УДК 519.863

С. Д. Штовба, к. т. н., доц.;
О. М. Рудий

МУРАШИНІ АЛГОРИТМИ ОПТИМІЗАЦІЇ

Вступ

Метою статті є ознайомлення читачів з теорією та практичним застосуванням мурашиних алгоритмів оптимізації — нового перспективного методу, що інтенсивно розроблюється на Заході, але є майже невідомим в Україні. Основні ідеї цього методу оптимізації ґрунтуються на принципах поведінки колонії біологічних мурах. Мурашині алгоритми запропоновані на початку дев'яностих [1]. Перша стаття з мурашиних алгоритмів в міжнародному науковому журналі надрукована в 1996 р. [2]. Знаковою подією у визнанні доцільності досліджень в області мурашиної оптимізації є рішення Європейської комісії про присудження в 2003 р. премії Фонду Марії Кюрі за видатні наукові дослідження в розмірі 50000 винахіднику мурашиних алгоритмів доктору Марко Доріго.

В статті наводиться теоретичне підґрунтя мурашиних алгоритмів, ілюструється використання мурашиних алгоритмів на прикладі розв'язання задачі комівояжера та дається огляд застосувань мурашиних алгоритмів оптимізації. Теоретична частина статті базується на [2—5].

1. Принципи поведінки мурах

Мурахи належать до так званих «соціальних комах», тобто комах, що живуть в межах певного колективу — сім'ї або колонії. На Землі біля двох відсотків комах є «соціальними», половина з яких припадає на мурах. Поведінка мурах під час транспортування їжі, обминання перешкод, побудови мурашника тощо наближається до теоретично оптимальної. Основу «соціальної» поведінки мурах складає *самоорганізація* — сукупність динамічних механізмів, за допомогою яких система досягає глобальної мети в результаті взаємодії елементів на низькому рівні. Принциповою особливістю такої низькорівневої взаємодії є використання елементами системи *лише локальної інформації*, без будь-якого централізованого управління та звернення до глобального образу, який репрезентує систему у зовнішньому світі. Самоорганізація є результатом взаємодії таких чотирьох компонентів: 1) випадковість; 2) додатний зворотний зв'язок; 3) від'ємний зворотний зв'язок; 4) багатократність взаємодій.

Мурахи використовують два способи передачі інформації: прямий — обмін харчами, мандибулярний, візуальний та хімічний контакти тощо, та непрямий — стигмержі (stigmergy). Стигмержі — це рознесений у часовому просторі тип взаємодії між елементами системи, коли один суб'єкт взаємодії змінює деяку частину оточуючого середовища, а решта використовує інформацію про її стан пізніше, коли знаходяться в її околі. Біологічно, стигмержі здійснюється через феромон (pheromone) — спеціальний секрет, який відкладається як слід під час руху мурахи. Чим вища концентрація феромону на стежці, тим більше мурах буде рухатись по ній. Феромон з часом випаровується, що дозволяє мурахам адаптувати свою поведінку до зміни оточуючого середовища. Розподіл феромону по простору пересування мурах є своєрідною глобальною пам'яттю, яка динамічно змінюється. Кожна мураха може сприймати та змінювати лише локальну чарунку цієї глобальної пам'яті мурашника.

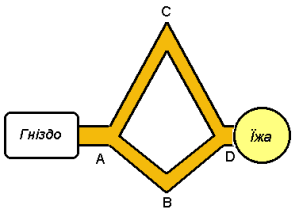

Рис. 1. Асиметричний міст (за матеріалами [6])

Розглянемо, як колективна поведінка біологічних мурах забезпечує знаходження найкоротшого шляху до їжі на прикладі експериментів на асиметричному мості [6]. Асиметричний міст (рис. 1) з'єднує гніздо мурах з джерелом їжі двома гілками різної довжини. Експерименти [6] проводилися за такою схемою: 1) будувався міст A-B-C-D; 2) відчинялися дверцята в точці A; 3) фіксувалась кількість мурах, які обрали довгий (A-C-D) та короткий (A-B-D) шляхи. На початку експериментів мурахи обирали обидві гілки з однаковою ймовірністю тому, що на мості не було феромонів. Через деякий час майже усі мурахи пересувались найкоротшим маршрутом A-B-D, що пояснюється таким чином. Мурахи, які обрали короткий маршрут A-B-D-B-A, скоріше повертались з їжею в гніздо, залишаючи феромонні сліди на короткій гілці мосту. При наступному виборі маршрута, мурахи віддавали перевагу короткій гілці мосту, тому що на ній вища концентрація феромонів. Таким чином, феромон швидше накопичується на гільці A-B-D, що підштовхує мурах до вибору найкоротшого маршруту.

2. Мурашиний підхід до розв'язання задачі комівояжера

Задача комівояжера полягає у виборі найкоротшого замкнутого шляху, що проходить через усі міста рівно один раз. Розглянемо, як реалізувати чотири основних компоненти самоорганізаційної поведінки мурах під час оптимізації маршруту комівояжера.

Багатократність взаємодії реалізується ітераційним пошуком маршруту комівояжера одночасного декількома мурахами.

Додатний зворотний зв'язок реалізується як імітація природної поведінки мурах типу «залишання слідів – пересування по слідах». Чим більше слідів залишено на стежці — ребрі графу, тим більше мурах буде рухатись по ній. При цьому на стежці з'являються нові сліди, які приваблюють додаткових мурах. Для задачі комівояжера додатний зворотний зв'язок реалізується таким стохастичним правилом: «ймовірність включення ребра графу в маршрут мурахи пропорційна кількості феромону на ній». Використання цього стохастичного правила забезпечує реалізацію і іншого компоненту поведінки мурах – *випадковості*. Кількість феромону, який відкладає мураха на ребрі графа, є зворотно пропорційною величиною до довжини відповідного маршруту комівояжера. Чим коротший маршрут комівояжера знайшла мураха, тим більше феромону буде відкладене на відповідних ребрах графу.

Використання лише додатного зворотного зв'язку призводить до передчасної збіжності алгоритму, тобто до випадку, коли усі мурахи рухаються одним і тим же субоптиальним маршрутом. Для уникнення цього використовується *від'ємний зворотний зв'язок* – випаровування феромону. Час

випаровування феромону не повинен бути дуже великим, бо при цьому виникає загроза збігання маршрутів усіх мурах до одного субоптимального розв'язку. З іншого боку, час випаровування не повинен бути і малим, щоб не призвести до некооперативної поведінки мурах через втрату пам'яті колонії.

Перехід мурахи з міста i в місто j на ітерації t алгоритму залежить від трьох складових: табу-списка, видимості та віртуального сліду феромону.

Табу-список — це перелік міст, які вже відвідані мурахою і заходити в які ще раз заборонено. Табу-список збільшується зі здійсненням маршруту та заповнюється нулями на початку кожної ітерації алгоритму. Позначимо через J_i^k список міст, які ще потрібно відвідати мурасі k, що перебуває у місті i. Зрозуміло, що об'єднання цих списків дає множину усіх міст з маршруту комівояжера.

Видимість – це величина обернена до відстані: $\eta_{ij} = 1/D_{ij}$, де D_{ij} — відстань між містами i та j. Видимість базується тільки на локальній інформації і являє собою «евристичну бажаність» вибору міста j, під час перебування у місті i. Чим ближче міста i та j, тим більше бажання відвідати їх.

Віртуальний слід феромону на ребрі $(i-j)$ являє собою «бажаність, підкріплену досвідом» переходу в місто i з міста j. Інформація, яку несе слід феромону, змінюється підчас оптимізації і відображає набутий мурахами досвід. Кількість віртуального феромону на ребрі $(i-j)$ на t-й ітерації алгоритму позначимо як $\tau_{ij}(t)$.

Ймовірність переходу k-ї мурахи з міста i у місто j на t-й ітерації розраховується за випадково-пропорційним правилом:

$$\begin{cases} P_{ij}^k(t) = \dfrac{\left[\tau_{ij}(t)\right]^\alpha \left[\eta_{ij}\right]^\beta}{\sum\limits_{l \in J_i^k} \left[\tau_{il}(t)\right]^\alpha \left[\eta_{il}\right]^\beta}, & \text{якщо } j \in J_i^k; \\ P_{ij}^k(t) = 0, \qquad \text{якщо } j \notin J_i^k. \end{cases} \tag{1}$$

де α і β – два регульовані параметри, які є вагами інтенсивності сліду феромону та видимості. Якщо $\alpha = 0$, то найвірогіднішим буде перехід у найближчі міста. У класичній теорії оптимізації це відповідає, так званому, пожадливому алгоритму. Якщо $\beta = 0$, тоді працює лише феромоне підсилення, що призводить до швидкого завершення роботи алгоритму через збігання маршрутів усіх мурах до одного субоптимального розв'язку.

Після завершення маршруту кожна мураха k відкладає на ребро $(i-j)$ таку кількість феромону:

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)}, & \text{якщо } (i,j) \in T^k(t) \\ 0, & \text{якщо } (i,j) \notin T^k(t) \end{cases},$$

де $T^k(t)$ – маршрут, зроблений мурахою k на ітерації t; $L^k(t)$ — його довжина; Q — регульований параметр, значення якого обирають одного порядку з довжиною оптимального маршруту.

Для забезпечення можливості експлуатації простору рішень потрібно забезпечити випаровування феромону — зменшення кількості відкладеного на попередніх ітераціях феромону. Інтенсивність випаровування феромону задається за допомогою коефіцієнта випаровування $p \in [0, 1]$. Кінцеве правило оновлення феромону, яке стосується всіх ребер, приймає вигляд

$$\tau_{ij}(t+1) \leftarrow (1-p)\,\tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{2}$$

де $\Delta\tau_{ij}(t) = \sum\limits_{k=1}^{m} \Delta\tau_{ij}^k(t)$; m – кількість мурах в колонії.

На початку оптимізації кількість феромону на ребрах приймається за малу додатну константу τ_0. Загальна кількість мурах в колонії приймається постійною на весь час розв'язання задачі. Забагато мурах призводить до швидкого підсилення субоптимальних маршрутів. Коли ж мурах замало, виникає небезпека втрати кооперативної поведінки через швидке випаровування феромону. Зазвичай кількість мурах приймають рівною числу міст — кожна мураха починає маршрут з окремого міста.

Для покращення характеристик мурашиного алгоритму використовують елітних мурах, які підсилюють ребра найкращого маршруту T^+, знайденого від початку пошуку. Кількість феромону, що відкладається на ребрах маршруту T^+, приймається рівною Q/L^+, де L^+ — довжина маршруту T^+. Підсилений слід феромону вздовж маршруту T^+ буде спрямовувати інших мурах до пошуку розв'язків, що містять декілька ребер найкращого на даний момент маршруту T^+. Якщо в мурашнику є e елітних мурах, то ребра найкращого маршруту T^+ отримають загальне підсилення $e\,Q/L^+$. Ідеї елітизму знайшли розвиток в рангових мурашиних алгоритмах (Rank-Based Ant Systems) [7], алгоритмах мурашиної колонії (Ant Colony Systems) [8], макс-мінних мурашиних алгоритмах (MAX-MIN Ant Systems) [9]. Ці алгоритми оптимізують швидше за рахунок збільшення ймовірностей вибору перспективних фрагментів маршрутів.

Під час розв'язання задач великої розмірності доцільно використовувати список кандидатів — короткий перелік рекомендованих вершин, в які може перейти мураха з даної вершини. Мураха обирає не рекомендовану вершину лише тоді, коли вона вже пройшла увесь список кандидатів. Список кандидатів формують евристично на основі апріорних знань про задачу, що вирішується, або на базі інформації, що динамічно оновлюється під час оптимізації. Список кандидатів дозволяє виключити заздалегідь неперспективні варіанти. Це дозволяє направити мурах на дослідження найобіцяніших маршрутів і тим самим значно скоротити область пошуку. Наприклад, для задачі комівояжера з 2392 містами «Pr2392» [10] оптимальний розв'язок можна знайти, якщо досліджувати продовження маршрутів у 8 сусідніх міст [11].

Для прискорення мурашиних алгоритмів залучають методи локального пошуку, які намагаються покращити знайдені мурахами розв'язки. Для задачі комівояжера часто застосовують процедури локального пошуку 2-opt, 3-opt та Лін-Кернігхана (Lin-Kernighan), які покращують маршрут заміною двох, трьох та змінного числа дуг, відповідно.

3. Мурашиний алгоритм оптимізації маршруту комівояжера

Нижче наводиться базовий мурашиний алгоритм оптимізації маршруту комівояжера, в якому втілені основні ідеї попереднього розділу.

Ініціалізація параметрів алгоритму α, β, e, Q, τ_0.
$m = n$ {Кількість мурах дорівнює кількості міст}
For i = 1 to n
  For j = 1 to n {Для кожного ребра}
    If $i <> j$
      $\eta(i,j) = 1/D(i,j)$ {Видимість}
      $\tau(i,j) = \tau_0$ {Феромон}
    Else $t(i,j) = 0$
    End
  End
End
For k = 1 to m
  Розмістити мураху k у випадково обране місто.
End
Обрати умовно найкоротший маршрут T^+ та обчислити його довжину L^+.
{Головна програма}
For t = 1 to t_{max} {Кількість ітерацій алгоритму}
  For k = 1 to m {Для кожної мурахи}
    Побудувати маршрут $T^k(t)$ за правилом (1) та обчислити його довжину $L^k(t)$.
  End
  If «Кращий розв'язок знайдено»
    Оновити T^+ та L^+.
  End
  For i = 1 to n
    For j = 1 to n {Для кожного ребра}

Оновити сліди феромону за правилом (2).

End

End

End

Вивести найкоротший маршрут T^+ та його довжину L^+.

4. Комп'ютерні експерименти

Наведений вище мурашиний алгоритм було протестовано на серії задач з бібліотеки [10]. Для задач невеликої розмірності мурашиний алгоритм швидко знаходить оптимальний розв'язок. Наприклад, для задачі про обхід 29 населених пунктів в Баварії Bays29 мурашиний алгоритм протягом 100 ітерацій знаходить глобальний розв'язок в чотирьох випадках з п'яти. Для складніших задач, наприклад Berlin52, за 100 ітерацій вдається знайти розв'язки, які близькі до оптимального маршруту комівояжера. Розв'язки можна покращити простим збільшенням кількості ітерацій до 1—2 тисяч.

Проведені нами експерименти свідчать, що популяція розв'язків ніколи не збігається до єдиного, спільного для усіх мурах маршруту. Навпаки, алгоритм продовжує створювати нові, можливо кращі рішення. На рис. 2 показані результати дослідження розв'язання задачі Bays29 алгоритмом з 5 елітними мурахами. Оптимальний розв'язок знайдено на 44-й ітерації (рис. 2а). Але і після цього алгоритм продовжує підтримувати диверсифікацію (різноманітність) розв'язків. На рис. 2б показано довжини найкоротших маршрутів, знайдених на кожній ітерації алгоритму. Графіки на рис. 2а та рис. 2б не співпадають, значить алгоритм продовжує синтез нових маршрутів. На рис. 2в показано середнє квадратичне відхилення довжин маршрутів на кожній ітерації. Графік проходить далеко від нуля, що також свідчить про різноманітність маршрутів на кожній ітерації. На рис. 2г показано середня по містах кількість розгалужень слідів феромону на кожній ітерації алгоритму. Кількість розгалужень визначається підрахунком для кожної вершини графа ребер, на яких слід феромону перевищує деякий поріг. Число розгалужень слідів феромону коливається біля 4, тобто в кожному місті мураха має декілька перспективних альтернатив продовження маршруту.

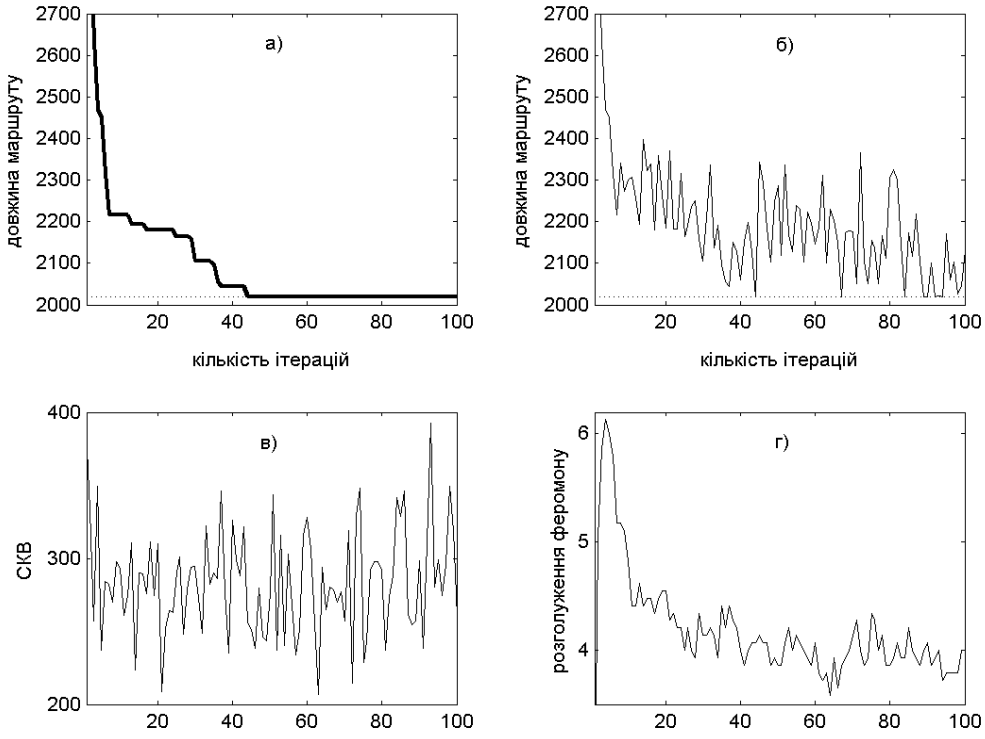

Рис. 2. Дослідження мурашиного алгоритму на задачі Bays29
а) еволюція найкращих розв'язків, знайдених від початку роботи алгоритму;
б) найкращі маршрути на кожній ітерації; в) розкид розв'язків;
г) середнє по містах число розгалужень слідів феромону.

В порівнянні з точними методами комбінаторної оптимізації, такими, як динамічне програмування та метод гілок та меж, мурашині алгоритми мають значну перевагу. Вже для порівняно невеликої розмірності задачі комівояжера мурашині алгоритми знаходять досить близький до оптимуму розв'язок за значно менший час. Час роботи мурашиного алгоритму є поліноміальним — $O(t,\cdot n^2,\cdot m)$, тоді як для точних методів ця залежність експоненційна [12].

В табл. 1 порівнюються розв'язки задач комівояжера з бібліотеки [10] такими мурашиними алгоритмами: AS – базовий мурашиний алгоритм; ASE – базовий мурашиний алгоритм з елітними мурахами; ASR – ранговий мурашиний алгоритм; ACS – алгоритм мурашиної колонії; MMAS — макс-мінний мурашиний алгоритм. Алгоритми синтезували однакову кількість маршрутів: $10000 \cdot n$ для симетричних задач Eil51, KroA100 і Dl98 та $20000 \cdot n$ для асиметричних задач Ry48p, Ft70, Kro124p і Ftv170. Числа в назвах задач вказують кількість міст (n). Числа в чарунках таблиці – усереднені за 25 прогонів довжини знайдених найкоротших маршрутів.

Таблиця 1

Розв'язки задач комівояжера різними мурашиними алгоритмами [9]

Задача	Eil51	Kroa100	Dl98	Ry48p	Ft70	Kro124p	Ftv170
Оптимум	426	21282	15780	14422	38673	36230	2755
MMAS	427,6	21320,3	15972,5	14553,2	39040,2	36773,5	2828,8
ACS	428,1	21420	16054	14565,4	39099	36857	2826,5
ASR	434,5	21746	16199,1	14511,4	39410,1	36973,5	2854,2
ASE	428,3	21522,8	16205	14685,2	39261,8	37510,2	2952,4
AS	437,3	22471,4	16702,1	15296,4	39596,3	38733,1	3154,5

5. Огляд застосувань мурашиних алгоритмів оптимізації

Наведений в статті мурашиний алгоритм оптимізації маршруту комівояжера після незначних модифікацій може використовуватися для розв'язання різних задач комбінаторної оптимізації. Для цього задачі комбінаторної оптимізації необхідно звести до пошуку найкоротшого шляху на деякому графі, обрати способи ініціалізації та оновлення феромону, та визначити евристичні правила вибору маршруту. На сьогодні вже отримані гарні результати розв'язання таких задач, як: квадратична задача про призначення, задача календарного планування, задача оптимізації транспортних маршрутів, задача календарного планування, задача розфарбування графа, задача про найкоротшу спільну суперпосідовність та інших. Мурашині алгоритми знаходять розв'язки дискретних задач не гірше за інші загальні метаевристичні технології оптимізації та деякі спеціалізовані методи. При цьому забезпечується добрий баланс між точністю розв'язку та часом оптимізації. Як приклад нижче порівнюються метаевристичні методи розв'язання квадратичної задачі про призначення (табл. 2) та задачі оптимізації транспортних маршрутів (табл. 3). Числа в чарунках таблиці є значеннями критерію оптимальності для розв'язків, знайдених відповідними методами оптимізації. Напівжирним шрифтом виділені найкращі на сьогодні розв'язки.

Таблиця 2

Порівняння евристичних методів дискретної оптимізації для квадратичної задачі про призначення [3]

Тестова задача	Nugent (12)	Nugent (15)	Nugent (20)	Nugent (30)	Elshafei (19)	Kraup (30)
Імітований відпал	**578**	**1150**	**2570**	6128	17937024	89800
Табу-пошук	**578**	**1150**	**2570**	**6124**	**17212548**	90090
Генетичні алгоритми	588	1160	2688	6748	17640584	108830
Еволюційні стратегії	598	1168	2654	6308	19600212	97880
Мурашині алгоритми без локального пошуку	**578**	**1150**	2598	6232	18122850	92490
Мурашині алгоритми з локальним пошуком	**578**	**1150**	**2570**	6128	**17212548**	**88900**

Таблиця 3

Порівняння результатів розв'язання складних задач оптимізації транспортних маршрутів [13]

Кількість клієнтів	Ранговий мурашиний алгоритм з локальним пошуком і декомпозицією задачі		Генетичний алгоритм	Гранульований табу-пошук
	Середній розв′язок за 10 прогонів	Найкращий розв′язок		
200	**6460,98**	**6460,98**	**6460,98**	6697,53
255	589,28	**586,87**	596,89	593,35
280	8437,44	8412,90	8412,90	8963,22
300	1007,81	**1007,07**	1018,74	1016,83
360	1368,92	**1367,20**	1385,6	1400,96
399	932,58	**927,27**	933,74	936,04
420	1836,87	**1834,79**	1846,55	1915,83
480	13958,68	13816,98	13728,8	14910,62

Серед прикладних застосувань мурашиних алгоритмів виділимо роботи з складання розкладів університетських занять [14], розміщення даних в пам'яті суперкомп'ютера [15], моделювання структури протеїна за його амінокислотним ланцюжкам [16], багатокритеріального проектування водних іригаційних мереж [17], навчання Байєсовських мереж [18] та логічних правил [19]. Високу ефективність демонструють мурашині алгоритми під час оптимізації розподілених систем з параметрами, що динамічно змінюються. Яскравим прикладом успішного застосування мурашиних алгоритмів до таких систем є знаходження оптимальних маршрутів трафіків в телекомунікаційних мережах [20]. В табл. 4 порівнюється ефективність алгоритмів маршрутизації в американській мережі NSFNET з інтенсивним завантаженням. Порівнювались такі алгоритми: AntNet — мурашиний алгоритм; OSRF — офіційний інтернетовський алгоритм маршрутизації; Daemon – апроксимація ідеального алгоритму маршрутизації; SRF — алгоритм, що використовує динамічну метрику під час оцінювання вартості з′єднань; BF — алгоритм Беллмана-Форда.

Таблиця 4

Порівняння алгоритмів маршрутизації для мережі NSFNET [20]

Алгоритм	AntNet	OSRF	SRF	Daemon	BF
Середня затримка передачі повідомлення, с	0,93	5,85	3,58	0,10	4,27
Пропускна здатність, 10^7, біт/с	2,392	2,100	2,284	2,403	1,410

Детальнішу інформацію про ці та інші застосування мурашиних алгоритмів оптимізації можна знайти в книзі [3], оглядових статтях [21, 22] та на сайті [5].

Висновки та майбутні дослідження

Основні ідеї мурашиних алгоритмів ґрунтуються на імітації самоорганізації «соціальних» комах — множині динамічних механізмів, за допомогою яких система досягає глобальної мети в результаті взаємодії елементів з використанням лише локальної інформації. Самоорганізація є результатом взаємодії таких чотирьох компонентів: випадковість, багатократність, додатний та від'ємний зворотні зв'язки. В статті, на прикладі задачі про комівояжера, показано як можна імплементувати компоненти самоорганізації мурах в алгоритми вирішення дискретних задач оптимізації. Проведені нами комп'ютерні експерименти свідчать, що застосування мурашиних алгоритмів забезпечує знаходження гарних, тобто близьких до оптимальних, розв'язків задачі комівояжера за значно менший час, в порівнянні з методами гілок і меж та динамічного програмування. Ефективність мурашиних алгоритмів збільшується зі зростанням розмірності задачі оптимізації.

Мурашині алгоритми дозволяють отримати розв'язки багатьох дискретних комбінаторних задач не гірше за інші загальні метаевристичні технології оптимізації та деякі проблемно-спеціалізовані методи. Надзвичайно гарні результати мурашиної оптимізації для розподілених систем, параметри яких змінюються у часі. Особливістю мурашиних алгоритмів є неконвергентність — навіть після багатьох ітерацій одночасно досліджується множина різних розв'язків, що дозволяє не застрягати в локальних оптимумах. Все це дозволяє рекомендувати застосування мурашиних алгоритмів для розв'язання складних комбінаторних задач дискретної оптимізації.

Перспективними шляхами покращення мурашиних алгоритмів, на нашу думку, є їх гібридизація з іншими методами природних обчислень, наприклад, з генетичними алгоритмами. Така гібридизація може бути реалізована за острівною схемою, коли мурашиний та генетичний алгоритми працюють паралельно – кожен на своєму острові, з обміном найкращими розв'язками через певний час. Заслуговує на увагу і дослідження з адаптації параметрів мурашиних алгоритмів безпосередньо під час вирішення задачі оптимізації. Одним з перспективних шляхів такої адаптації є застосування бази нечітких правил, за аналогією з нечітким управлінням параметрами генетичних алгоритмів, запропонованого в статті [23].

СПИСОК ЛІТЕРАТУРИ

1. Dorigo M. Optimization, Learning and Natural Algorithms. PhD Thesis, Dipartimento di Elettronica, Politechnico Di Milano, Italy. — 1992. — 140 p. [Італійською].
2. Dorigo M., Maniezzo V., Colorni A. The Ant System: Optimization by a Colony of Cooperating Agents // IEEE Trans. on Systems, Man and Cybernetics. Part B. – 1996. — № 1. — Vol. 26. — P. 29—41.
3. Bonavear E., Dorigo M. Swarm Intelligence: from Natural to Artificial Systems. — Oxford University Press, 1999. — 307 p.
4. Dorigo M. Swarm Intelligence, Ant Algorithms and Ant Colony Optimization // Reader for CEU Summer University Course «Complex System», Budapest, Central European University. — 2001. — P. 1—38.
5. Ant Colony Optimization: http://iridia. ulb. ac. be/~mdorigo/ACO/ACO. html
6. Goss S., Aron S., Deneubourg J. L., Pasteels J. M. Self-Organized Shortcuts in the Argentine Ant // Naturwissenshaften. — 1989. — № 76.— P. 579—581.
7. Bullnheimer B., Hartl R. F., Strauss C. A New Rank-Based Version of the Ant System: A Computational Study // Central European Journal for Operations Research and Economics. — 1999. — № 1(7). — P. 25—38.
8. Dorigo M., Gambardella L. M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem // IEEE Trans. on Evolutionary Computation. — 1997. — № 1(1). — P. 53—66.
9. Stutzle T., Hoos H. H. MAX-MIN Ant System // Future Generation Computer Systems. — 2000. — № 8(16). — P. 889—914.
10. TSPLIB: http://www. iwr. uni-heidelberg. de/groups/comopt/software/TSPLIB95.
11. Reinelt G. The Traveling Salesman: Computational Solutions for TSP Applications. Lecture Notes in Computer Science. Vol. 840. — Berlin: Springler-Verlag. — 1994.
12. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность: Пер. с англ. — М.: Мир, 1985. — 512 с.
13. Reimann M. Ant Based Optimization in Good Transportation. PhD Thesis. University of Vienna. — Vienna, Austria. — 2002. — 149 p.
14. Socha K., Knowles J., Smples M. A MAX-MIN Ant System for the University Course Timetabling Problem. In Proc. of the Third International Workshop on Ant Algorithms «ANTS 2002». Lecture Notes in Computer Science 2463: Springer-Verlag, 2002. — P. 1—13.
15. Rodrigues A. Application of Ant Colony Optimization to Data Distribution in Memory in Computer Systems. In Abstracts of 7th Annual Swarm Researchers Meeting «SwarmFest 2003». USA, Notre Dame. — 2003 (повний текст на http://www. nd. edu/~arodrig6/).
16. Shmygelska A., Hoos H. An Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem. In Proc. of the Third International Workshop on Ant Algorithms «ANTS 2002». Lecture Notes in Computer Science 2463: Springer-Verlag, 2002.
17. Mariano C. E., Morales E. MOAQ: An Ant-Q Algorithm for Multiple Objective Optimization Problems. In Proc. of Genetic and Evolutionary Computation Conference (GECCO-99). USA, San-Francisco, 1999, Vol. 1. — P. 894—901.
18. De Campos L. M., Gamez J. A., Puerta J. M. Learning Bayesian Networks by Ant Colony Optimisation: Searching in Two Different Spaces // Mathware & Soft Computing. — 2002. — № 9.
19. Raspinelli J. M., Lopes H. S., Freitas A. A. Data Mining with an Ant Colony Optimization Algorithm // IEEE Trans. on Evolutionary Computation. Special issue on Ant Colony Algorithms. — 2002. —V. 6. — № 4. — P. 321—332.
20. Caro G. D., Dorigo M. Anet: a Mobile Agents Approach to Adaptive Routing // IRIDA – Universite Libre de Brusseles. — Brussels, Belgium. Technical Report IRIDA 97-12. — 27 p.
21. Dorigo M., Stutzle T. The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. In Handbook of Metaheuristics (Eds. Glover F. and Kochenberger G.). Norwell, MA: Kluwer Academic Publishers. — 2002.
22. Cordon O., Herrera F., Stutzle T. A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends // Mathware & Soft Computing. — 2002. — № 9.
23. Наместников А. М., Ярушкина И. Г. Эффективность генетических алгоритмов для задач автоматизированного проектирования // Известия РАН. Теория и системы управления. — 2002. — № 2. — С. 127—133.

Рекомендована кафедрою комп'ютерних систем управління

Надійшла до редакції 24.06.03
Рекомендована до опублікування 24.03.04

Штовба Сергій Дмитрович – докторант, ***Рудий Олег Миколайович*** – інженер
Кафедра комп'ютерних систем управління. Вінницький національний технічний університет