УДК 519.854

ЗАСТОСУВАННЯ *H*-МЕТОДУ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ НА ПЕРЕСТАНОВКАХ

Л.Ф. ГУЛЯНИЦЬКИЙ, Д.А. ГОБОВ

Запропоновано клас гібридних алгоритмів розв'язання задач комбінаторної оптимізації (*H*-метод), який побудовано на основі синтезу алгоритму прискореного ймовірнісного моделювання (*G*-алгоритм) та модифікованого дискретного методу деформованих багатогранників. Обгрунтовано алгоритми побудови відрізків та напівінтервалів у просторі перестановок для розв'язання квадратичної задачі про призначення. Наведено результати обчислювального експерименту, що демонструють ефективність розроблених алгоритмів у порівнянні з деякими відомими.

ВСТУП

Труднощі розв'язання задач комбінаторної оптимізації загальновідомі. Практично всі вони належать до класу NP-складних задач. Внаслідок швидко зростаючої обчислювальної складності задач при збільшенні розмірності існує необхідність у розробці методів і алгоритмів, які б дозволяли отримувати придатний для застосування розв'язок задачі за умови використання обмежених обчислювальних ресурсів. Одним із перспективних напрямків розвитку алгоритмів розв'язання задач комбінаторної оптимізації є розробка метаевристичних (гібридних) алгоритмів.

Термін *метаевристика (metaheuristics)* вперше був застосований Гловером у 1986 р. як поєднання двох грецьких слів: *евристика* походить від дієслова *heuriskein*, що означає шукати, суфікс *мета* — над, на верхнім рівні. Перед тим як цей термін набув поширення у зарубіжній літературі, замість нього часто використовували термін *сучасні евристики.* Існує велика кількість означень для терміну метаевристика. Підсумовуючи варіанти означень із [1–3] можна сказати, що метаевристики є високорівневими стратегіями для дослідження простору розв'язків за допомогою використання різних методів. Більшість відомих метаевристик базується на поєднанні популяційних алгоритмів, перш за все генетичних (ГА), та алгоритмів пошуку в околах (локальний пошук, табу-пошук, імітаційний відпал) [4, 5]. Найпоширенішими є меметичні алгоритми (деякі автори називають їх гібридними генетичними), що оперують у просторі локальних екстремумів. ГА виступає в

якості метаевристики, а вбудованою евристикою є процедура локального пошуку, яка оперує з усіма точками популяції.

У роботі розглядаються гібридні алгоритми розв'язання задач комбінаторної оптимізації, названі *Н*-алгоритмами. Вони побудовані на основі синтезу алгоритму прискореного ймовірнісного моделювання (*G*-алгоритм) та дискретного методу, названого методом деформованих багатогранників [6, 7].

Під задачею комбінаторної оптимізації далі будемо розуміти пошук такого $x \in X$, при якому задана цільова функція $f(x)$ досягає оптимуму, де X —деякий скінченний простір. Більш детально розглянемо задачі на перестановках. Ефективність алгоритмів досліджується при розв'язуванні однієї із найвідоміших моделей комбінаторної оптимізації — квадратичної задачі про призначення [1, 4].

ПОБУДОВА НАПІВІНТЕРВАЛІВ У ПРОСТОРІ ПЕРЕСТАНОВОК

Введемо означення метрики d у просторі перестановок X. Відстань між двома довільними перестановками a та b $d(a,b)$, $a,b \in X$ визначимо як мінімальну кількість транспозицій, що перетворюють перестановку a в b (під транспозицією розуміється обмін місцями двох компонентів перестановки).

Введемо означення відрізка у просторі перестановок X з метрикою d.

Послідовність перестановок $L=(l^1, ..., l^q)$, де $l^1 = a$, $l^q = b$, $d(l^i, l^{i+1}) = 1$, $i = 1,..., q-1$, є d -відрізком між перестановками $a = (a_i, ..., a_n)$ та $b = (b_1, ..., b_n)$ у просторі перестановок X з метрикою d, якщо $q = d(a,b)$.

Наведемо алгоритм побудови d -відрізка $< a,b >$. Нехай він складається з перестановок $L = (l^1, ..., l^q)$, де $l^1 = a$, $l^q = b$, а l^i, $i = 2,..., q$, які породжуються шляхом транспозиції двох елементів перестановки l^{i-1}.

Алгоритм 1.

Будуємо *d*-відрізок $< a,b >$ таким чином:

$l^1 = a$, $k = 1$.

l^2. Якщо $l_1^1 \neq b_1$, то $l^2 = (b_1, a_2, ..., a_1, ..., a_n)$, покладаємо $k := k+1$, а елемент a_1 переміщуємо у позицію, в якій стояв елемент b_1.

l^w. Якщо $l_i^{w-1} \neq b_i$, то $l^w = (b_1, ..., b_i, l_{i+1}^{w-1}, ..., l_i^{w-1}, ..., l_n^{w-1})$, $k := k+1$, l_i^{w-1}, переміщується на позицію, на якій стояв елемент b_i. Кінець алгоритму.

Для кожної пари перестановок a та b алгоритм будує один відрізок з множини можливих відрізків між a та b .

Нехай маємо перестановку (a, b, c, d, e, f), яку необхідно перетворити у перестановку (c, d, f, b, e, a). Циклічний запис цієї процедури буде

мати вигляд $(a\,c\,f)\ (bd)$, що означає «a переходить в c, c в f, f в a, b в d, d в b». Тобто цикл $(x_1, x_2, ..., x_n)$ означає, що x_1 переходить в x_2, x_2 в $x_3, ..., x_{n+1}$ в x_n, x_n в x_1. Тому що елемент e є фіксованим (тобто переходить сам у себе), він не з'являється у циклічному записі, оскільки одиничні цикли записувати не прийнято.

Легко побачити, що будь-яку процедуру по перетворенню однієї перестановки в іншу можна подати у циклічному записі.

Розглянемо таку процедуру. У просторі перестановок розмірності n потрібно перетворити перестановку a в перестановку b, причому кількість пар $\{a_i, b_i \mid a_i = b_i\}$ дорівнює r. Позначимо елементи, з яких складаються перестановки a та b, як $x_1, ..., x_n$. Нехай, починаючи з елемента x_1, маємо, що x_1 переходить у x_2, x_2 в x_3 і так далі, доки не прийдемо до елементу x_{p+1}, який вже зустрічався серед $x_1, ..., x_p$. Цей елемент x_{p+1} повинен дорівнювати x_1. Припустимо, що $x_{p+1} = x_3$. Але це неможливо, бо в x_3 переходить x_2, а за припущенням в x_{p+1} переходить $x_p \neq x_2$. Ці міркування можуть бути повторені для будь-якого x_w, $w \neq 1$. Таким чином $x_{p+1} = x_1$, і ми отримуємо цикл $(x_1, x_2, ..., x_p)$, що є частиною процедури по перетворенню a в b. Якщо $p \neq n - r$, то існує елемент y_1 такий, що $y_1 \neq x_i, i - 1, ..., p$, для якого за допомогою аналогічних міркувань отримуємо цикл $(y_1, y_2, ..., y_p)$. Жоден з y_j, $j = 1, ..., p$ не може дорівнювати довільному x_i, $i = 1, ..., p$, тому що з $x_i = y_j$ випливає $x_{i+1} = y_{j+1}$ і т.д. Для деякого k отримуємо $x_k = y_1$, що суперечить припущенню при виборі y_1. Діючи подібним чином, можна знайти всі цикли, які виникають при перетворенні перестановки a у перестановку b.

Теорема 1. Для того щоб при перетворенні перестановки a у перестановку b здійснити переміщення всіх елементів, які належать одному циклу розмірності k, необхідно виконати щонайменше $k - 1$ крок.

Доведення. Використаємо метод математичної індукції.

1. Для $k = 1$ процедура є виродженою, тобто елемент x з перестановки a необхідно перемістити в елемент x перестановки b, що виконується за очевидним правилом.

2. Виходячи з того, що для $\forall k \leq n$ теорема виконується, доведемо справедливість для $k = n + 1$. Маємо: a_1 переходить в $b_1, ..., a_n$ в b_n. Перевизначимо елементи: x_1 переходить в $x_2, ..., x_i$ в $x_{i+1}, ..., x_j$ в $x_{j+1}, ..., x_n$ в x_1.

Нехай у перестановці a поміняли місцями елементи x_i та x_j. Внаслідок цієї транспозиції отримуємо, що x_1 переходить в $x_2, ..., x_i$ в x_{j+1}, x_{j+1} в $x_{j+2}, ..., x_n$ в x_1, x_{i+1} в $x_{i+2}, ..., x_j$ в x_i.

У такий спосіб отримаємо два цикли розмірності $n-j+i+1$ та $j-i$. Для першого циклу, оскільки його розмірність менша за $n+1$, для переведення всіх його елементів потрібно зробити $n-j+1$ крок, а для другого циклу — відповідно $j-i-1$ крок. Таким чином будуть зроблені $1+(n-j+i)+(j-i-1)=n$ кроків. Теорему доведено.

Теорема 2. Процедура перетворення перестановки a у перестановку b шляхом транспозицій елементів перестановки b при умові, що задача містить t циклів розмірності $k_1,k_2,...,k_t$, де $\sum_{i=1}^{t} k_i = N$, має щонайменше $n-t$ кроків, де n — розмірність простору перестановок.

Доведення. Зрозуміло, що в результаті транспозиції двох елементів a_i та a_j, можливі два випадки: або елементи a_i та a_j належать двом циклам, або одному.

Розглянемо перший випадок. Нехай маємо два цикли: x_1 перетворюється в $x_2,...,x_u$ в $x_{u+1},...,x_p$ в x_1, де $x_{u+1}=a_i$ та y_1 в $y_2,...,y_h$ в $y_{h+1},...,y_q$ в y_1, де $y_{h+1}=a_j$.

Поміняємо місцями елементи x_{u+1} та y_{h+1}. Отримаємо: x_1 перетворюється в $x_2,...,x_u$ в $y_{h+1},...,y_q$ в y_1, y_1 в $y_2,...,y_h$ в $x_{u+1},...,x_p$ в x_1. Тепер маємо один цикл розмірності $p+q$. Для перетворення всіх елементів, які належать до цього циклу, згідно із теоремою 1 необхідно зробити $p+q-1$ крок, а для вихідних циклів — $p+q-2$. Таким чином, кожна транспозиція елементів, що належать до різних циклів, збільшує кількість необхідних транспозицій для перетворення перестановки a в b на 2.

Якщо на кожному кроці будемо змінювати місцями елементи, які належать до одного циклу, то для перетворення всіх елементів цього циклу, необхідно буде зробити кількість кроків, яка дорівнює розмірності циклу зменшену на одиницю (виходячи з теореми 1), тобто $\sum_{i=1}^{t}(k_i-1)=$

$=\sum_{i=1}^{t} k_i - t = N - t$. Теорему доведено.

Теорема 3. Алгоритм 1 будує послідовність перестановок найменшої довжини, а саме перетворює перестановку a в перестановку b за мінімально можливу кількість транспозицій, тобто будує між ними d -відрізок.

Доведення. На кожному кроці алгоритм 1 міняє місцями два елементи, які належать до одного циклу. Це виходить з того, що l_i^j перетворюється в b_i, і ми шукаємо таке l_d^j, що дорівнює b_i і яке належить до одного циклу з l_i^j. Повторюючи відповідні міркування з доведення теореми 2, отримаємо,

що алгоритм 1 перетворить перестановку a в b за $\sum_{i=1}^{t}(k_i - 1)$, де $k_1, k_2, ..., k_t$ — розмірності циклів у вихідній задачі. Виходячи з теореми 2, ця кількість кроків є мінімальною. Теорему доведено.

Розглянемо поняття напівінтервала у просторі перестановок. Виходячи з теореми 1 та 2, точка c є найбільш віддаленою від точки a, якщо $d(a,c) = n - 1$.

Теорема 4. При використанні алгоритму 1 максимальна кількість транспозицій для перетворення будь-якої перестановки a в c дорівнює $n - 1$, де n — розмірність простору перестановок.

Доведення. Нехай процедура по перетворенню перестановки a в c має t циклів. Тоді, згідно із теоремою 2, необхідно зробити $\sum_{i=1}^{t}(k_i - 1)$ кроків, де $\sum_{i=1}^{t} k_i = N$. Перетворимо цей вираз:

$$\sum_{i=1}^{t}(k_i - 1) = \sum_{i=1}^{t} k_i - t = N - t.$$

Через те що кількість циклів більша або дорівнює 1, цей вираз набуває максимуму для $t = 1$. Таким чином отримали максимум, який дорівнює $n - 1$.

Розглянемо алгоритм побудови напівінтервала через дві задані точки. Нехай вихідна задача перетворення перестановки a в b має t циклів. Тоді за теоремою 2 кількість кроків буде дорівнювати $n - t$. Теорему доведено.

Оберемо t елементів з перестановки b, з кожного циклу вихідної задачі по одному елементу та модифікуємо алгоритм 1.

Алгоритм 2.

Будуємо множину перестановок L таким чином:

$l^1 = a,\ k = 1,\ p = 1$.

l^2. Якщо $l_1^1 \neq b_1$, то $l^2 = \{b_1, a_2, ..., a_1, ..., a_n\}$, покладаємо $k := k + 1$, a_1 переміщуємо в позицію, в якій стояв елемент b_1; якщо $l_1^1 = b_1$, то $m_p = l_1^1,\ p := p + 1$.

Провівши аналогічні дії, завершимо їх для l^w.

l^w. Якщо $l_i^{w-1} \neq b_i$, то $l^w = \{b_1, ..., b_i, l_{i+1}^{w-1}, ..., a_i, ..., l_n^{w-1}\}$, $k := k + 1$, a_i переміщуємо в позицію, в якій стояв елемент b_i; якщо $l_i^{w-1} = b_i$, то $m_p = l_i^{w-1},\ p := p + 1$.

Зрозуміло, що $p = T$. Всі $m_p,\ p = 1, ..., T$, належать до різних циклів вихідної задачі.

Очевидно, що у випадку транспозиції двох елементів m_i буде отримана перестановка, яка не зустрічалась раніше.

Формуємо множину S всіх пар (m_i, m_j), $i \neq j$, $i, j \in [1,...,T]$, $i = 1$.

Виконуємо T раз такі дії:

1. Обираємо випадково чи за певним правилом з множини S пару (m_i, m_j).

2. Міняємо місцями ці елементи у перестановці l^{w+i-1} і отримуємо перестановку l^{w+i}.

3. Виключаємо з множини S пари (m_i, m_j) та (m_j, m_i).

4. $i := i + 1$.

Таким чином ми отримали послідовність l^i, $i = 1,...,n-1$, тобто побудували напівінтервал, який проходить через точки *a* та *b*.

Наведений алгоритм дозволяє будувати напівінтервали, причому з множини всіх можливих напівінтервалів завжди вибирається лише один, що дозволяє відбраковувати пари точок, які були вже задіяні у побудові напівінтервалу. Слід зауважити: як параметр зазначеного алгоритму може використовуватись номер елемента у перестановці, з якого починається дія алгоритма побудови напівінтервалу. У такому випадку можуть відбраковуватись вже дві-три перестановки та номер елемента, з якого починає діяти такий алгоритм.

Відзначимо, що подібні результати можна отримати і для інших метрик.

ГІБРИДНИЙ МЕТОД ДЕФОРМАЦІЙ (*H*-МЕТОД)

Основна ідея *H*-методу полягає в тому [8], що формується початкова множина розв'язків, які грають роль віддалено подібну ролі багатогранника у методі пошуку по деформованому багатограннику (метод Нелдера-Міда [9]). Із цієї множини певним чином вибираються декілька пар точок. Для кожної з них будується напівінтервал у просторі варіантів розв'язку задачі, який проходить через вибрані точки, і на ньому відшукується найкраща для цільової функції точка. Знайдений варіант передається як початкове наближення для покращення *G*-алгоритмом. У результаті таких дій знаходиться сукупність покращуючих точок. Ці точки включаються у багатогранник замість найгірших для вибраного критерію точок, і описана процедура повторюється до виконання критерію завершення.

Наведемо більш формально обчислювальну схему *H*-методу, використовуючи термінологію еволюційних обчислень.

procedure *H*-Algorithm (x);
  begin
  $h := 0$; $P^0 = \varnothing$;
  for $j := 1$ to m do

$x :=$ ГенераціяДопустимогоВаріанту;

$x := G\_Search(x)$;

$P^h := P^h \cup x$;

endfor; {формування початкової популяції}

repeat

$P := P^h$

for $i := 1$ to k do

ВідбірДляВаріації $(x, y \in P : f(x) > f(y))$;

$z := \arg\min\{f(u) : u \in < x, x^\infty \wedge L(y)\},\ y \in < x, x^\infty /\}$;

$z := G\_Search(z)$;

$P := P^h \cup z$;

endfor; {сформовано тимчасовий багатогранник із $(m + k)$ точок}

for $i := 1$ to l do

ВідбірДляМутації $(x \in P)$;

$x :=$ Мутація (x);

$x := G\_Search(x)$;

$P^h := P^h \cup x$;

endfor; {сформовано тимчасовий багатогранник із $m + k + l$ точок}

$P^h :=$ ВідбірПопуляції (P);

$h := h + 1$;

until виконується умова завершення;

$x := \arg\min\{f(y) : y \in P\}$;

return x;

end.

Позначення.

$< a, b /$ — напівінтервал у просторі перестановок від перестановки a до b, виключаючи перестановку b;

$L(y)$ — заданий окіл точки $y \in P$.

Параметри.

m — число вершин багатогогранника (число осіб у популяції в термінах ГА);

k — кількість пар вершин, які вибираються для дослідження (проведення напівпрямих від поточної точки до максимально віддаленої та пошуку мінімуму на цих прямих);

l — кількість вершин, які підлягають мутації;

x^∞ — точка, максимально віддалена у просторі варіантів розв'язку від початкової точки x (діаметрально протилежна у просторі X);

G_Search — процедура G-алгоритму;

ВідбірПопуляції — процедура відбору, наприклад, $(m+(k+l))$-стратегія: вибір m точок із $(m+k+l)$ точок тимчасового багатогогранника (популяції).

Використання наведеної схеми може породжувати сімейство алгоритмів комбінаторної оптимізації за рахунок можливих способів конкретизації (із урахуванням досвіду застосування та специфіки задачі) таких її аспектів.

1. Правила відбору чергової пари точок $x, y \in P$ для варіації:

- вибір таких точок, які відповідають найкращому і найгіршому значенню цільової функції серед точок P;
- випадковий вибір точок;
- вибір «найгіршої» точки x і випадковий вибір точки y.

2. Відбір для мутації. Використовується випадковий відбір l точок. Сама мутація полягає у збуренні декількох компонентів вибраного варіанта розв'язку.

3. Стратегія формування нового багатогранника, що повинен складатися із m точок тимчасового багатогранника обсягом $(m+k+l)$ точок:

- відбір m кращих точок;
- включення нащадків на основі критерію Метрополіса [6];
- заміна всіх m батьків кращими із $(k+l)$ нащадків, $m \le k+l$.

4. Критерієм завершення обчислювального процесу може бути:

- перевищення заданої кількості $H_{\max}$ оновлення множини P: алгоритм завершує роботу, коли $h > H_{\max}$;
- прагнення до нуля розкиду значень цільової функції у точках множини P:

$$\sigma = \max\{f(x) : x \in P\} - \min\{f(x) : x \in P\};$$

- всі пари точок вже прийняли участь у формуванні напівінтервалів, якщо використовується однозначний спосіб побудови $< x, x^{\infty} /$.

5. Спосіб побудови напівінтервалів $< x, x^{\infty} /$. Скінченні комбінаторні метричні простори, які розглядаються, мають дві характерні особливості:

- для довільного $x \in P$ «максимально віддаленою» є точка, що задовольняє умові: $x^{\infty} = \max\{d(x,u) : u \in X\}$, де $d(x,u)$ — метрика на X (часто це значення дорівнює діаметру простору X);
- між двома несусідніми точками $x, y \in P$ можна побудувати не один, а декілька d-відрізків, тому при реалізації H-алгоритмів слід розрізняти випадок, коли із множини можливих d-відрізків $< x, y >$, $d(x,y) > 2$ завжди за конкретним правилом/способом побудови вибирається лише один чи коли можуть розглядатися/будуватися всі можливі відрізки. У залежності від цього може модифікуватися і правило «ВідбірДляВаріації»: якщо розглядається один можливий напівінтервал $< x, x^{\infty} /$, причому $y \in < x, x^{\infty} /$, то

вершини x, y після відбору слід помітити як «відпрацьовані», оскільки їх повторний вибір уже стає недоречним. Якщо ж є можливість будувати всі чи декілька можливих напівінтервалів $< x, x^{\infty} /$, то повторний відбір пари x, y стає доцільним, якщо при цьому буде побудований новий напівінтервал.

ОБЧИСЛЮВАЛЬНИЙ ЕКСПЕРИМЕНТ

Оцінювання ефективності H-алгоритму проводилося у два етапи. Перший передбачав розв'язання квадратичних задач про призначення, для яких відомі розв'язки [10]. На другому етапі розв'язувались серії квадратичних задач про призначення (КЗП) різних розмірностей з невідомим точним розв'язком.

Ефективність H-алгоритму порівнювалась з алгоритмами детермінованого пошуку, а саме з методом вектора спаду з кільцевим алгоритмом перебору точок в околі (АКП) [11]), з алгоритмами імітаційного відпалу та мультистартового G-алгоритму [6, 11]. Під мультистартовим G-алгоритмом розуміється така процедура: обирається випадкова перестановка, що подається як початкове наближення на вхід G-алгоритму, а отриманий варіант розв'язку додається до множини розв'язків. Подібні дії повторюються задану кількість разів і з отриманої множини розв'язків обирається найкращий. Крім того були реалізовані ГА у загальному вигляді (ЗГА), меметичний алгоритм (Mem), гібридний генетичний алгоритм на основі G-алгоритму (G-ГА), мультистартовий G-алгоритм (MG), метод деформованих багатогогранників (МДМ), гібридний метод деформованих багатогранників (МДМ+G). Обчислювальний експеримент проводився на робочій станції з характеристиками: процесор Athlon 900 Mhz, RAM 256 Mb. Результати розв'язання відомих задач наведені в табл. 1, де для кожного методу розв'язку вказано процент відхилення від глобального оптимуму, n — розмірність задачі, t — обмеження на час роботи алгоритмів в c. Параметри H-алгоритму та вбудованого G-алгоритму подані у табл. 2, де M — кількість переходів в одному прогоні [6]. Назва задач в таблиці — це їх ідентифікатори в роботі [10].

При опрацюванні поточного багатогранника будувались п'ять напівінтервалів. Мутації підлягали три вершини багатогранника, в кожній з яких випадковим чином змінювали місця три елемента. Критерій завершення роботи алгоритму — час роботи більше заданого обмеження. Для побудови кожного напівінтервалу обирались дві довільні точки з множини P, які ще не використовувались для напівінтервалів. Якщо при заданому параметрі μ виконано k прогонів і отримані значення $f_1, ..., f_k$, то вважалося, що досягнута рівновага, якщо $\frac{|f_{k+1} - f|}{f} \le \varepsilon$, де f — краще середнє значення цільової функції за k прогонів.

Таблиця 1. Точність розв'язання КЗП

Назва задачі	n	АКП	Відпал	ГА	Mem	ГА+G	MG	МДМ	МДМ +G	H-алг.	t
Els19	19	4,21	4,21	16,03	0,00	0,00	0,90	32,12	12,00	0,00	10
Chr20*a*	20	56,48	20,53	46,26	0,00	1,82	1,37	14,96	9,76	0,00	10
Chr20*b*	20	35,16	14,19	15,58	7,31	6,44	7,40	6,53	5,13	3,48	10
Chr20c	20	83,04	34,44	45,34	0,00	0,00	0,00	44,68	28,60	0,00	10
Nug21	21	2,38	0,00	0,00	0,00	0,00	0,00	5,09	0,90	0,00	10
Nug22	22	2,84	0,50	1,33	0,00	0,00	0,00	1,84	1,00	0,00	10
Lip40*a*	40	1,41	1,14	1,78	1,19	1,13	1,14	1,68	1,02	1,01	40
Lip40*b*	40	18,85	0,00	20,48	0,00	0,00	0,00	19,97	16,59	0,00	40
Lipa50*a*	50	1,41	0,97	1,62	1,15	1,00	0,99	1,71	0,98	0,97	63
Lipa50	50	18,11	17,08	20,30	0,00	0,00	0,00	22,20	17,40	0,00	63
Esc64*a*	64	5,17	0,00	15,52	0,00	0,00	0,00	8,62	0,00	0,00	10
Sco81	81	1,98	0,97	10,54	1,10	0,49	1,17	12,24	1,91	0,34	400
Tai100*b*	100	6,51	5,20	10,14	1,84	1,52	2,16	24,34	3,17	1,34	850
Tai150*b*	150	2,79	3,92	1,73	0,65	1,48	1,75	1,73	1,58	1,18	1900

Таблиця 2. Параметри G та H-алгоритмів

m	H	γ	M	ε
20	0,01	2,5	5	0,00001

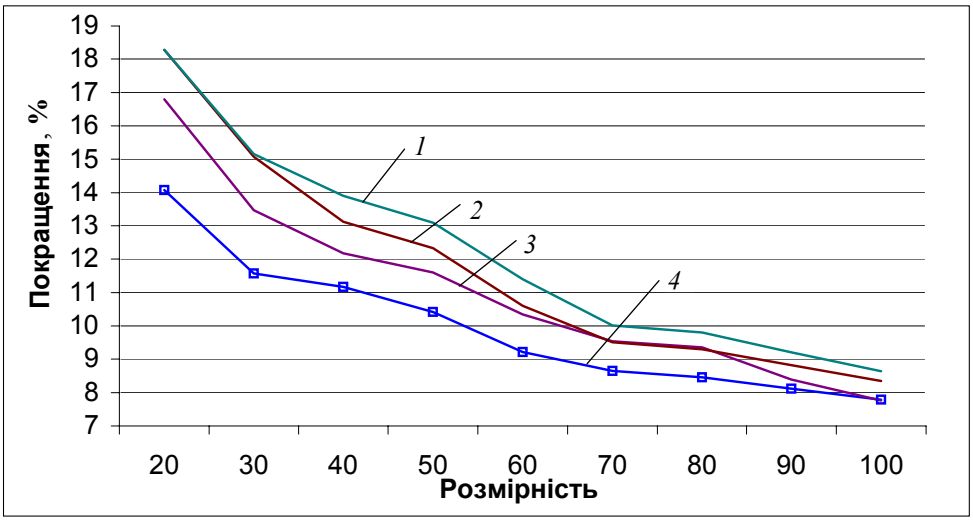
Як обмеження у часі для G -алгоритму використовувався час

$$t_G = t_{\text{клп}} * 5 + 0{,}5\,,$$

де $t_{\text{клп}}$ —— час роботи алгоритма АКП.

Результати розв'язання випадково згенерованих серій задач наведені на рисунку.

Генерування проводилось на основі датчика випадкових чисел з рівномірним розподілом. Розв'язувались задачі розмірності від 20 до 100 з кроком 10. Для кожної розмірності — по 20 задач. При опрацюванні поточного багатогранника будувались три напівінтервали. При відборі популяції бралися кращі m точок. Критерій завершення роботи алгоритму — це час роботи більше 1000с або переглянуті всі пари точок у поточному багатограннику. У табл. 3 наведено середній час роботи алгоритмів. Для побудови напівінтервалу обирались дві точки, що відповідають найкращому і найгіршому значенням цільової функції серед точок P. На кожному прогоні для мутації обирались три точки з багатогранника. Мутація вплинула на три елементи перестановки. Відсоток покращення рахувався від початкового розв'язку, який генерувався випадково. Обсяг багатогогранників дорівнював 50; $H = 0{,}1$; $\gamma = 1{,}4$.



Результат розв' язання серії задач: *1* — *H*-алгоритм; *2* — *MG*-алгоритм; *3* — імітаційний відпал; *4* — АКП

Таблиця 3. Середній час роботи алгоритмів

n	АКП	Імітаційний відпал	MG	*H*-алгоритм
20	0,02	9,66	86,38	86,38
30	0,07	64,62	423,21	423,21
40	0,35	140,74	1000	1000
50	0,53	264,33	1000	1000
60	1,18	1000	1000	1000
70	2,43	1000	1000	1000
80	4,47	1000	1000	1000
90	7,56	1000	1000	1000
100	12,43	1000	1000	1000

У табл. 4 наведено значення середньоквадратичного відхилення σ та величина довірчого інтервалу δ для покращання значення цільової функції.

Таблиця 4. Дисперсія ефективності алгоритмів

n	АКП		Імітаційний відпал		*G*-алгоритм		*H*-алгоритм	
	σ	δ	σ	δ	σ	δ	σ	δ
20	2,48	1,75	2,39	1,69	2,11	1,49	2,12	1,5
30	1,66	1,17	1,56	1,1	1,46	1,03	1,34	0,95
40	1,34	0,95	1,34	0,95	1,32	0,93	1,27	0,9
50	0,81	0,57	0,9	0,64	0,85	0,6	0,77	0,54
60	0,77	0,54	0,69	0,49	0,64	0,45	0,63	0,44
70	0,68	0,48	0,69	0,49	0,48	0,34	0,5	0,35
80	0,29	0,2	0,31	0,22	0,4	0,28	0,26	0,18
90	0,44	0,31	0,34	0,24	0,31	0,22	0,42	0,3
100	0,36	0,25	0,31	0,22	0,25	0,18	0,21	0,15

Як бачимо, довірчі інтервали достатньо малі, тому оцінки ефективності алгоритмів можна вважати достовірними.

ВИСНОВКИ

Обчислювальний експеримент показав високу ефективність *H*-алгоритмів як відносно алгоритмів локального пошуку, так і імітаційного відпалу, мультистартового *G*-алгоритму, модифікацій генетичних алгоритмів та алгоритмів методу деформацій. При проведенні досліджень виявлено, що великий вплив на ефективність *H*-алгоритму мають параметри використовуваного *G*-алгоритму, і тому вибір їх значень стає актуальним.

Потребує додаткового вивчення залежність ефективності *H*-алгоритму від кількості напівінтервалів, які будуються на кожному кроці алгоритму, а також залежності оптимальної розмірності популяції від розмірності задачі. Цікавим для дослідження є теоретичний аналіз ефективності *H*-алгоритмів з різними правилами вибору точок із популяції для побудови напівінтервалу.

Важливий напрямок подальших розробок — теоретичні дослідження трудомісткості алгоритму та оцінок збіжності при розв'язанні конкретних типів задач комбінаторної оптимізації.

Реалізовано клас нових ефективних гібридних алгоритмів розв'язання задач комбінаторної оптимізації на перестановках, побудований на основі методу деформованих багатогогранників та *G*-алгоритмів. Запропоновано алгоритми побудови відрізків та напівінтервалів у просторі перестановок, які проходять через дві задані точки.

Досліджено ефективність одного алгоритму з розробленого класу при розв'язанні відомих квадратичних задачах про призначення та серії згенерованих задач у порівнянні з методами локального пошуку, ймовірнісними методами, модифікаціями генетичного алгоритму. Продемонстровано конкурентоспроможність розробленого класу алгоритмів у порівнянні з наведеними відомими алгоритмами.

ЛІТЕРАТУРА

1. *Meta-Heuristics* – Advances and Trends in Local Search Paradigms for Optimization / S. Vob, S. Martello, I.H. Osman, C. Roucaisol. — Dordrecht: Kluwer Academic Publishers. — 1999. — 528 p.
2. *Osman I. H., Laporte G.* Metaheuristics: A bibliography// Ann. Oper. Res. — 1996. — № 63. — P. 513–623.
3. *Metaheuristics* network website // http: www.metaheuristics.net.
4. *Hoos H.H., Stützle T.* Stochastic Local Search: Foundations and Applications. — San Francisco: Morgan Kaufmann Publ., 2005. — 658 p.
5. *Blum C., Roli A.* Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison // ACM Computing Surveys. — 2003. — **35**, №. 3. — P. 268–308.

6. *Гуляницкий Л.Ф.* Решение задач комбинаторной оптимизации алгоритмами ускоренного вероятностного моделирования // Компьютерная математика. — 2004. — № 1. — С. 64–72.
7. *Гуляницкий Л.Ф.* Метод деформаций в дискретной оптимизации // Исследование операций и АСУ. — 1989. — Вып. 34. — С. 30–33.
8. *Гуляницький Л.Ф*. Один гібридний алгоритм комбінаторної оптимізації // Abstract of Int. Ukrainian-Polish Workshop «Problems of Stochastic and Discrete Optimization» (May 10 – 15, 2005, Kaniv, Ukraine). — Kaniv, 2005. — С. 63–65.
9. *Химильблау Д.* Прикладное линейное программирование. — М.: Мир, 1974. — 534 с.
10. *QAPLIB* // http://www.opt.math.tu-graz.ac.at/qaplib/.
11. *Гуляницкий Л.Ф., Гобов Д.А.* О сравнении эффективности алгоритмов решения одного класса задач размещения прямоугольников на полубесконечной ленте // Компьютерная математика. — 2003. — № 1. — С. 64–72.

Надійшла 20.07.2006