Застосування паралельного генетичного алгоритму для вирішення задач оптимізації складних динамічних систем

Трубаров В.А., Теплинський К.С., Бабенко І.В.
Кафедра ЕОМ ДонНТУ
trubarov@cs.dgtu.donetsk.ua

Abstract

Truvarov V.A., Teplinskiy K.S., Babenko I.V. Parallel genetic algorithm for solving optimization problems of the complex dynamical systems. *Parallel realization of the genetic algorithm is proposed, implemented, tested and described for solving optimization problems of the complex dynamical systems, parameter fitting of the complex biological and chemical models in particular. Different parallel (distributed) technologies like MPI and CORBA are compared. The influence of the built-in GA cache on parallel realization effectiveness is analyzed.*

Вступ

В цій статті розглядається створений паралельний генетичний алгоритм на базі еволюційних обчислень. Він вирішує проблеми оптимізації моделювання складних динамічних систем. Основною метою є розглядання різних засобів розпаралелювання алгоритмів оптимізації та обирання найкращого з них для покращання ефективності при вирішенні проблеми ідентифікації параметрів для складних моделей біологічних та хімічних процесів. Особливістю таких моделей є складність, нелінійність та мультимодальність цільових функцій та велика кількість параметрів та обмежень. Такий характер цільових функцій потребує вирішення проблеми глобальної оптимізації. Для цього використовується один з найкращих методів еволюційних обчислень для задач оптимізації – генетичний алгоритм (ГА) [1]. Завдяки складності обчислення цільової функції для вказаних задач та необхідності в деяких випадках обчислювати велику кількість поколінь для знаходження глобального оптимуму, процес оптимізації може зайняти багато часу (дні або навіть місяці). Для зменшення часу роботи була розроблена паралельна модель ГА, яка і розглядається. Об’єктом дослідження виступають модель Лотка-Вольтерра та модель реактора Хафке. У якості засобів розпаралелювання розглянуті стандарти MPI [8] та CORBA [9], інтерфейси для яких було впроваджено у паралельну підсистему оптимізації у складі моделюючого середовища Diana. Було виконано апріорну оцінку розпаралелювання, проведено серію експериментів на кластері та на багатоядерному процесорі, а також аналіз отриманих результатів.

Задачі оптимізації та ідентифікації параметрів

Останнім часом з розвитком комп'ютерних технологій важливішим стає питання моделювання процесів в різних галузях науки та техніки. Моделювання дозволяє більш детальніше дослідити процеси в якомусь об'єкті. Також при моделюванні можна скоротити час і кошти на побудову об'єкта, тому що не потрібно буде на практиці проводити експерименти для знаходження найкращих параметрів для цього об'єкту. Особливо важливим постає питання моделювання в біології та хімії.

В даній роботі розглядається оптимізація складних динамічних систем, тобто вирішується задача ідентифікації параметрів. В такому випадку цільовою функцією є різниця між результатами роботи якоїсь розробленої моделі та експериментальними результатами. Метою оптимізації є мінімізація цієї різниці. Особливістю вирішення таких задач є висока складність моделі (більше 10 параметрів та ресурсоємне моделювання), а також складність цільової функції (велика кількість локальних оптимумів).

Тестові задачі

У якості тестових задач для паралельної оптимізації були обрані деякі штучні цільові функції (сфера), модель Лотка-Вольтерра та модель реактора Хафке. Всі вони мають різну складність, від простих штучних цільових функцій до складної моделі реактора Хафке. Тому на кожній з цих задач можна оцінити ефективність паралельної оптимізації в різних умовах (залежно від складності задачі).

Модель Лотка-Вольтерра

Одними з найбільш вдалих математичних моделей в екології є моделі, які описують спряжені коливання чисельності популяцій хижака та жертви. А. Лотка (Alfred J. Lotka, 1925) і В. Вольтерра (Vito Volterra, 1926) запропонували таку математичну модель [4]:

$$\begin{aligned} \frac{dx}{dt} &= x(\alpha - \beta y) \\ \frac{dy}{dt} &= -y(\gamma - \delta x) \end{aligned} \qquad (1)$$

де x та y – щільності відповідно жертви й хижака;

α – питома миттєва швидкість популяційного росту жертви;

β – константа, що пов'язує смертність жертв з щільністю хижака;

γ – питома смертність хижака (що нерідко вважається постійною);

δ – константа, що пов'язує народжуваність у популяції хижака з щільністю жертви.

Миттєва швидкість росту популяції хижака dy/dt в цій моделі дорівнює різниці народжуваності (яка у свою чергу залежить від інтенсивності споживання хижаком жертв) і постійній смертності.

Згідно наведених рівнянь кожна з популяцій, що взаємодіють в своєму збільшенні обмежена лише іншою популяцією, тобто зріст числа жертв лімітується пресом хижаків, у свою чергу зростання чисельності хижаків – недостатньою кількістю жертв. Слід наголосити, що в цій моделі ніякого самообмеження популяцій не передбачається. Вважається, наприклад, що їжі для жертви завжди вдосталь. Також не передбачається і вихід з-під контролю хижака популяції жертв.

Модель реактора Хафке

Модель описує роботу хімічного сумішного реактора (СР), у якому відбувається окислення етанолу, що каталізується нітратом заліза (III), з перекисом оцтового альдегіду й оцтової кислоти [11]. Усі експерименти проводяться у скляному реакторі об'ємом 3,5 літри (рисунок 1). Реагенти надходять до верхньої частини реактору двома вхідними потоками (вода, перекис водню та вода, етанол, нітрат заліза (III) відповідно), що керуються толоковими помпами.

Наступна схема реакцій, що проходять у реакторі, була запропонована К. Хафке та Е. Гіллєсом [5]:

$$
\begin{aligned}
&CH_3CH_2OH + H_20_2 \rightarrow (cat)\ CH_3CHO + 2\ H_20\ (-\Delta hR)_1 = 302\ kJ/mol\\
&CH_3CHO + H_20_2 \rightarrow (cat)\ CH_3COOH + H_20\ (-\Delta hR)_2 = 389\ kJ/mol\\
&H_20_2 \rightarrow (cat)\ 0.5\ O_2 + H_20\ (-\Delta hR)_3 = 95\ kJ/mol \qquad (2)\\
&cat + CH_3COOH \rightarrow cat^*(-\Delta hR)_4 = 0\\
&cat^* \rightarrow cat + CH_3COOH\ (-\Delta hR)_5 = 0
\end{aligned}
$$

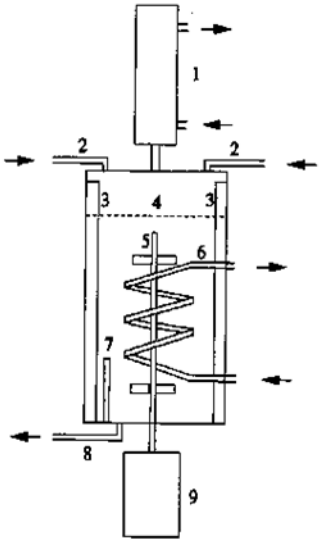

Рисунок 1 – Структурна схема реактора Хафке

При оптимізації моделі головним завданням обирається максимізація вихідної концентрації продукту, що отримується, – оцтового альдегіду (CH_3CHO). Але для поставленого завдання ідентифікації параметрів у якості шуканих параметрів обрані концентрації реагентів на виході, а у якості експериментальних даних – значення температур суміші та охолоджувальної речовини у реакторі у різних часових точках.

Модель є нелінійною та представлена у вигляді системи з 6 диференціально-алгебраїчних рівнянь [10]. Реакція проходить із коливаннями температур та концентрацій речовин, що ускладнює завдання оптимізації.

Генетичний алгоритм

Опис алгоритму

Основною ідеєю ГА [6] є пошук у багатомірному просторі, який імітує еволюцію біологічних індивідуумів (мається на увазі науковий підхід, який базується на еволюційній теорії Дарвіна). Береться певна кількість індивідуумів (популяція), кожен з яких є рішенням задачі оптимізації. Цільова функція обчислюється для кожного індивідууму та є його пристосованістю.

ГА починається з ініціалізації початкової популяції (в більшості випадків випадковими значеннями). Потім різноманітні генетичні оператори, такі як кросовер, мутація та добір, застосовуються до поточної популяції. Еволюція продовжується, доки усі індивідууми будуть мати одне й те саме значення цільової функції (це рішення задачі оптимізації) або буде досягнуто максимальну кількість поколінь (у цьому випадку індивідуум з кращім значенням цільової функції буде рішенням).

Для всіх цих генетичних операторів існує велика кількість модифікацій [2, 7]. Представлення індивідуумів (яке є бінарним у ГА) також може бути різним (стандартний код або код Грея [2]).

Мутація – це оператор, який змінює випадкові біти індивідууму інвертуючи їх. Мутація може застосовуватися до випадкового біту або до кожного біту із заданою імовірністю p_m.

Кросовер є найважливішим оператором ГА [2]. Він реалізує механізм генетичного спадкування важливих якостей батьків нащадками. Використовується два основних типи кросоверу: багатоточковий та однорідний. Перший обирає $z \geq 1$ точок кросоверу та обмінює кожен другий сегмент між обраними батьками, утворюючи двох нащадків. Однорідний кросовер обмінює кожен біт між батьками з імовірністю p_c.

Оператор добору втілює механізм природного добору. Існують такі модифікації добору, як "рулетка", "лінійне упорядкування" та "турнамент" [2]. Вони або обчислюють імовірність виживання залежно від відносної

пристосованості індивідууму, або імітують внутрішню боротьбу за виживання в популяції. Додатковий елемент алгоритму під назвою "елітизм" може бути доданий до будь-якого добору; він забезпечує виживання кращого індивідууму. Будь-який оператор добору може бути застосовано тільки на нащадках ([μ, λ] добір), або ще й на батьках ([μ + λ] добір).

Принцип паралельної роботи ГА

На кожному кроці ГА виконується операція обчислення цільової функції для всіх елементів популяції. Ці обчислення є незалежними (обчислення ЦФ для різних рішень не потребує ніякого обміну між ними на кожному кроці). Операція розрахунку ЦФ потребує дуже високих процесорних ресурсів для складних задач (якою й є оптимізація параметрів розглянутих моделей). Тому розпаралелювання обчислювання ЦФ є найбільш ефективним, оскільки потребує менше затрат на реалізацію та обмін даними. Паралельний ГА працює на сервері та запускає необхідну кількість клієнтів, які використовується для обчислення ЦФ (рис. 2).

Розроблена підсистема оптимізації є універсальним паралельним моделюючим середовищем [3] та її може бути застосовано для вирішення різноманітних класів задач.

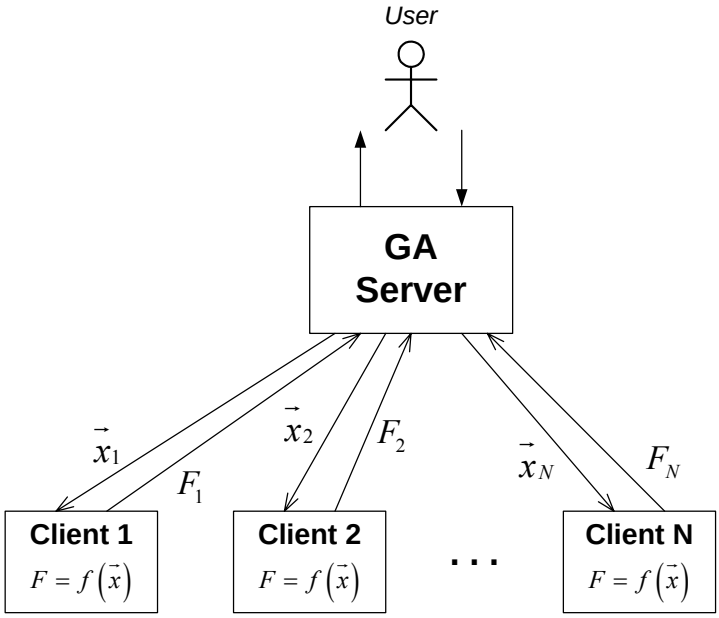

Рисунок 2 – Паралельна модель ГА

Паралельний інтерфейс (ПІ) ГА спочатку розподіляє по одній заявці для кожного клієнта, а потім, коли якийсь з них завершить обчислення ЦФ, надає йому наступну. Враховуючи те, що ПІ повертає керування ГА тільки після того, як буде розраховано ЦФ для всіх членів популяції, а кратність кількості індивідуумів до числа клієнтів в переважній більшості випадків не дотримується, то частина клієнтів на останньому кроці розрахунку

кожної генерації може не використовуватися. Особливо часто ця "проблема кратності" виникає при невеликій кількості необхідних обчислень ЦФ. Крім того, як окремий випадок, особливо при достатньо великій кількості клієнтів, існує ситуація, коли загальна кількість необхідних обчислень ЦФ є меншою за кількість клієнтів на якомусь кроці, тоді певна кількість клієнтів не буде використовуватися взагалі.

Результати досліджень

Апріорна оцінка розпаралелювання

Час вирішення задачі оптимізації за допомогою ГА на локальної машині, тобто без використання паралельного алгоритму (ПА), може бути обчислено як час ініціалізації (побудова початкової популяції), час виконання шагів ГА – селекція хромосом, застосування генетичних операторів (схрещування та мутація), побудова нової популяції та розрахунок ЦФ для всіх хромосом в популяції. Отже, час виконання ГА, як сума усіх раніше перерахованих часових інтервалів, дорівнює:

$$T_1 = t_{ініціалізація} + \sum_{i=1}^{n}(t^i_{шагГА} + \sum_{j=1}^{m} t^j_{рцф1хр}), \quad (3)$$

де $t_{ініціалізація}$ – час ініціалізації;

n – кількість шагів генетичного алгоритму;

$t^i_{шагГА}$ – час виконання одного шагу генетичного алгоритму без розрахунку функції пристосованості для популяції;

m – кількість елементів в популяції, тобто кількість хромосом;

$t^j_{рцф1хр}$ – час розрахунку ЦФ для однієї хромосоми.

Для оцінки ефективності ПА звичайно використовуються наступні показники. Візьмемо T_p – час виконання ПА на паралельній обчислювальній системі з числом процесів $p > 1$. T_1 – час роботи "найкращого" послідовного алгоритму. Тоді:

$$S_p = \frac{T_1}{T_p} \le p; \quad W_p = \frac{S_p}{p} \le 1, \quad (4)$$

де S_p – прискорення; W_p – ефективність паралельного алгоритму.

Розглянемо час виконання ГА на паралельній машині з числом процесорів $p > 1$. Із усього ГА розпаралеленню підлягає та частина коду, де йде розрахунок ЦФ для усій популяції.

$$T_p = t_{ініціалізація} + \sum_{i=1}^{n}(t^i_{шагГА} + T^i_{обміну} + \sum_{j=1}^{\left\lceil \frac{m}{p-1} \right\rceil} t^j_{рцф1хр}) \quad (5)$$

Таким чином маємо:

$$S_p = \frac{t_{ініціалізація} + \sum_{i=1}^{n}(t^i_{шагГА} + \sum_{j=1}^{m} t^j_{рцф1хр})}{t_{ініціалізація} + \sum_{i=1}^{n}(t^i_{шагГА} + T^i_{обміну} + \sum_{j=1}^{\left\lceil \frac{m}{p-1} \right\rceil} t^j_{рцф1хр})} \le p, \quad (6)$$

де $T^i_{обміну}$ – сумарний час обміну даними між процесорами на кожному кроці ГА.

Розглянемо більше детально відношення часових інтервалів. Найбільш довгим по часу виконання в ГА для "складних" задач є час розрахунку ЦФ ($t^j_{рцф1хр}$); далі йде час виконання одного шагу ГА і час ініціалізації, при цьому час $t^j_{рцф1хр} >> t^i_{шагГА} > t_{ініціалізація}$. Таким чином час $t_{ініціалізація}$ не є значним у порівнянні з $\sum_{i=1}^{n}(t^i_{шагГА} + \sum_{j=1}^{m} t^j_{рцф1хр})$, особливо якщо вважати, що для сходження ГА треба зробити дуже велику кількість кроків і в популяції дуже велика кількість хромосом. Час $T^i_{обміну}$ складається з часу посилки сервером інформації о хромосомах клієнтам і посилки клієнтами розрахованої ЦФ цих хромосом серверу. Цей час є незначним у порівнянні зі складними та довгими розрахунками ЦФ та пересилка інформації між сервером та клієнтом йде послідовно, тобто в разі потреби. Якщо паралельна система буде працювати на кластері, то міжпроцесорний час зведено до мінімуму, а в локальної мережі час на передачу не є значним. Таким чином, $T^i_{обміну} << \sum_{j=1}^{\left\lceil \frac{m}{p-1} \right\rceil} t^j_{рцф1хр}$. Із вище викладеного виходить, що:

$$S_p = \frac{t_{ініціалізація} + \sum_{i=1}^{n}(t^i_{шагГА} + \sum_{j=1}^{m} t^j_{рцф1хр})}{t_{ініціалізація} + \sum_{i=1}^{n}(t^i_{шагГА} + T^i_{обміна} + \sum_{j=1}^{\left\lceil \frac{m}{p-1} \right\rceil} t^j_{рцф1хр})} \cong$$

$$\cong \frac{\sum_{i=1}^{n}(t^i_{шагГА} + \sum_{j=1}^{m} t^j_{рцф1хр})}{\sum_{i=1}^{n}(t^i_{шагГА} + T^i_{обміна} + \sum_{j=1}^{\left\lceil \frac{m}{p-1} \right\rceil} t^j_{рцф1хр})} = \frac{t^i_{шагГА} + \sum_{j=1}^{m} t^j_{рцф1хр}}{t^i_{шагГА} + T^i_{обміна} + \sum_{j=1}^{\left\lceil \frac{m}{p-1} \right\rceil} t^j_{рцф1хр}} \cong \frac{\sum_{j=1}^{m} t^j_{рцф1хр}}{\sum_{j=1}^{\left\lceil \frac{m}{p-1} \right\rceil} t^j_{рцф1хр}} = \frac{m}{\left\lceil \frac{m}{p-1} \right\rceil} \cong p-1$$

$$W_p = \frac{S_p}{p} = \frac{p-1}{p} = 1 - \frac{1}{p}$$

Припущення, які було зроблено в цій формулі є приблизними. Таким чином, виходить, що прискорення дорівнює числу процесорів-клієнтів, а ефективність розпаралелювання буде наближуватися до одиниці при збільшенні числа процесорів. Ефект від ПА буде дуже добрий, якщо час розрахунку ЦФ, кількість шагів ГА і число процесорів є дуже

великими.

Слід також зробити дві поправки, які пов'язані з ГА. По-перше, для ефективності роботи даного ГА потрібно використовувати число процесорів-клієнтів, яке менше чи дорівнює розміру популяції. При використані кількості процесорів більше, ніж число елементів в популяції зайві процесори не будуть використовуватись. Тобто число процесорів кількістю *p-1-m* не буде використовуватись взагалі. По-друге, якщо час $T_{ГА} + T_{обміну}$ порівняний з часом розрахунку ЦФ $t_{рцф1хр}$, то ефективність використання ПА значно зменшується; якщо $T_{ГА} + T_{обміну} >> t_{рцф1хр}$, то прискорення буде менше одиниці $S < 1$, тобто відбувається навіть сповільнення при використанні ПА і програма буде працювати довше, ніж без використання ПА.

Засоби й особливості реалізації ПА

Метою тестування було порівняльне дослідження ефективності організації паралельної оптимізації складних динамічних систем на базі технологій CORBA та MPI, а також дослідження впливу кеша ГА як на послідовну, так і на паралельну реалізацію ГА.

Для тестування використовувалися безкоштовні версії бібліотек MPI та CORBA: MPICH2 та omniORB версії 4.1 відповідно. Тестування проводилося на кластері з 30-ти машин на базі двоядерного Intel Pentium D 3GHz, що об'єднані локальною мережею за технологією Ethernet 100, під керуванням операційної системи SuSE Linux 10.1.

Для обміну даними використовувались масиви змінюваної довжини елементів типу double. Для бібліотеки MPI, природнім форматом зберігання динамічного масиву можна вважати ділянку пам'яті, у якій розташований цей масив. Стандарт об'єктно-орієнтованої CORBA рекомендує використання шаблонного класу-контейнеру sequence для операцій над динамічними масивами. Саме цей клас був взятий за основу при проведенні тестувань.

У рамках паралельного модуля, що побудований на базі CORBA технології також були реалізовані наступні додаткові можливості: динамічне підключення нових обчислювальних вузлів (ОВ), відновлення після "втрати" одного чи більше ОВ, динамічне балансування навантаження за принципом мінімального середнього часу обчислення ЦФ обчислювальним вузлом.

Специфіка паралельного ГА полягає у тому, що розрахунок ЦФ здійснюється виключно засобами ОВ, при цьому основна гілка ГА очікує, доки усі ЦФ будуть обчислені. Це робить можливим запуск основної гілки ГА та одного з ОВ на тому самому процесорі, адже конкуренція за обчислювальний ресурс процесора у цих двох процесів зведена до

мінімуму (окрім того, для проведення експериментів використовувались двоядерні процесори, отже ці два процеси будуть виконуватись на різних ядрах процесора). Саме тому для основної гілки ГА під час тестування не відводилося окремого процесору.

Експериментальна оцінка роботи ПА

Конфігурація обчислювальних ресурсів позначена у форматі *C*x*P*, де *C* – кількість комп'ютерів, а *P* – кількість процесорів на кожному комп'ютері, що застосовувались для обчислень.

Тестування простої штучної ЦФ "сфера" (рис. 3) виявило, що її розпаралелювання не є ефективним: прискорення не зростає, більш того, воно завжди є меншим за одиницю (0.7 для MPI, 0.4 для CORBA – ПА працює довше за послідовний варіант для сфери). Такі результати пов'язані з тим, що ця ЦФ є занадто простою, тобто час на її обчислювання є дуже незначним, разом з тим, кількість обчислень є досить великою. Тому в даному випадку, час, який витрачається на обмін, порівняний з часом обчислювання ЦФ, що приводить до неефективності ПА для обчислювання простих функцій.

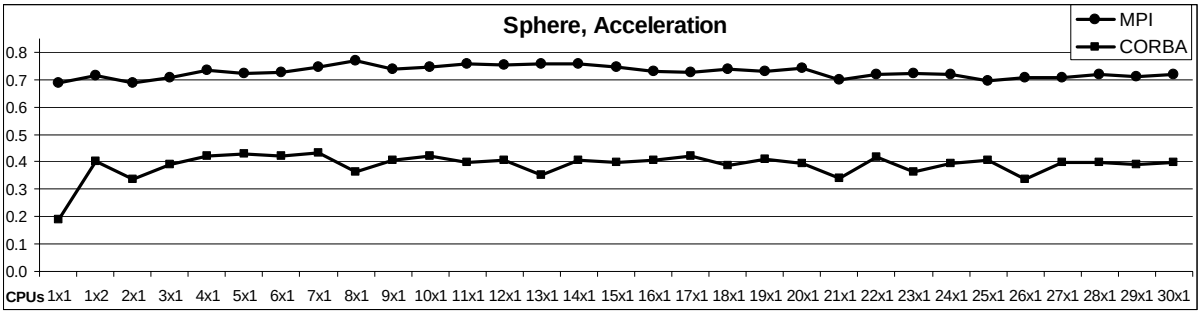

Рисунок 3 – Прискорення ПА для штучної ЦФ "сфера"

Результати тестування моделі Лотка-Вольтерра (рис. 4) свідчать про можливість її розпаралелювання, але в певних межах. Для цієї моделі спостерігається насичення прискорення (воно більше не зростає) на значенні приблизно 6.5 при 14 та більше процесорах. Проаналізувавши роботу ГА по шагах, виявилося, що середня кількість необхідних обчислень ЦФ на кожному кроці ГА при оптимізації параметрів цієї моделі складає в середньому 17. Оскільки розпаралелюванню підлягає кожний крок ГА окремо, то очевидно, що, враховуючи час обміну та втрати на "проблемі кратності" (кількості обчислень на кожному кроці ГА та кількості процесорів), це і є межею для прискорення. Таким чином, розпаралелювання для цієї моделі є досить ефективним до 14 використаних процесорів, після цього збільшення кількості процесорів для обчислення є недоцільним.

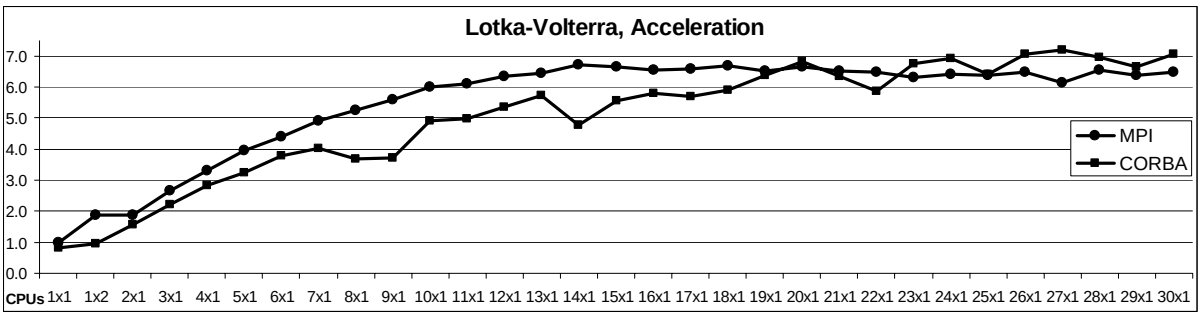

Рисунок 4 – Прискорення ПА для моделі Лотка-Вольтерра

Для моделі реактора Хафке розпаралелювання дало найбільший ефект. Зростання прискорення спостерігалося упродовж всіх тестів (рис. 5). Це пояснюється по-перше складністю моделі (час обчислення ЦФ є досить суттєвим, послідовний ГА виконує оптимізацію на вказаному типі машин майже 12 хвилин), тому час обміну майже не впливає на прискорення. По-друге, для ідентифікації параметрів цієї моделі ГА потребує більшої кількості обчислень ЦФ на кожному кроці (ніж для моделі Лотка-Вольтерра), тому "проблема кратності" виникає не так часто, і її негативний вплив також зменшується. Отже, для оптимізації моделі реактора Хафке для 30 процесорів маємо прискорення 19, і воно ще продовжує зростати, на відміну від моделі Лотка-Вольтерра, не зважаючи на те, що середня кількість необхідних обчислень ЦФ для моделі реактора Хафке складає 29. Для цієї моделі також можливе існування точки насичення прискорення, але вона є значно дальше, не раніше ніж на 50 або більше процесорах.

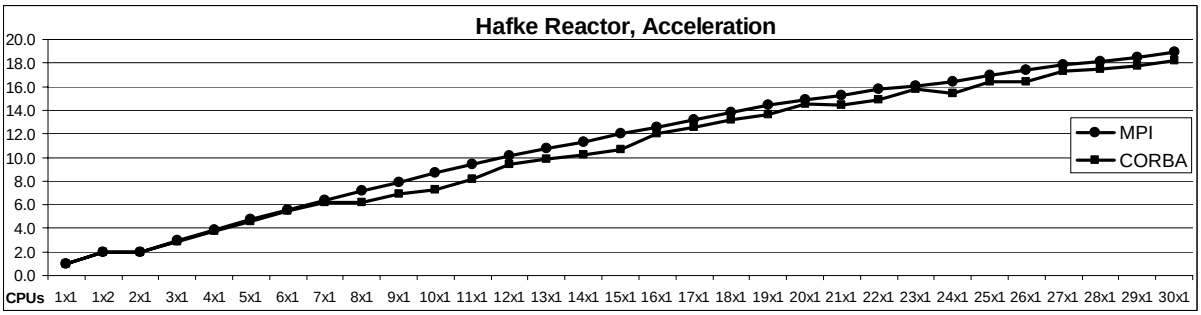

Рисунок 5 – Прискорення ПА для моделі реактора Хафке

Порівнюючи ефективність розпаралелювання для моделей Лотка-Вольтерра та реактора Хафке (рис. 6), можна ще раз зазначити, що розпаралелювання моделі реактора Хафке є більш ефективним (навіть для 30 процесорів ефективність ПА все ще складає більше 60% на відміну від 20% для моделі Лотка-Вольтерра). Більше падіння ефективності для моделі Лотка-Вольтерра пов’язано із зазначеним насиченням прискорення та більшим впливом часу обміну, оскільки ця модель є досить простою та моделюється значно швидше за модель реактора Хафке.

Загальне зменшення ефективності зі зростанням кількості

процесорів пояснюється тим, що при цьому зростає вплив "проблеми кратності", а також збільшується кількість операцій обміну.

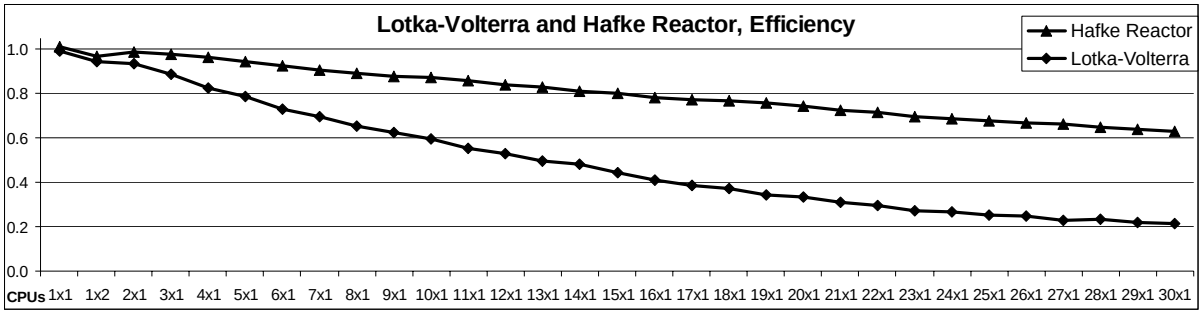

Рисунок 6 – Ефективність ПА для моделей Лотка-Вольтерра та реактора Хафке

З проведених тестів можна зробити такі висновки відносно використаних технологій для розпаралелювання. MPI має дуже незначний час обміну між машинами, тоді як витрати на обмін при використанні технології CORBA виявилися більш суттєвими. Але при виконанні великих обсягів обчислень (для складних ЦФ, таких як для моделі реактора Хафке) вплив обміну не є таким важливим. Також з результатів тестів випливає, що час роботи паралельного ГА (а разам з ним прискорення і ефективність) при використанні MPI є досить стабільним, тоді як при використанні CORBA спостерігаються досить великі відхилення, які пов’язані перш за все з нерівномірним розподіленням значень часу обміну клієнтів з сервером (це краще видно на моделі Лотка-Вольтерра, рис. 4). Але технологія CORBA дозволяє динамічне додавання клієнтів під час виконання та зберігає стабільність системи в цілому при виході з ладу якогось клієнту. Отже, потрібно використовувати технологію MPI для більшості тестів у зв’язку зі стабільністю її результатів та незначним часом на обмін, та використовувати технології CORBA там, де фактор стабільності системи взагалі упродовж тривалого часу грає суттєву роль.

Із графіків також виходить, що ефективність паралельного ГА при використанні двох комп’ютерів є вищою, ніж ефективність використання комп’ютера з двоядерним процесором (конфігурації 2x1 та 1x2 відповідно). Оскільки обсяг даних, що передавалися, був незначний, очевидно, що така перевага пов’язана з обчислювальною потужністю комп’ютерів. Дійсно, двоядерний процесор поступається у потужності двом окремим процесорам. Не слід також забувати про основний потік ГА, який в останньому випадку виконувався на окремому ядрі. Також, для CORBA-технології, з її досить суттєвим часом на обмін, різниця в ефективності ще збільшується, оскільки при конфігурації 1x2 весь обмін виконується на одному комп’ютері.

Також було проведено тести для виявлення впливу реалізованого внутрішнього кешу ГА на ефективність ПА. Кеш виконує збереження вже

обчислених ЦФ, і якщо на якійсь генерації виникає такий же самий елемент популяції, то для нього ЦФ вже не обчислюється а береться з кешу, відповідно цей елемент вже не передається на обчислення (в тому числі не передається і ПА). Це досить суттєво зменшує кількість необхідних обчислень ЦФ (див. табл. 1).

Таблиця 1. Порівняння кількості обчислень ЦФ при використанні кешу ГА

	Кількість генерацій	Кількість оптимізованих параметрів	Кількість обчислень ЦФ	
			Без кешу	З кешем
Функція "сфера"	160	10	12190	9530
Модель Лотка-Вольтерра	75	2	1372	451
Модель реактора Хафке	90	4	2712	1647

Зменшення кількості необхідних обчислень ЦФ звісно зменшує час роботи як послідовного, так і паралельного ГА. Але разом з цим, для ПА зменшуються показники прискорення та ефективності (рис. 7).

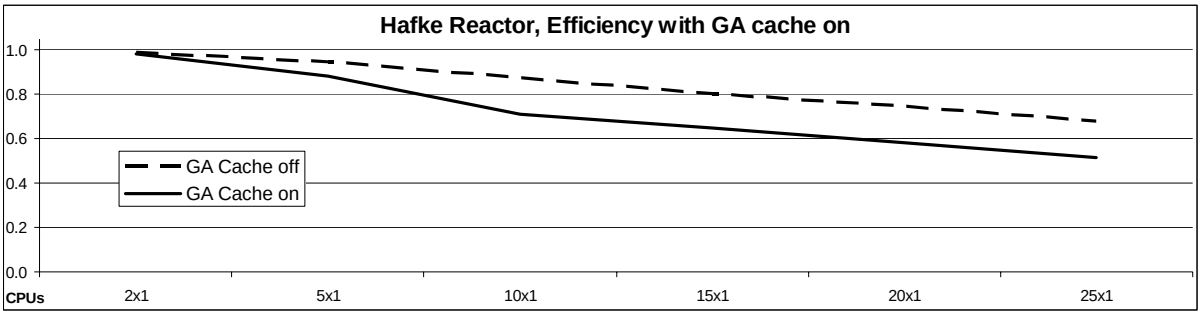

Рисунок 7 – Ефективність ПА для моделі реактора Хафке при включеному кеші ГА

Це пов’язано з тим, що загальне зменшення обчислень ЦФ є результатом зменшення кількості обчислень ЦФ на кожному кроці ГА, що призводить до частішого виникнення "проблеми кратності". Для більш детального дослідження цього питання було розглянуто цикл оптимізації ГА по кроках. Для кожного кроку було виконано заміри часу на його виконання, а потім виконано розрахунок прискорення та ефективності для кожного кроку при паралельному виконанні відносно його послідовного аналога. Результати одного з таких тестів зображені на рис. 8.

Виходячи з результатів (рис. 8), існують досить сильні коливання обчислень ЦФ в кожній генерації, а разом з ними і коливання ефективності ПА (як із включеним кешем, так і без нього). Форма коливань ефективності достатньо близько відповідає формі коливань обчислень ЦФ (за рахунок зменшення загального часу обчислення при меншій кількості обчислень), але має деякі відхилення, обумовлені зміною впливу "проблеми кратності" (можливе як зменшення, так і збільшення ефективності) та зменшенням кількості операцій обміну при зменшенні кількості обчислень ЦФ.

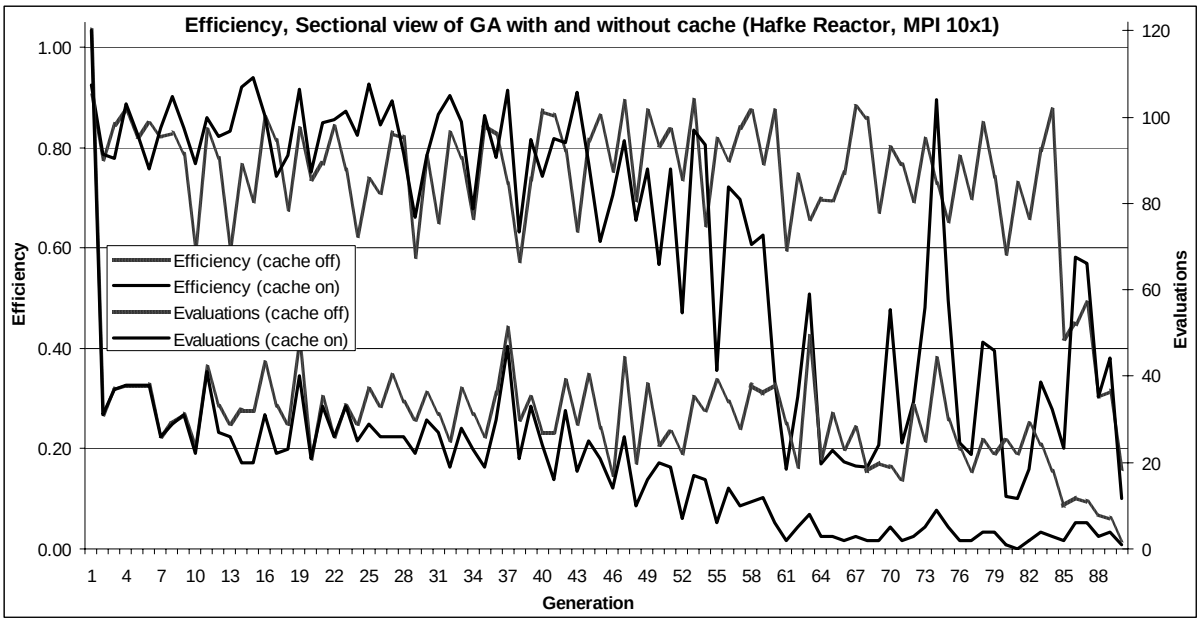

Рисунок 8 – Ефективність ПА по генераціях порівняно до кількості обчислень ЦФ

Звертаючи увагу на порівняння результатів з включеним кешем та без нього, можна помітити, що протягом приблизно першої половини роботи алгоритму з включеним кешем кількість обчислень ЦФ не дуже змінюється в порівнянні з відключеним кешем (зменшується, але не досить сильно). При цьому ефективність у більшості випадків тільки зростає (адже зменшується загальний час на обчислення). Але в другій половині роботи ГА спостерігається значне зменшення необхідних обчислень ЦФ за рахунок включеного кешу. Це обумовлено тим, що, по-перше, кеш набрав уже достатньо велику базу обчислених значень ЦФ, а по-друге, сама специфіка роботи ГА обумовлює сходження рішень, тобто вони частіше за все вже обираються не з такого широкого кола можливих значень, отже вони частіше повторюються в генераціях. При цьому суттєвому зменшенні кількості обчислень ЦФ спостерігається також суттєве зниження ефективності (в окремих випадках ефективність може повернутися або навіть перевищити значення без кешу, але взагалі ефективність все ж таки знижується). За рахунок цього суттєвого зниження ефективності у другій половині роботи ГА його загальна (середня) ефективність також зменшується, що і пояснює результати на рис. 7.

Висновки

Таким чином, було апріорно оцінено та експериментально доведено ефективність розробленого паралельного генетичного алгоритму для вирішення задач оптимізації складних динамічних систем, особливо для ідентифікації параметрів складних моделей, таких як модель реактора Хафке. Було проаналізовано ефективність різних засобів

розпаралелювання, таких як стандарти MPI та CORBA, з яких MPI виявився більш швидким та ефективним (оскільки має менший час на обмін даними), проте CORBA можна використовувати там, де фактор стабільності системи взагалі упродовж тривалого часу грає суттєву роль, оскільки вона дозволяє динамічне додавання клієнтів під час виконання та зберігає стабільність системи в цілому при виході з ладу якогось клієнту. Також було проаналізовано та обґрунтовано вплив кешу ГА на ефективність ПА (зменшується загальний час роботи алгоритму, але при цьому дещо зменшується ефективність ПА).

У цілому, використання ПА для вирішення задач оптимізації складних динамічних систем є дуже ефективним. Планується використання розробленої системи для оптимізації систем ще більшої складності.

Література

1. K. Teplinskiy, V. Trubarov, V. Svjatnyj. Optimization Problems in the Technological-Oriented Parallel Simulation Environment. 18-th ASIM-Symposium Simulationtechnique, pages 582-587. SCS Publishing House, Erlangen, 2005.
2. Th. Bäck. Evolutionary Algorithms in Theory and Practice. Oxford University Press, Inc., New York, 1996.
3. Svjatnyj V.A., Rasinkov V.V., Bräunl T., Reuter A., Zeitz M. Problemorientierte massiv parallele Simulationsumgebung für dynamische Netzobjekte. Tagungsband 10. Symposium Simulationstechnik ASIM '96 in Dresden, pp. 515-518.
4. Lotka-Volterra equation (http://en.wikipedia.org/wiki/Lotka_volterra).
5. Hafke, C; Gilles, E. D. Mess., Steuern, Regeln 1968, 11, 204.
6. J.H. Holland. Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, MI, 1975.
7. D.E. Goldberg. Genetic Algorithms in Search, Optimizations and Machine Learning. Addison Wesley, Reading, MA, 1989.
8. Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, 1995.
9. William R. Stanek. Distributed Programming with CORBA and IIOP.
10. K.-P. Zeyer, M. Mangold, S. Shah, A. Kienle, E.D. Gilles.Yield Effects in Single and Coupled Nonliniear Thermokinetic Reaction Systems. Z.Phys. Chem. 216 (2002) 403-433
11. K.-P. Zeyer, M. Mangold, T. Obertopp, E.D. Gilles. Iron(III)-Catalyzed Oxidation of Ethanol by Hydrogen Peroxide: A. Thermokinetic Oscillator. J. Phys. Chem. A 1999, 103, 5515-5522

Дата надходження до редакції 07.11.2007 р.