РЕСУРСІВ В ІНФОРМАЦІЙНИХ СИСТЕМАХ, БЛИЗЬКОГО ДО ОПТИМАЛЬНОГО, З ВИКОРИСТАННЯМ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Генетичні алгоритми (*Genetic Algorithms*) є варіантами еволюційних методів пошуку, за якими створюється популяція елементів (особин), де в задачі оптимізації кожна особина відповідає одному з можливих розв'язків. Для пошуку найкращого розв'язку використовується цільова функція або пов'язана з нею функція пристосування, значення якої показують, наскільки особина відповідає розв'язку задачі. Для забезпечення процесу еволюційно-

го пошуку до початкової популяції застосовуються основні генетичні операції (селекції, схрещування, мутації та клонування), за результатами яких генерується нова популяція за допомогою додавання нових елементів з кращими показниками цільової функції або функції пристосування.

Генетичні алгоритми поділяють на дві групи: генетичні алгоритми з бінарним кодуванням [17, 18] та генетичні алгоритми з дійсним кодуванням [19]. Перша група використовує двійкову систему числення для кодування розв'язків на множині допустимих значень. Це забезпечує високу ефективність пошуку екстремального значення на множині допустимих розв'язків, що досягається використанням теореми про схеми [18].

Друга група виникла як результат відмови від ідеї кодування, тобто розв'язок в особині подається у вигляді набору дійсних чисел. У цьому випадку реалізація алгоритмів змінюється, а операції кодування–декодування відсутні.

Далі пропонується реалізація генетичних алгоритмів за двома підходами, що зумовлюється структурами математичних моделей.

Підхід 1 — для задач бінарної оптимізації

Крок 1. *Формування початкової популяції.*

1. Задається номер популяції $np = 0$, максимальна кількість популяцій *pmax*, номер ітерації $itern = 0$.

2. Задається розмір популяції q і випадковим чином формується початкова популяція розміру q. Для цього за допомогою рівномірного розподілу генерується q матриць розміру $m \times n$ (наприклад, для задачі (11)–(12)) — особин популяції, елементи яких є випадкові значення з одиничного відрізка $(0,1)$. Однак усі елементи матриці мають бути цілими числами з множини $\{0,1\}$. Тому в роботі пропонується вводити матрицю самонавчання алгоритму $\mu = \{\mu_{ij}\}$, що буде засобом цілеспрямованого впливу на характеристики пошуку, де елементи цієї матриці спочатку також розраховуються випадковим чином з відрізка $(0,1)$ із застосуванням рівномірного розподілу. Зауважимо, що за такого підходу до кодування розв'язку особиною популяції є матриця, стовпці якої є хромосомами особини.

3. Для кожної матриці — особини з популяції перед розрахунком цільової функції — застосовуватиметься процедура дискретизації: якщо значення елемента матриці менше або дорівнює відповідному значенню матриці коригування, то значення зменшується до нуля, у противному випадку — зростає до одиниці. Якщо за умовою обмежень оптимізаційної задачі тільки одне значення в стовпці (хромосомі) матриці популяції має дорівнювати одиниці, то збільшується до одиниці тільки перше значення стовпця, що є більшим від відповідного значення елемента матриці самонавчання.

4. Для кожної дискретизації матриці з популяції розраховується значення цільової функції (або функції пристосування) і перевіряється задоволення додаткових обмежень, сформованих в оптимізаційній моделі. Якщо у сформованій популяції відсутні варіанти, що задовольняють обмеження оптимізаційної моделі, то популяція має бути перестворена заново.

Крок 2. *Селекція*. Особини для схрещування пропонується обирати відповідно до стратегії панміксії — випадкового рівномірного вибору двох батьківських особин — двох матриць з популяції.

Крок 3. *Схрещування* — формування нових нащадків у популяції. Для формування нащадків пропонується застосування одноточкового та багатоточкового схрещування. За першого варіанта кожна пара стовпців (хромосом) матриць батьків випадково розривається в одній точці і формуються два нові стовпці матриць нащадків з використанням двох частин хромосоми кожної батьківської особини: перший нащадок отримує, наприклад, кожну верхню частину вектора-стовпчика першого батька і нижню другого, тоді як другий нащадок — навпаки.

За другого варіанта схрещування випадково вибираються дві або більше точок розриву для кожної пари хромосом з матриць батьків, і нащадки отримають нові хромосоми, що складатимуться із сегментів, розміщених між цими точками. Зауважимо, що варіант вибору кількості сегментів розбиття також визначається випадковим чином для кожної нової популяції

Результатом цього кроку є два нащадки, що отримуються з використанням вибраних на другому кроці алгоритму батьківських особин.

Крок 4. *Мутація* — перетворення хромосоми, що випадково змінює один або декілька з її генів. Застосовується для підтримання різноманітності хромосом у популяції.

У роботі пропонується для мутації використовувати оператор інверсії, за яким кожен стовпець (хромосома) матриці особини випадково ділиться на дві частини, які потім обмінюються місцями — нижня частина стає верхньою і навпаки.

Як наслідок, результатом цього кроку є два нащадки-мутанти, що були отримані застосуванням оператора мутації до двох нових особин, отриманих на третьому кроці алгоритму.

Крок 5. *Формування нової популяції*. З отриманих на попередньому кроці особин вибирається одна, результат застосування якої до цільової функції є найкращим. Вона замінить у вихідній популяції особину, що має найгіршу пристосованість. Зауважимо, що значення пристосовуваності обраховується з використанням матриці самонавчання алгоритму за схемою, наведеною на *кроці 3*.

Далі перевіряються умови:

- якщо $itern < q$, то покласти $itern = itern + 1$ і перейти до *кроку 2*;
- якщо $itern = q$, то покласти $np = np + 1$ і перейти до *кроку 6*.

Крок 6. *Перевірка умови завершення роботи генетичного алгоритму*. Умовою завершення роботи генетичного алгоритму є формування заданої кількості популяцій $np > ptax$:

- якщо умову не виконано, то покладаємо $itern = 1$ і переходимо до *кроку 2*. Уточнюється матриця самонавчання алгоритму. Для цього значення матриці μ обраховуються за формулою

$$\mu_{ij} = \begin{cases} 0, & \mu_{ij} \pm (1 - F_0 / F_1) \le 0; \\ \mu_{ij} \pm (1 - F_0 / F_1), & 0 < \mu_{ij} \pm (1 - F_0 / F_1) < 1; \\ 1 & \mu_{ij} \pm (1 - F_0 / F_1) \ge 1, \end{cases}$$

де знак плюс вибирається для оптимізаційних задач на максимум, мінус — для задач пошуку мінімуму цільової функції. Значення F_1 — найкраще значення функції пристосовуваності для поточного покоління популяції; F_0 — відповідно найкраще значення функції пристосовуваності для попереднього покоління. Очевидною є вимога побудови двох поколінь популяцій перед

застосуванням цієї процедури уточнення для матриці самонавчання. Уведення такої залежності робить процес самонавчання генетичного алгоритму більш чутливим до змін якості розв'язку. Дійсно, за малих змін функції пристосовуваності значення елементів матриці змінюються незначно і навпаки;

- якщо умову завершення роботи виконано, то як розв'язок (наближений) вибираємо особину з найбільшою пристосованістю з поточної популяції, застосувавши процедуру дискретизації (із кроку 3).

Як показали проведені числові експерименти побудови розв'язку задачі (11)–(12), уведення у генетичний алгоритм процедури самонавчання алгоритму дозволило у середньому зменшити кількість поколінь при формуванні популяцій на п'ятнадцять відсотків, а кількість задач, для яких розв'язок не був знайдений за десять тисяч поколінь популяцій, зменшився на п'ять відсотків (табл. 2).

Таблиця 2. Оцінка точності використання генетичного алгоритму в задачі (11)–(12) з урахуванням процедури самонавчання алгоритму

Функція пристосування (13)	**Середня кількість поколінь генетичного алгоритму без самонавчання**	**Кількість задач, розв'язків для яких не було знайдено за десять тисяч поколінь за алгоритмом без самонавчання**	**Середня кількість поколінь генетичного алгоритму із самонавчанням**	**Кількість задач, розв'язків для яких не було знайдено за десять тисяч поколінь за алгоритмом із самонавчанням**
1	1220	11	1040	10
2	1300	21	1110	19
3	1450	8	1230	7
4	1180	45	1000	40

Функція пристосування визначалася такими залежностями:

$$1)\ F_p(\{x_{ij}\}) = \frac{1000^{F(\{x_{ij}\})}}{E(\{x_{ij}\})};\ 2)\ F_p(\{x_{ij}\}) = \frac{e^{F(\{x_{ij}\})}}{E(\{x_{ij}\})};$$

$$3)\ F_p(\{x_{ij}\}) = \frac{F(\{x_{ij}\})}{E(\{x_{ij}\})};\ \ 4)\ F_p(\{x_{ij}\}) = \frac{F(\{x_{ij}\})}{e^{E(\{x_{ij}\})}},$$

де вирази для функцій $F(\{x_{ij}\})$ та $E(\{x_{ij}\})$ наведено у формулі (11).

Підхід 2 — для задач оптимізації з дійсними розв'язками

Крок 1. *Формування початкової популяції.*

1. Як і для задачі бінарної оптимізації задається номер популяції $np = 0$, максимальна кількість популяцій *pmax*, номер ітерації $itern = 0$.

2. Задається розмір популяції q і випадковим чином формується початкова популяція розміру q. Для цього за допомогою рівномірного розподілу генерується q матриць $\{x_{ij}^0\}$ розміру $m \times n$ (наприклад, для задачі (1)–(3)) — особин, елементи яких є випадковими значеннями з одиничного відрізка $[0,1]$. Із використанням лінійного перетворення кожне значення елемента

матриці відображається на відповідний проміжок $[\alpha_{ij}, \beta_{ij}]$: $x_{ij} = (\beta_{ij} - \alpha_{ij})x_{ij}^0 + \alpha_{ij}$, коли, наприклад, за умовою (1) $\alpha_{ij} = 0$, $\beta_{ij} = T_i$.

3. Обчислюється значення функції пристосування для кожної матриці $\{x_{ij}\}$ — особини популяції і перевіряється задоволення додаткових обмежень, що сформовані в оптимізаційній моделі. Якщо у сформованій популяції немає варіантів, що задовольняють обмеження оптимізаційної моделі, то популяція має бути перестворена заново.

Крок 2. *Селекція*. Особини для схрещування, як і у випадку бінарної оптимізації, пропонується обирати із застосуванням стратегії панміксії — випадкового рівномірного вибору двох батьківських особин — двох матриць з популяції.

Крок 3. *Схрещування* — формування нових нащадків у популяції. Для формування нащадків пропонується застосування кросоверу, що імітує двійковий [20], коли нащадки формуються з особин батьків за формулою

$$x_{ij}^{c1} = \frac{1}{2}[(1-\chi_1)x_{ij}^{p1} + (1+\chi_1)x_{ij}^{p2}];\ x_{ij}^{c2} = \frac{1}{2}[(1+\chi_2)x_{ij}^{p1} + (1-\chi_2)x_{ij}^{p2}],$$

де $\{x_{ij}^{p1}\}$, $\{x_{ij}^{p2}\}$ — батьківські особини; $\{x_{ij}^{c1}\}$, $\{x_{ij}^{c2}\}$ — матриці нащадків. Значення χ_r, $r = 1,2$ обчислюються за формулою

$$\chi_r = \begin{cases} (2u)^{\frac{1}{\eta+1}}, & u \le \frac{1}{2}; \\ (2(1-u))^{\frac{-1}{\eta+1}}, & u > \frac{1}{2}, \end{cases}$$

де значення u обирається випадковим чином з інтервалу $(0,1)$ на кожній ітерації алгоритму. Значення η за працею [20] має належати інтервалу $\eta \in [2,5]$. Надалі, як і за варіантом бінарної оптимізації, пропонується застосовувати підхід самонавчання алгоритму, тобто змінювати коефіцієнт схрещування η відповідно до кращого пристосування популяції. Тобто, зафіксувавши на початку формування першої популяції деяке початкове значення η з інтервалу $[2,5]$, для кожної наступної, починаючи з другої, буде застосовуватися схема самонавчання алгоритму

$$\eta = \begin{cases} 2, & \eta \pm (1 - F_0 / F_1) \le 2; \\ \eta \pm (1 - F_0 / F_1), & 2 < \eta \pm (1 - F_0 / F_1) < 5; \\ 5 & \eta \pm (1 - F_0 / F_1) \ge 5, \end{cases} \qquad (13)$$

де, як і раніше, знак плюс вибирається для оптимізаційних задач на максимум, знак мінус — для задач пошуку мінімуму цільової функції. Значення F_1 — найкраще значення функції пристосовуваності для поточного покоління популяції, F_0 — відповідно найкраще значення функції пристосовуваності для попереднього покоління.

Очевидно, що за такого вибору оператора схрещування значення матриць особин-нащадків також будуть задовольняти обмеження відповідної

оптимізаційної задачі, якщо координати батьківських особин також задовольняли ці обмеження.

Крок 4. *Мутація*. У роботі пропонується для мутації використовувати випадкову мутацію: випадковим чином вибирається декілька елементів кожної матриці нащадків і їх значення замінюються випадковим чином на нові за схемою формування особини 1.2. Як наслідок, результатом цього кроку є два нащадки-мутанти, що були отримані застосуванням оператора мутації до двох нових особин, отриманих на третьому кроці алгоритму.

Крок 5. *Формування нової популяції.* З отриманих на попередньому кроці особин вибирається одна, результат застосування якої до цільової функції є найкращим. Вона замінить у вихідній популяції особину, що має найгіршу пристосованість.

Далі перевіряються умови:

- якщо $itern < q$, то покласти $itern = itern + 1$ і перейти до *кроку 2*;
- якщо $itern = q$, то покласти $np = np + 1$ і перейти до *кроку 6*.

Крок 6. *Перевірка умови завершення роботи генетичного алгоритму.* Умовою завершення роботи генетичного алгоритму є формування заданої кількості популяцій $np > ptax$:

- якщо умову не виконано, то покладаємо $itern = 1$ і переходимо до *кроку 2*. При цьому відбувається процедура (13) самонавчання алгоритму — уточнення коефіцієнта схрещування η ;
- якщо умову завершення роботи виконано, то як розв'язок (наближений) вибираємо особину з найкращою пристосованістю з поточної популяції.

Як наслідок, модифікація генетичного алгоритму — уведення параметрів самонавчання алгоритму — забезпечує коригування популяцій у напрямі найкращої пристосованості. Тобто рівноймовірний процес генетичного алгоритму отримує властивість пам'яті, що зменшує кількість генерацій поколінь популяцій для досягнення необхідної точності результату.

ВИСНОВКИ

Інтеграція інформаційних та обчислювальних ресурсів у єдине середовище та організація ефективного доступу до них є одним з основних напрямів розвитку сучасних інформаційних технологій. На перший план виходить проблема ефективного використання обчислювальних ресурсів кожного вузла мережі для вирішення складних наукових, виробничих і технологічних завдань. Тому одним з актуальних завдань сьогодення є ефективне керування обчислювальними ресурсами у розподіленому середовищі. Зростання кількості ресурсних центрів, що складають розподілену інфраструктуру, за відсутності або низької оптимальності підсистеми планування, яка забезпечує керування потоком задач, не тільки знижує ефективність використання усієї розподіленої інфраструктури, але й може зробити беззмістовним її створення. Тому розширення класу математичних моделей та інструментарію числових методів, які можуть бути залучені у комп'ютерні системи адміністрування розподілу ресурсів, є актуальною науковою проблемою, що і

зумовило значущість досліджень цієї роботи; здійснено побудову та досліджено математичні моделі оптимального оброблення даних у розподілених інформаційних системах.

Побудовано наближені обчислювальні алгоритми з використанням процедури «жадібного» вибору. Для зниження обчислювальної похибки розв'язку, що отримується за сформованими алгоритмами, запропоновано процедуру покращення розв'язку, яка зменшує відносну обчислювальну похибку до рівня, прийнятного у прикладних дослідженнях.

Наведено дві схеми розрахунку наближеного розв'язку задач оптимізації за бінарного та дійсного кодування параметрів моделі. Запропоновано модифікацію генетичного алгоритму — уведення параметрів самонавчання алгоритму, що забезпечує коригування популяцій у напрямі найкращої пристосованості. Тобто рівноймовірний процес генетичного алгоритму отримує властивість пам'яті, що зменшує кількість генерацій поколінь популяцій для досягнення необхідної точності. Як показали числові експерименти, уведення в генетичний алгоритм процедури самонавчання алгоритму дозволило у середньому зменшити кількість поколінь у формуванні популяцій на двадцять відсотків, а кількість задач, для яких розв'язок не був знайдений за десять тисяч поколінь популяцій, зменшився на п'ять відсотків.

Як наслідок, сформульовані та досліджені математичні моделі оптимізації розподілу ресурсів та наближені методи розв'язання відповідних оптимізаційних задач можуть бути покладені в основу створення програмних комплексів керування розподіленою комп'ютерною інфраструктурою, що можна розглядати предметом подальших досліджень.

ЛІТЕРАТУРА

1. *Sabin G.* Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment / G. Sabin, R. Kettimuthu // Job Scheduling Strategies for Parallel Processing (JSSPP'03): Proceedings of the 9th International Workshop (Seattle, WA, USA, June 24, 2003). — Springer, Lecture Notes in Computer Science, 2003. — Vol. 2862. — P. 87–104.
2. *Arora M.* A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments / M. Arora, S.K. Das, R. Biswas // International Conference on Parallel Processing Workshops (ICPPW'02): Proceedings of International Conference (Vancouver, British Columbia Canada, August 20–23, 2002). — IEEE Computer Society, 2002. — P. 499–505.
3. *El-Rewini H.* Task Scheduling in Parallel and Distributed Systems / H. El-Rewini, T. Lewis, H. Ali. — Prentice Hall, 1994. — 290 p.
4. *Buyya R.* The Grid Economy / R. Buyya, D. Abramson, S. Venugopal // Proceedings of the IEEE. — 2005. — Vol. 93, N 3. — P. 698–714.
5. *Zhu Y.* Incentive–based P2P Scheduling in Grid Computing / Y. Zhu, L. Xiao // Proceedings of the 3rd International Conference on Grid and Cooperative Computing (GCC2004), Wuhan, China, October 21–24, 2004. Springer, Lecture Notes in Computer Science, 2004. — P. 209–216.
6. *Young L.* Scheduling Architecture and Algorithms within the ICENI Grid Middleware / L. Young, S. McGough // Proceedings of UK e–Science All Hands Meeting. — Nottingham, UK: IOP Publishing Ltd., 2003. — P. 5–12.
7. *You S.Y.* Task Scheduling Algorithm in GRID Considering Heterogeneous Environment / S.Y. You , H.Y. Kim // Proceedings of the International Conference on

Parallel and Distributed Processing Techniques and Applications (PDPTA ‘04), June 21–24, 2004. — Nevada, USA: CSREA Press, 2004. — P. 240–245.

8. *Kim S.* A Genetic Algorithm Based Approach for Scheduling Decomposable Data Grid Applications / S. Kim, J.B. Weissman // Proceedings of the 2004 International Conference on Parallel Processing (ICPP’04), Montreal, Quebec Canada, August 15–18, 2004). IEEE Computer Society, 2004. — P. 406–413.
9. *Song S.* Security–Driven Heuristics and A Fast Genetic Algorithm for Trusted Grid Job Scheduling / S. Song, Y. Kwok, K. Hwang // Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS’05), Denver, Colorado USA, April 25–29, 2005. IEEE Computer Society, 2005. — P. 65–74.
10. *Braun R.* A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems / R. Braun, H.Siegel // Journal of Parallel and Distributed Computing. — 2001. — Vol. 61, N 6. — P. 810–837.
11. *Young L.* Scheduling Architecture and Algorithms within the ICENI Grid Middleware / L. Young, S.McGough // Proceedings of UK e–Science All Hands Meeting, September 2003). — Nottingham, UK: IOP Publishing Ltd., 2003. — P. 5–12.
12. *Abraham A.* Nature’s Heuristics for Scheduling Jobs on Computational Grids / A. Abraham, R. Buyya, B. Nath // Proceedings of 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), Cochin, India, December 14–16, 2000. IEEE Computer Society, 2000. — P. 45–52.
13. *Ахо А.В.* Построение и анализ вычислительных алгоритмов / А.В. Ахо, Д.Э. Хонкрофт, Д.Д. Ульман. — М.: Мир, 1979. — 380 с.
14. *Ахо А.В.* Структуры даных и алгоритмы / А.В. Ахо, Д.Э. Хонкрофт, Д.Д. Ульман. — М.: Издательский дом «Вильямс», 2003. — 426 с.
15. *Новиков Ф.А.* Дискретная математика для программистов / Ф.А. Новиков. — Спб.: Питер, 2011. — 526 с.
16. *Сигал И.Х.* Введение в прикладное дискретное программирование: модели и вычислительные методы / И.Х. Сигал, А.П. Иванова. — М.: Физматлит, 2002. — 320 с.
17. *Chambers D.L.* Practical Handbook of Genetic algorithms, Applications / D.L. Chambers. — Chapman and Hall; CRC Press, 2001. — 650 p.
18. *Halland J.N.* Adaption in natural and artificial systems / J.N. Halland. — Ann Arbor, Michigan: University of Michigan Press, 1975. — 726 p.
19. *Michalewicz Z.* Genetic algorithms, numerical optimization and constraints / Z. Michalewicz // Proceedings of the 6th Intern. Coference on Genetic Algorithms, 1995. — P. 151–158.
20. *Deb K.* Realcoded genetic algorithms with simulated binary crossover: Studies on multimodal and multiobjective problems / K. Deb, A. Kumar // Complex Systems, 1995. — Vol. 9, N 6. — P. 431–454.

Надійшла 12.02.2018