УДК 519.854
DOI https://doi.org/10.24144/2616-7700.2025.47(2).303-311

**С. В. Чупов[1], А. В. Федорішко[2]**

[1] ДВНЗ «Ужгородський національний університет»,
доцент кафедри системного аналізу та теорії оптимізації,
кандидат фізико-математичних наук, доцент
serhii.chupov@uzhnu.edu.ua
ORCID: https://orcid.org/0000-0002-7715-3924

[2] ДВНЗ «Ужгородський національний університет»,
аспірант кафедри системного аналізу та теорії оптимізації
anton.fedorishko@uzhnu.edu.ua
ORCID: https://orcid.org/0009-0002-8921-0882

# ГІБРИДНИЙ АЛГОРИТМ РОЗВ'ЯЗАННЯ КВАДРАТИЧНОЇ ЗАДАЧІ ПРО ПРИЗНАЧЕННЯ

Квадратична задача про призначення (QAP) є однією з найбільш вивчених комбінаторних задач оптимізації з різними практичними застосуваннями. У цій статті ми представляємо наближений гібридний алгоритм (HBMA) для розв'язання QAP. HBMA досліджує простір пошуку шляхом використання відомих алгоритмів 2-OPT, BLS та BMA на окремих етапах розв'язання задачі. Експериментальні оцінки на множині еталонних задач з QAPLIB показують, що запропонований підхід здатний досягти найвідоміших на даний момент результатів для всіх екземплярів, крім окремих задач типу ***taia***, із середнім часом обчислення менше 1 години. Також надаються порівняння, щоб показати конкурентоспроможність запропонованого підходу відносно алгоритмів BLS та BMA для QAP.

**Ключові слова:** Квадратична задача про призначення, гібридний алгоритм наближеного пошуку, локальний алгоритм наближеного пошуку, випадкове збурення компонент розв'язку, ефективність алгоритмів.

**1. Вступ.** Відомо, що багато задач дискретної оптимізації, зокрема задача QAP, є NP-складними. Основною проблемою при їх розв'язанні є експоненціальний ріст обчислювальних витрат при збільшенні розмірності задачі. Це обмежує можливості розв'язання таких задач у реальному масштабі часу. Крім того, обчислювальна складність задачі QAP викликана значним об'ємом обчислень (порядку $O(n^2)$), необхідних для отримання значення цільової функції та дослідження околу поточного розв'язку. Нагадаємо, що в алгоритмах для задачі QAP окіл $N(\pi)$ розв'язку $\pi$ складається з усіх перестановок, які отримані шляхом обміну позиціями між усіма парами об'єктів. Таким чином, окіл містить $C_n^2 + 1$ перестановку, а обчислення значень цільової функцій для усіх перестановок з $N(\pi)$ потребує об'єму обчислень порядку $O(n^4)$. Якщо $n \geq 100$, то виконання хоча б $n^2$ ітерацій локального пошуку вже стає важкою обчислювальною задачею.

Практична важливість та обчислювальна складність задачі QAP підтверджується хронологією її дослідження. Сформульована вона була Koopmans та Beckmann [9] у 1957 р. У 1976 р. Sahni та Gonzalez довели [13], що ця задача належить до класу NP-складних задач. Burkard [4] у 1984 р. відзначив, що не можна побудувати точних методів для ефективного розв'язання QAP розмірністю, більшою за 20. І тільки з кінця 80-х років минулого сторіччя розпочався

бурхливий розвиток наближених методів відшукання якісних розв'язків квадратичної задачі про призначення.

Сьогодні більшість ефективних наближених алгоритмів для QAP базується на використанні різноманітних схем методу табу пошуку (**TS**), запропонованого F. Glover [7, 8]. Варто згадати такі версії методу, як надійний табу пошук [15] (**RoTS**), реактивний табу пошук [1] (**ReTS**). В останні роки найбільш якісні результати отримано такими наближеними алгоритмами як Breakout Local Search, **BLS** [2], Memetic Algorithm for QAP, **BMA** [3] Improved Hybrid Genetic Algorithm, **IHGA** [11], **ITS for QAP** [12] та багатьма іншими.

Як правило схеми сучасних наближених методів розв'язання задач оптимізації включають дві фази: інтенсифікація пошуку та диверсифікація пошуку. На першій фазі здійснюється пошук локального екстремуму задачі, а на другій — робиться спроба вийти з його околу та перейти у іншу, ще не досліджену, область множини допустимих розв'язків задачі. Розглянемо більш детально деякі з ефективних алгоритмів розв'язання QAP.

Ітеративний локальний пошук на основі популяції (**PILS**) [14] — це один з ефективних алгоритмів. Базовий алгоритм ітеративного локального пошуку (ILS) починається з випадкового розв'язку та застосовує процедуру локального пошуку першого покращення на основі 2-opt. Щоб пришвидшити процес пошуку, алгоритм використовує стратегію «не дивись біт», запропоновану раніше для прискорення алгоритмів локального пошуку для TSP. Після досягнення локального оптимуму ILS застосовує збурення, яке полягає в обміні k випадково вибраними елементами У запропонованому PILS, популяція складається з $\mu$ рішень, і на кожній ітерації генеруються нові рішення. Потім використовується стратегія вибору, що базується як на якості, так і на відстані між рішеннями, для формування нової популяції $\mu$ рішень.

Ітеративний заборонений пошук (**ITS**) [12] дотримується загальної схеми метаевристики ітерованого локального пошуку. Він використовує традиційний заборонений пошук для досягнення локальних оптимумів та запускає фазу збурень (реконструкції) щоб уникнути досягнутого локального оптимуму. «Зруйнований» розв'язок стає новою відправною точкою для базової процедури TS. ITS використовує механізм збурень, який адаптивно змінює кількість випадкових рухів збурень в деякому інтервалі. ITS отримує чудові результати на неструктурованих екземплярах та реальних екземплярах.

У [18, 19] представлено ефективний алгоритм для розв'язання QAP, названий **RITSR** (повторюваний ітерований алгоритм табу). Фаза інтенсифікації, яка названа дослідницьким пошуком реалізує схему ітерованого локального табу пошуку (**ITSL**). Цей пошук призначений для виявлення кращих розв'язків, що містяться на незначній відстані від межі поточного околу локального мінімуму. Якщо отримано кращий розв'язок, він запам'ятовується та починається детальний пошук нового, кращого за знайдений, розв'язку в околі знайденої перестановки. Після цього цикл повторень у режимі дослідницького пошуку розпочинається заново. Друга фаза пошуку — фаза диверсифікації – виконується кожен раз перед початком нового дослідницького пошуку у циклі повторень. У цій фазі збурюється поточний локальний мінімум так, щоб наступний дослідницький пошук відбувався на певній відстані від околу поточного розв'язку.

У даній статті представлено новий гібридний алгоритм розв'язання назва-

ний **HBMA**. У своїй роботі він використовує комбінацію алгоритмів **2-OPT** [17], **BLS**, **BMA**. Наведена формальна схема алгоритму та результати числових експериментів. У висновках коротко оцінюється ефективність запропонованого алгоритму. Пропонуються подальші напрямки досліджень.

**2. Постановка задачі.** Змістовна постановка квадратичної задачі про призначення (QAP) наступна. Дано $n$ об'єктів, які потрібно розташувати у $n$ різних локаціях (місць призначень). Відомі величини потоків ресурсів $a_{ij}$ між об'єктами $i$ та $j$, $i, j = 1, \ldots, n$, і відстані $b_{rs}$ між локаціями (пунктами) $r$ та $s$, $r, s = 1, \ldots, n$. Потрібно знайти таке розподілення об'єктів по локаціях, щоб сума відстаней, помножена на відповідні потоки, була б мінімальною.

Отже, математичну постановку задачі QAP можна сформулювати наступним чином: знайти:

$$\min f(\pi)_{\pi \in \Pi^n} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{\pi_i \pi_j},$$

де $A = [a_{ij}]$ та $B = [b_{rs}]$ — квадратні матриці порядку $n$, $\Pi^n$ — множина усіх перестановок розмірності $n$ і $\pi_i$ задає номер локації об'єкту $i$.

Квадратичні задачі про призначення можливо точно розв'язати тільки для не великих размірностей. У зв'язку з цим актуальними є розробка, дослідження нових і вдосконалення існуючих наближених алгоритмів розв'язання задачі QAP.

**Алгоритм 2-OPT.** Серед простих алгоритмів локального пошуку найвідомішим є **2-OPT** [17]. Алгоритм **2-OPT** вперше був запропонований Кроесом (1958) для **TSP**. Цей алгоритм використовує попарну транспозицію об'єктів. Якщо кількість місць розташування об'єктів позначена як n, кількість транспозицій у кожній ітерації буде $n(n-1)/2$. Спочатку алгоритм розглядає транспозицію об'єктів 1 та 2. Якщо значення цільової функції отриманого розв'язку менше, ніж значення цільової функції початкового розв'язку, то воно зберігається як кандидат для подальшого розгляду. В іншому випадку воно відкидається, і алгоритм розглядає транспозицію засобів 1 та 3. Якщо цей обмін генерує краще рішення, то воно зберігається як кандидат для подальшого розгляду; в іншому випадку, воно відкидається і так далі. Таким чином, щоразу, коли знайдено краще рішення, алгоритм відкидає попереднє найкраще рішення. Ця процедура продовжується, доки не будуть розглянуті всі попарні обміни.

**Breakout Local Search (BLS). BLS** [2] використовує оператор заміни, який полягає в обміні значеннями з двох різних позицій у $\pi$, тобто, перестановці розташування двох об'єктів (будемо називати такий оператор заміни ходом). Він використовує процедуру найкращого покращення спуску для використання всієї області обміну $N(\pi)$, яка оцінюється за час $O(n^2)$ завдяки ефективній стратегії оцінки області, запропонованій у [15].

Після того, як процедура локального спуску повертає найкращий локальний оптимум, *BLS* запускає механізм багато типової диверсифікації, який адаптивно визначає тип $T$ рухів збурень та кількість $L$ збурень (яку називають величиною стрибка), враховуючи деяку інформацію, пов'язану зі станом пошуку. Цей механізм поєднує три типи збурень: прямі, засновані на «нещодавності» та випадкові.

Збурення на основі табу використовує правило вибору, яке надає перевагу ходам, що мінімізують зниження вартості, за умови, що хід не був застосований протягом останніх $\gamma$ ітерацій.

Збурення на основі «нещодавності» спирається лише на інформацію з історії ходів. Воно надає перевагу найменшому нещодавно виконаному ходу, незалежно від зниження вартості, спричиненого цим ходом.

Зрештою, випадкове збурення просто виконує ходи, які вибираються рівномірно випадковим чином.

**Breakout Memetic Algorithm (BMA).** Враховуючи початкову популяцію, яка складається з локально оптимальних рішень, меметичний підхід генерує нові рішення, застосовуючи кросовер та/або мутацію, а потім фазу локального пошуку для покращення кожного рішення-нащадка. Правильний вибір оператора кросовера залежить від структури задачі та властивостей «ландшафту» конкретної задачі. Більше того, успіх меметичного підходу обумовлений ефективністю процедури локального пошуку. Хоча головна роль кросовера полягає у виявленні недосліджених перспективних областей простору пошуку, локальний пошук в основному спрямований на пошук хороших рішень шляхом концентрації пошуку навколо цих областей.

Меметичний алгоритм для QAP (**BMA**) [3] використовує оператор рівномірного кросовера, процедуру локального пошуку з проривом (**BLS**), стратегію заміщення популяції на основі придатності та адаптивний механізм мутації. Кожний розв'язок для популяції, згенерований за допомогою рівномірного кросовера, покращується за допомогою процедури **BLS**. Даний меметичний підхід потім застосовує стратегію оновлення популяції, щоб можливо замінити найгірший об'єкт з популяції покращеним розв'язком. Щоб уникнути передчасної конвергенції, BMA запускає адаптивний механізм мутації для всієї популяції, якщо найкраще рішення, знайдене під час пошуку, не було покращено протягом фіксованої кількості поколінь. Це зміщує пошук у віддалені регіони щоразу, коли виявляється застій пошуку.

**Гібридний алгоритм HBMA для QAP.** В загальному, нехай маємо множину *Algs* з $k$ алгоритмів для розв'язання задачі оптимізації $\{A_1,\ A_2,\ \ldots,\ A_k\}$, причому кожен алгоритм використовується у формі $(\pi, f) = A_k(\pi_0,\ f_0)$, де $\pi_0$, $f_0$ — початковий розв'язок та значення цільової функції відповідно. Розіб'ємо процес розв'язання цієї задачі на окремі етапи відповідно до значень цільової функції $f_0$. На першому етапі завдяки обраному певним чином алгоритмом намагаємося покращити початковий розв'язок за обраним алгоритмом . Якщо, за прийнятний час, не вдається отримати покращення, формуємо інше початкове значення та продовжуємо роботу обраного алгоритму або обираємо новий. Як тільки виконується умова завершення етапу, переходимо до другого. І так далі. Якщо на певному етапі не вдається отримати покращення, переходимо до першого етапу. Врешті решт можемо отримати придатний розв'язок задачі. Обираючи різні підмножини з множини *Algs* та умови переходу від одного етапу до іншого можемо отримувати різні схеми гібридного алгоритму.

У даній версії гібридного алгоритму **HBMA** множина алгоритмів *Algs* містить алгоритми **2-Opt** [17], **BLS** [2], **BMA** [3].

Нехай $f = f(\pi)$ — поточне значення цільової функції задачі. Значення $bound_1$, $bound_2$ та $bound_3$ задають межі застосування алгоритмів **2-Opt, BLS**

або **BMA**. Якщо $bound_1 \leq f < +\infty$, тоді використовується алгоритм **2-Opt** для здійснення дослідницького пошуку (*investigated search*). Якщо $bound_3 < f < bound_1$, тоді використовується алгоритм **BLS** для здійснення уточнюючого пошуку (*advanced search*). Коли значення цільової функції міститься у межах $[bound_2, bound_3)$, тоді здійснюється накопичення гарних розв'язків для роботи алгоритму **BMA**. Якщо $f \leq bound_3$, тоді використовується алгоритм **BMA** для здійснення детального пошуку (*detailed search*). Коли досягнуто відомого рекорду задачі, або максимального часу розв'язання, алгоритм припиняє свою роботу.

**Algorithm 1** `General scheme of the hybrid algorithm for QAP`

**Require: Set of** `algorithms Algs` $\leftarrow$ {`2-Opt, Bls, Bma`};
**Ensure:** $(\pi^*,\ f^*)$

```
1  while not stopping condition do
2  π ← GetInitialSolution; π* ← π
3  f ← f(π); f* ← f
4    do
5      do
6        do
7          (π, f) ← 2-Opt(π*, f*)
8          if f < f* then
9            π* ← π; f* ← f
10           if f* ≤ f^bks then
11             SaveBestSolution(π*, f*)
12             return
13           end if
14         end if
15       while f > bound_1
16       (π, f) ← Bls(π*, f*)
17       if f < f* then
18         π* ← π; f* ← f
19         if f* ≤ f^bks then
20           SaveBestSolution(π*, f*)
21           return
22         end if
23       end if
24     while f > bound_2
25     (π, f) ← Bma(π*, f*)
26     if f < f* then
27       π* ← π; f* ← f
28       if f* ≤ f^bks then
29         SaveBestSolution(π*, f*)
30         return
31       end if
32     end if
33   while f > bound_3
34 end while
35 SaveBestSolution(π*, f*)
```

Лістинг 7: Формальна схема гібридного алгоритму HBMA.

**3. Результати числових експериментів.** Найбільш використовуваними штучними тестовими задачами, є дві серії тестів Taillard (*Taia* та *Taib*) [5]. В однорідних екземплярах (*Taia uniform distributed problems*) матриця відстаней є евклідовою матрицею відстаней між випадковими точками на колі, а матриця

потоку є випадковою матрицею з цілим числом, випадково вибраним між двома межами. Реально-подібні екземпляри (*Taib real-like problems*) створені на основі певних реальних проблем та імітують деякі їхні властивості. Матриця відстаней також є евклідовою матрицею, але де точки кластеризовані (розбиваються на групи), а значення матриці потоку розподілені експоненціально.

Були запропоновані й інші екземпляри. Серія екземплярів *Taie* та *Dre* була спеціально розроблена для складних метаевристик [6]. Крім того, створено екземпляри, які систематично змінюють два параметри тестової задачі; вони пов'язані з домінуванням та розрідженістю значень потоків [14]. Також було запропоновано окремий випадок QAP, який є поліноміально розв'язуваним [10], для тестування алгоритмів «чорної скриньки».

Числові експерименти проводилися на реально-подібних тестових задачах (*Taib*). У наступних таблицях використані такі позначення: *Dim* — розмірність задачі, *Bks* — відомий рекорд, *Solved* — кількість розв'язаних задач для яких досягнуто рекорд із загальної кількості задач, яка дорівнює 10, $\Delta_f$ — середнє відхилення цільової функції від відомого рекорду у процентах, $\Delta_{time}$ — середній час розв'язання у секундах, $T_{max}$ — максимальний час одного експерименту у секундах. Задачі розмірності 150, 100, 80 та 60 було взято з бібліотеки тестових задач QAPLIB [5]. Задачі інших розмірностей було створено на основі структури, наведеної у [16].

У таблицях 1, 2, 3 наведені результати розв'язання тестових задач ***taiXXb***, алгоритмами **HBMA**, **BMA** та **BLS** відповідно.

*Таблиця 1.*

Алгоритм HBMA

| **Dim** | **Bks** | **Solved** | $\Delta_f$ | $\Delta_{time}$ | $T_{max}$ |
|---|---|---|---|---|---|
| 150 | 498896643 | 10 | 0.0000% | 1449.781 | 640 |
| 140 | 2161522017 | 10 | 0.0000% | 644.044 | 4500 |
| 130 | 1751013618 | 10 | 0.0000% | 1445.184 | 3600 |
| 120 | 1411349647 | 10 | 0.0000% | 688.203 | 3000 |
| 110 | 1262728142 | 10 | 0.0000% | 89.341 | 1800 |
| 100 | 1185996137 | 10 | 0.0000% | 99.064 | 900 |
| 90 | 369834423 | 10 | 0.0000% | 202.881 | 900 |
| 80 | 818415043 | 10 | 0.0000% | 45.231 | 900 |
| 70 | 172783096 | 10 | 0.0000% | 163.544 | 600 |
| 60 | 608215054 | 10 | 0.0000% | 15.004 | 300 |

На рис. 1 візуально наводиться процес розв'язання тестової задачі **tai150b**. Зеленим кольором визначається робота алгоритму 2-OPT, синім — робота алгоритму BLS, коричневим — робота алгоритму. Як видно з рисунку, знадобилося по два проходи за алгоритмами 2-OPT та BLS для досягнення рекорду. Вісь ординат містить значення $\Delta_f$, вісь абсцис — час у секундах.

*Таблиця 2.*

Алгоритм BMA

| **Dim** | **Bks** | **Solved** | $\Delta_f$ | $\Delta_{time}$ | $T_{max}$ |
|---|---|---|---|---|---|
| 150 | 498896643 | 7 | 0.0074% | 4065.802 | 640 |
| 140 | 2161522017 | 10 | 0.0000% | 1271.875 | 4500 |
| 130 | 1751013618 | 8 | 0.0224% | 2619.210 | 3600 |
| 120 | 1411349647 | 10 | 0.0000% | 836.137 | 3000 |
| 110 | 1262728142 | 10 | 0.0000% | 244.235 | 1800 |
| 100 | 1185996137 | 10 | 0.0000% | 119.038 | 900 |
| 90 | 369834423 | 9 | 0.0158% | 375.815 | 900 |
| 80 | 818415043 | 10 | 0.0000% | 78.368 | 900 |
| 70 | 172783096 | 9 | 0.0031% | 201.342 | 600 |
| 60 | 608215054 | 10 | 0.0000% | 12.181 | 300 |

*Таблиця 3.*

Алгоритм BLS

| **Dim** | **Bks** | **Solved** | $\Delta_f$ | $\Delta_{time}$ | $T_{max}$ |
|---|---|---|---|---|---|
| 150 | 498896643 | 1 | 0.1088% | 6215.109 | 640 |
| 140 | 2161522017 | 2 | 0.2400% | 4214.208 | 4500 |
| 130 | 1751013618 | 2 | 0.1802% | 3463.286 | 3600 |
| 120 | 1411349647 | 3 | 0.0947% | 2880.278 | 3000 |
| 110 | 1262728142 | 4 | 0.0811% | 1355.981 | 1800 |
| 100 | 1185996137 | 4 | 0.1116% | 766.291 | 900 |
| 90 | 369834423 | 3 | 0.0669% | 726.472 | 900 |
| 80 | 818415043 | 7 | 0.2287% | 575.560 | 900 |
| 70 | 172783096 | 5 | 0.1796% | 394.416 | 600 |
| 60 | 608215054 | 9 | 0.1077% | 139.153 | 300 |

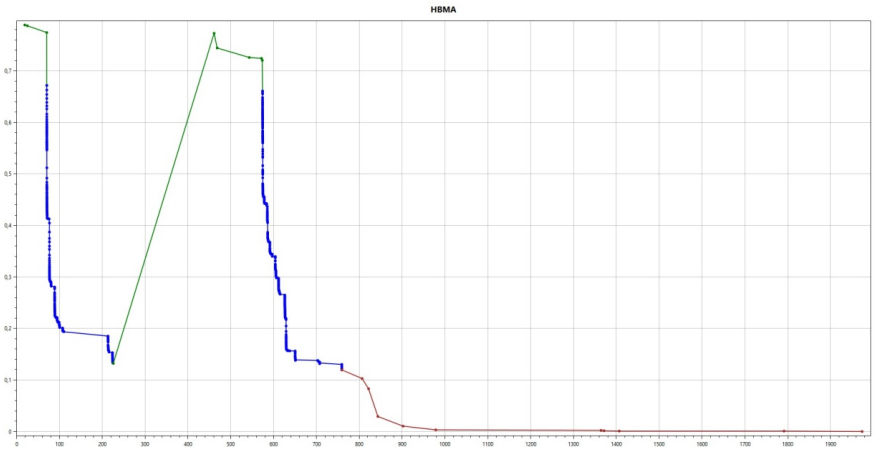


Рис. 1. Траєкторія руху за алгоритмом HBMA для задачі **tai150b**.

**4. Висновки та перспективи подальших досліджень.** У цій статті обговорюється ефективна реалізація гібридного алгоритму **HBMA** для розв'я-

зання задачі QAP. Метою цього алгоритму є покращення продуктивності пошуку шляхом спроби мінімізувати негативний вплив явища стагнації, що є однією з головних перешкод ітераційного пошуку, особливо у випадках, коли необхідно виконати значну кількість кроків на кожному етапі пошуку. Кожен з етапів пошуку алгоритму **HBMA** базується на парадигмі інтенсифікації та диверсифікації, де інтенсифікація спрямована на пошук кращих розв'язків поблизу поточного, тоді як диверсифікація відповідає за вихід з поточного локального оптимуму та переміщення до нових невивчених областей простору пошуку. На кожному етапі пошуку використовується різні алгоритми. На етапі дослідницького пошуку використовується алгоритм **2-OPT**, на етапі розширеного пошуку — **BLS**, на етапі детального пошуку — **BMA**. Кожен алгоритм задає свою траєкторію пошуку, тобто процесу переходу від одного розв'язку до іншого. Тому, якщо використовується тільки один конкретний алгоритм при розв'язанні задачі, то дуже часто такий пошук призводить до переходу у режим стагнації (застою), особливо для "важких" задач, у яких при наймі розмірність є більшою за 100. Комбінація застосування алгоритмів дозволяє більш ефективно здійснювати процедуру диверсифікації, що значно підвищує ефективність всього алгоритму. Про це яскраво свідчать результати числових експериментів. Алгоритм **HBMA** показує кращі результати як за кількістю розв'язаних задач та і за середнім часом розв'язання у порівнянні з використанням "чистих" алгоритмів **BLS** та **BMA**.

У подальших дослідженнях планується:

1. Застосувати алгоритм **HBMA** до задачі про максимальний розріз графу.
2. Застосувати алгоритм **HBMA** до багатовимірної булевої задачі про ранець.
3. Використати комбінації інших алгоритмів, таких як **3-OPT**, **RITSR, ITS, ReTS, IHGA** разом із **2-OPT, BLS** та **BMA**.
4. Дослідити траєкторії руху на для різних алгоритмів з метою визначення більш придатних значень параметрів $bound_1$, $bound_2$ та $bound_3$ для конкретних тестових задач.
5. Розробити механізм адаптації параметрів алгоритму в залежності від ходу процесу розв'язання.

## Список використаної літератури


1. Battiti, R., & Tecchiolli, G. (1994). The Reactive Tabu Search. *ORSA J. on Computing,* 6, 126–140.
2. Benlic, U., & Hao, J. K. (2013). Breakout local search for the quadratic assignment problem. *Applied Mathematics and Computation,* 219(9), 4800–4815.
3. Benlic, U., & Hao, J. K. (2015). Memetic search for the quadratic assignment problem. *Expert Systems with Applications,* 42(1), 584–595.
4. Burkard, R. E. (1984). Quadratic assignment problems. *European J. of Oper. Res.,* 15, 283–289.
5. Burkard, R. E., Karisch, S. E., & Rendl, F. (1997). Qaplib — a quadratic assignment problem library. *Journal of Global optimization,* 10, 391–403.
6. Drezner, Z., Hahn, P. M., & Taillard, E. D. (2005). Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for metaheuristic methods. *Annals of Operations research,* 139, 65–94.
7. Glover, F. (1989). Tabu Search — Part I. *ORSA J. on Computing,* 1(3), 190–206.
8. Glover, F. (1990). Tabu Search — Part II. *ORSA J. on Computing,* 2(1), 4–32.
9. Koopmans, T. C., & Beckmann, M. J. (1957). Assignment problems and the location of economic activities. *Econometrica,* 25, 53–76.

10. Laurent, M., & Seminaroti, M. (2015). The quadratic assignment problem is easy for robinsonian matrices with toeplitz structure. *Operations Research Letters,* 43(1), 103–109.
11. Misevicius, A. (2004). An improved hybrid genetic algorithm: New results for the quadratic assignment problem. *Knowledge Based Systems,* 17(2–4), 65–73.
12. Misevicius, A. (2012). An implementation of the iterated tabu search algorithm for the quadratic assignment problem. *OR Spectrum,* 34(3), 665–690.
13. Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *J. of the ACM,* 23, 555–565.
14. Stutzle, T., & Fernandes, S. (2004). New benchmark instances for the qap and the experimental analysis of algorithms. In: *European Conference on Evolutionary Computation in Combinatorial Optimization.* Springer, 199–209.
15. Taillard, E. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing,* 17, 443–455.
16. Taillard, E. D. (1995). Comparison of iterative searches for the quadratic assignment problem. *Location science,* 3(2), 87–105.
17. Zhang, Q., Sun, J., Tsang, E., & Ford, J. (2006). Estimation of Distribution Algorithm with 2-opt Local Search for the Quadratic Assignment Problem. In: Lozano, J. A., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.). *Towards a New Evolutionary Computation.* Studies in Fuzziness and Soft Computing. Springer. Berlin: Heidelberg, 192. https://doi.org/10.1007/3-540-32494-1_12
18. Sergienko, I. V., Shylo, V. P., Chupov, S. V., & Shylo, P. V. (2020). On solving the quadratic assignment problem. *Cybernetics and Systems Analysis,* (1), 64–69 [in Ukrainian].
19. Shylo, V. P., Chupov, S. V., Boyarchuk, D. O., & Shylo, P. V. (June 3–5, 2018). New approaches to solving the quadratic assignment problem. In: *VII International Scientific and Practical Conference "Mathematics. Information Technologies. Education."* Abstracts of reports. Lutsk: Lesya Ukrainka Eastern European National University. Lutsk: Svityaz, 121–122 [in Ukrainian].

**Chupov S. V., Fedorishko A. V.** Hybrid algorithm for solving the quadratic assignment problem.

The quadratic assignment problem (QAP) is one of the most studied combinatorial optimization problems with various practical applications. In this paper, we present a hybrid approximation algorithm (HBMA) for solving QAP. HBMA explores the search space by using the well-known algorithms 2-OPT, BLS, and BMA at separate stages of solving the problem. Experimental evaluations on a set of benchmark problems from QAPLIB show that the proposed approach is able to achieve the best-known results to date for all instances, except for some ***taia***-type problems, with an average computational time of less than 2 hours. Comparisons are also provided to show the competitiveness of the proposed approach with respect to the BLS and BMA algorithms for QAP.

**Keywords:** Quadratic assignment problem, hybrid approximate search algorithm, local approximate search algorithm, random perturbation of solution components, algorithm efficiency.

Одержано 05.09.2025