КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.711: 519.711.3: 519.81 DOI:10.34229/2707-451X.22.2.3

І.О. ЛУК'ЯНОВ, Ф.А. ЛИТВИНЕНКО

ПРО ВИБІР ЧИСЛА ПРОЦЕСОРІВ ДЛЯ ПАРАЛЕЛЬНОГО БАГАТОПОПУЛЯЦІЙНОГО ГЕНЕТИЧНОГО АЛГОРИТМУ

В роботі розглянуті деякі особливості паралельної реалізації багатопопуляційного генетичного алгоритму, а також підходи до його оптимізації. Представлені результати експериментів з використанням різного числа процесорів і різними способами генерації початкових популяцій з метою оптимізації алгоритму по декільком критеріям (оцінці використання обчислювальних і часових ресурсів). На прикладі конкретної тестової задачі приведені оцінки з вибору оптимального числа процесорів для отримання необхідного результату.

Ключові слова: *паралельний генетичний алгоритм, генерація початкової популяції, вибір числа процесорів (популяцій), оптимізація алгоритму.*

Вступ. У роботах [1–3] описано паралельний багатопопуляційний генетичний алгоритм (ПБГА), призначений для проведення оптимізаційно-імітаційних експериментів при дослідженні складних стохастичних систем великої розмірності.

Алгоритм розроблявся відповідно до сучасного підходу в комп'ютерному моделюванні, що базується на методології Data Farming [4–6], яка використовує інтеграційні можливості засобів імітації, методи оптимізації, високопродуктивні обчислення та методи інтелектуального аналізу даних.

Основними вимогами до алгоритму були простота, що не вимагає високої математичної підготовки, та універсальність щодо природи діючих факторів та можливої форми поверхні реакції комп'ютерної моделі. Ефективність алгоритму, яка визначалася швидкістю його збіжності, досліджувалася на різних тестових задачах.

Були отримані результати, що встановлюють залежність швидкості збіжності ПБГА від значень його параметрів та особливостей реалізації. Таких, наприклад, як ймовірності схрещування і мутації, процедури вибору початкової популяції, стратегій схрещування та стратегій обміну перспективними хромосомами між популяціями.

Як одна з детермінованих тестових задач використовувалася (проста з точки зору оптимізації) задача пошуку заданого (цільового) рядка довжини $\boldsymbol{N}$ в кінцевому алфавіті розмірності $\boldsymbol{K}$. При проведенні експериментів були прийняті наступні значення: $\boldsymbol{N} = 100$, $\boldsymbol{K} = 33$.

Особливість цієї задачі – єдиний мінімум, поліпшення значення фітнес-функції виключно при знаходженні єдиного правильного значення для гена (оптимального значення для фактору), незалежність кожного з генів.

Враховуючи особливості класу задач, при розробці алгоритму особливе значення надавалося ефективному використанню операцій схрещування з метою накопичення і збереження "правильних" генів (символів, значення яких збігається зі значеннями цільового рядку) [3]. У результаті вдалося розробити багатопопуляційний генетичний алгоритм, що здатен досягти 98% від оптимуму (отримувати хромосому-рішення, що містить 98% «правильних» генів), використовуючи тільки операції схрещування, без використання операцій мутації.

Запропонований алгоритм умовно складається із трьох послідовних етапів.

На ***початковому*** етапі випадковим чином генеруються початкові популяції (множина хромосом-рішень заданої довжини ***N***) з яких за допомогою операції схрещування формується елітна частина. Мета цього етапу – збереження генетичного матеріалу кожної популяції, тобто концентрація якомога більшої кількості "правильних" генів у кількох хромосомах-рішеннях. Цей етап займає приблизно 10 % ітерацій, кожна з яких відповідає зміні одного покоління на всіх процесорах.

На ***обмінному*** етапі популяції обмінюються кращими хромосомами-рішеннями таким чином, щоб з одного боку максимально наблизитися до оптимального рішення, а з іншого – не допустити виродження популяцій (приблизно 70% ітерацій).

І, нарешті, на ***заключному*** етапі, коли операції схрещування та обміну практично вичерпують себе, подальший процес оптимізації здійснюється з використанням операцій мутації. З їх допомогою алгоритм заповнює недостатній генетичний матеріал, який був загублений у процесі роботи алгоритму або був відсутній у початкових популяціях (приблизно 20% ітерацій). Зазначимо, що на цьому етапі відбувається пошук останніх відсутніх «правильних» генів – близько 2% від оптимуму.

Необхідно зазначити, що для вирішення даної тестової задачі можна застосувати алгоритм перебору, який буде найефективнішим, проте він може працювати тільки з незалежними факторами. Такий алгоритм полягає у послідовному переборі символів алфавіту для кожного з генів, умовою переходу до наступного гену є знаходження «правильного» значення для поточного, що визначається покращенням значення фітнес-функції на одиницю. У його роботі у середньому розглядається приблизно ***N*K***/2 альтернатив.

Мета даної роботи – проведення експериментів із заданою тестовою задачею з різним числом процесорів та альтернативними способами генерації початкової популяції для оцінки ефективності ПБГА. Більшість раніше описаних експериментів проводилися на 16 процесорах. Така кількість процесорів була обумовлена розмірністю тестової задачі та результатами початкових експериментів.

Випадкова генерація початкової популяції. У роботі [3] наведені формули для розрахунку розміру початкової популяції з метою генерації загального числа «правильних» генів у 16 популяціях на рівні 98% і вище. Відповідно, при зменшенні числа процесорів розмір початкової популяції зростає. У табл. 1 наведені результати експериментів і значення розміру початкової популяції у залежності від числа процесорів згідно з формулою, наведеною в [3].

ТАБЛИЦЯ 1

Число процесорів	Розмір початкової популяції	Кількість ітерацій	Кількість розглянутих альтернатив
16	20	140	25445
12	28	128	24517
8	40	134	24249
4	92	149	30605
2	272	138	42282

Як видно з табл. 1, результати експериментів для 16, 12 і 8 процесорів є схожими за кількістю ітерацій та розглянутих альтернатив (розглянутих хромосом-рішень). Для 4 та 2 процесорів показник розглянутих альтернатив значно гірший.

На рис. 1 показано кількість розглянутих альтернатив для досягнення певного відсотка оптимуму (від 90 % до 100 %) для різної кількості процесорів. Найбільш ефективним на всьому інтервалі виявилися експерименти з використанням 8 процесорів, для 12 і 16 процесорів кількість розглянутих альтернатив була трохи більшою, але відмінність між ними не суттєва. Результати для 4 процесорів гірші на всьому інтервалі, зі значним погіршенням при наближенні до оптимуму.

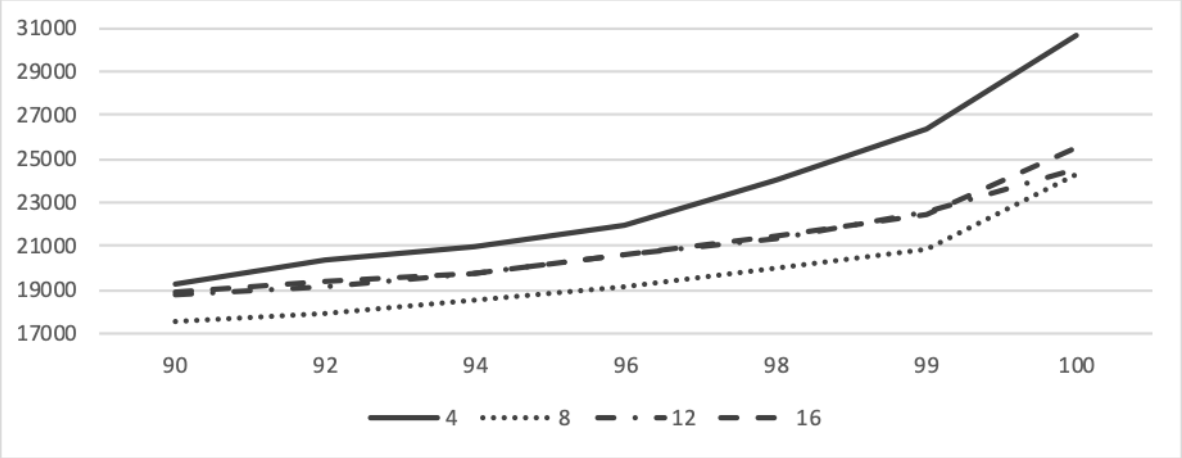

РИС. 1

Альтернативний метод генерації початкової популяції. В роботі [2] запропоновано альтернативний спосіб генерації початкових популяцій для ПБГА, названий алгоритмом рівномірного сканування простору значень факторів, в результаті застосування якого у кожній початковій популяції присутні всі можливі значення факторів оптимальної хромосоми-рішення. За умови використання цього алгоритму розмір початкової популяції завжди дорівнює розміру алфавіту K = 33, незалежно від кількості процесорів. У табл. 2 наведені результати експериментів з даним способом генерації початкової популяції.

ТАБЛИЦЯ 2

Число процесорів	Розмір початкової популяції	Кількість ітерацій	Кількість розглянутих альтернатив
16	33	119	22185
12	33	139	19029
8	33	219	19371
4	33	507	21545

Для алгоритму рівномірного сканування простору значень факторів на 8, 12 та 16 процесорах знадобилося приблизно на 20–25% менше альтернатив, ніж для випадкового способу генерації початкової популяції. Для 4 процесорів результат краще приблизно на 30%.

На рис. 2 показано результати експериментів для алгоритму рівномірного сканування простору значень факторів для досягнення певного відсотка оптимуму (від 90 % до 100 %) з різною кількістю процесорів. Для зручності порівняння на даному рисунку також присутні результати експери-

ментів для попереднього способу генерації початкової популяції (випадкового) з використанням 8 процесорів – найкращий результат для випадкового способу генерації початкової популяції.

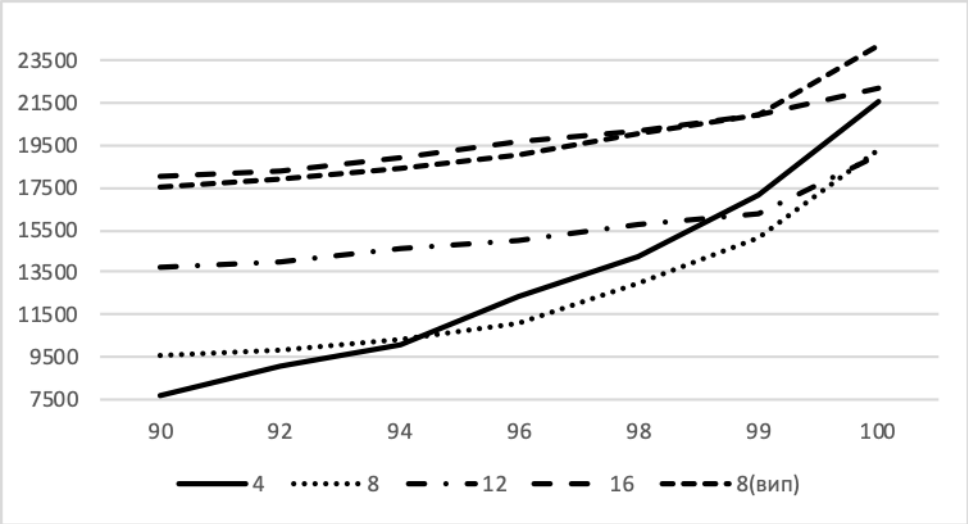

РИС. 2

Експерименти з 4 процесорами показують найбільш ефективні результати при досягненні 94 % від оптимуму включно. Для досягнення 95–99 % – використання 8 процесорів було ефективніше. Для досягнення 100 % трохи ефективнішими були 12 процесорів. Зазначимо, що всі експерименти з альтернативним способом генерації початкової популяції показали зменшення кількості розглянутих альтернатив під час використання однакової кількості процесорів у порівнянні з випадковим способом. Але при цьому ефективність випадкової генерації початкової популяції та 8 процесорів була приблизно такою самою, як у алгоритму рівномірного сканування простору значень факторів з 16 процесорами.

Необхідно зазначити, що для цієї задачі операція мутації є не ефективною, в порівнянні з операцією схрещування, тому що ймовірність «погіршити» результат значно перевищує ймовірність знайти «правильний» ген. Були проведені експерименти без заключного етапу (для отримання нових хромосом використовувалися тільки операції схрещування) та з різною кількістю процесорів для порівняння частки експериментів, що не досягли оптимуму через виродження популяції. Результати показано на рис. 3.

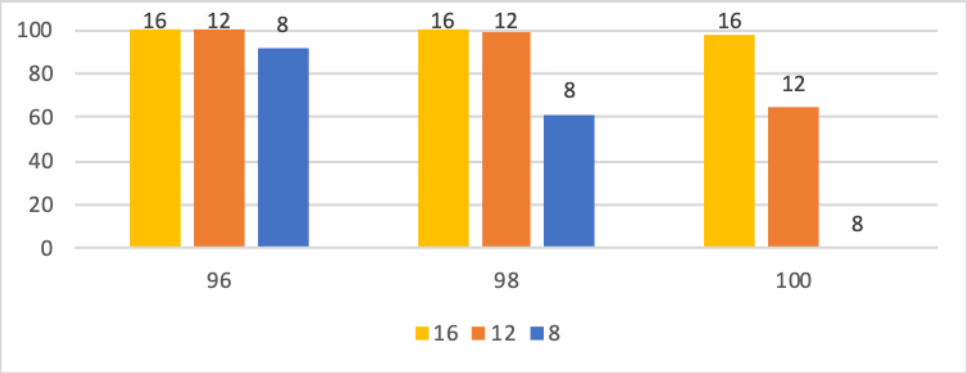

РИС. 3

Як видно із цього рисунку, при наближенні до оптимуму та зі зменшенням кількості процесорів значно зменшується відсоток експериментів, що досягли заданого результату (вісь абсцис – відсоток від оптимуму, вісь ординат – відсоток експериментів, що досягли заданого результату). Для 16 процесорів операцію мутації можна взагалі не використовувати до досягнення 99 % від оптимуму (для 100 % від оптимуму – всього 3 % експериментів не досягли заданого результату).

Оцінка ефективності алгоритму за різними критеріями. Оцінюючи ефективність алгоритму можна розглядати кілька критеріїв, основними з яких є кількість використаних обчислювальних і часових ресурсів.

Показник загальної кількості розглянутих альтернатив (K_1) можна інтерпретувати як кількість витрачених обчислювальних ресурсів, що знадобилися для знаходження оптимуму. В свою чергу, кількість розглянутих альтернатив на процесорі «лідері» (K_2) – процесорі, де було знайдено оптимум – можна розглядати, як витрачений часовий ресурс, необхідний для знаходження оптимуму. Як грубу оцінку можна вважати, що $K_2 \approx K_1$/*кількість процесорів*.

Для завдань комп'ютерного моделювання великий інтерес представляє оптимізація за критерієм K_2, оскільки найчастіше прогін моделі – це дуже ресурсомістка операція з погляду витраченого часу.

Таким чином, при обчислювальній вартості однієї альтернативи C_1 загальна обчислювальна вартість дорівнює C_1*K_1. А при часовій вартості однієї альтернативи C_2 загальна часова вартість дорівнює C_2*K_2.

У табл. 3 наведено показники K_1 та K_2 для випадкової генерації початкової популяції та алгоритму рівномірного сканування простору значень факторів для досягнення 100 % від оптимуму. Для обох способів зі збільшенням кількості процесорів покращується показник K_2. Але водночас при використанні 16 і більше процесорів це поліпшення досить невелике, і, враховуючи збільшення K_1, може не мати сенсу в залежності від поставленого завдання. З огляду на це, можна сказати, що при оптимізації по K_1 та K_2 ефективніше використовувати 12 чи 16 процесорів, залежно від C_1 і C_2, оскільки подальше поліпшення по K_2 занадто незначне відносно погіршення по K_1.

ТАБЛИЦЯ 3

Число процесорів	K_1 для випадкового способу	K_2 для випадкового способу	K_1 для альтернативного способу	K_2 для альтернативного способу
4	30605	7651	21545	5386
8	24249	3031	19371	1614
12	24517	2043	19029	1585
16	25445	1590	22185	1386
20	28237	1411	26571	1328
24	32759	1364	31774	1323
28	37530	1340	34818	1312
32	42603	1331	41835	1307

Висновок. Проведені експерименти показують, що алгоритм рівномірного сканування простору значень факторів є ефективнішим за випадковий спосіб генерації початкової популяції. Також експерименти показали, що для досягнення максимальної ефективності ПБГА кількість популяцій (процесорів) необхідно обирати відносно бажаної точності результату. Так, для даної тестової задачі, для отримання результату 90–94 % від оптимуму найбільш ефективним з точки зору обчислювальних ресурсів є використання 4 процесорів та алгоритмом рівномірного сканування простору значень факторів. Для досягнення результату, що перевищує 94% і оптимізації за K_1 кращий ре-

зультат показали 8 процесорів і алгоритм рівномірного сканування простору значень факторів. Якщо враховувати ще й критерій часових ресурсів K_2, то для досягнення 90–98 % від оптимуму потрібно використовувати 8 процесорів, а для 99–100 % 12 чи 16 процесів, залежно від C_1 та C_2.

Список літератури

1. Литвиненко Ф.А., Лук'янов І.О., Криковлюк О.А. Особливості реалізації паралельної версії багатопопуляційного генетичного алгоритму. *Комп'ютерна математика*. 2018. № 2. С. 21–29. http://dspace.nbuv.gov.ua/handle/123456789/161882
2. Лук'янов І.О., Литвиненко Ф.А., Криковлюк О.А. Про підвищення ефективності паралельної версії багатопопуляційного генетичного алгоритму. *Теорія оптимальних рішень*. 2019. № 18. С. 116–122. http://dspace.nbuv.gov.ua/handle/123456789/161683
3. Литвиненко Ф.А., Лук'янов И.О., Криковлюк О.А. Використання різноманітності початкової популяції у багатопопуляційному генетичному алгоритмі. *Комп'ютерна математика*. 2019. № 1. С. 116–123. http://dspace.nbuv.gov.ua/handle/123456789/161941
4. Horne G.E., Meyer T.E. Data Farming: Discovering Surprise. *Proc. of the Winter Simulation Conference*. 2005. P. 1082–1087.
5. Horne G.E., Schwierz K.-P. Data Farming around the world overview. *Proc. of the Winter Simulation Conference*. 2008. P. 1442–1447.
6. Пепеляєв В.А., Чорний Ю.М. Про можливості застосування генетичних алгоритмів в оптимізаційно-імітаційних експериментах. *Теорія оптимальних рішень*. 2019. № 18. С. 69–77. http://dspace.nbuv.gov.ua/handle/123456789/161681

Одержано 01.09.2022

Лук'янов Ігор Олегович,
молодший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
ihorlukianov@gmail.com

Литвиненко Федір Антонович,
молодший науковий співробітник
Інституту кібернетики імені В.М. Глушкова НАН України, Київ.
fedirlytvynenko@gmail.com

UDC 519.711: 519.711.3: 519.81

Ihor Lukianov, Fedir Lytvynenko

About Selecting the Number of Processors for Parallel Multipopulation Genetic Algorithm

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
Correspondence: ihorlukianov@gmail.com, fedirlytvynenko@gmail.com

Introduction. The paper considers some features of the parallel implementation of a multipopulation genetic algorithm, as well as approaches to its optimization. The results of experiments with the use of a different number of processors and different methods of generating initial populations are presented in order to optimize the algorithm according to several criteria (assessment of the use of computational and time resources). On the example of a specific test problem, estimates are given for choosing the optimal number of processors to obtain the desired result.

The purpose of this work is to conduct experiments with a given test problem with a different number of processors and alternative methods for generating the initial population to evaluate the effectiveness of the algorithm.

Results. For the test problem, to obtain a result of 90–94 % of the optimum, the most efficient in terms of computing resources is the use of 4 processors with an algorithm for uniform scanning of the space of factor values. To achieve a result exceeding 94 % and optimize by K_1 (computational resources), 8 processors and an

algorithm for uniform scanning of the space of factor values showed the best result. If we also take into account the criterion of time resources K_2, then to achieve 90–98 % of the optimum, it is necessary to use 8 processors, for 99–100 % 12 or 16 processes, depending on C_1 and C_2 (cost of computational and time resources respectively).

Conclusions. Performed experiments show that the algorithm of uniform scanning of the space of factor values is more efficient than the random method of generating the initial population. Experiments also showed that in order to achieve the maximum efficiency of PMGA, the number of processors must be chosen depending on the desired result precision.

Keywords: parallel genetic algorithm, initial population generation, choice of the number of processors (populations), algorithm optimization.