УДК 004.023, 004.75

М.Ю. Юрич, аспір.
Запорізький національний технічний університет

ЗАБЕЗПЕЧЕННЯ ЕФЕКТИВНОСТІ РОЗПОДІЛУ ЗАВДАНЬ В ОБЧИСЛЮВАЛЬНІЙ СИСТЕМІ ЗА ДОПОМОГОЮ ГЕНЕТИЧНОГО АЛГОРИТМУ

(Представлено д.т.н., проф. Карпуковим Л.М.)

Розглянуто можливості використання методів еволюційного моделювання для розв'язання задач оптимізації. Розроблено генетичний алгоритм для розподілу завдань в обчислювальній системі. Наведено результати експериментальних досліджень роботи генетичного алгоритму для розв'язання поставленої задачі.

Постановка задачі в загальному вигляді та її актуальність. На даний час дуже актуальним став розвиток GRID-технологій. У зв'язку з цим загострилась проблема розподілу завдань, що мають бути вирішенні в системі таким чином, щоб час на їх вирішення був максимально скороченим, а задіяні при цьому апаратні ресурси якомога менше завантажені. Тобто стає проблема оптимального розподілу завдань в обчислювальній системі.

Як відомо, одним із структурних підрозділів GRID-системи є саме локальні комп'ютерні мережі. І на відміну від «верхнього» рівня цієї структури, де працюють вже відомі брокери, на «нижньому» рівні є потреба у застосуванні ефективного та доступного засобу розподілу завдань в середині такого сегмента загальної системи GRID.

З метою вирішення цієї проблеми було розроблено власну систему розподілу завдань. В основу математичної моделі цієї системи покладено математичну модель класичної транспортної задачі [1]. Проте для пристосування до конкретного випадку – розподілу завдань в обчислювальній системі – необхідним стало введення до існуючої математичної моделі класичної транспортної задачі додаткових обмежень, які, звісно, випливають із специфіки задачі, що розв'язується. Таким чином, в основі математичної моделі системи розподілу завдань було покладено модифіковану транспортну задачу [2].

Проте, модифікація транспортної задачі не зняла з неї основних труднощів, що виникають при розв'язанні класичної транспортної задачі.

Як відомо, саме транспортна задача відноситься до класу NP – повних задач, що означає, що розв'язати її (а саме знайти оптимальний розв'язок) можна лише за експоненціальний час. Звісно, при збільшенні вхідних даних знаходження такого розв'язання ускладнюється. Тому постає задача знайдення розв'язку, максимально наближеного до оптимального (але не оптимального), проте за реальний час.

Існують багато способів розв'язання транспортної задачі класичними методами (точними або наближеними). Точні методи – забирають занадто багато часу, або, взагалі їх не можна пристосувати через велику кількість вхідних даних, а наближені – не завжди можуть задовольнити своїми результатами. Тому стає потреба в застосуванні чогось нового, що зможе більш ефективно розв'язувати задачі такого класу.

З метою розв'язання поставленої задачі було запропоновано методи еволюційного моделювання, а саме застосування генетичного алгоритму.

Класичний генетичний алгоритм. Батьком сучасної теорії генетичних алгоритмів (ГА) вважається Холланд (J. Holland), чия робота [3] стала класикою в цій області. В ній Холланд вперше ввів термін «генетичний алгоритм». Зараз описаний там алгоритм називають «класичний ГА» (іноді «канонічний ГА», canonical GA), а поняття «генетичні алгоритми» стало дуже широким, і часто до них відносять алгоритми, сильно відмінні від класичного ГА. Учні Холланда Кенет Де Йонг (Kenneth De Jong) і Девід Голдберг (David E. Goldberg) внесли величезний внесок у розвиток ГА (представлені у [4]) [5].

Генетичні алгоритми засновані на еволюції особин, що намагаються «вижити» в поставлених умовах, спираючись на закони природного відбору [6].

Алгоритм роботи генетичних алгоритмів доволі простий. Необхідно дотримуватися деякої послідовності операцій:

– створити початкову популяцію хромосом;

– вибрати кращі рішення для повторної комбінації хромосом шляхом операції кросовера (схрещування);

– виконати мутацію;

– замінити гірші рішення на кращі;

– створити нову популяцію.

І такий процес повторюється до тих пір, поки не буде знайдено найкраще рішення за допомогою використання функції пристосовності та виконання умови зупинення алгоритму.

Загальний вигляд роботи класичного генетичного алгоритму представлено на рис.1.

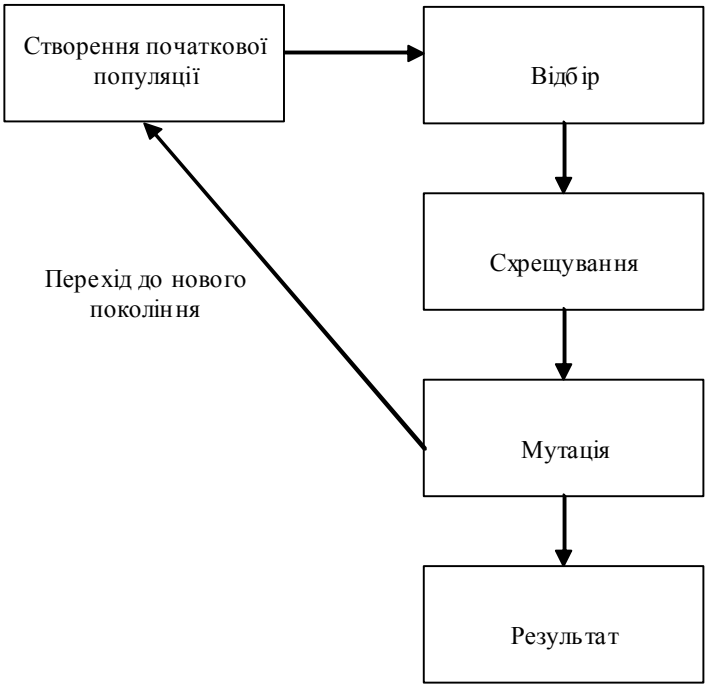

Рис. 1. Робота класичного генетичного алгоритму

Розробка власного генетичного алгоритму. Розроблений генетичний алгоритм має наступні етапи. Спочатку проводимо заповнення початкових даних, яке складається з таких елементів:

1. Матриця, що включає загальну кількість задач *i*-го потоку ($i = \overline{1,m}$);
2. Матриця, що включає загальну кількість задач, що будуть розв'язані на *j*-му ($j = \overline{1,n}$). комп'ютері при оптимальному розбитті *m* потоків задач серед *n* комп'ютерів;
3. Матриця, що включає час розв'язання однієї задачі з *i*-го потоку на *j*-му комп'ютері;
4. Загальна кількість задач, що надійшли до системи;
5. Кількість потоків задач;
6. Кількість комп'ютерів, що розв'язують задачі з потоків.

Далі формуємо P початкових хромосом, якими є матриці, що складаються з кількості задач з *i*-го потоку, які планується розв'язати на *j*-му комп'ютері. Створюємо випадковим чином 2 точки кросовера. Потім схрещуємо начальні хромосоми за межами точок кросовера. Між кросоверами заповнюємо значення випадково або нулями. Потім вилучаємо від'ємні значення в нових хромосомах (нащадках). Створюємо матрицю похибок для нормалізації за кількістю задач. Проводимо нормалізацію за кількістю задач. Нормалізація виконується, доки матриця похибок не обнулиться повністю. Після нормалізації нащадків їх можна схрещувати знову.

З метою уникнення «звироднілості» популяції проводимо операцію мутації. Після цього знову проводимо нормалізацію (якщо в тому є потреба). Далі визначаємо найкраще серед отриманих рішення, за допомогою використання функції пристосовності. У даному випадку, це мінімізація суми добутку матриці, що включає час розв'язання однієї задачі з *i*-го потоку на *j*-му комп'ютері, на матрицю, що складається з кількості задач з *i*-го потоку, які планується розв'язати на *j*-му комп'ютері:

$$T(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \to \min . \quad (1)$$

Найбільш вдалі рішення створюють нове покоління. Весь перелік описаних вище операцій повторюється, створюючи нові популяції, доки не виконається умова зупинення алгоритму. Зупиняємо процес у випадку, коли кількість проведених ітерацій L дорівнює добутку кількості потоків задач на кількість комп'ютерів, що розв'язують задачі з потоків, помноженому на 2 (проте, це значення може бути змінено у разі необхідності).

Таким чином, укрупнена блок-схема роботи розробленого генетичного алгоритму має вигляд, представлений на рис. 2.

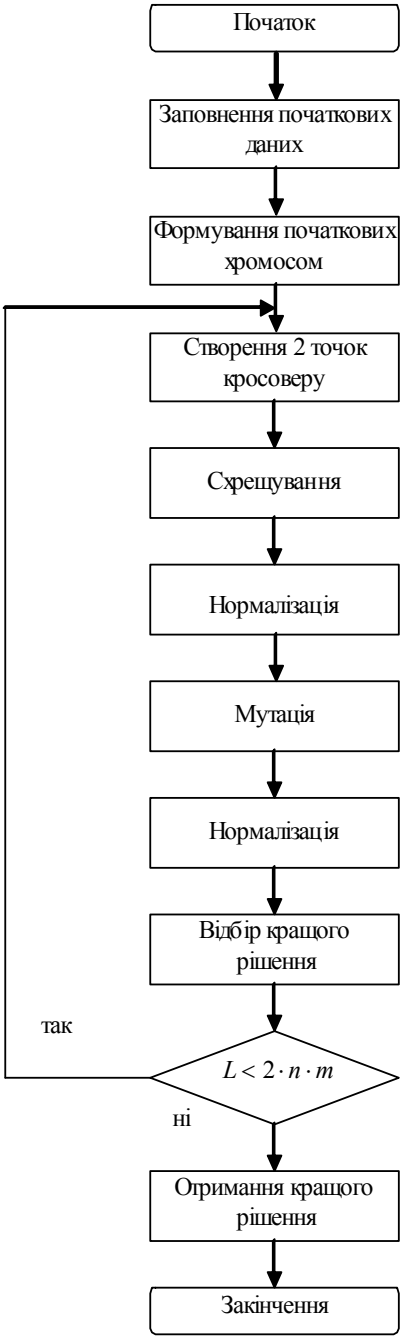

Рис. 2. Алгоритм роботи розробленого генетичного алгоритму

Приклад роботи розробленого ГА. З метою наочності роботи розробленого алгоритму розглянемо приклад частини його роботи на системі з малою кількістю вхідних даних.

Нехай маємо 5 потоків задач і 8 комп'ютерів, на які їх необхідно розподілити. Відома матриця, що включає час розв'язання однієї задачі з *i*-го потоку на *j*-му комп'ютері:

7 1 3 4 1 9 8 2
8 9 2 7 6 1 3 1
1 3 8 5 4 3 6 2
8 9 2 7 6 7 3 6
1 3 8 5 4 8 4 2

Відомо кількість задач у кожному потоці

40 25 15 10 10

І кількість задач, які необхідно розв'язати на кожному з 8 комп'ютерів.

20 15 5 10 20 10 15 5

Для початку вибирається Р випадкових рішень. Наприклад, нехай сформовані 2 хромосоми 5х8.

15	11	3	2	2	4	1	2
1	4	0	1	14	5	0	0
0	0	0	3	4	0	5	3
3	0	0	0	0	0	7	0
1	0	2	4	0	1	2	0

16	9	3	9	0	1	0	2
4	1	0	0	4	9	4	3
0	5	2	0	3	0	5	0
0	0	0	0	4	0	6	0
0	0	0	1	9	0	0	0

Далі випадковим чином обираються 2 точки кросовера, наприклад 26 і 30.

Тоді з цих хромосом можна сформувати дві нові. Спочатку з першого батька вибираємо всі значення, що містяться до першої точки кросовера та занотовуємо їх першому нащадку. Йому також повинні передатися гени другого з батьків, тому, починаючи з 2-ї точки кросовера, першому нащадку записуються значення другого з батьків. Ділянка хромосоми між кросоверами заповнюється нулями. Для другого нащадка робимо те ж саме, тільки змінивши місцями першого і другого з батьків. Таким чином, у нас вийде:

15	11	3	2	2	4	1	2
1	4	0	1	14	5	0	0
0	0	0	3	4	0	5	3
3	0	|0	0	0	0	|6	0
0	0	0	1	9	0	0	0

16	9	3	9	0	1	0	2
4	1	0	0	4	9	4	3
0	5	2	0	3	0	5	0
0	0	|0	0	0	0	|7	0
1	0	2	4	0	1	2	0

На наступному кроці необхідно виконати нормалізацію хромосом за кількістю задач у потоці. Наприклад, у 1-й хромосомі необхідно розподілити ще 1 задачу, а в 2-й хромосомі ще 3. Відсутні елементи заповнюються випадково між точками кросовера. Можливий варіант, коли після схрещування з *i*-го потоку розподіляється товару більше, ніж у нього є, тоді між кросоверами можливі негативні елементи. Нехай після нормалізації за кількістю задач у потоці у нас вийшов такий результат:

15	11	3	2	2	4	1	2
1	4	0	1	14	5	0	0
0	0	0	3	4	0	5	3
3	0	|1	0	0	0	|6	0
0	0	0	1	9	0	0	0
16	9	3	9	0	1	0	2
4	1	0	0	4	9	4	3
0	5	2	0	3	0	5	0
0	0	|1	1	1	0	|7	0
1	0	2	4	0	1	2	0

Якщо між кросоверами були негативні елементи, то на наступному кроці від них позбавляємося, віднімаючи його від будь-яких позитивних елементів в цьому рядку.

Наші хромосоми не нормалізовані за кількістю задач, що необхідно розподілити на комп'ютери. Для нормалізації спочатку будуємо масив, який визначає різницю між значеннями *i*-го елемента масиву кількості задач, що необхідно розподілити на комп'ютер *j*, та сумою *i*-го стовпчика сформованих хромосом.

У нашому випадку отримаємо такі значення:

(1 0 1 3 –9 1 3 0)

(–1 0 –3 –4 12 –1 –3 0)

Тепер потрібно нормалізувати кожен стовпчик.

Наприклад, в перший стовпчик першої хромосоми потрібно додати 1. Додаємо до елемента [1] [3], бо у нього мінімальний час розв'язання однієї задачі. Для того, щоб не змінювати вже нормалізованих даних за кількістю задач в потоці, від елемента, що міститься в цьому рядку і має найбільший час розв'язання однієї задачі, віднімаємо 1.

Так послідовно нормалізуємо всі стовпчики і всі хромосоми. Після чого проводимо операцію мутації та усі інші операції, розглянуті у пунктах вище, при описі роботи розробленого генетичного алгоритму.

Результати експериментальних досліджень. З метою доведення ефективності розробленого алгоритму відносно існуючих класичних методів було розв'язано задачі з однаковими вхідними даними з використанням симплекс-методу та розробленого генетичного алгоритму. Дослідження проводились при різній кількості елементів матриць початкових даних. Аналізуючи час, затрачений на розв'язання поставленої задачі обома способами, було зроблено висновок, що чим більша матриця, що включає час розв'язання однієї задачі з *i*-го потоку на *j*-му комп'ютері, тим швидше розв'язується задача у порівнянні з симплекс-методом. При цьому відхилення від оптимального результату склало 3–9 %.

Для прикладу наведемо такі цифри. При порівнянні часу роботи з використанням матриці, що включає час розв'язання однієї задачі з *i*-го потоку на *j*-му комп'ютері, розміром 100x100. Час роботи розробленого алгоритму склав 78 секунд при використанні 6 нащадків. Таку ж саму задачу, застосовуючи симплекс-метод, було розв'язано за 482 секунди. Проте, такий виграш у часі був отриманий за рахунок меншої точності отриманого результату – відхилення від оптимального результату склало 6 %.

Висновки. Дослідження, проведені під час роботи, довели, що застосування генетичного алгоритму для розподілу завдань в обчислювальній системі є ефективним (особливо з великою кількістю комп'ютерів у розподіленій системі). Свідченням цього стали результати, отримані у порівнянні із найбільш розповсюдженим точним методом розв'язання поставленої задачі.

Так, розроблений під час роботи генетичний алгоритм для розподілу завдань в обчислювальній системі дозволяє отримати результати за час в середньому у 7–8 разів менший за час, необхідний для розв'язання тієї ж самої задачі шляхом застосування класичних методів оптимізації, при цьому відхилення від оптимального результату склало в середньому 6 %.

Підвищенням вже отриманих результатів для розподілу задач в обчислювальній системі може стати застосування розпаралеленого генетичного алгоритму при розв'язанні поставленої задачі.

ЛІТЕРАТУРА:

1. *Триус Е. Б.* Задачи математического программирования транспортного типа. – М.: Сов. радио, 1967. – 208 с.
2. *Юрич М.Ю.* Подход к оптимальному распределению заданий в вычислительной системе // Комп'ютинг. – Тернопіль: Економічна думка, 2008. – Том 7. – Випуск 1. – С. 27–34.
3. *Holland J.* Adaptation in Natural and Artificial Systems. – Univ. of Michigan, Press, Ann Arbor, 1975.
4. *Goldberg, D. E.,* Genetic Algorithms in Search, Optimization, and Machine Learning, Addison–Welsey Publishing Company Inc., 1989.
5. *Редько В.Г., Цой Ю.Р.* Оценка эффективности эволюционных алгоритмов // Доклады АН № 3. – 2005. – С. 312–315.
6. *Рутковская Д., Пилинский М., Рутковский Л.* Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И. Д. Рудинского. – М.: «Горячая линия – Телеком», 2004. – 452 с.: ил.

ЮРИЧ Марія Юріївна – аспірант кафедри «Комп'ютерні системи та мережі» Запорізького національного технічного університету

Наукові інтереси:

- нейронні мережі, еволюційне моделювання, штучний інтелект;
- балансування навантаження, розподілені обчислювальні системи.

Тел.: 8(066)7335578; 8(097)7305279.

E-mail: masheryka@mail.ru; mashery@zntu.edu.ua

Подано 11.11.2008

Юрич М.Ю. Забезпечення ефективності розподілу завдань в обчислювальній системі за допомогою генетичного алгоритму
Юрич М.Ю. Обеспечение эффективности распределения заданий в вычислительной системе с помощью генетических алгоритмов
Yurich M.Yu. Providing of efficiency of distribution of tasks in the computing system by genetic algorithms

УДК 004.023, 004.75

Обеспечение эффективности распределения заданий в вычислительной системе с помощью генетических алгоритмов / М.Ю.Юрич

Рассмотрены возможности использования методов эволюционного моделирования для решения задач оптимизации. Разработан генетический алгоритм для распределения заданий в вычислительной системе. Приведены результаты экспериментальных исследований работы генетического алгоритма для решения поставленной задачи.

УДК 004.023, 004.75

Providing of efficiency of distribution of tasks in the computing system by genetic algorithms // M.Yu. Yurich

Opportunities of use of methods of evolutionary modelling for the decision tasks of optimization are considered. The genetic algorithm for distribution of tasks in the computing system is developed. Results of experimental researches of work of genetic algorithm for the decision of a task in view are resulted.