УДК 519.854.2
DOI https://doi.org/10.24144/2616-7700.2021.38(1).123-136

Л. Ф. Гуляницький[1], А. В. Дубіна[2]

[1] Інститут кібернетики ім. В.М. Глушкова НАН України,
завідувач відділу,
доктор технічних наук, старший науковий співробітник
leonhul.icyb@gmail.com
ORCID: https://orcid.org/0000-0002-1379-4132

[2] НТУУ «Київський політехнічний інститут імені Ігоря Сікорського»,
магістрант
tasyana.ch@gmail.com
ORCID: https://orcid.org/0000-0001-8069-9664

РОЗВ'ЯЗУВАННЯ ЗАДАЧІ РОЗМІЩЕННЯ ПРЯМОКУТНИКІВ НА НАПІВСКІНЧЕННІЙ СТРІЧЦІ АЛГОРИТМАМИ ЛОКАЛЬНОГО ТА ТАБУЙОВАНОГО ПОШУКУ

В роботі розглянуто алгоритми стандартного локального та табуйованого пошуку для розв'язування задачі розміщення прямокутників на напівнескінченній стрічці. Особливостями задачі є наявність заборонених областей (дірок), які впливають на ефективність роботи алгоритмів. Досліджувана задача має значну теоретичну цінність і важливе прикладне значення. Ця задача належить до задач NP-повних і більшість методів розв'язування є наближеними.

Експериментально досліджено ефективність запропонованих алгоритмів для задачі розміщення прямокутників. Визначено рекордні значення цільової функції, дисперсію, довірчі інтервали та час роботи алгоритмів для задач з різними параметрами.

Ключові слова: оптимізація розміщення, локальний пошук, табуйований пошук, комбінаторна оптимізація.

1. Вступ. Використання локального пошуку (ЛП) в комбінаторній оптимізації почалося у 1956 році при розв'язуванні задачі комівояжера [1], в наступні роки сфера застосування була розширена, а його концепція покладена в основу багатьох алгоритмів розв'язування задач комбінаторної оптимізації [2-4].

Для NP-складних задач алгоритми локального пошуку дозволяють знаходити наближені розв'язки і найчастіше використовуються для розв'язування задач, коли точні методи безсилі через надто великий обсяг обчислювальних затрат.

2. Постановка задачі. Розглянемо задачу розміщення прямокутників із заданою шириною та різними довжинами на напівнескінченній стрічці. Стрічка розділена на декілька горизонтальних рівнів, які мають задану ширину та однакову довжину. На кожному рівні можуть бути заборонені області (дірки), на які не можна розміщати прямокутники. Необхідно розмістити прямокутники таким чином, щоб загальна довжина використаної частини стрічки була найменшою. Математична постановка задачі наведена у роботах [5-6].

3. Алгоритм локального пошуку. Для багатьох задач теорії розкладів, розміщення, покриття та інших оптимізаційних задач алгоритми локального пошуку дозволяють отримувати кращі результати у порівнянні з деякими іншими алгоритмами, особливо за умов обмеження у часі та великої розмірності задач.

Розглянемо загальну схему алгоритму [6,7]:

1. Генерація початкового припустимого розв'язку x, який обираємо як поточний варіант.

2. Чергова ітерація.

Формуємо окіл $L(x)$ поточного варіанта й точно чи наближено знаходимо елемент $y \in L(x)$, який є субоптимальним розв'язком у цьому околі. Якщо $y \neq x$, то знайдений елемент оголошується черговим поточним варіантом (здійснюється переприсвоєння $y \leftarrow x$) і починається чергова ітерація, інакше – повернення на п. 3.

3. Завершення роботи алгоритму: x – локальний розв'язок, якщо на останній ітерації здійснюється вичерпний пошук в околі.

Принциповими моментами реалізації конкретних алгоритмів локального пошуку є:

1) визначення околів $L(x)$ у просторі розв'язків задачі;

2) генерація чергової точки околу $y \in L(x)$;

3) критерій завершення перегляду точок у поточному околі та переходу до наступного;

4) спосіб обчислення величини зміни цільової функції при переході до нового поточного варіанта;

5) критерій завершення;

6) формування початкового наближення.

Ефективність алгоритмів локального пошуку істотно залежить від вибору відповідного типу околу $L(x)$. Чим більший окіл, тим імовірніше отримати кращий результат, але розширення околів швидко стає непрактичним з обчислювальної точки зору.

У п.1 використовуються метричні околи чи околи, які утворені алгоритмічно. Пошук в околі може бути здійснений шляхом повного перебору точок або ж завершуватися переходом до першого розв'язку, який поліпшує значення цільової функції.

Для організації перебору точок околу при переході до нової ітерації використовують лінійний генератор або кільцевий генератор [6].

Критерієм завершенням локального пошуку може бути одна із умов: відсутність поліпшуючої точки в поточному околі, вичерпання ліміту часу або заданої кількості ітерацій.

Розглянемо задачу розміщення прямокутників на напівнескінченній стрічці, що містить заборонені зони, і яка буде зведена до задачі на перестановках, для чого перенумеруємо прямокутники.

Наведемо псевдокод алгоритму.

```
1   procedure local_search
2    x^0:= деякий припустимий розв'язок;
3    h:=0;
4    while окіл L(x^h) поточного варіанта x^h не переглянутий
     повністю або не вичерпаний часовий ліміт do
5      y:= генерація чергової точки околу L(x^h);
6      Δ = f(x^h) − f(y);
7      if Δ > 0 then
8       h:=h+1;
9       x^h := y;
10     end if;
11   end while;
12   return x = x^h;
13  end
```

У рядку 2 припустимий розв'язок – початкова перестановка.

У рядку 4 окіл поточного варіанта визначається як множина перестановок, отриманих шляхом транспозиції елементів поточної перестановки.

Перегляд точок поточного околу завершується і здійснюється перехід до наступної ітерації, коли знаходиться перший же розв'язок, який покращує значення цільової функції, що і відображено у рядках 7-10.

Значенням цільової функції для даного розв'язку (перестановки) є правий край найвіддаленішого від початку прямокутника у розміщенні, яке побудоване спеціальною процедурою (placement), про яку піде мова далі.

В якості критерія завершення роботи алгоритму використано такі умови:

1) відсутність поліпшуючої точки в околі поточного розв'язку;

2) вичерпання заданого ліміту часу.

Генерування точок околу поточного розв'язку здійснюється, як зазначалося, шляхом транспозицій його компонентів. При переході до нової ітерації процес генерування продовжується, починаючи з тієї транспозиції, на якій завершився процес у попередньому околі, а при досягненні останньої пари компонент транспозиції починаються із компонент 1 і 2, якщо перегляд точок в околі не завершено.

Нехай на ітерації h маємо перестановку $x^h = (x_1, ..., x_n)$, а $y \in L(x^h)$ утворюється із x^h шляхом транспозиції компонент i та j, $i \neq j$. Тоді покладаємо $x^{h+1} = y$ і генерування точок із околу $L(x^{h+1})$ починаємо із транспозиції i та j+1, якщо $j < n$, або ж присвоюємо $i = 1$, $j = 2$ в іншому разі. При цьому слід підраховувати кількість згенерованих точок околу, щоб вона не перевищила ${C_n}^2$ – кількості всіх точок околу при такому його означенні, що і використовується в одній із умов завершення роботи алгоритму.

Псевдокод алгоритму побудови розміщення прямокутників (placement) по заданій перестановці, що реалізує принцип "розміщувати вліво-вниз", подано далі. Його використання, як і оберненої процедури, яка по розміщенню будує перестановку, дозволяє встановити взаємно однозначну відповідність між перестановками та підмножиною розміщень, яка обов'язково містить глобальний розв'язок задачі.

```
1   procedure placement
2    x:= перестановка, для якої необхідно побудувати розміщення;
3    i :=1;
4    while i ≤ n do {доки всі прямокутники у перестановці x не
     розглянуто}
5     обрати прямокутник x_V з перестановки x;
6     s_j :=відстань до правого краю прямокутника x_V на рівні j, якщо
     його розмістити найближче до останнього прямокутника чи дірки,
     дотримуючись умови неперетинання;
7     l := найменший номер рівня, де s_l мінімальне;
8     на рівень l помістити прямокутник x_V;
9     i :=i +1;
10    end while;
11    return розміщення прямокутників;
12  end
```

У рядках 6 та 8 прямокутник розміщається впритул до лівого краю останнього прямокутника на обраному рівні (чи дірки, якщо лівіше неї розмістити прямокутник не можна).

4. Алгоритм табуйованого пошуку. Для того, щоб запобігти повторному перегляду неефективних розв'язків, доцільно використовувати алгоритми табуйованого пошуку (ТП).

Головна ідея табуйованого пошуку полягає в зміні поточного варіанта розв'язку задачі та запам'ятовуванні послідовності таких змін та їх використанням [4,7]. Всі зроблені модифікації розв'язків фіксуються в списку заборон.

Пошук починається з початкового варіанта розв'язку, який поступово поліпшується локальними модифікаціями. Розв'язок, який був відвіданий нещодавно, включається в список заборон і далі не розглядається як кандидат на відвідування.

Для поданої задачі розглянемо метод строгого табуювання, який запам'ятовує всі розв'язки з метою заборони потрапляння в уже відвідані точки простору розв'язків.

Псевдокод алгоритму можна подати так.

```
1   procedure tabu_search
2    x^0:= деякий припустимий розв'язок;
3    T:= ∅;
4    h:=0;
5    while окіл L(x^h) поточного варіанта x^h не переглянутий
     повністю або не вичерпаний часовий ліміт do
6     y:= генерація чергової точки околу L(x^h), що не належить списку заборон T;
7     Δ = f(x^h) − f(y);
8     if Δ > 0 then
9      h:=h+1;
```

```
10      x^h := y;
11      T:=T ∪x^h;
12     end if;
13    end while;
14    return x = x^h;
15   end
```

У рядку 2, як і в алгоритмі локального пошуку, припустимий розв'язок – початкова перестановка.

У рядку 3 вказується множина розв'язків, що потрапили у список заборон T і не розглядатимуться повторно.

У рядку 6 відбувається генерація нової точки околу, що відсутня у списку заборон.

Якщо нова точка околу покращує значення цільової функції, то вона зберігається у списку заборон у рядку 11.

Процедури обчислення значення цільової функції, генерування нової точки околу та критерій завершення роботи алгоритму використовуються ті самі, що і наведені для алгоритму локального пошуку.

5. Експериментальне дослідження ефективності алгоритмів. Експериментально перевіримо ефективність використання алгоритмів локального та табу пошуку для виділених задач розміщення прямокутників на напівнескінченній стрічці з діркам. Метою досліджень є оцінка ефективності алгоритмів, їх порівняння, виявлення їх недоліків і переваг.

Алгоритми реалізовано в рамках інформаційної системи PlaceIT (платформа .NET, мова C#). Експерименти виконувались на персональному комп'ютері з наступними характеристиками: процесор Intel Core i7-8565U, 1.8 GHz з оперативною пам'яттю 8 GB та операційною системою Microsoft Windows 10 Home (версія 10.0.18363).

Для цього сформуємо задачі розміщення прямокутників, кількість яких вибиралася із діапазону 10-100, на напівнескінченних стрічках з кількістю рівнів від 5 до 20. Для кожного типу задачі згенеруємо 1000 індивідуальних задач, для яких застосуємо пропоновані алгоритми локального та табуйованого пошуку. Згенеровані довжини прямокутників не перевищують 15, кількість дірок на кожному рівні не перевищує 3, а довжина кожної дірки є 1. Для генерації довжин прямокутників та положення дірок використовувався датчик випадкових чисел з рівномірним розподілом. Для дослідження алгоритмів використовуються однакові початкові наближення.

Результатами експерименту є час роботи алгоритмів в секундах, відсоток середнього покращення значення цільової функції (наведені у таблиці 1). На початку та після завершення роботи кожного алгоритму для кожної задачі визначається *найдовша* відстань від початку стрічки на кожному з рівнів до правого краю прямокутників (довжина найдовшого занятого рівня). Середнє покращення вказує на те, на скільки відсотків зменшується довжина найдовшого рівня у порівнянні із початковим варіантом розміщення.

Для заданих алгоритмів не проводилися експерименти для тих задач, де кількість прямокутників не перевищувала кількості рівнів. Це зумовлено тим, що на кожному рівні в такому разі буде розміщений лише 1 прямокутник і

отже значення цільової функції є довжиною найдовшого прямокутника. Для поданих результатів подано графіки залежності часу роботи алгоритмів від параметрів задач та відсотку середнього покращення значення цільової функції від параметрів задач для кожного з алгоритмів.

На рисунках 1-2 подані результати експериментів для задач з 5, 10, 15 та 20 рівнями та кількістю прямокутників від 5 до 100.

Таблиця 1. Результати експерименту, частина 1

Параметри задачі		**Час (с)**		**Середнє покращення (%)**	
Рівні	**Прямокутники**	**ЛП**	**ТП**	**ЛП**	**ТП**
5	10	7,8	7,0	6,5	21,3
5	15	12,7	16,1	0,4	17,8
5	20	6,2	10,0	1,1	16,3
5	25	15,2	4,2	1,0	4,1
5	30	21,0	16,8	4,4	8,7
5	35	34,4	9,8	3,0	11,0
5	40	55,5	5,0	2,7	12,3
5	45	62,3	13,2	3,4	5,5
5	50	99,9	5,7	1,3	3,8
5	55	144,5	3,5	2,5	5,0
5	60	150,0	4,8	1,3	1,9
5	65	190,4	4,9	3,6	5,1
5	70	236,5	4,4	0,9	1,2
5	75	313,9	2,7	3,1	5,0
5	80	356,2	4,7	0,4	2,7
5	85	479,4	3,8	0,3	1,0
5	90	572,7	2,1	0,9	2,9
5	95	786,	1,4	0,7	2,6
5	100	865,1	0,7	0,1	0,4
10	10	0	0	0	0
10	15	8,4	0,1	0,7	16,9
10	20	12,9	15,8	1,0	0,1
10	25	24,4	8,3	2,4	12,1
10	30	34,8	16,6	8,5	13,0
10	35	68,0	14,9	4,1	5,7
10	40	100,1	8,8	3,0	9,1
10	45	155,5	1,4	0,4	4,0
10	50	189,5	16,7	1,7	11,5
10	55	243,9	7,8	0,3	0,9
10	60	335,5	12,3	1,5	3,4
10	65	456,5	5,5	1,4	0,2
10	70	540,8	3,0	3,5	7,4
10	75	601,1	7,0	3,7	7,6

10	80	715,1	9,4	4,6	6,1
10	85	786,6	5,8	3,0	6,7
10	90	975,3	7,4	5,8	7,8
10	95	1091,5	3,9	3,5	9,8
10	100	1165,1	5,7	1,7	5,1
15	10	0	0	0	0
15	15	0	0	0	0
15	20	16,9	0,8	1,0	0,7
15	25	36,6	18,3	2,6	8,1
15	30	72,0	10,7	4,0	10,8
15	35	109,6	26,3	6,4	11,4
15	40	155,9	6,0	8,7	12,6
15	45	237,7	16,0	4,5	10,7
15	50	299,7	21,0	10,8	14,7
15	55	357,6	5,2	2,7	4,0
15	60	524,5	13,8	7,5	8,5
15	65	651,2	8,4	7,1	15,5
15	70	848,6	15,1	3,7	4,9
15	75	1007,1	7,6	4,9	6,0
15	80	1252,6	7,2	4,7	6,1
15	85	1521,1	6,7	5,6	6,3
15	90	1641,4	9,3	7,2	8,6
15	95	1968,0	2,8	7,9	8,5
15	100	2259,2	10,8	0,6	2,0
20	10	0	0	0	0
20	15	0	0	0	0
20	20	0	0	0	0
20	25	50,1	11,6	1,1	2,6
20	30	85,6	10,4	2,0	3,3
20	35	99,9	8,1	2,2	3,4
20	40	185,7	15,8	4,8	5,4
20	45	251,9	8,5	5,9	8,0
20	50	352,1	12,4	6,3	8,4
20	55	456,1	7,0	11,3	11,2
20	60	600,2	11,6	16,9	19,4
20	65	837,5	23,5	10,1	23,7
20	70	1058,0	10,4	10,7	17,1
20	75	1322,3	9,4	13,3	13,8
20	80	1641,8	12,6	3,4	4,8
20	85	1764,0	13,2	3,4	8,1
20	90	2162,8	8,5	13,0	13,5
20	95	2550,5	11,4	5,7	11,6
20	100	2816,9	14,3	13,0	13,1

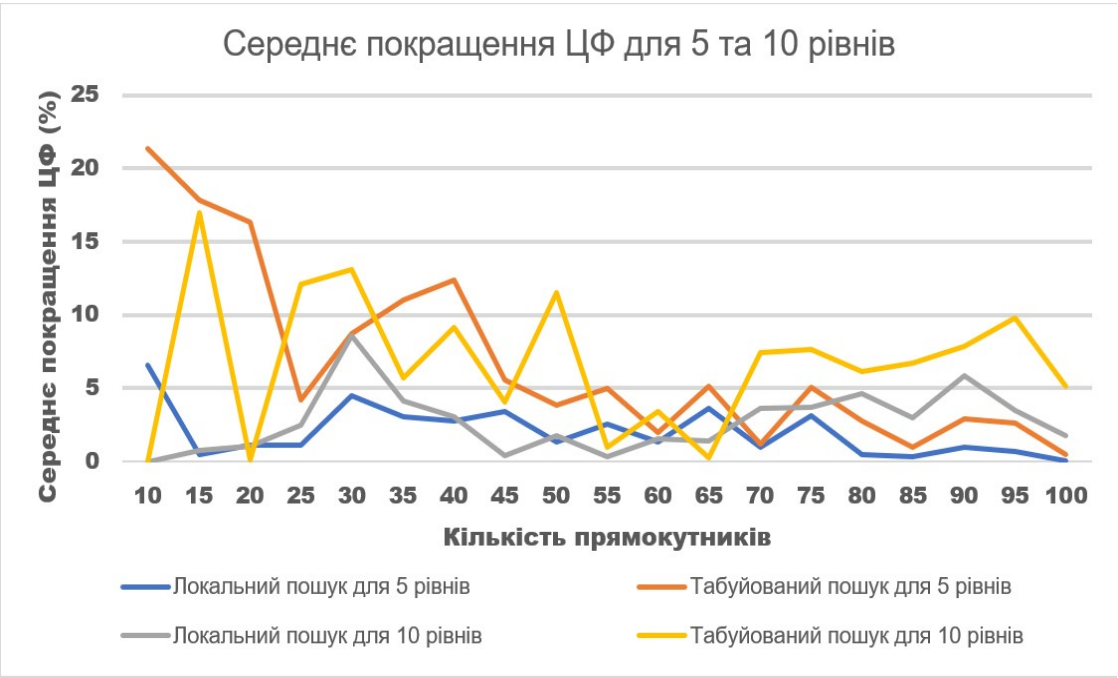

Рис. 1. Середнє покращення значення цільової функції (ЦФ) для 5 та 10 рівнів

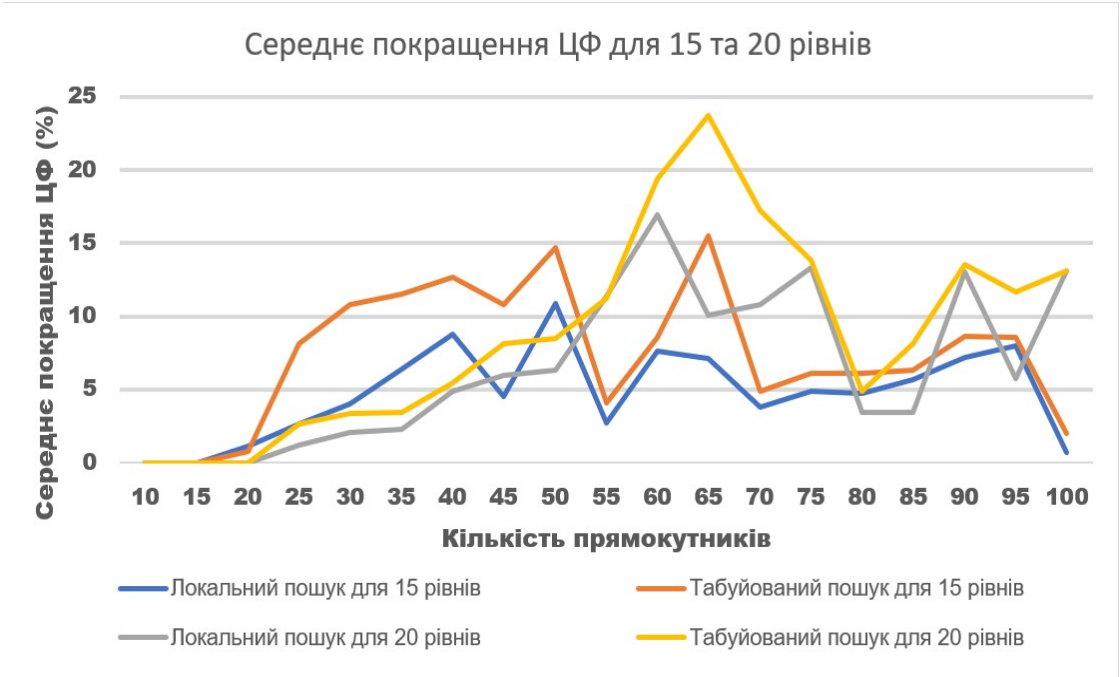

Рис. 2. Середнє покращення значення цільової для 15 та 20 рівнів

Отже, алгоритм локального пошуку програє алгоритму табуйованого пошуку у відсотку середнього покращення значення цільової функції для задач з кількістю прямокутників менші за 60. Це зумовлено тим, що для задач невеликої розмірності алгоритм табуйованого пошуку не розглядає околи із табу списку, тим самим збільшуючи кількість різних околів, які будуть переглянуті.

Для задач з кількістю прямокутників більше 60 та кількістю рівнів 5 – відсоток середнього покращення значення цільової функції не перевищує 5% кожним алгоритмом, а для задач із кількістю рівні 10 – не перевищує 10%. Це зумовлено тим, що генерується велика кількість прямокутників з невеликою різницею довжин сторін, тому кількість околів, які можна переглянути, перевищує задані часові ліміти.

Найкращі результати у середньому показав алгоритм табуйованого пошуку для задачі, яка складається з 20 рівнів та кількістю прямокутників 55-70, проте в інших випадках переваги у середньому покращенні значення цільової функції алгоритм табуйованого пошуку не очевидні.

Для задачі із 15 рівнями алгоритми табуйованого пошуку має незначне середнє покращення у порівнянні з алгоритмом локального пошуку для кількості прямокутників, що перевищує 50, окрім задачі для 65 прямокутників.

На рисунках 3 та 4 наведемо графіки залежності часу роботи алгоритмів від параметрів задачі (кількості рівнів стрічки та прямокутників).

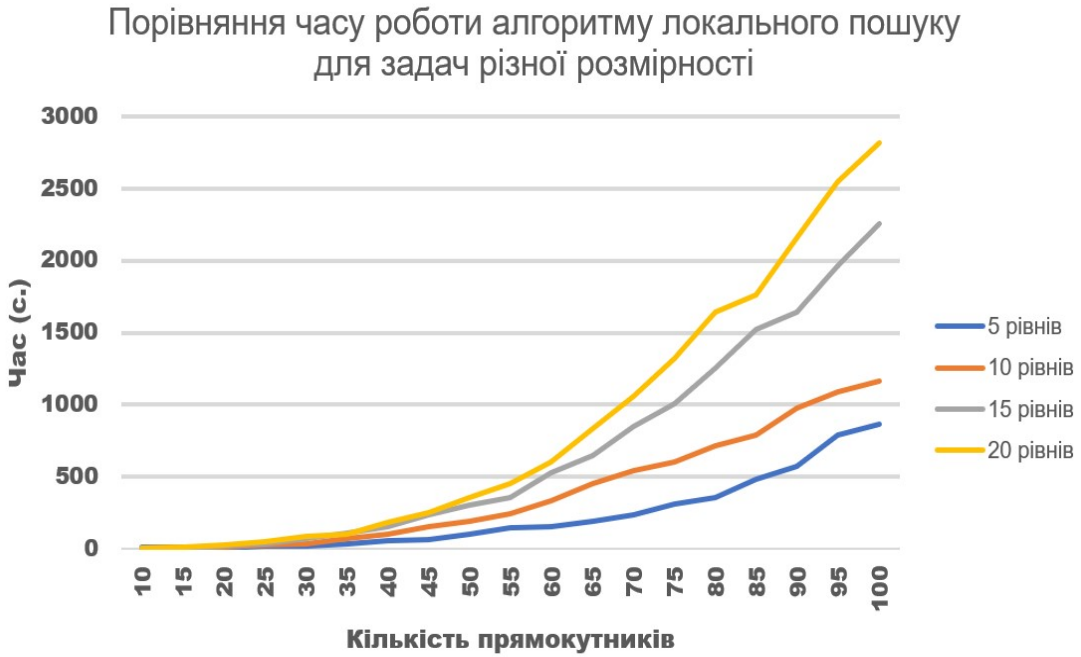

Рис. 3. Час роботи алгоритму локального пошуку

Час роботи алгоритму локального пошуку значно більший за час роботи алгоритму табуйованого пошуку для задач з кількістю прямокутників більше за 30. Це зумовлено тим, що алгоритм табуйованого пошуку повторно не розглядає околи і тому, отримуючи нові околи без покращення – зупиняє свою роботу, в той час, коли алгоритм локального пошуку продовжує розглядати інші околи, доки не закінчиться заданий ліміт ітерацій для околу.

Із рисунку 4 випливає, що для час роботи алгоритму табуйованого пошуку буде найменшим для будь-якої кількості прямокутників у випадку, коли стрічка має 5 рівнів. Максимальний час роботи алгоритму табуйованого пошуку більший для задач, де стрічка має 15 та 20 рівнів, проте не перевищує час роботи алгоритму локального пошуку.

Рис. 4. Час роботи алгоритму табуйованого пошуку

У таблиці 2 подані інші результати експерименту: рекордне (найкраще знайдене) значення цільової функції, середньоквадратичне відхилення (СКВ), дисперсія та довірчі інтервали.

Таблиця 2. Результати експерименту, частина 2

Задача		**Локальний пошук**					**Табуйований пошук**				
Рівні	**Прямокутники**	**Рекорд**	**СКВ**	**Дисперсія**	**Довірчі інтервали**		**Рекорд**	**СКВ**	**Дисперсія**	**Довірчі інтервали**	
5	10	10	2.9	8.5	13.0	18.9	9	3.3	11.0	11.6	18.3
5	15	18	2.5	6.7	19.4	24.5	19	2.5	6.5	19.4	24.5
5	20	24	3.0	9.1	24.9	31.0	22	4.8	23.3	25.1	34.8
5	25	30	3.8	14.7	34.1	41.8	29	3.8	15.1	33.1	40.8
5	30	36	5.7	32.7	38.2	49.7	34	5.6	32.4	37.3	48.6
5	35	47	5.3	28.1	47.6	58.3	48	3.8	15.0	47.1	54.8
5	40	54	5.6	31.9	56.3	67.6	51	3.8	14.7	55.1	62.8
5	45	60	4.8	23.4	65.1	74.8	57	5.5	31.0	60.4	71.5
5	50	65	5.5	30.8	70.4	81.5	67	4.0	16.6	69.9	78.0
5	55	73	4.8	23.8	74.1	83.8	73	5.0	25.5	76.9	87.
5	60	78	7.9	63.0	84.0	99.9	80	6.2	39.3	86.7	99.2
5	65	86	8.5	72.3	86.4	103.5	77	7.6	58.9	88.3	10.6
5	70	94	6.3	40.8	101.6	114.3	82	9.1	83.4	94.8	113.1
5	75	102	6.6	44.2	104.3	117.6	95	8.1	66.5	105.8	122.1
5	80	112	5.7	32.7	113.2	124.7	102	7.1	51.4	108.8	123.1

5	85	120	5.0	25.3	121.9	132.0	113	8.3	70.0	113.6	130.3
5	90	116	7.5	56.4	125.4	140.5	117	6.8	47.5	127.1	140.8
5	95	125	9.1	83.2	126.8	145.1	130	7.1	50.9	133.8	148.1
5	100	136	10.9	120.8	139.0	160.9	132	9.4	88.5	136.5	155.4
10	10	0	0	0	0	0	0	0	0	0	0
10	15	12	1.1	1.4	11.8	14.1	10	1.8	3.3	11.1	14.8
10	20	13	2.3	5.7	14.6	19.3	12	2.9	8.8	13.0	18.9
10	25	18	2.4	5.9	18.5	23.4	17	1.8	3.4	18.1	21.8
10	30	20	2.1	4.6	21.8	26.1	20	2.5	6.4	19.4	24.5
10	35	24	2.9	8.7	25.0	30.9	23	2.4	6	24.5	29.4
10	40	25	2.9	8.6	27.0	32.9	28	2.6	7.2	28.3	33.6
10	45	27	2.8	8.1	31.1	36.8	31	2.4	5.8	32.5	37.4
10	50	32	2.9	8.5	36.0	41.9	32	3.3	11.5	35.6	42.3
10	55	39	2.5	6.2	39.4	44.5	33	3.7	14.0	39.2	46.7
10	60	44	1.4	2	45.5	48.4	39	3.0	9.2	43.9	50.0
10	65	41	3.5	12.2	46.4	53.5	41	2.7	7.3	45.2	50.7
10	70	50	3.0	9.5	51.9	58.0	44	3.4	12.0	49.5	56.4
10	75	54	3.2	10.8	53.7	60.2	51	3.6	13.4	53.3	60.6
10	80	53	4.7	22.2	53.2	62.7	54	4.6	21.4	54.3	63.6
10	85	61	2.6	7.0	62.3	67.6	57	3.3	10.9	57.6	64.3
10	90	62	3.3	10.9	63.6	70.3	63	3.8	14.9	65.1	72.8
10	95	63	4.4	20	67.5	76.4	65	3.9	15.6	68.0	75.9
10	100	72	2.7	7.5	71.2	76.7	69	3.2	10.5	72.7	79.2
15	10	0	0	0	0	0	0	0	0	0	0
15	15	0	0	0	0	0	0	0	0	0	0
15	20	12	0.2	0.1	12.7	13.2	11	0.7	0.5	12.2	13.7
15	25	13	1.9	3.9	13.0	16.9	13	1.9	3.7	13.0	19.9
15	30	15	1.7	3.0	16.2	19.7	15	1.1	1.4	15.8	18.1
15	35	18	1.4	2.1	18.5	21.4	16	2.3	5.4	18.6	23.3
15	40	20	1.6	2.5	21.3	24.6	20	2.0	4.3	21.9	26.0
15	45	22	1.9	3.8	24.0	27.9	20	1.9	3.8	23.0	26.9
15	50	25	1.7	2.9	26.2	29.7	24	2.1	4.5	25.8	30.1
15	55	27	2.0	4.2	28.9	33.0	26	1.7	2.8	27.2	30.7
15	60	29	1.8	3.2	30.1	33.8	29	1.6	2.8	30.3	33.6
15	65	31	2.2	4.9	32.7	37.2	31	2.1	4.7	32.8	37.1
15	70	33	1.7	3.1	35.2	38.7	32	3.1	9.9	33.8	40.1
15	75	35	2.3	5.6	35.6	40.3	35	2.6	7.2	37.3	42.6
15	80	36	3.0	9.4	39.9	46.0	39	2.8	8.2	38.1	43.8
15	85	41	2.7	7.7	42.2	47.7	37	2.6	7.2	40.3	45.6
15	90	42	3.2	10.4	43.7	50.2	45	2.0	4.0	44.9	49.0
15	95	45	3.1	9.7	45.8	52.1	46	2.6	6.8	48.3	53.6
15	100	50	2.1	4.4	50.9	55.1	48	3.5	12.9	48.4	55.5
20	10	0	0	0	0	0	0	0	0	0	0
20	15	0	0	0	0	0	0	0	0	0	0
20	20	0	0	0	0	0	0	0	0	0	0
20	25	11	0.5	0.3	12.4	13.5	12	0.8	0.6	12.1	13.8

20	30	11	1.4	2.1	11.5	14.4	12	1.4	2.0	11.5	14.4
20	35	13	2.3	5.3	13.6	18.3	13	1.8	3.5	14.1	17.8
20	40	17	1.5	2.2	17.4	20.5	16	0.9	0.9	17.0	18.9
20	45	18	1.4	2.2	19.5	22.4	16	1.8	3.3	17.1	20.8
20	50	18	2.5	6.3	19.4	24.5	18	2.0	4.3	18.9	23.0
20	55	20	1.3	1.8	21.6	24.3	21	1.7	2.9	22.2	25.7
20	60	23	1.0	1.0	23.9	26.0	22	2.0	4.2	22.9	27.0
20	65	24	2.2	5.2	24.7	29.2	23	1.7	3.1	25.2	28.7
20	70	25	2.4	5.8	26.5	31.4	26	1.1	1.4	26.8	29.1
20	75	28	1.4	2.2	29.5	32.4	28	2.6	6.8	28.3	33.6
20	80	30	1.3	1.7	31.6	34.3	29	2.1	4.8	30.8	35.1
20	85	31	1.4	1.9	32.5	35.4	32	1.7	3.0	32.2	35.7
20	90	34	3.0	9.5	33.9	40.0	33	1.4	2	33.5	36.4
20	95	33	2.0	4.2	36.9	41.0	36	1.7	2.8	36.2	39.7
20	100	36	2.1	4.5	38.8	43.1	38	1.8	3.5	38.1	41.8

На рисунку 5 подано графік залежності рекордного значення цільової функції від параметрів задачі для кожного алгоритму.

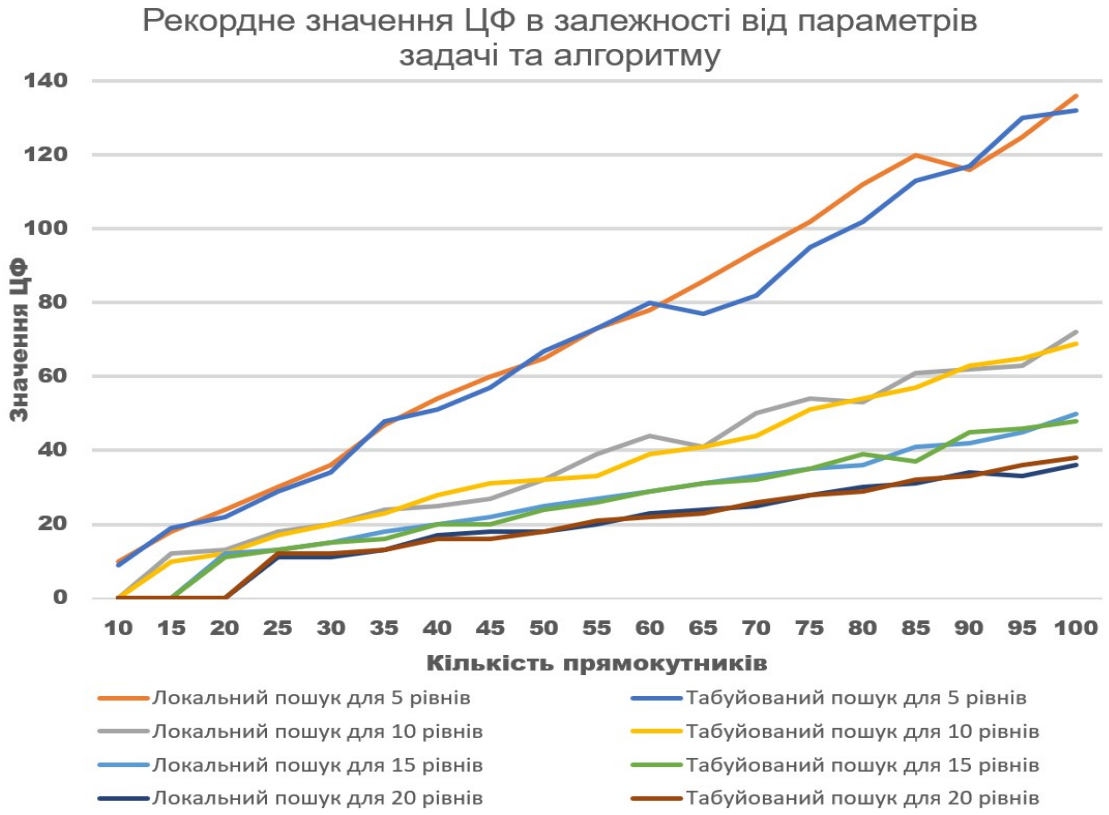

Рис. 5. Рекордне значення цільової функції

Рекордні значення цільової функції для задач з кількістю рівнів 15 та 20 майже однакові, проте, якщо кількість прямокутників більша за 90 – кращі рекордні значення отримано алгоритмом табуйованого пошуку.

Отже, для розглянутої задачі розміщення прямокутників на напівнескінченній стрічці алгоритм табуйованого пошуку показав кращі результати, ніж

алгоритм локального пошуку.

До переваг алгоритму локального пошуку можна віднести такі: можливість перегляду всіх точок в околі, для деяких задач – найкраще значення цільової функції. Недоліками алгоритму є час роботи алгоритму та гірше значення середнього покращення значення цільової функції.

До переваг алгоритму табуйованого пошуку можна віднести час роботи алгоритму, середнє покращення значення цільової функції. Недоліком алгоритму табуйованого пошуку є менше знайдене рекордне значення цільової функції.

Час роботи алгоритму локального пошуку є суттєвим недоліком, оскільки у реальних задачах часто є обмеження у часових ресурсах, тому за вказаний час можна не отримати найкращого розв'язку. Для таких випадків доцільно використовувати алгоритм табуйованого пошуку, оскільки середнє покращення вище, ніж у алгоритмі локального пошуку.

6. Висновки. В роботі запропоновано алгоритми локального та табуйованого пошуку для розв'язування задачі задач розміщення прямокутників на напівнескінченній стрічці з забороненими областями. Досліджено експериментально ефективність алгоритмів та зазначено переваги та недоліки кожного із них.

Перспективи подальших досліджень полягають у модифікації алгоритмів локального та табуйованого пошуку для підвищення їх ефективності, а також використання популяційних метаевристик [4], зокрема, із включенням в них цих алгоритмів.

Список використаної літератури

1. Fuentes G.E.A., Gress E.S.H., Mora J.C.S.T., Marin J.M. Solution to travelling salesman problem by clusters and a modified multi-restart iterated local search metaheuristic. *PLoS ONE.* 2018. Vol. 13(8). e0201868. DOI: `https://doi.org/10.1371/journal.pone.0201868`
2. Hoos H.H., Stützle T. Stochastic local search: Foundations and applications. Elsevier, 2004.
3. Song T., Liu S., Tang X., Peng X., Chen M. An iterated local search algorithm for the University Course Timetabling Problem. *Applied Soft Computing.* 2018. Vol. 68. P. 597-608. DOI: `https://doi.org/10.1016/j.asoc.2018.04.034`
4. Handbook of metaheuristics. Cham: Springer, 2019.
5. Сергиенко И.В., Гуляницкий Л.Ф., Малышко С.А. О решении задач размещения одного класса. *Экономика и математические методы.* 1989. XXV, № 3. С. 560-564.
6. Дубіна А., Гуляницький Л. Алгоритми локального пошуку та табу-пошуку для задач розміщення прямокутників. Наукове забезпечення технологічного прогресу XXI сторіччя: матеріали міжнародної наукової конференції (Т.2). 1 травня, 2020 рік. Чернівці, Україна: МЦНД. С. 48-53.
7. Гуляницький Л. Ф., Мулеса О. Ю. Прикладні методи комбінаторної оптимізації: навч. посібник. К.: Видавничо-поліграфічний центр "Київський університет". 2016.

Hulianytskyi L. F., Dubina A. V. Solve problems of placing rectangles on a semi-infinite tape of local and taboo search algorithms.

The paper considers the algorithms of standard local and taboo search for solving the problem of placing rectangles on a semi-infinite tape. The peculiarities of the problem are the presence of forbidden areas (holes) that affect the efficiency of algorithms. The researched problem has significant theoretical value and important applied value. This problem belongs to the problems of NP-complete and most of the methods of solution are approximate.

The efficiency of the proposed algorithms for the problem of placing rectangles is experimentally investigated. Record values of the objective function, variance, confidence intervals

and algorithm runtime for problems with different parameters are determined.

Keywords: placement optimization, local search, tabu search, combinatorial optimization.

Список використаної літератури

1. Fuentes, G.E.A., Gress, E.S.H., Mora, J.C.S.T., & Marin, J.M. (2018). Solution to travelling salesman problem by clusters and a modified multi-restart iterated local search metaheuristic. PLoS ONE 13(8), e0201868. https://doi.org/10.1371/journal.pone.0201868.
2. Hoos, H.H., & Stützle, T. (2004). Stochastic local search: Foundations and applications. Elsevier.
3. Song, T., Liu, S., Tang, X., Peng, X., & Chen, M. (2018). An iterated local search algorithm for the University Course Timetabling Problem. Applied Soft Computing, 68, 597-608. https://doi.org/10.1016/j.asoc.2018.04.034.
4. Handbook of metaheuristics. (2019). Cham: Springer.
5. Sergienko, I.V., Hulianitskyi, L.F., & Malyshko, S.A. (1989). About the decision of problems of placement of one class. Economics and Mathematical Methods, XXV, 3, 560-564 [in Russian].
6. Hulianitskyi, L., & Dubina, A. (2020). Local search algorithms and taboo search for the task of placing rectangles. Scientific support of technological progress of the XXI century: materials of the international scientific conference (Vol. 2), May 1, 2020. Chernivtsi, Ukraine, 48-53 [in Ukrainian].
7. Hulianitskyi, L.F., & Mulesa, O. Y. (2016). Applied methods of combinatorial optimization. Publishing and Printing Center "Kyiv University"[in Ukrainian].

Одержано 06.04.2021