ДК 621.372

КОРНАЧЕВСЬКИЙ Я.І.,
ЛАДОГУБЕЦЬ В.В.

ГЕНЕТИЧНІ АЛГОРИТМИ В САПР

Розглядаються особливості паралельної реалізації генетичних алгоритмів для розв'язання задачі оптимізації. Дана оцінка алгоритмів з точки зору їх застосування в різних галузях САПР.

Article is de voted to the features of parallel implementation of genetic algorithms for solution of the optimization problem. Application of these algorithms for CAD systems is evaluated.

Вступ

Генетичні алгоритми (ГА) використовуються в САПР не так часто (порівняно з іншими алгоритмами). Разом з цим, ГА мають багато переваг (наприклад, відсутність обмежень на характер цільової функції, зокрема нечутливість до "яружності" та перервності), які роблять їх дуже зручними для задач, які характерні для систем автоматизованого проектування. Тому оцінка можливості застосування ГА для САПР є дуже важливою.

1. Елементи еволюційної теорії

Генетичні алгоритми були запропоновані в 1975 році Джоном Голландом [1]. Прототипом обрано природний добір, який в рамках еволюційної теорії постійно покращує види, або, умовно кажучи, перетворює їх до оптимальної форми. Суть природного добору полягає у тому, що найбільш пристосовані представники краще виживають і приносять більше потомства, ніж менш пристосовані. Відзначимо, що сам по собі природний відбір ще не забезпечує розвитку біологічного виду. Справді, якщо припустити, що всі нащадки народжуються приблизно однаковими, то різні покоління будуть відрізнятися тільки по чисельності, але не по пристосованості. Тому дуже важливо вивчити, яким чином відбувається спадкування, тобто як властивості нащадка залежать від властивостей батьків.

Основний закон спадкування полягає в тому, що нащадки схожі на батьків. Зокрема, нащадки більш пристосованих батьків будуть, напевне, одними з найбільш пристосованих у своєму поколінні.

Майже в кожній клітині тварини є набір хромосом, що несуть у собі всю спадкову інформацію. Інформація кодується генами – відрізками ланцюга ДНК, відповідальними за певну визначену властивість особини, наприклад, за колір очей чи тип волосся. В процесі утворення клітини нового організму відбувається таке: нитки ДНК розриваються в декількох випадкових місцях і хромосоми обмінюються своїми частинами (рис 1). Цей процес забезпечує появу нових варіантів хромосом і називається "кросинговер".

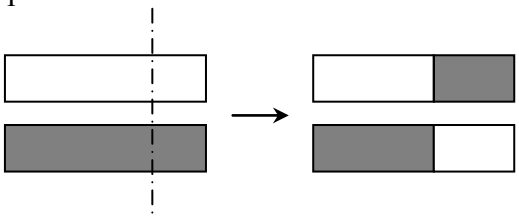

Рис. 1. Умовна схема кросинговеру

На спадковість впливають також мутації, які виражаються в зміні деяких ділянок ДНК. Мутації випадкові.

2. Задача оптимізації

Як уже було відзначено вище, еволюція – це процес постійної оптимізації біологічних видів. Природний добір гарантує, що найбільш пристосовані особини дадуть багато нащадків. А завдяки генетичному спадкуванню ми можемо бути впевнені, що частина цих нащадків не тільки збереже високу пристосованість батьків, але буде володіти і деякими новими властивостями. Якщо ці нові властивості виявляться корисними, то, з великою імовірністю, вони перейдуть і в наступне покоління. Таким чином, відбувається нагромадження корисних якостей і поступове підвищення пристосовності біологічного виду в цілому. Знаючи, як розв'язується задача оптимізації видів у природі, ми тепер застосуємо схожий метод для вирішення різних реальних задач.

Введемо позначення і приведемо кілька класичних прикладів. Як правило, в задачі оптимізації ми можемо керувати декількома параметрами (позначимо їх через x_1, x_2, ..., x_n), а нашою метою є максимізація (чи мінімізація) деякої цільової функції, $f(x_1, x_2, ..., x_n)$, що залежить від цих параметрів. Наприклад, якщо потрібно максимізувати цільову функцію "прибуток компанії", то керованими параметрами будуть число співробітників компанії, обсяг виробництва, витрати на рекламу, ціни на кінцеві продукти і т.д. Важливо відзначити, що ці параметри зв'язані між собою – наприклад, при зменшенні числа співробітників швидше всього зменшиться й обсяг виробництва.

Звичайно, математики здавна займалися подібними задачами. Розроблено багато методів їхнього рішення. У випадку, якщо цільова функція досить гладка і має тільки один локальний мінімум (унімодальна), то оптимальне рішення можна одержати градієнтними або квазіньютонівськими методами. Недоліком таких підходів є порівняно невелика швидкість збіжності, досить велика трудоємкість одного кроку, порівняно жорсткі вимоги до властивостей цільових функцій Аналогічні проблеми виникають і з застосуванням інших математичних методів. У багатьох важливих задачах параметри можуть приймати лише визначені значення, причому, у всіх інших точках цільова функція не визначена. Звичайно, у такому випадку не може бути і мови про її гладкість. Тому вимагаються принципово інші підходи.

Найпростіший спосіб знайти оптимальне рішення – перебрати всі можливі значення параметрів. При цьому не потрібно робити ніяких припущень про властивості цільової функції, а задати її можна просто за допомогою таблиці. Однак, щоб вирішити таким способом задачу комівояжера хоча б для 20 міст, буде потрібно перебрати близько 10^{19} маршрутів, що зовсім нереально для будь-якого обчислювального центру. Отже, виникає необхідність у якому-небудь новому методі оптимізації, придатному для практики. Покажемо, як можна застосувати механізми еволюційного процесу до таких задач. Фактично ж організуємо штучну еволюцію в спеціально побудованому світі.

3. Робота ГА

Уявимо собі штучний світ, населений безліччю істот (осіб), причому кожна істота – це деяке рішення нашої задачі. Будемо вважати особу тим більше пристосованою, чим краще відповідне рішення (чим більше значення цільової функції воно дає). Тоді задача максимізації цільової функції зводиться до пошуку найбільш пристосованої істоти. Звичайно, ми не можемо оселити в наш віртуальний світ всі істоти відразу, тому що їх дуже багато. Замість цього ми будемо розглядати багато поколінь, що змінюють одне одного. Тепер, якщо ми зуміємо ввести в дію природний добір і генетичне спадкування, то отриманий світ буде підкорятися законам еволюції. Відзначимо, що, відповідно до нашого визначення пристосованості, метою цієї штучної еволюції буде саме створення найкращих рішень. Очевидно, еволюція – нескінченний процес, у ході якого пристосованість осіб поступово підвищується. Примусово зупинивши цей процес через досить довгий час після його початку і вибравши найбільш пристосовану особу у поточному поколінні, ми одержимо не абсолютно точний, але близький до оптимального розв'язок. Перейдемо тепер до точних визначень і опишемо робо-

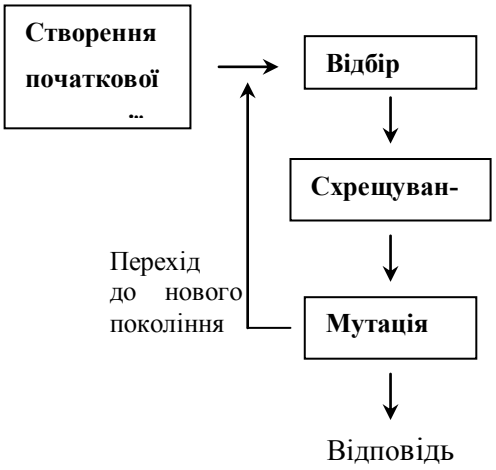

Рис. 2. Блок-схема генетичного алгоритму

ту ГА більш детально.

Для того, щоб говорити про генетичне спадкування, потрібно постачати наші істоти хромосомами. У ГА хромосома – це деякий числовий вектор, що відповідає параметру, який підбирається, а набір хромосом даної особи визначає рішення задачі. Які саме вектори варто розглядати в конкретній задачі, вирішує сам користувач. Кожна з позицій вектора хромосоми називається геном.

Визначимо тепер поняття, які відповідають мутації і кросинговеру в ГА.

Мутація – це перетворення хромосоми, що випадково змінює одну чи декілька її позицій

(генів). Найбільш розповсюджений вид мутацій – випадкова зміна лише одного з генів хромосоми [1, 2].

Кросинговер (у літературі по ГА також вживається назва **кросовер** чи **схрещування**) – це операція, при якій із двох хромосом породжується одна чи декілька нових хромосом. У найпростішому випадку кросинговер у ГА реалізується так само, як і в біології (див. рис. 1). При цьому хромосоми розрізаються у випадковому місці і обмінюються частинами між собою. Наприклад, якщо хромосоми (1, 2, 3, 4, 5) і (0, 0, 0, 0, 0) розрізати між третім і четвертим генами і обміняти їх частини, то вийдуть нащадки (1, 2, 3, 0, 0) і (0, 0, 0, 4, 5).

Блок-схема ГА зображена на рис. 2. Спочатку генерується початкова популяція осіб (індивідуумів), тобто деякий набір рішень задачі. Як правило, це робиться випадковим способом. Потім ми повинні змоделювати розмноження всередині цієї популяції. Для цього випадково відбираються кілька пар індивідуумів, відбувається схрещування між хромосомами в кожній парі, а отримані нові хромосоми вміщуються в популяцію нового покоління. У ГА зберігається основний принцип природного добору – чим більш пристосований індивідуум (чим більше відповідне йому значення цільової функції), тим з більшою імовірністю він буде брати участь у схрещуванні. Тепер моделюються мутації – у декількох випадково обраних особах нового покоління змінюються деякі гени. Потім стара популяція частково чи цілком знищується і ми переходимо до розгляду наступного покоління. Популяція наступного покоління в більшості реалізацій ГА містить стільки ж осіб, скільки й початкова, але в силу добору пристосованість у ній у середньому вище. Тепер описані процеси добору, схрещування і мутації повторюються уже для цієї популяції і т.д.

4. Особливості ГА для САПР

Найбільш істотні особливості задача параметричної оптимізації наступні:

1. Складний багатоекстремальний характер цільових функцій. Найчастіше це приводить до того, що значення цільової функції у знайденій точці локального оптимуму не задовольняє вимогам завдання, і виникає задача визначення координат іншого, більш придатного локального екстремуму. Крім цього, виникає велика залежність результату оптимізації від координат початкової точки оптимізації.

2. Складність одержання аналітичної залежності цільової функції від варійованих параметрів, що виключає можливість використання прямих методів визначення екстремуму.

3. Істотний перепад в абсолютних значення варійованих параметрів (наприклад, ємності можуть мати значення порядку 10^{-9}-10^{-12}, тоді як опір – 10^{3}-10^{6}). Це висуває підвищені вимоги до чисельної стійкості процедур, які використовуються при оптимізації і нерідко вимагає застосування спеціальних методів масштабування.

4. Неможливість або дуже висока трудомісткість оцінки значень похідних цільової функції. Це ускладнює застосування для пошуку оптимальних точок швидкозбіжних процедур, які використовують градієнти та похідні вищих порядків.

5. Яружний характер цільових функцій електронних схем вимагає розробки підходів для просування "вздовж ярів", а ще краще – усунення такої залежності.

6. Необхідність працювати з цільовими функціями для задач, які взагалі важко формалізувати, і які внаслідок цього вимагають розробки спеціальних методів.

З появою потужних багатоядерних обчислювальних систем виникла можливість досить ефективно використовувати методи випадкового пошуку. Однак останні мають суттєві обмеження [3, 4], тому пропонується побудувати оптимізаційну процедуру на основі генетичного алгоритму, який буде працювати на системах із масовим паралелізмом (MPP), зокрема на кластерних системах.

Оскільки основна частина часу в САПР витрачається на розрахунок цільових функцій, пропонується розробити планувальник, який буде реалізовувати генетичний алгоритм і завантажувати вузли кластеру розрахунком цільових функції.

Будемо діяти таким чином: нехай у розмір популяції N, а кількість процесів (ядер, вузлів) – p. Організуємо ітераційний процес, як зображено на рис. 3.

Після старту процедури організуємо цикл від 1 до N з кроком p (блок 2). Для кожного процесу з k=2,p (блок 3) передамо значення вектора X для розрахунку цільової функції (блок 4). Потім запустимо процес розрахунку цільової функції для мастер-процесу (блок 5). Після цього

зберемо дані від процесів з k=2,p(блоки 6, 7) і запишемо їх у масив значень цільових функцій індивідів.

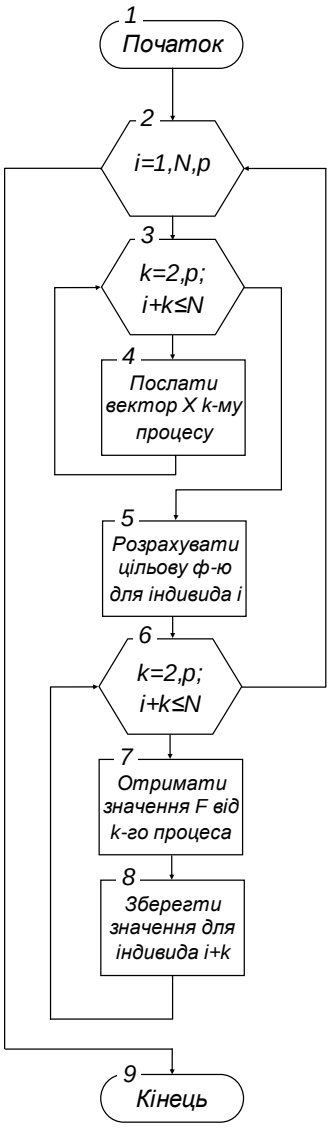

Рис. 3. Схема роботи паралельного ГА

Для пересилання даних використано функції MPI_Send() та MPI_Recv(), Спочатку здійснюється обмін даними розміром NV· ·MPI_DOUBLE, де NV – кількість параметрів цільової функції (MPI_DOUBLE – розмір даних подвоєної точності), потім – 1·MPI_DOUBLE, оскільки назад пересилається лише значення цільової функції. Таким чином, при роботі з p процесами на кожній ітерації ми пересилаємо (NV·MPI_DOUBLE+MPI_DOUBLE)(p-1) = MPI_DOUBLE·(1+ NV)(p-1) байт. Навіть при роботі з відносно повільною електронною мережею затрати на пересилання такої кількості даних мізерно малі порівняно з можливим часом розрахунку цільової функції. Однак слід зазначити, що перед початком розрахунку необхідно на кожен вузол завантажити всі необхідні для початку моделювання дані (наприклад, опис оптимізованої схеми). Цей процес може зайняти чималий час, хоча і виконується лише один раз.

Перед запуском генетичного алгоритму необхідно розрахувати кілька службових змінних. Перша – це довжина хромосоми, точніше довжини генів, які припадають на кожну змінну (рис. 4).

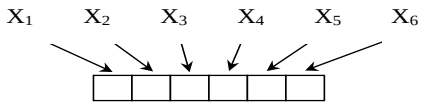

Рис. 4. Хромосома

Перед стартом користувач задає точність дискретизації ділянки пошуку, діапазон зміни кожного параметра (наприклад, ε=0.001, [-2…3; -1...8]). На основі цих даних знаходиться ширина ділянки пошуку (у прикладі відповідно 3-(-2) = 5; 8-(-1) = 9), далі розраховується кількість відліків (5/ 0.001 = 5000; 9/ 0.001 = 9000 відліків), а потім – довжини генів (log 2 (5000) ≈12,287 → 13; log 2 (9000) ≈13,135 → 14). Довжина хромосоми очевидно буде сумою довжин генів.

Оскільки майже завжди довжина генів в хромосомі буде надлишковою для заданого діапазону (наприклад, 213 = 8192), то необхідно перерозрахувати значення ε для кожного гена так, щоб максимальне його значення відповідало правому кінцю діапазону зміни параметра (наприклад, εн = 5/8192 = 0,00061).

Генерація стартової популяції полягає в генерації певної кількості індивідів. Після генерації кожного індивіду відбувається перевірка на його присутність в популяції. Якщо такий індивід в популяції вже існує – він не додається. Замість "дубліката" генерується новий індивід. Побудова індивіда полягає формуванні хромосоми. Побудова гена хромосоми полягає в генерації випадкового числа (від 0 до кількості відліків для поточного гена), перетворенні цього числа в код Грея і в подальшому записі його на позицію, яка відповідає позиції гена в хромосомі.

Після формування популяції і оцінки функції пристосованості індивіди сортуються по спаданню своєї пристосованості. Після цього починається операція кросинговеру.

Спочатку в нову популяцію записується певна кількість кращих старих індивідів (для забезпечення стратегії елітизму). Потім відбирається два різних індивіда, випадковим чином визначається позиція в хромосомі і відбувається-

ся схрещування – обмін бітами (одноточковий кросинговер, див. Розділ 2).

Відбір виконується так, щоб з більшою ймовірністю в подальшому кросинговері приймали участь ті індивіди, які мають кращу пристосованість. Для цього рівномірний закон розподілу ймовірності (вбудований в математичну бібліотеку С) перетворено на гіперболічний. Типова частота використання індивідів приведена на рис. 5.

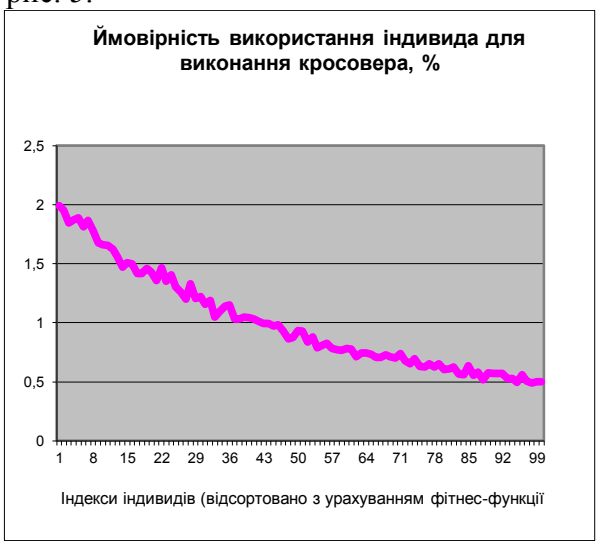

Рис. 5. Типова частота використання індивідів в залежності від фітнес-функції

Після кросинговеру генерується випадкове число (ймовірність мутації), і, якщо воно менше порогу мутації, один з нащадків мутує (двійкова мутація, див. розділ 2). Причому, якщо це число вдвічі менше порогу мутації, то мутує перший нащадок, інакше – другий. Якщо обидва індивіди нові (не існують в новій популяції), то їх додають в нову популяцію.

Індивіди сортуються і алгоритм знову готовий розпочати генерацію нової популяції. Але перед цим відбувається перевірка на значення кращої хромосоми. Якщо хромосома не змінюється протягом певної кількості ітерацій, то, очевидно, що це значення і є розв'язком. У такому випадку алгоритм зупиняє розрахунок і видає результати користувачеві.

5. Тестування ГА для САПР

Для тестування ефективності алгоритму запропоновано виконати оптимізацію для тестової функції (час на розрахунок одного значення цільової функції) і визначити витрати часу для різної кількості процесів. Для симуляції тривалого часу розрахунку складної схеми в програму введено спеціальну змінну, яка уповільнює розрахунок цільової функції. При цьому було отримано наступні результати (при роботі на дво- та чотириядерній платформі):

Табл. 1. Двоядерна платформа

1 потік, с	2 потоки, с	прискорення, разів
0,089	0,315	0,283
0,092	0,315	0,292
0,291	0,332	0,877
0,869	0,630	1,379
1,830	1,052	1,740
3,207	1,804	1,778
4,930	2,670	1,846
20,633	10,506	1,964
80,003	40,196	1,990

Рис. 6. Коефіцієнт прискорення ГА (2 ядра)

Табл. 2. Чотириядерна платформа

Кількість потоків / час виконання (секунди)				Кількість потоків / прискорення (рази)		
1	2	3	4	2	3	4
1,00	0,62	0,50	0,45	1,61	2,02	2,25
2,00	1,13	0,84	0,70	1,77	2,38	2,86
3,00	1,64	1,20	0,96	1,83	2,50	3,13
4,00	2,16	1,53	1,23	1,85	2,61	3,25
5,00	2,59	1,87	1,42	1,93	2,67	3,52
10,00	5,11	3,57	2,68	1,96	2,80	3,73
20,00	10,43	7,18	5,34	1,92	2,79	3,75
30,00	15,06	10,32	7,66	1,99	2,91	3,92
40,00	20,15	13,72	10,16	1,99	2,92	3,94
50,00	25,07	17,10	12,67	1,99	2,92	3,95

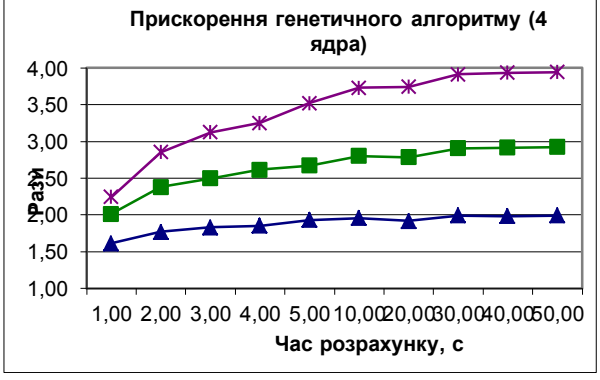

Рис. 7. Коефіцієнт прискорення ГА (4 ядра)

Висновки

Приведені результати повністю підтверджують закон Амдала – при малому часі розрахунку цільових функцій значну частину загального часу займає сам генетичний алгоритм і обмін даними між процесами. Коли ж час розрахунку великий, то прискорення досягає практично теоретичної межі, що свідчить про правильність припущень і запропонованих та реалізованих підходів до побудови паралельного генетичного алгоритму.

Список літератури

1. John H. Holland. Adaptation in Natural and Artificial Systems / John H. Holland. – Ann Arbor : University of Michigan Press, 1975. – 183 p.
2. Goldber, David E. Genetic algorithms in search, optimization, and machine learning / Goldber, David E. – Addison-Wesley Publishing Company, Inc, 1989. – 414 p.
3. Cantu-Paz E. Efficient and accurate parallel genetic algorithms / Cantu-Paz E. – Springer, 2000. – 162 p.
4. Lance Chambers. The Practical Handbook of GENETIC ALGORITHMS / Lance Chambers. – CRC Press, Inc, 1998. – 592 p.