**УДК 519.16**

**Р.П. Базилевич, Р.К. Кутельмах**
Національний університет "Львівська політехніка",
кафедра програмного забезпечення

# АЛГОРИТМ ПРИЄДНАННЯ ЧАСТКОВИХ РОЗВ'ЯЗКІВ У ПІДМНОЖИНАХ ПРИ ДЕКОМПОЗИЦІЇ ЗАДАЧІ КОМІВОЯЖЕРА



**Запропоновано новий метод приєднання часкових розв'язків у підмножинах при декомпозиції задачі комівояжера. Алгоритм передбачає розбиття вхідної множини точок на підмножини. Процес розв'язання починається в певній підмножині, де розв'язок відомий. Розв'язок у наступній (сусідній) підмножині утворюється за допомогою розширення існуючого розв'язку.**

**Ключові слова – задача комівояжера, транспортна задача, декомпозиція, підмножина.**

**New algorithm is suggested for union partial TSP solutions in subsets, created with the decomposition. The algorithm presupposes splitting the initial vertex set into subsets. The solution begins in a certain initial subset, where solution is found. The solution in the next subset is found by expanding the existing initial one.**

**Keywords – traveling salesman problem, vehicle routing problem, decomposition, subarea.**

## Вступ

Алгоритми, пов'язані з транспортними задачами, мають велике прикладне застосування [1, 2] – окрім транспортних систем, їх використовують у системах телекомунікацій, виробництві друкованих плат, лазерній нарізці пластмас і металів тощо. З часом вимоги до швидкості обчислень та якості підвищуються, а також зростають розмірності задач, що приводить до необхідності розроблення спеціалізованих алгоритмів для швидкого та якісного розв'язування задач великих розмірностей.

Задача комівояжера – одна із базових транспортних задач. Нині методи її розв'язання розглядаються у багатьох наукових та методично-навчальних працях. Задача є широко дослідженою, але простота її формулювання поєднується з великою складністю розв'язування навіть для задач малих розмірностей. Актуальним є розроблення ефективних декомпозиційних алгоритмів для задачі комівояжера, які б забезпечили отримання якісних розв'язків для задач великих розмірностей з обчислювальною складністю, що не є не вищою за $O(N\log N)$.

Існує небагато алгоритмів, що забезпечують одержання якісних розв'язків задачі комівояжера, особливо за малих часових затрат [3]. Для розв'язування задачі комівояжера алгоритм Ліна–Кернігана є одним з найефективніших [4, 5]. Його обчислювальна складність – $O(n^2)$. Одержані результати – у межах 1–3 % від оптимального [3]. Впродовж останніх років було отримано нову версію алгоритму Ліна–Кернігана – алгоритм Ліна–Кернігана–Гельсгауна [6]. Він забезпечує оптимальний розв'язок задачі для 7397 точок із бібліотеки тестів для транспортних задач – TSPLIB[7]. Як показали результати тестування відомих методів розв'язування задачі комівояжера DIMACS TSP Challenge [3], він є найточнішим евристичним алгоритмом [3,8]. Обчислювальна складність алгоритму – $O(n^{2,2})$.

Групою вчених [9–12] розроблено пакет програмного забезпечення для точного розв'язування задачі комівояжера – Concorde TSP Solver [13]. Він забезпечив одержання оптимальних розв'язків для усіх тестів із бібліотеки TSPLIB, розмірністю, зокрема, 85900 точок. Розв'язання задачі

розмірністю 85900 точок зайняло майже 136 років процесорного часу на кластері з ПК з процесорами Intel Xeon та AMD Opteron.

### Формулювання задачі

Задача комівояжера подається як граф $G=(V, E)$, де $V$ – множина вершин графа, а $E$ – множина його ребер. Вага (або довжина) $c_{i,j}$ кожного ребра $e_{i,j} \in E$ вважається заданою. Задача вважається симетричною, якщо $c_{i,j} = c_{j,i}, \forall i, j \in V$. Крім того, задачу вважають евклідовою за умови, що $c_{i,j} + c_{j,k} \geq c_{i,k}\ \forall i, j, k \in V$. Необхідно знайти гамільтонів цикл мінімальної ваги, де гамільтонів цикл – це закритий цикл у графі, що включає усі вершини та передбачає відвідування кожної вершини лише один раз.

Розглядається симетрична евклідова задача комівояжера, де заданими вважають множину $N$ з $n$ точок ($|N|=n$), які описані їх координатами $(x_i, y_i)$. Необхідно знайти маршрут $S^*$, що проходить по одному разу через кожну точку, довжина якого $L^*(S^*)$ є мінімальною:

$L^*(S^*) = \Sigma_{ij}\ l_{ij}^* \rightarrow \min \Sigma_{ij}\ l_{ij}^*\ \forall\ l_{ij}^* \in l_{ij}^{\ell}$,

де $l_{ij}^{\ell}$ – деяка з допустимих за заданими обмеженнями ділянка між двома суміжними точками $i$ та $j$ виділеного маршруту.

Запропонований алгоритм передбачає розділення вхідної множини точок на підмножини ($N_0$, $N_1$,..., $N_k \in N$). Розв'язання задачі починається в певній підмножині – початковій підмножині $N_0$. Маючи розв'язок в початковій підмножині, у певний спосіб знаходять розв'язок у наступній, після чого їх розв'язки об'єднуються. Операція триває до знаходження розв'язку у всіх підмножинах. Задача з $n$ точками замінюється $m$ задачами у підмножинах. Для розв'язання задачі комівояжера у певній підмножині застосовується деякий класичний алгоритм.

Для задачі комівояжера з кластерним розподілом точок [14, 15] запропоновано декомпозиційні алгоритми, що розглядають кластери близько розміщених точок як підмножини, між якими розміщений макромаршрут. Також запропоновано стратегії знаходження початкового маршруту та його оптимізації [16–18].

### Формування підмножин

*Вхідна область точок ділиться на підмножини за заданими шириною та висотою підмножин або кількістю точок у ній. Рис. 1 містить вхідну область точок та початкову підмножину* $N_0$.

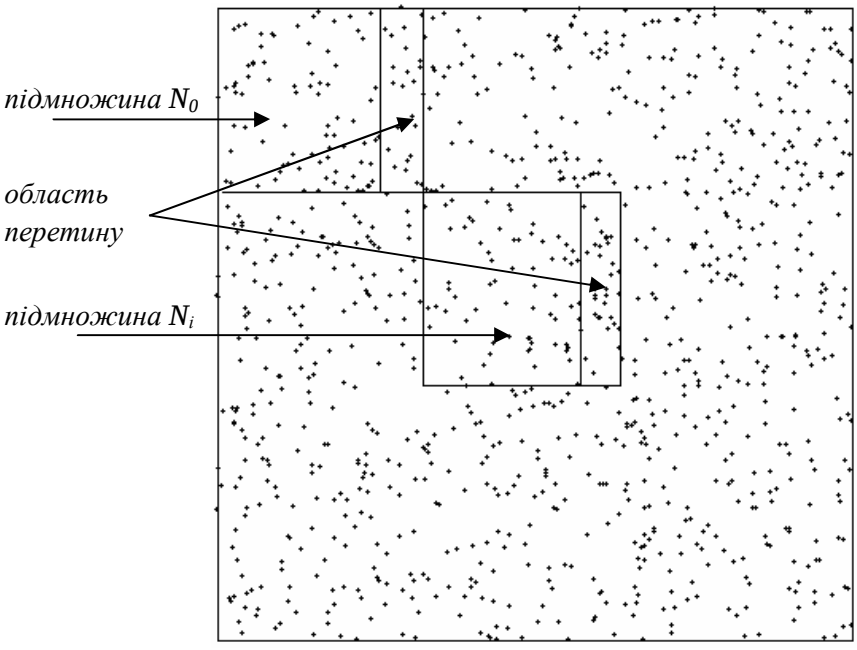


*Рис. 1. Формування підмножин*

**Алгоритм приєднання**

Опишемо процес приєднання часткових розв'язків задачі комівояжера. Розглянемо підмножини $N_0$ та $N_1$. Допустимо, що в підмножині $N_0$ розв'язок відомий (рис. 2).

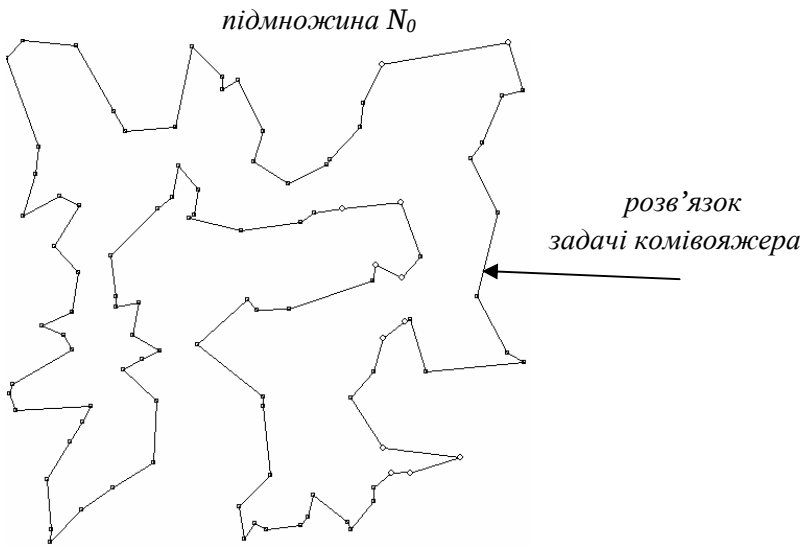


*Рис. 2. Розв'язок задачі комівояжера у підмножині $N_0$*

Необхідно знайти розв'язок у підмножині $N_1$ та об'єднати розв'язки підмножин $N_0$ та $N_1$.

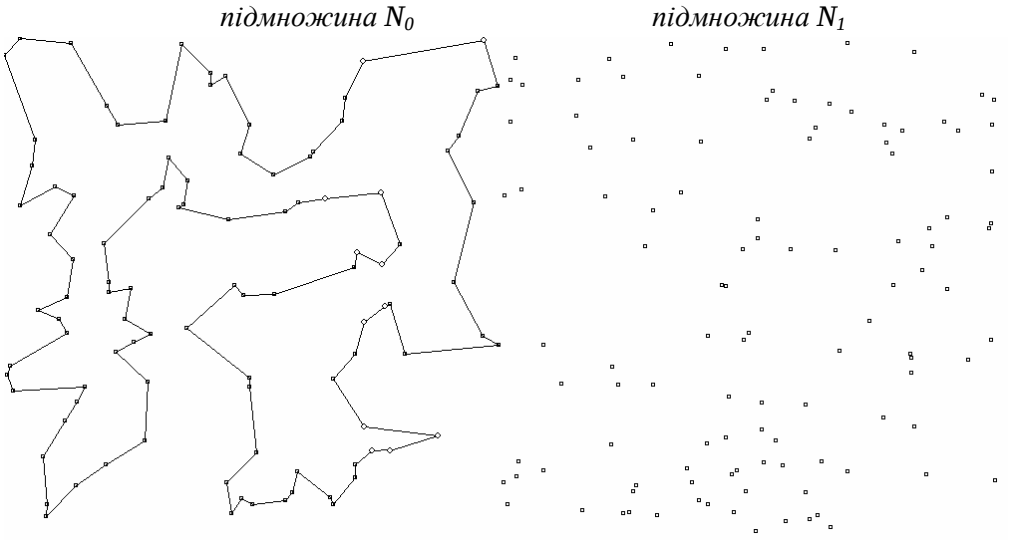


*Рис. 3. Підмножини $N_0$ та $N_1$*

Розглянемо такий параметр, як величина області "перетину" $\alpha$. Він може задаватися або у числовому вигляді, або в процентному відносно розміру підмножини. Рекомендоване значення параметра – у межах 20–70 % від розміру підмножини. Рис. 4 зображає підмножини $N_0$ та $N_1$ з областю перетину.

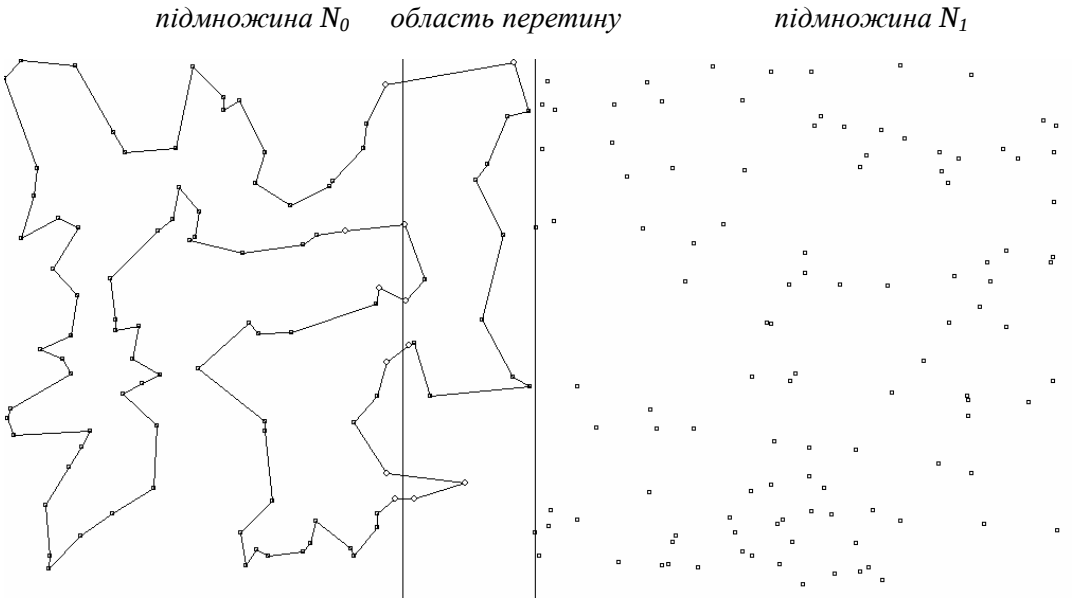


*Рис. 4. Підмножини $N_0$, $N_1$ та область перетину*

Для знаходження єдиного розв'язку задачі комівояжера для двох підмножин $N_0$ та $N_1$ розглянемо множину $N^*$, що містить усі точки з множини $N_1$ та точки з множини $N_0$, які належать ребрам з області перетину $\alpha$. Розв'язок задачі комівояжера для $N^*$ міститиме розв'язок для всієї множини $N_1$ та тієї частини множини $N_0$, яка є наближчою до підмножини $N_1$. Для того, щоб на наступному кроці можна було об'єднати розв'язки задачі комівояжера, у підмножинах $N_0$ та $N^*$ застосовуються фіксовані ребра, які встановлюються між першою відвіданою точкою та останньою, а також між парами вхідних та вихідних точок у послідовності їх відвідування. Маючи усі необхідні вхідні дані (точки та умовні ребра), можемо розв'язати задачу комівояжера для підмножини $N^*$. Рис. 5 ілюструє розв'язок задачі комівояжера для неї.

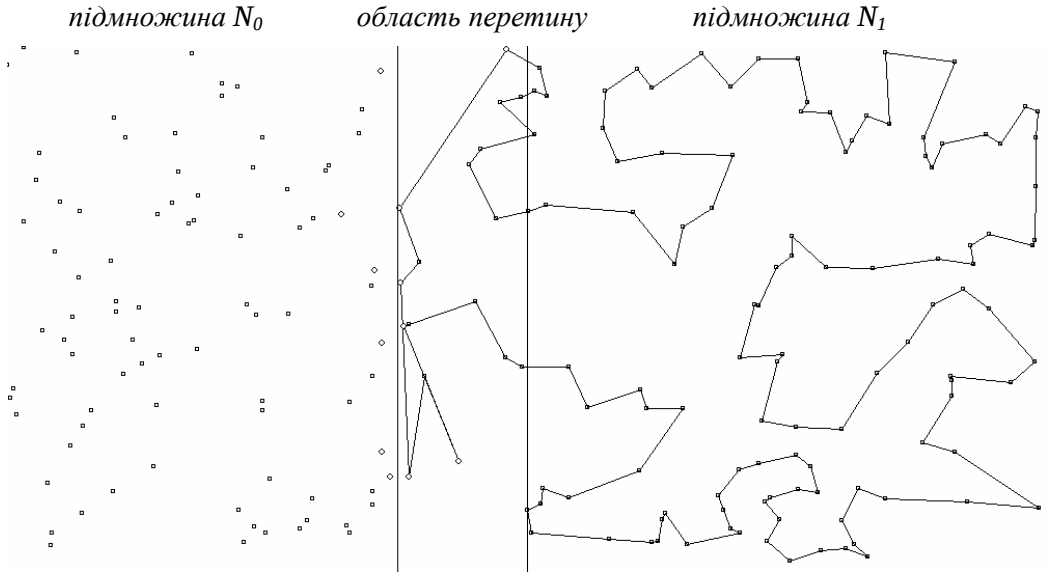


*Рис. 5. Розв'язок задачі комівояжера для підмножини $N^*$*

Останній крок – вилучення фіксованих ребер зі знайденого розв'язку та об'єднання його з розв'язком для підмножини $N_0$. У результаті одержуємо розв'язок задачі комівояжера для усіх точок підмножин $N_0$ та $N_1$. Рис. 6 показує результат об'єднання розв'язків у підмножинах $N_0$ та $N_1$.

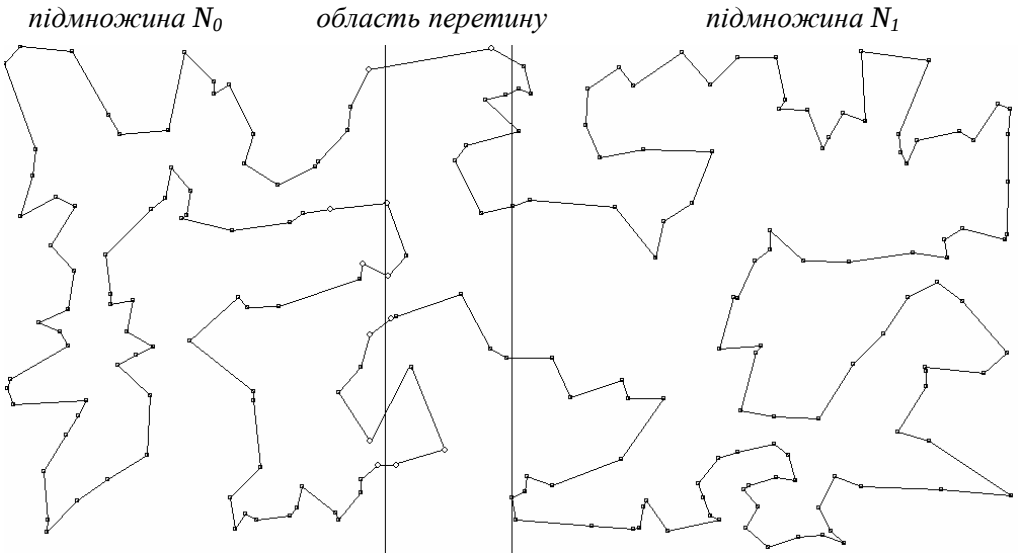


*Рис. 6. Об'єднання розв'язків у підмножинах $N_0$ та $N^*$ в розв'язок задачі для підмножин $N_0$ та $N_1$*

Алгоритм триває, доки всі підмножини не приєднаються до повного розв'язку вхідної множини точок. Табл. 1 містить псевдокод описаного алгоритму.

*Таблиця 1*

| |
|---|
| **Алгоритм** Приєднання підмножин |
| **Вхідні дані**: підмножини $N_i \in N$ та область перетину $\Delta N$ |
| **Вихідні дані**: Розв'язок задачі $S^*$ |
| 1) ЗНАЙТИ розв'язок задачі $S^*_0$ в множині $N_0$<br>2) $S^* \leftarrow S^*_0$<br>3) ДЛЯ УСІХ інших підмножин $N_i$:<br>a. ВИБРАТИ підмножину $N_i$<br>b. ВИБРАТИ множину $Neighbors(N_i)$ усіх сусідніх підмножин $N_k,\ldots, N_l$ для яких задача вже розв'язана<br>c. ВИБРАТИ множину точок з $Neighbors(N_i)$ які належать області перетину $\Delta N$<br>*d.* ВСТАНОВИТИ граничні точки, що належать існуючому маршруту $S^*$ та $\Delta N$<br>e. ВИБРАТИ підмножину $N^* = N_{ij} \cup \Delta N$<br>f. ЗНАЙТИ розв'язок задачі $S^*_i$ для підмножини $N^*$<br>g. ОБ'ЄДНАТИ існуючий маршрут $S^*$ та $S^*_i$ |

Після того, як розв'язок одержано, застосовується його оптимізація. Для покращання якості отриманого початкового маршруту пропонується використовувати різні методи оптимізації – послідовної та геометричної [19]. Послідовна оптимізація передбачає покращання маршруту на його ділянках, що вибираються послідовно вздовж існуючого маршруту. Розмір ділянки задається розміром елементарної області.

Як вхідні дані алгоритму подається маршрут, що потрібно оптимізувати, та додаткові параметри, що охоплюють:

- розмір області оптимізації (scanning area size – SAsize) – кількість точок, що входять до однієї області оптимізації;
- розмір області перетину (overlapping area size – OAsize) – кількість спільних точок, що належать двом "сусіднім" областям оптимізації;
- базовий алгоритм;

Оптимізація відбувається так. У вхідному маршруті вибирають ділянку з кількістю точок SAsize. Перша та остання точки ділянки вважаються граничними. За допомогою певного вибраного базового алгоритму розв'язується незамкнута задача комівояжера для множини точок ділянки із заданим умовним ребром (що з'єднує його граничні точки). Довжина отриманого маршруту порівнюється із довжиною існуючого і, якщо є покращання, маршрут замінюється на новий. Далі вибирається наступна ділянка з такою самою кількістю точок. Операція триває доти, доки не буде розглянуто усі точки вхідного маршруту.

Геометрична оптимізація полягає у оптимізації декількох маршрутів одночасно – частин загального маршруту, розміщених геометрично близько одна до одної. У певний спосіб вибирається геометрична область довільної форми (квадратна, прямокутна, кругла тощо), у якій розпізнаються окремі ділянки загального маршруту, та за допомогою дещо модифікованого базового алгоритму розв'язання задачі комівояжера розв'язується задача сумарної мінімізації маршрутів. Якщо сума довжин нових ділянок менша від суми старих, то старі ділянки маршрутів замінюються новими.

Як вхідні дані для алгоритму подається загальний маршрут, який потрібно оптимізувати, та параметри:

- розмір геометричної області сканування (якщо область прямокутна, то ширина і висота, якщо кругла – то радіус і т.д.);
- розмір області перетину (у відсотках) – розмір спільної геометричної області, що належить двом сусіднім областям сканування;
- базовий алгоритм.

Метод може бути застосовано для покращання будь-якого маршруту – початкового чи вже оптимізованого. Пропонується повторне проходження алгоритму оптимізації (із зміненими параметрами, наприклад, із зміненим розміром області сканування), де очікується ще деяке покращання якості маршруту.

## Експерименти

Здійснено дослідження задач розмірністю 1000, 2000, 3000, 4000, 5000, 7000 та 10000 точок. Як базовий було взято алгоритм Ліна–Кернігана–Гельгауна, що є найкращим сьогодні евристичним алгоритмом. Вхідну множину точок було поділено на підмножини 50, 100 та 200 точок. Розмір області перетину задавався згідно з такими значеннями: 10 %, 20 %, 30 %, 40 %, 50 %, 60 %, 70 %, 80 %.

Час роботи базового алгоритму наведено в табл. 2.

*Таблиця 2*

| **Алгоритм LKH** | | | | | | | |
|---|---|---|---|---|---|---|---|
| Розмірність задачі, точки | 1000 | 2000 | 3000 | 4000 | 5000 | 7000 | 10000 |
| Час роботи, секунди | 24 | 74 | 334 | 784 | 920 | 2644 | 6520 |

Табл. 3 містить усі результати тестування запропонованого підходу (час та якість обчислень відносно базового) для задач розмірністю 1000, 2000, 3000, 4000, 5000, 7000 та 10000 точок.

*Таблиця 3*

| Область перетину / Кількість точок підмножини | 10 % | 20 % | 30 % | 40 % | 50 % | 60 % | 70 % | 80 % |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Розмірність задачі – 1000 точок. Час обчислень, секунди | | | | | | | | |
| 50 | 1,1 | 1,7 | 2,0 | 2,6 | 3,0 | 3,2 | 4,2 | 4,3 |
| 100 | 1,5 | 2,2 | 2,8 | 2,7 | 3,2 | 5,0 | 4,6 | 5,5 |
| 200 | 2,4 | 2,7 | 2,3 | 2,6 | 3,2 | 4,8 | 6,4 | 6,0 |
| Розмірність задачі – 1000 точок. Якість маршруту | | | | | | | | |
| 50 | 4,67 % | 2,26 % | 2,23 % | 1,90 % | 1,05 % | 0,92 % | 0,94 % | 0,58 % |
| 100 | 2,28 % | 1,48 % | 1,27 % | 0,87 % | 0,67 % | 0,58 % | 0,43 % | 0,44 % |
| 200 | 1,25 % | 0,81 % | 0,49 % | 0,38 % | 0,32 % | 0,23 % | 0,26 % | 0,23 % |
| Розмірність задачі – 2000 точок. Час обчислень, секунди | | | | | | | | |
| 50 | 4,3 | 5,4 | 5,8 | 6,6 | 8,7 | 9,7 | 10,8 | 12,2 |
| 100 | 6,0 | 7,0 | 7,5 | 9,4 | 11,8 | 15,0 | 12,9 | 15,3 |
| 200 | 6,4 | 7,3 | 8,5 | 12,0 | 13,3 | 15,2 | 15,6 | 19,0 |
| Розмірність задачі – 2000 точок. Якість маршруту | | | | | | | | |
| 50 | 7,02 % | 3,38 % | 1,82 % | 1,54 % | 1,00 % | 0,91 % | 0,53 % | 0,47 % |
| 100 | 2,38 % | 1,57 % | 0,85 % | 0,44 % | 0,31 % | 0,25 % | 0,19 % | 0,19 % |
| 200 | 1,13 % | 0,50 % | 0,31 % | 0,25 % | 0,19 % | 0,16 % | 0,03 % | 0,00 % |
| Розмірність задачі – 3000 точок. Час обчислень, секунди | | | | | | | | |
| 50 | 7,5 | 12,8 | 13,8 | 17,4 | 17,2 | 20,9 | 25,6 | 27,2 |
| 100 | 11,5 | 13,2 | 17,3 | 17,0 | 21,9 | 29,9 | 31,1 | 39,0 |
| 200 | 17,7 | 21,5 | 20,1 | 20,7 | 21,5 | 27,5 | 34,2 | 43,6 |
| Розмірність задачі – 3000 точок. Якість маршруту | | | | | | | | |
| 50 | 5,08 % | 2,96 % | 1,73 % | 1,42 % | 0,88 % | 0,72 % | 0,52 % | 0,57 % |
| 100 | 2,37 % | 1,50 % | 1,03 % | 0,80 % | 0,64 % | 0,46 % | 0,39 % | 0,23 % |
| 200 | 0,90 % | 0,44 % | 0,21 % | 0,10 % | 0,05 % | 0,03 % | 0,03 % | 0,08 % |
| Розмірність задачі – 4000 точок. Час обчислень, секунди | | | | | | | | |
| 50 | 11,9 | 14,4 | 18,5 | 24,4 | 25,6 | 29,4 | 35,8 | 45,1 |
| 100 | 19,5 | 19,8 | 24,2 | 26,5 | 30,8 | 37,7 | 41,7 | 47,6 |
| 200 | 21,3 | 29,8 | 26,3 | 35,6 | 43,1 | 46,2 | 49,8 | 65,0 |
| Розмірність задачі – 4000 точок. Якість маршруту | | | | | | | | |
| 50 | 5,44 % | 3,10 % | 2,08 % | 1,74 % | 1,11 % | 0,95 % | 0,70 % | 0,57 % |
| 100 | 3,42 % | 1,81 % | 1,06 % | 0,68 % | 0,52 % | 0,43 % | 0,29 % | 0,18 % |
| 200 | 1,72 % | 0,79 % | 0,45 % | 0,32 % | 0,27 % | 0,27 % | 0,18 % | 0,14 % |
| Розмірність задачі – 5000 точок. Час обчислень, секунди | | | | | | | | |
| 50 | 16,9 | 21,3 | 23,6 | 28,6 | 38,6 | 40,2 | 46,3 | 53,0 |
| 100 | 20,5 | 25,3 | 32,0 | 36,4 | 43,8 | 51,1 | 55,7 | 63,1 |
| 200 | 24,8 | 30,6 | 34,8 | 44,6 | 54,4 | 67,6 | 72,6 | 72,7 |
| Розмірність задачі – 5000 точок. Якість маршруту | | | | | | | | |
| 50 | 7,62 % | 4,53 % | 3,00 % | 2,25 % | 1,53 % | 1,33 % | 0,98 % | 0,73 % |
| 100 | 3,96 % | 1,71 % | 1,41 % | 0,98 % | 0,80 % | 0,57 % | 0,41 % | 0,27 % |
| 200 | 1,55 % | 0,86 % | 0,45 % | 0,33 % | 0,20 % | 0,22 % | 0,22 % | 0,12 % |
| Розмірність задачі – 7000 точок. Час обчислень, секунди | | | | | | | | |
| 50 | 26,2 | 28,7 | 35,5 | 42,6 | 52,2 | 58,0 | 67,3 | 83,2 |
| 100 | 32,9 | 35,1 | 37,9 | 44,5 | 56,2 | 66,9 | 73,6 | 85,6 |
| 200 | 34,7 | 42,5 | 52,8 | 55,9 | 69,3 | 88,9 | 98,4 | 112,8 |
| Розмірність задачі – 7000 точок. Якість маршруту | | | | | | | | |
| 50 | 6,65 % | 3,65 % | 2,26 % | 1,40 % | 1,04 % | 0,85 % | 0,62 % | 0,38 % |
| 100 | 3,77 % | 2,07 % | 0,92 % | 0,60 % | 0,45 % | 0,38 % | 0,31 % | 0,17 % |
| 200 | 1,16 % | 0,57 % | 0,36 % | 0,26 % | 0,21 % | 0,12 % | 0,10 % | 0,09 % |

*Продовження табл. 3*

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| Розмірність задачі – 10000 точок. Час обчислень, секунди | | | | | | | | |
| 50 | 36,9 | 39,7 | 52,1 | 61,3 | 67,6 | 76,9 | 87,4 | 101,0 |
| 100 | 37,7 | 46,4 | 57,5 | 63,2 | 78,2 | 93,3 | 118,3 | 145,6 |
| 200 | 51,3 | 66,7 | 81,3 | 94,4 | 112,2 | 139,0 | 160,5 | 182,0 |
| Розмірність задачі – 10000 точок. Якість маршруту | | | | | | | | |
| 50 | 6,26 % | 2,96 % | 1,96 % | 1,12 % | 0,65 % | 0,45 % | 0,28 % | 0,22 % |
| 100 | 2,87 % | 1,18 % | 0,55 % | 0,36 % | 0,22 % | 0,13 % | 0,06 % | 0,06 % |
| 200 | 1,28 % | 0,46 % | 0,16 % | 0,15 % | 0,07 % | 0,06 % | 0,03 % | 0,04 % |

Наступні рисунки (7 та 8) містять графіки залежностей часу обчислень від розмірності задачі для базового алгоритму без декомпозиції та запропонованого алгоритму.

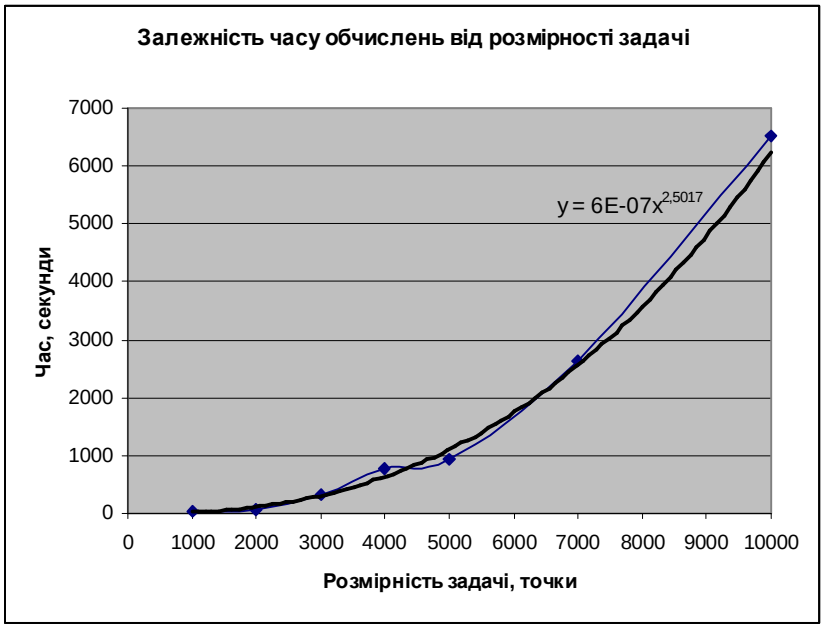


*Рис. 7. Залежність часу обчислень від розмірності задачі для алгоритму Ліна–Кернігана–Гельсгауна*

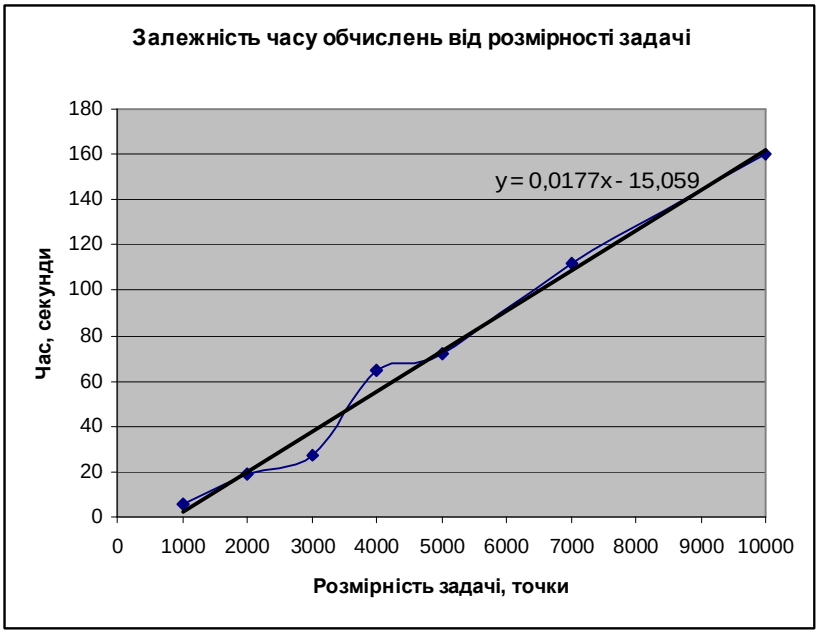


*Рис. 8. Залежність часу обчислень від розмірності задачі для декомпозиційного алгоритму*

Рис. 9 містить графік залежності якості розв’язку від розмірності задачі для запропонованого алгоритму.


*Рис. 9. Залежність якості розв’язку від розмірності задачі для декомпозиційного алгоритму*

## Висновки

Запропонований алгоритм приєднання часткових розв’язків задачі комівояжера демонструє погіршення якості маршруту в межах 0,03–0,20 % (за великих розмірів підобластей та області перетину) відносно базового алгоритму без декомпозиції. В експериментах використовувався найкращий нині алгоритм, що забезпечує якість маршруту, в межах 1 % від оптимального. За незначної втрати якості (на 0,03 %) спостерігається виграш у часі (в 40 разів для задачі розмірністю 10000 точок), що вказує на доцільність використання алгоритму для задач великих розмірностей. Із зростанням розмірності задачі якість одержаного розв’язку не погіршується.

Необхідне подальше дослідження стратегій оптимізації одержаного маршруту та тестування алгоритму на задачах великих розмірностей.

*1. Reinelt, Gerhard (1994), The Traveling Salesman: Computational Solutions for TSP Applications. Lecture Notes in Computer Science 840, Springer–Verlag, Berlin. 2. Reinelt, Gerhard (1992), Fast heuristics for large geometric traveling salesman problems. ORSA Journal on computing, 4:206-217 3. David S. Johnson and Lyle A. McGeoch. Experimental Analysis of Heuristics for the STSP. In Gutin and Punnen, editors, The Traveling Salesman Problem and its Variations. Kluwer Academic Publishers, 2002. 4. Lin S. and Kernighan B. W. An effective heuristic algorithm for the Traveling salesman problern. Operations Research, 21:498–516. 1973. 5. Lin S. Computer solutions of the travelling salesman problem. Bell System Technical Journal 44, pages 2245–2269, 1965. 6. K. Helsgaun, “An effective implementation of the Lin–Kernighan Traveling Salesman Heuristic”, 2002. 7. Reinelt, Gerhard(1991) TSPLIB – A traveling salesman problem library, ORSA Journal on Computing 3, 376–384. 8. http://www.research.att.com/~dsj/chtsp/ 9. Applegate D., Bixby R.E., Chvátal V. and Cook W. On the solution of traveling salesman problems. Documenta Mathematica, Extra Volume ICM III:645–656, 1998. 10. Applegate D., Cook W., Rohe A.. Chained Lin–Kernighan for large traveling salesman problems. INFORMS J. Computing, to appear. 11. Applegate D., Bixby R.E., Chvátal V. and Cook W. Findong tours in the TSP.1998. 12. Applegate D., Bixby R. E., Chvátal V., and Cook W. Findong cuts in the TSP.1995 13.*

*http://www.tsp.gatech.edu/concorde.html. 14. Neto D. Efficient cluster compensation for Lin–Kernighan Heuristics. PhD thesis, Department of Computer Science, University of Toronto, 1999. 15. Laporte G., Potvin J-Y., Quilleret F. A Tabu Search using Genetic Diversification for the Clustered Traveling Salesman Problem // Journal of Heuristics, Vol 2 (3), p. 187-200, 1996. 16. Базилевич Р. П., Кутельмах Р. К. Алгоритми динамічного формування моделі робочого поля для задачі комівояжера з кластерним розподілом точок // Вісник Нац. ун-ту "Львівська політехніка", Львів, 2006. 17. Базилевич Р. П., Ремі Дюпа, Кутельмах Р. К. Використання алгоритмів локальної оптимізації для розв'язування задачі комівояжера з кластерним розподілом точок // Вісник Нац. ун-ту "Львівська політехніка". – Львів, 2006. 18. Bazylevych R., Dupas R, Kutelmakh R. Scanning-area algorithms for clustered TSP // Proceedings of International conference "Comp. science and Information Technologies", Lviv, Polytechnic University, 2006, pp. 148–152. 19. Bazylevych R., Prasad B., Kutelmakh R., L.Bazylevych. Decomposition and Scanning Optimization Algorithms for TSP, Proceedings of the International Conference on Theoretical and Mathematical Foundations of Computer Science (TMFCS-08), pp. 110–116, Florida, USA, 2008.*

**УДК 004.89**

**А.Ю. Берко, О.М. Явлінський**
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж

# ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ПРИ УПРАВЛІННІ НЕПРИБУТКОВИМИ ПРОЕКТАМИ



**Описано створену інтелектуальну систему, метою якої є визначати ймовірність успішності неприбуткових проектів, що зазвичай організовуються неприбутковими та неурядовими організаціями. Науково обґрунтованим є використання теорії ймовірності та методу ядрового згладжування при визначенні ймовірностей з оперуванням малими обсягами наявних даних.**

**Ключові слова – системи підтримки прийняття рішень, інтелектуальні системи, теорія ймовірності, прогнозування, проектний менеджмент, прийняття рішень в умовах невизначеності.**

**In the abstract it is showed the example of the implemented intelligence system which aim is to predict the estimated successfulness of the unprofitable projects, which are mostly realized by the nonprofit and non-governmental organizations. The usage of probability theory and the kernel smoothing method while proceeding the predictions operating with a small number of available data is scientifically grounded.**

**Keywords – decision making support systems, intelligence systems, probability theory, prediction, project management, decision making in uncertain terms.**

### Постановка проблеми управління неприбутковими проектами

Сьогодні громадянське суспільство України з кожним днем все міцніше "стає на ноги". Яскравими показниками цього є достатньо впевнений розвиток в Україні так званого "третього сектора" – поява та активізація все більшої кількості неурядових організацій, молодіжних рухів, благодійних фондів, а також все більше бажання населення займатися громадською діяльністю, організовуючи неприбуткові, соціальні події та акції, та отримуючи взамін щось більше ніж просто матеріальну винагороду. Кількість зареєстрованих громадських об'єднань постійно зростає.