УДК 519.854

С. В. Чупов(ДВНЗ «Ужгородський нац. ун-т»)

БІНАРНІ АЛГОРИТМИ ПОШУКУ ЛЕКСИКОГРАФІЧНИХ ЕКСТРЕМУМІВ МНОЖИН У ЗАДАЧАХ ПРО ПОКРИТТЯ ТА УПАКОВКУ СКІНЧЕННОЇ МНОЖИНИ.

One of the possibilities to find the solutions to the problems of covering and packing of the set are the algorithms of lexicographical search. A key component of these algorithms is the search for the lexicographical extrema of sets that are subsets of feasible solutions set of the problem. The coefficients of the constraint matrix in the problems of covering and packing are boolean. That is why the columns of the constraint matrix are boolean vectors which can be considered as the whole numbers written in binary notation. The paper describes and justifies the search algorithms of the lexicographical extrema set for the above mentioned problems in which, in contrast to standard lexicographic search algorithms, all actions are binary operations "and" and "or" over the columns of the constraint matrix of the corresponding problem. The analysis of the effectiveness of the proposed algorithm demonstrates a significant advantage compared with the standard algorithms of lexicographical search.

Однією з можливостей відшукання розв'язків задач про покриття та упаковку множини є алгоритми лексикографічного пошуку. Ключовою складовою у таких алгоритмах є пошук лексикографічних екстремумів множин, які є підмножинами множини допустимих розв'язків задачі. Коефіцієнти матриці обмежень у задачах про покриття та упаковку – булеві. Тому стовпці матриці обмежень задачі це булеві вектори, які можна розглядати як цілі числа, записані у двійковій системі числення. У роботі описуються та обґрунтовуються алгоритми пошуку лексикографічних екстремумів множин для вищевказаних задач у яких, на відміну від стандартних алгоритмів лексикографічного пошуку, усі дії – порозрядні операції "and" і "or" над стовпцями матриці обмежень відповідної задачі. Аналіз ефективності роботи запропонованих алгоритмів свідчить про значну перевагу у порівнянні з стандартними алгоритмами лексикографічного пошуку.

1. Вступ. До задач про зважене покриття та упаковку скінченної множини зводиться багато прикладних задач. На сьогоднішній день розмірність таких задач є дуже великою. Враховуючи NP - складність цих задач, які відносяться до задач булевого лінійного програмування, на перший план виходять питання розробки нових алгоритмів пошуку їх розв'язків, або підвищення ефективності роботи вже існуючих методів. Однією з можливостей відшукання розв'язків задач про покриття та упаковку є алгоритми лексикографічного пошуку [1]. Ключовою складовою у таких алгоритмах є пошук лексикографічних екстремумів множин, що є підмножинами множини допустимих розв'язків задачі [2,3]. Коефіцієнти матриці обмежень та вектору вільних членів у задачах про покриття та упаковку є булевими. Тому стовпці матриці обмежень задачі це булеві вектори, які можна розглядати як цілі числа, записані у двійковій системі числення. У роботі описуються та обґрунтовуються алгоритми пошуку лексикографічних екстремумів множин для вищевказаних задач, у яких, на відміну від стандартних алгоритмів лексикографічного пошуку, усі дії – порозрядні (бінарні) операції "and" та "or" над стовпцями матриці обмежень відповідної задачі. Аналіз ефективності роботи запропонованих алгоритмів свідчить про значну перевагу над стандартними алгоритмами лексикографічного пошуку.

2. Постановка задачі. Нехай $V = \{v_1, \ldots, v_m\}$ — скінченна множина елементів довільної природи та $V^j \subset V$, $j = 1, \ldots, n$ — сімейство підмножин

множини V. Кожну множину V^j, $j = 1, \ldots, n$ можна однозначно представляти за допомогою булевого вектору $A^j \in B^m$, де $a_{ij} = \begin{cases} 1, v_i \in V^j, \\ 0, v_i \notin V^j. \end{cases}$

У такому представленні математична модель класичної задачі про зважене покриття (упаковку) скінченної множини формулюється наступним чином: мінімізувати(максимізувати)

$$x_0 = \sum_{j=1}^{n} w_j x_j, \tag{1}$$

за умов

$$\sum_{j=1}^{n} a_{ij} x_j \geq (\leq) 1, i = 1, \ldots, m, \tag{2}$$

$$x_j \in \{0, 1\}, j = 1, \ldots, n, \tag{3}$$

де $a_{ij} \in \{0, 1\}$, $w_j > 0$, $i = 1, \ldots, m$, $j = 1, \ldots, n$.

Множину допустимих розв'язків задачі про покриття, яка визначається умовами (2), (3), позначатимемо через X^{Cover}:

$$X^{Cover} = \left\{ x \in B^n | \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m \right\},$$

множину допустимих розв'язків задачі про упаковку — X^{Pack}:

$$X^{Pack} = \left\{ x \in B^n | \sum_{j=1}^{n} a_{ij} x_j \leq 1, i = 1, \ldots, m \right\}.$$

У роботі розглядаються питання підвищення ефективності пошуку лексикографічного мінімуму множини $\left\{ x \in X^{Cover} \middle| x \geq^L \bar{x} \right\}$ для задачі про покриття та пошуку лексикографічного максимуму множини $\left\{ x \in X^{Pack} \middle| x \leq^L \bar{x} \right\}$ у задачі про упаковку, де $\bar{x} \in B^n$ – фіксований вектор.

3. Властивості допустимих розв'язків. Систему нерівностей (2) для задачі про покриття представимо у векторній формі: $\sum\limits_{j=1}^{n} A^j x_j \geq \bar{1}$, де $A^j \in B^m$, $j = 1, \ldots, n$, — булеві вектори розмірності m, $\bar{1} \in B^m$ — вектор, усі координати якого рівні 1. У такій формі булевий вектор A^j, $j = 1, \ldots, n$ можна розглядати як ціле число без знаку, що записане у двійковій системі числення. Для довільного булевого вектору x через S_1^x позначатимемо множину індексів цього розв'язку для яких $x_j = 1$, $S_1^x = \{ j \in \{1, \ldots, n\} | \, x_j = 1 \}$.

Теорема 1. *У задачі про покриття розв'язок $x \in B^n$ допустимий тоді і тільки тоді коли* $\bigvee\limits_{j \in S_1^x} A^j = \bar{1}$.

Доведення. Знаком "$\vee$" позначено порозрядну операцію "or". Нехай x — допустимий розв'язок. Це означає, що $\sum\limits_{j=1}^{n} A^j x_j = \sum\limits_{j \in S_1^x} A^j \geq \bar{1}$. Для довільних двох булевих значень b_1 та b_2 таких, що $b_1 + b_2 \geq 1$ випливає $b_1 \vee b_2 = 1$. Це факт

залишається вірним і при довільній кількості булевих значень. Отже, для кожного $i = 1, \ldots, m$, з того, що $\sum\limits_{j \in S_1^x} a_{ij} \geq 1$ слідує $\bigvee\limits_{j \in S_1^x} a_{ij} = 1$. З іншого боку, нехай $\bigvee\limits_{j \in S_1^x} A^j = \bar{1}$. Тоді для кожного $i = 1, \ldots, m$ маємо $\bigvee\limits_{j \in S_1^x} a_{ij} = 1$. Виходячи з властивостей порозрядної операції "or", остання рівність має місце тільки тоді коли $\sum\limits_{j \in S_1^x} a_{ij} \geq 1$. Тому має виконуватись $\sum\limits_{j \in S_1^x} A^j = \sum\limits_{j=1}^{n} A^j x_j \geq \bar{1}$, або $x \in X^{Cover}$.

Систему нерівностей (2) для задачі про упаковку також можна представити у вигляді: $\sum\limits_{j=1}^{n} A^j x_j \leq \bar{1}$. Крім того, через n_x позначимо потужність множини S_1^x, $n_x = |S_1^x|$.

Теорема 2. *У задачі про упаковку розв'язок $x \in B^n$ допустимий тоді і тільки тоді коли для усіх $k = 2, \ldots, n_x$ виконується* $\left(\bigvee\limits_{j=1}^{k-1} A^{S_{1,j}^x}\right) \wedge A^{S_{1,k}^x} = \bar{0}$.

Доведення. Знаком "$\wedge$" позначено порозрядну операцію "and", $\bar{0} \in B^m$ — нульовий вектор . Нехай x — допустимий розв'язок. Це означає, що $\sum\limits_{j=1}^{n} A^j x_j = \sum\limits_{j \in S_1^x} A^j \leq \bar{1}$. Для довільних двох булевих значень b_1 та b_2 умови $b_1 + b_2 \leq 1$ та $b_1 \wedge b_2 = 0$ є еквівалентними. Для більшої кількості булевих значень це не так. З того, що $\sum\limits_{j=1}^{K} b_j \leq 1$ $(K > 2)$ випливає, що $\bigwedge\limits_{j=1}^{K} b_j = 0$, але навпаки ні. Умова $\bigwedge\limits_{j=1}^{K} b_j = 0$ вказує лише на те, що серед значень $b_1, \ldots, b_K$ є хоча б одне, яке рівне 0. Для того щоб серед значень $b_1, \ldots, b_K$ було не більше одного із значенням 1 потрібно, щоби для усіх пар індексів j та s, таких що $j \neq s$, виконувалося б $b_j \wedge b_s = 0$, або $\bigvee\limits_{\substack{(j,s) \\ j \neq s}} b_j \wedge b_s = 0$. Але $\bigvee\limits_{\substack{(j,s) \\ j \neq s}} b_j \wedge b_s = \bigvee\limits_{s=2}^{K} \left(\left(\bigvee\limits_{j=1}^{s-1} b_j \right) \wedge b_s \right)$. Таким чином умови $\sum\limits_{j=1}^{K} b_j \leq 1$ та $\bigvee\limits_{s=2}^{K} \left(\left(\bigvee\limits_{j=1}^{s-1} b_j \right) \wedge b_s \right) = 0$ — еквівалентні. На підставі цього можна казати що, система нерівностей $\sum\limits_{j \in S_1^x} a_{ij} \leq 1$, $i = 1, \ldots, m$, або $\sum\limits_{j=1}^{n} A^j x_j \leq \bar{1}$ еквівалентна умові $\bigvee\limits_{k=2}^{n_x} \left(\left(\bigvee\limits_{j=1}^{k-1} A^{S_{1,j}^x} \right) \wedge A^{S_{1,k}^x} \right) = \bar{0}$, що і доводить теорему.

Теореми 1 та 2 дозволяють побудувати нові схеми перевірки розв'язків на допустимість у задачах про покриття та упаковку скінченної множини. На рис. 1 та рис. 2 представлені відповідні алгоритми.

4. Пошук лексикографічного мінімуму множини у задачі про покриття. Основною складовою алгоритмів лексикографічного пошуку є відшукання лексикографічних екстремумів множин, які є підмножинами множини допустимих розв'язків задачі оптимізації. У стандартному алгоритмі пошуку лексикографічного мінімуму множини [2] на кожному кроці k, $k = 1, \ldots, n$, здійснюється процес скалярної мінімізації значення змінної x_k. В результаті

```
bool CoverFeasible(x ∈ B^n){
 s = 0̄;
  for(int j = 1; j ≤ n; j++)
    if(x_j == 1)s = s ∨ A^j;
 return s == 1̄;
}
```

Рис. 1. Алгоритм перевірки допустимості розв'язку у задачі про покриття.

```
bool PackFeasible(x ∈ B^n){
 s = 0̄;
  for(int j = 1; j ≤ n; j++)
    if(x_j == 1){
      if(s ∧ A^j == 0̄)
        s = s ∨ A^j;
      else
        return false;
    }
 return true;
}
```

Рис. 2. Алгоритм перевірки допустимості розв'язку у задачі про упаковку.

отримуємо значення $\delta_k = \max\left\{1 - \sum\limits_{j=1}^{k-1} a_{ij}x_j^* - \sum\limits_{j=k+1}^{n} a_{ij} \,\middle|\, a_{ik} = 1\right\}$ та $x_k^* = \max\{0, \delta_k\}$. Крім того, слід зауважити, що завжди $\delta_k \leq 1$ [2].

Використовуючи вказані властивості допустимих розв'язків у задачі про покриття з попереднього пункту, виникає можливість побудувати нову схему пошуку лексикографічного мінімуму множини допустимих розв'язків задачі про покриття скінченної множини. Визначимо булеві вектори: $U^k = \bigvee\limits_{j=k}^{n} A^j$, $k = 1, \ldots, n$, $U^{n+1} = \bar{0}$, $Y^k = \bar{0}$, $k = 0, 1, \ldots, n$. На початку пошуку $x^* = \bar{1}$. На кожному кроці k, $k = 1, \ldots, n$, пошуку лексикографічного мінімуму перевіряється умова: чи вектор $Y^{k-1} \vee U^{k+1}$ рівний $\bar{1}$. Якщо умова виконується, тоді значення x_k^* стає рівним 0, якщо ні, тоді значення x_k^* залишається рівним 1. При цьому, якщо умова виконується, тоді $Y^k = Y^{k-1}$, інакше $Y^k = Y^{k-1} \vee A^k$.

Теорема 3. *Визначення вектору $Y^{k-1} \vee U^{k+1}$ на кожному k-му кроці, $k = 1, \ldots, n$, еквівалентно обчисленню значення δ_k у стандартному алгоритмі пошуку лексикографічного мінімуму множини для задачі про покриття.*

Доведення. Значення δ_k є мінімально можливим значенням при якому розв'язок $\left(x_1^*, \ldots, x_{k-1}^*, \delta_k, 1, \ldots, 1\right)$ задовольняє умову (2). Для допустимості розв'язку це значення коригується так, щоб воно було булевим. В решті решт, можна казати, що значення δ_k дозволяє визначити чи є допустимим розв'язок $\left(x_1^*, \ldots, x_{k-1}^*, 0, 1, \ldots, 1\right)$. Не важко помітити, що на кожному кроці k, $k = 1, \ldots, n$, розглянутих вище дій, $Y^{k-1} = \bigvee\limits_{\substack{j=1 \\ x_j^*=1}}^{k-1} A^j$, $U^{k+1} = \bigvee\limits_{j=k+1}^{n} A^j$ та $Y^{k-1} \vee U^{k+1} = \bigvee\limits_{\substack{j=1 \\ x_j^*=1}}^{k-1} A^j \vee \bigvee\limits_{j=k+1}^{n} A^j = \bigvee\limits_{j=1}^{n} A^j x_j^k$, де $x^k = \left(x_1^*, \ldots, x_{k-1}^*, 0, 1, \ldots, 1\right)$. Якщо

$\bigvee_{j=1}^{n} A^j x_j^k = \bar{1}$, тоді, згідно теореми 1, це означає, що розв'язок x^k — допустимий.

Слід відзначити, що коли на деякому кроці k, $k = 1, \ldots, n$, значення $x_k^* = 0$, тоді вектор Y^k не змінюється. Тому для підвищення ефективності роботи запропонованого бінарного алгоритму пошуку лексикографічного мінімуму у задачі про покриття, заміть того щоб кожен раз визначати вектор Y^k, може використовуватись масив індексів iy, у якому кожне значення iy_k містить останній номер кроку на якому відбувалася зміна вектору Y^{iy_k}, тобто $iy_k = \max\{j \in \{1, \ldots, k-1\} \,|\, Y^j = Y^{j-1} \vee A^j\}$. Нехай цілочислове значення $lastK$ містить номер останнього кроку, на якому відбувалася зміна вектору Y^k. На початку роботи алгоритму значення $lastK = 0$.

При такому уточненні, на кожному кроці k, $k = 1, \ldots, n$, пошуку лексикографічного мінімуму:

1) $iy_k = lastK$;

2) перевіряється умова: чи дорівнює вектор $Y^{k-1} \vee U^{k+1}$ вектору $\bar{1}$;

3) якщо так, тоді значення $x_k^* = 0$;

4) якщо ні, тоді $Y^k = Y^{lastK} \vee A^k$, $lastK = k$.

На рис. 3 представлена структурна схема алгоритму пошуку лексикографічного мінімуму множини допустимих значень задачі про покриття $LexMinCover$.

Рис. 3. Алгоритм пошуку лексикографічного мінімуму у задачі про покриття.

```
int[] LexMinCover(){
  int[] x* = 1̄; int lastK = 0;
  for(int k = 1; k ≤ n; k++){
    iy_k = lastK;
    if(Y^lastK ∨ U^{k+1} == 1̄)
      x*_k = 0;
    else
      {Y^k = Y^lastK ∨ A^k; lastK = k;}
  }
  return x*;
}
```

Алгоритм $LexMinCover$ призначений для пошуку лексикографічного мінімуму x^* множини X^{Cover}, $x^* = \min^L X^{Cover}$. Але, враховуючи те, що в процесі пошуку лексикографічного мінімуму усі зміни у послідовності векторів Y^k, $k = 1, \ldots, n$, фіксуються, алгоритм $LexMinCover$ може бути використаний для пошуку лексикографічного мінімуму множини $\left\{x \in X^{Cover} \middle| \, x \geq^L \bar{x}\right\}$, де $\bar{x} \in B^n$ — фіксований булевий вектор. Для цього цикл пошуку у алгоритмі $LexMinCover$ слід починати не з першої координати, а із координати, що забезпечує виконання прямого лексикографічного обмеження $x \geq^L \bar{x}$.

5. Пошук лексикографічного максимуму множини у задачі про упаковку. У стандартному алгоритмі пошуку лексикографічного максимуму

множини [3], враховуючи специфічність коефіцієнтів обмежень задачі про упаковку, на кожному кроці $k,\ k = 1, \ldots, n$, здійснюється процес скалярної максимізації за змінною x_k. В результаті якого обчислюється $\delta_k = \min\left\{ 1 - \sum\limits_{j=1}^{k-1} a_{ij}x_j^* \,\middle|\, a_{ik} = 1 \right\}$ та $x_k^* = \min\{1, \delta_k\}$. Слід зауважити, що значення δ_k завжди невід'ємне, $\delta_k \geq 0$ [3].

Спираючись на властивість допустимих розв'язків у задачі про упаковку, що обґрунтовується теоремою 2, опишемо схему бінарного пошуку лексикографічного максимуму множини допустимих розв'язків задачі про упаковку X^{Pack}. На початку пошуку $x^* = \bar{0},\ Y^k = \bar{0},\ k = 0, 1, \ldots, n$. На кожному кроці пошуку лексикографічного максимуму $k,\ k = 1, \ldots, n$ перевіряється умова: чи є нульовим вектор $Y^{k-1} \wedge A^k$. Якщо умова виконується, тоді значення x_k^* стає рівним 1, якщо ні, тоді значення x_k^* залишається рівним 0. При цьому, якщо умова виконується, тоді $Y^k = Y^{k-1} \vee A^k$, інакше $Y^k = Y^{k-1}$.

Теорема 4. *Визначення вектору $Y^{k-1} \wedge A^k$ на кожному k-му кроці, $k = 1, \ldots, n$, еквівалентно обчисленню значення δ_k у стандартному алгоритмі пошуку лексикографічного максимуму множини для задачі про упаковку.*

Доведення. Значення δ_k є максимально можливим значенням при якому розв'язок $\left(x_1^*, \ldots, x_{k-1}^*, \delta_k, 0, \ldots, 0\right)$ задовольняє умову (2) для задачі про упаковку. Для допустимості розв'язку це значення коригується так, щоб воно не перевищувало 1. Таким чином, значення δ_k дозволяє визначити допустимість розв'язку $\left(x_1^*, \ldots, x_{k-1}^*, 1, 0, \ldots, 0\right)$. Не важко помітити, що на кожному кроці $k,\ k = 1, \ldots, n$, розглянутих вище дій, $Y^{k-1} = \bigvee\limits_{\substack{j=1 \\ x_j^* = 1}}^{k-1} A^j = \bigvee\limits_{j=1}^{k-1} A^j x_j^k$ та $Y^{k-1} \wedge A^k = \left(\bigvee\limits_{j=1}^{k-1} A^j x_j^k \right) \wedge A^k$, де $x^k = \left(x_1^*, \ldots, x_{k-1}^*, 1, 0, \ldots, 0\right)$. Усі розв'язки вигляду $\left(x_1^*, \ldots, x_p^*, 0, \ldots, 0\right)$, $p = 1, \ldots, k-1$ є допустимим, тому $Y^{k-1} \leq \bar{1}$. Якщо $Y^{k-1} \wedge A^k \neq \bar{0}$, тоді знайдеться така координата $q > k$, для якої $Y_q^{k-1} = 1$, $A_q^k = 1$ та $Y_q^{k-1} \wedge A_q^k = 1$, тобто обмеження задачі з номером q для розв'язку x^k містить два не нульові значення, що порушує допустимість цього розв'язку. Якщо ж $Y^{k-1} \wedge A^k = \bar{0}$, тоді розв'язок x^k — допустимий.

По аналогії з бінарним алгоритмом пошуку лексикографічного мінімуму у задачі про покриття, для підвищення ефективності роботи бінарного алгоритму пошуку лексикографічного максимуму у задачі про упаковку, використаємо масив індексів iy у якому кожне значення iy_k містить останній номер кроку на якому відбувалася зміна вектору Y^k, $iy_k = \max\{j \in \{1, \ldots, k-1\} \,|\, Y^j = Y^{j-1} \vee A^j\}$. Нехай цілочислове значення $lastK$ містить номер останнього кроку на якому відбувалася зміна вектору Y^k. На початку роботи алгоритму значення $lastK = 0$.

Враховуючи вище сказане, на кожному кроці $k,\ k = 1, \ldots, n$, пошуку лексикографічного максимуму виконаємо дії:

1) $iy_k = lastK$;

2) перевіряється умова: чи є вектор $Y^{k-1} \wedge A^k$ нульовим;

3) якщо так, тоді значення $x_k^* = 1$, $Y^k = Y^{lastK} \vee A^k$, $lastK = k$.

На рис. 4 представлена структурна схема алгоритму пошуку лексикографічного максимуму множини допустимих значень задачі про упаковку скінченної множини $LexMaxPack$.

Рис. 4. Алгоритм пошуку лексикографічного максимуму у задачі про упаковку.

```
int[] LexMaxPack(){
  int[] x* = 0̄; int lastK = 0;
  for(int k = 1; k ≤ n; k++){
    iy_k = lastK;
    if(Y^lastK ∧ A^k == 0̄)
     {x*_k = 1; Y^k = Y^lastK ∨ A^k; lastK = k;}
  }
 return x*;
}
```

Алгоритм $LexMaxPack$ призначений для пошуку лексикографічного максимуму x^* множини X^{Pack}, $x^* = \max^L X^{Pack}$. Враховуючи те, що в процесі пошуку лексикографічного максимуму усі зміни у послідовності векторів Y^k, $k = 1, \ldots, n$, фіксуються, алгоритм $LexMaxPack$ також може бути використаний для пошуку лексикографічного максимуму множини $\left\{ x \in X^{Pack} \middle| \, x \leq^L \bar{x} \right\}$, де $\bar{x} \in B^n$ — фіксований булевий вектор. Для цього цикл пошуку у алгоритмі $LexMaxPack$ слід починати не з першої координати, а із координати $startK$, що забезпечує виконання прямого лексикографічного обмеження $x \leq^L \bar{x}$. Її значення легко відшукується за правилом: $startK = \max \left\{ j \in \{1, \ldots, n\} | \, \bar{x}_j = 1; \bigvee_{i=1}^{j-1} A^i \bar{x}_i \leq \bar{1} \right\} + 1$.

6. Аналіз ефективності бінарних алгоритмів пошуку лексикографічних екстремумів у задачах про покриття та упаковку. Сучасні процесори мають можливість здійснювати порозрядні операції з 128-и, 256-и і навіть 512-и розрядними двійковими числами. У реальних задачах про покриття або упаковку скінченної множини кількість лінійних обмежень коливається від декількох сотень до декількох тисяч. Тому для представлення одного стовпця матриці обмежень у двійковому вигляді одного бінарного числа навіть максимальної розмірності може виявитись не достатньо. У програмній реалізації стовпець матриці обмежень задачі представляється як послідовність двійкових чисел, розмірність яких підтримується процесором. Наприклад, якщо кількість обмежень задачі $m = 500$, а процесор має можливість працювати з 128-и розрядними числами, тоді стовпець матриці обмежень задачі A^k може бути представлений як послідовність з 4-х 128-и розрядних двійкових чисел:

$$A^k = (\underbrace{a_{1,k}, \ldots, a_{128,k}}_{128bit}, \underbrace{a_{129,k}, \ldots, a_{256,k}}_{128bit}, \underbrace{a_{257,k}, \ldots, a_{384,k}}_{128bit}, \underbrace{a_{385,k}, \ldots, a_{500,k}, 0, \ldots, 0}_{128bit})$$

У такому представленні одна порозрядна операція над стовпцями матриці обмежень реалізується як послідовність з 4-х порозрядних операцій над 128-и розрядними двійковими числами.

У роботі розроблено два способи представлення стовпців матриці обмежень задачі, як послідовність 128-и розрядних двійкових чисел ($Bin128$) та як 256-и розрядних двійкових чисел ($Bin256$). Бінарний пошук лексикографічних екстремумів множин у цих представленнях порівнювався з роботою стандартного алгоритму пошуку (Std) [2,3]. Оскільки алгоритми $LexMaxPack$ та $LexMinCover$ базуються на одних і тих самих принципах, то ефективність їх роботи для матриць обмежень з однаковою структурою у порівнянні із стандартним алгоритмом теж однакова. Тому аналіз результатів представлено тільки для алгоритму $LexMinCover$. Дослідження проводилося для задач про покриття з випадково побудованою матрицею обмежень, щільність не нульових елементів якої становила 5%. Для отримання середнього часу роботи алгоритмів, генерувалося 10 задач однієї розмірності. У кожній з 10 задач 1000 разів викликався кожен з 3-х алгоритмів пошуку лексикографічного мінімуму множини: стандартний (Std), $LexMinCover$ з 128-и розрядною реалізацією ($Bin128$) та $LexMinCover$ з 256-и розрядною реалізацією ($Bin256$). Таким чином для кожної розмірності кожен з 3-х алгоритмів викликався 10000 разів.

На рис. 5 представлено графік залежності часу роботи алгоритмів від кількості обмежень задачі. При цьому кількість обмежень у задачі про покриття змінювалась від 50 до 2000 з кроком 50. Кількість змінних n задачі була фіксованою і дорівнювала 20000, $n = 20000$.

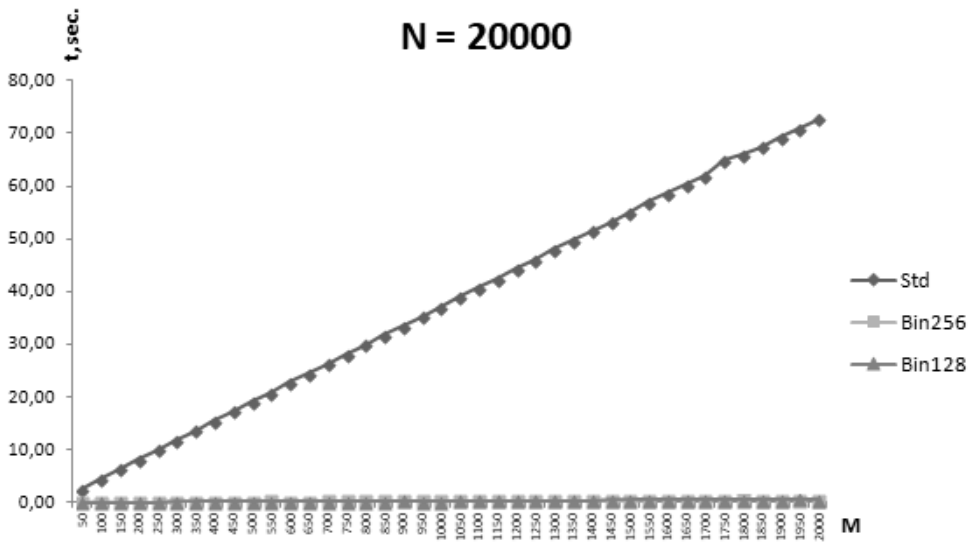

Рис. 5. Залежність між кількістю обмежень задачі про покриття та середнім часом роботи стандартного алгоритму та алгоритму $LexMinCover$.

Легко бачити, що із збільшенням кількості обмежень час роботи стандартного алгоритму пошуку лексикографічного мінімуму множини лінійно зростає і при $m = 2000$ становить $72,71$ сек. В той же час, у порівнянні із стандартним алгоритмом, ефективність бінарних алгоритмів змінюється не значно. Для більшої деталізації, на рис. 6 представлено графік залежності часу роботи тільки бінарних алгоритмів пошуку від кількості обмежень задачі.

Час роботи алгоритму $LexMinCover$ у реалізаціях $Bin256$ та $Bin128$ із збільшенням m теж зростає майже лінійно, але значно повільніше у порівнянні

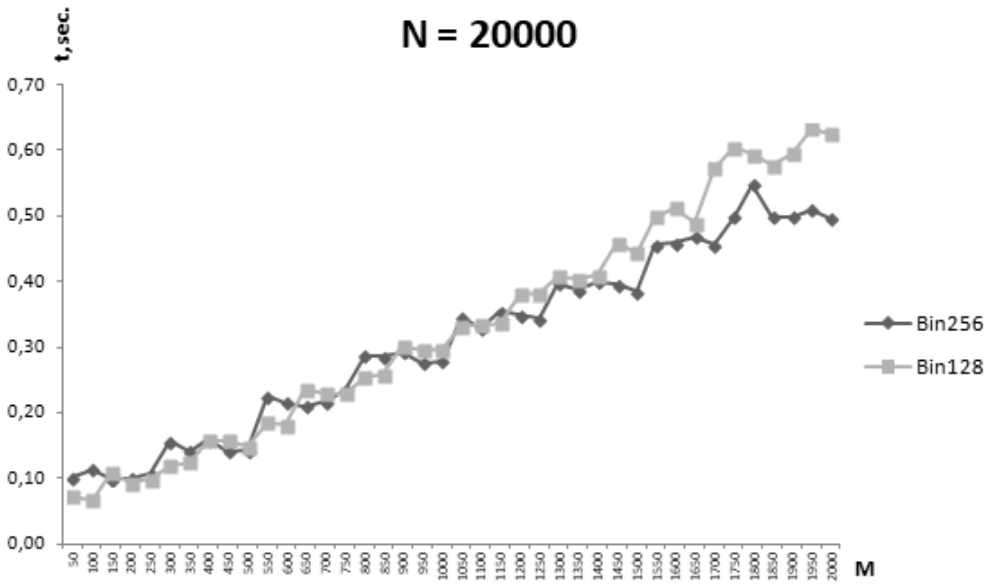

Рис. 6. Залежність між кількістю обмежень задачі про покриття та середнім часом роботи алгоритму $LexMinCover$ для різних представлень стовпців матриці обмежень.

із стандартним алгоритмом. Так, у реалізації $Bin256$ найбільший середній час роботи становив $0,548$ сек. при $m = 1800$, а у реалізації $Bin128$ — $0,633$ сек. при $m = 1950$. Варто відзначити, що ефективність бінарних алгоритмів пошуку лексикографічних екстремумів множини залежить від фактичної кількості двійкових розрядів, що використовуються для представлення стовпців матриці обмежень задачі та реальної кількості, що виділяється для вмісту послідовності двійкових значень на рівні процесору. Чим більше двійкових розрядів з виділеної кількості використовується, тим ефективніше працюють алгоритми бінарного пошуку. Наприклад, при $m \geq 1800$, у реалізації $Bin256$, потрібно виділити 8 256-и розрядних двійкових значень для збереження одного стовпця матриці обмежень. При $m = 1800$ не використаними лишаються 248 розрядів з 2048, а при $m = 2000$ — 48. На графіку з рис. 6 для $Bin256$ видно, що при $m = 2000$ середній час роботи бінарного алгоритму становив $0,496$ сек. В той же час, при $m = 1800$ — $0,548$ сек.

На рис. 7 представлено графік залежності часу роботи алгоритмів від кількості змінних задачі. При цьому кількість змінних у задачі про покриття змінювалась від 500 до 20000 з кроком 500. Кількість обмежень m задачі була фіксованою і дорівнювала 1000, $m = 1000$.

Із збільшенням кількості змінних час роботи стандартного алгоритму пошуку лексикографічного мінімуму множини лінійно зростає і при $n = 20000$ становить $37,1$ сек. В той же час, у порівнянні із стандартним алгоритмом, середній час роботи бінарних алгоритмів змінюється не значно. Так, при $n = 20000$ середній час роботи алгоритму $LexMinCover$ у 256-и розрядній реалізації становив $0,282$ сек., а у 128-и розрядній реалізації – $0,293$ сек. Зауважимо, що збільшення кількості змінних задачі менше впливає на ефективність роботи алгоритмів пошуку лексикографічного мінімуму. При зміні кількості обмежень

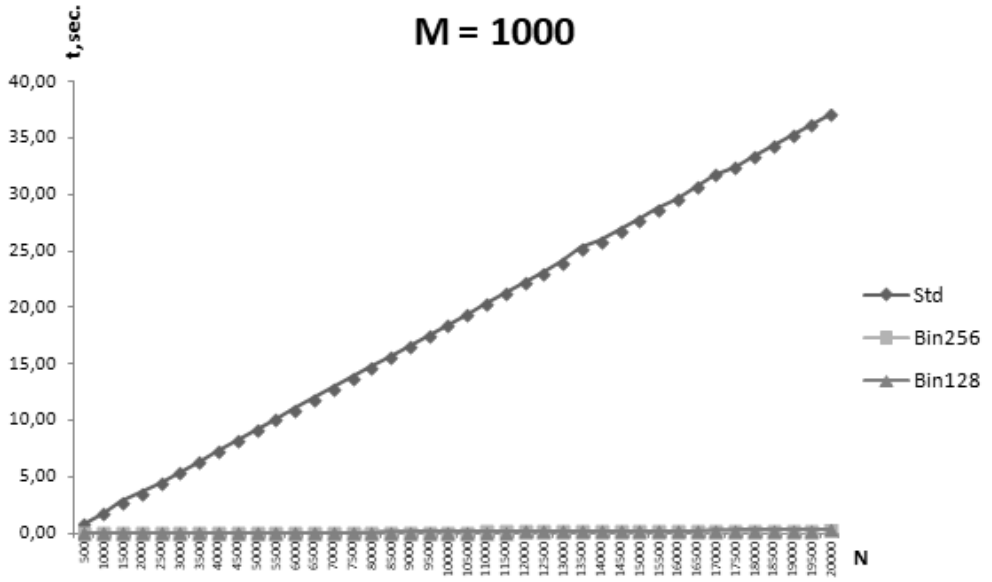

Рис. 7. Залежність між кількістю змінних задачі про покриття та середнім часом роботи стандартного алгоритму та алгоритму $LexMinCover$.

задачі лінійна залежність часу у стандартному алгоритмі може бути виражена формулою: $t = 0.036m+t_0$. Коли ж міняється кількість змінних, тоді залежність виражається як $t = 0.0018n + t_0$.

7. Висновки. Побудовані та досліджені нові алгоритми пошуку лексикографічних екстремумів множин, які є підмножинами множини допустимих розв'язків задач про покриття та упаковку скінченої множини. Особливості матриці обмежень та вектору вільних членів цих задач дозволили представити схему стандартного алгоритму пошуку лексикографічного екстремуму множини у нових термінах – через бінарні або порозрядні операції над булевими векторами. Порівняння ефективності роботи бінарних алгоритмів з стандартним алгоритмом пошуку здійснювалось на основі задач, отриманих випадковим чином. Аналіз результатів показав, що в залежності від розмірності задачі, в першу чергу від кількості обмежень, бінарний алгоритм працює у 50 – 150 разів швидше ніж стандартний алгоритм пошуку. Причому, із збільшенням розмірності задачі ефективність роботи бінарного алгоритму зростає.

Список використаної літератури

1. *Чупов С.В.* Новые подходы к решению задач дискретного программирования на основе лексикографического поиска // Кибернетика и систем. анализ. – 2016.– № 4.– С. 43–54.
2. *Чупов С.В.* Эффективные алгоритмы поиска лексикографического минимума множества// Компьютерная математика. – К.: Ин-т кибернетики имени В.М. Глушкова НАН Украины, 2015.– № 2.– С. 123–131.
3. *Чупов С.В.* Модифікації алгоритму пошуку лексикографічного максимуму множини// Наук. вісник Ужгород. ун-ту. Сер. матем. і інформ. – 2015.– Вип. № 2(27). – С. 168–173.

Одержано 22.01.2017