УДК 519.854.33

С.В. ЧУПОВ

НАБЛИЖЕНИЙ АЛГОРИТМ ЛЕКСИКОГРАФІЧНОГО ПОШУКУ У БАГАТЬОХ ПОРЯДКАХ РОЗВ'ЯЗКУ БАГАТОВИМІРНОЇ БУЛЕВОЇ ЗАДАЧІ ПРО РАНЕЦЬ

Анотація. Запропоновано нову схему наближеного лексикографічного пошуку розв'язку багатовимірної булевої задачі про ранець. Основна ідея алгоритму полягає у поступовому визначенні лексикографічного порядку (впорядкування змінних), у якому «якісні» розв'язки задачі належать прямому двосторонньому лексикографічному обмеженню, верхня межа якого є лексикографічним максимумом множини допустимих розв'язків задачі у цьому порядку. Оскільки пошук «якісних» розв'язків у кожному порядку здійснюється на обмеженому лексикографічному інтервалі, запропонований алгоритм названо обмеженим лексикографічним пошуком. Якість роботи наближеного методу обмеженого лексикографічного пошуку досліджується за допомогою розв'язання тестових задач з відомих наборів Beasley та F. Glover–G.A. Kochenberger.

Ключові слова: лексикографічний порядок, лексикографічний максимум, багатовимірна булева задача про ранець, алгоритм лексикографічного пошуку.

ВСТУП

Багатовимірна булева задача про ранець (ББЗР) добре відома та має чимало практичних застосувань, а саме: задачі планування виробництва, розкроювання матеріалів, розміщування процесорів та баз даних у розподілених комп'ютерних системах та ін. Крім того, вона є підкласом задачі цілочислового лінійного програмування. Оскільки ББЗР є NP-складною, точні алгоритми для розв'язання сучасних практичних задач можна використовувати тільки за їхньої невеликої розмірності. Враховуючи практичну важливість цієї задачі, зараз розроблено велику кількість наближених методів, які для її довільної розмірності дозволяють за прийнятний час отримувати якісні розв'язки. У працях [1, 2] наведено детальний огляд точних та евристичних методів розв'язання ББЗР. Зокрема, високою ефективністю відзначаються генетичні алгоритми [3], метод табу [4], метод глобального рівноважного пошуку, що належить до класу адаптивних методів з використанням симуляції відпалу [1, 5], гібридні алгоритми, які інтегрують ідеї генетичних методів, методу табу, лінійного програмування [6–8], та багато інших евристичних алгоритмів [9].

На відміну від точного алгоритму лексикографічного пошуку оптимального розв'язку задачі булевого програмування [10, 11], згідно з яким, починаючи з лексикографічного максимуму множини допустимих розв'язків задачі, здійснюється поступовий рух у наперед заданому напрямку лексикографічного спадання розв'язків, у цій роботі розглянемо новий підхід, названий обмеженим лексикографічним пошуком у багатьох порядках. Зазначимо, що для ББЗР існує такий лексикографічний порядок, для якого лексикографічний максимум множини допустимих розв'язків задачі збігається з її оптимальним розв'язком. Отже, основна ідея алгоритму полягає у пошуку такого лексикографічного порядку, за яким якісний розв'язок задачі знаходиться близько (у лексикографічному розумінні) до лексикографічного максимуму множини допустимих розв'язків задачі у цьому порядку. У кожному з отриманих у процесі роботи алгоритму лексикографічних порядків пошук кращих розв'язків здійснюють на обмеженому лексикографічному проміжку. Під обмеженим лексикографічним проміжком розуміємо множину векторів, що задовольняє пряме двостороннє лексикографічне обмеження

$\{x \in R^n \mid \bar{x} \le^L x \le^L \hat{x}\}$ [10]. Верхньою межею такого проміжку є лексикографічний максимум множини допустимих розв'язків ББЗР у заданому порядку, нижню межу визначають параметри алгоритму, зміна яких дозволяє або прискорювати пошук, або детальніше здійснювати аналіз цього інтервалу. На сьогодні стандартом обчислювальних систем є багатопроцесорні системи. Тому розроблення нових паралельних алгоритмів пошуку або трансформація наявних послідовних методів у паралельні — надзвичайно важлива задача. Оскільки обмежені лексикографічні інтервали у різних порядках, на яких здійснюється пошук, здебільшого не перетинаються, запропоновану схему обмеженого лексикографічного пошуку представлено в асинхронному варіанті, відповідно до якого окрема гілка алгоритму аналізує один лексикографічний проміжок у певному порядку незалежно від інших гілок. До того ж у кожен момент часу працює декілька таких гілок. Для аналізу ефективності роботи наближеного алгоритму обмеженого лексикографічного пошуку використано два відомі тестові набори задач: набір Beasley [12] та набір, запропонований F. Glover і G.A. Kochenberger [4]. У процесі розв'язання багатьох тестових задач отримано покращення відомих рекордів. Детальний опис результатів числових експериментів наведено в останньому розділі роботи.

ПОСТАНОВКА ЗАДАЧІ

Математичну модель багатовимірної булевої задачі про ранець задамо так: максимізувати

$$f_0(x) = \sum_{j=1}^{n} c_j x_j \tag{1}$$

за умов

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \ i = 1, \dots, m, \tag{2}$$

$$x \in \{0,1\}^n, \tag{3}$$

де $c_j > 0$, $b_i > 0$, $a_{ij} \ge 0$, $i = 1, \dots, m$, $j = 1, \dots, n$. Множину розв'язків, які задовольняють умови (2), (3), позначимо X^D.

Ґрунтуючись на лексикографічному впорядкуванні векторів як повному порядку та використовуючи поняття лексикографічного максимуму множини [10], точний алгоритм лексикографічного пошуку будує вкладену послідовність множини $X^D = X^0 \supset X^1 \supset \dots \supset X^k \supset \dots$ таким чином, щоб лексикографічні максимуми $x^k = \max^L X^k$, $k = 0, 1, \dots$, відповідних множин утворювали лексикографічно спадну послідовність допустимих розв'язків ББЗР $x^0 >^L x^1 >^L \dots >^L x^k >^L \dots$, якій би відповідала зростаюча послідовність значень цільової функції (1) $f_0(x^0) < f_0(x^1) < \dots < f_0(x^k) < \dots$ Пошук завершується, як тільки деяка множина X^k, $k = 1, 2, \dots$, виявиться порожньою. Стандартний алгоритм лексикографічного пошуку здійснює рух лише в одному лексикографічному порядку. Але у більшості задач оптимальний розв'язок знаходиться на значній лексикографічній відстані від початкового. Тому, якщо рухатись лише в одному напрямку лексикографічного спадання, час досягнення оптимального розв'язку може бути дуже тривалим. У наступних розділах описано нову схему алгоритму лексикографічного пошуку, згідно з якою лексикографічний рух здійснюється паралельно у декількох різних порядках і його межі у кожному порядку визначаються обмеженим лексикографічним проміжком.

НАБЛИЖЕНИЙ ПОШУК НА ОБМЕЖЕНОМУ ЛЕКСИКОГРАФІЧНОМУ ПРОМІЖКУ

У роботі [11] детально описано та обґрунтовано алгоритм стохастичного лексикографічного пошуку розв'язку задачі булевого програмування загального вигляду, а в [13] висловлено ідеї та наведено способи його використання для такої задачі. У цьому розділі детально описано алгоритм стохастичного лексикографічно-

го пошуку якісних розв'язків ББЗР $APrLexMax(X^D, x^0, i_1, i_2)$, який адаптовано для роботи на обмеженому лексикографічному проміжку.

У формальній схемі алгоритму використано такі позначення:

• f_{lp} — оптимальне значення цільової функції для лінійної релаксації задачі (1)–(3);

• $countT$ — масив значень розмірності n, у якому кожен елемент з індексом t містить кількість виконаних етапів алгоритму з використанням цієї координати;

• $sumT$ — масив значень розмірності n, у якому кожен елемент з індексом t містить суму значень цільової функції у точці y^r, отриманої при виконанні етапу r з використанням координати t, $sumT_t = \sum_{k=1}^{countT_t} f_0^{t,k}$;

• $tabuX$ — масив значень розмірності n, у якому кожна координата t містить значення, що дозволяє або забороняє використовувати цей індекс на поточному етапі; при цьому, якщо $tabuX_t = 0$, то індекс t можна використовувати на поточному етапі, у протилежному випадку — ні;

• $tabuMax$ — цілочислове значення, яке визначає кількість заборон використання певного індексу на етапах алгоритму; якщо значення $tabuMax = 0$, то усі заборони скасовуються і алгоритм перетворюється в детермінований; із збільшенням значення $tabuMax$ точність пошуку зменшується, але швидкість роботи алгоритму зростає;

• $f_t^{\lim}$ — дійсне значення, що визначає можливість заборони використання поточного індексу на наступних етапах; якщо для індексу t маємо $avgT_t < f_t^{\lim} f_{lp}$ [11], де $avgT_t$ — середнє арифметичне значень цільової функції, отримане після зміни координати t з 0 на 1, то застосування змінної з цим індексом забороняється; значення $f_t^{\lim}$ повинно бути меншим за 1, інакше використання усіх індексів забороняється; із зменшенням значення $f_t^{\lim}$ можливість заборони використання поточного індексу зменшується;

• f_0^* — рекорд для задачі (1)–(3), отриманий на поточний момент.

Таким чином, параметрами алгоритму $tabuMax$ та $f_t^{\lim}$ можна регулювати ступінь деталізації аналізу лексикографічного проміжку алгоритмом стохастичного пошуку.

Алгоритм стохастичного пошуку працює на обмеженому лексикографічному інтервалі $\tilde{x} \le^L x \le^L \bar{x}$ за умови впорядкування змінних або у порядку, що визначається початковою допустимою точкою x^0. Новий порядок отримуємо після перестановки змінних розв'язку x^0 так, щоб на початку розташовувались змінні, значення яких дорівнюють 1, а потім — інші. Отже, розв'язок x^0 у новому порядку матиме такий вигляд: $x^0 = (\underbrace{1, \dots, 1}_{s_x}, \underbrace{0, \dots, 0}_{n-s_x}) = (\underbrace{1, \dots, 1}_{s_x - l_x}, \underbrace{1, \dots, 1}_{l_x}, \underbrace{0, \dots, 0}_{n-s_x})$.

Зауважимо, що за умови переходу до нового порядку потрібно також змінити порядок розміщення стовпців матриці обмежень задачі та вектора коефіцієнтів цільової функції.

Алгоритм $\boldsymbol{APrLexMax(X^D, x^0, i_1, i_2)}$

Крок 0. Будуємо лексикографічний інтервал $\tilde{x} \le^L x \le^L \bar{x}$, на якому відбуватиметься пошук, де $\tilde{x} = (\underbrace{1, \dots, 1}_{s_x - l_x}, 1, \dots, \underset{i_1}{0}, \dots, \underset{i_2}{0}, \dots, 1, \underbrace{0, \dots, 0}_{n-s_x})$, $\bar{x} = (\underbrace{1, \dots, 1}_{s_x - l_x}, 1, \dots \dots, \underset{i_1}{0}, \dots, \underset{i_2}{0}, \dots, 1, \underbrace{1, \dots, 1}_{n-s_x})$.

Визначаємо лексикографічний максимум множини X^D на обмеженому лексикографічному проміжку $\widetilde{x} \le^L x \le^L \bar{x}$, $z^0 = \max^L \{x \in X^D \mid \widetilde{x} \le^L x \le^L \bar{x}\}$ [14]. Вектор x^0 допустимий, тому розв'язок $\widetilde{x}$ теж допустимий, а отже, z^0 завжди існує. Знаходимо значення $f_0^0 = f_0(z^0)$ і покладаємо $tabuX = (\underbrace{0, \ldots, 0}_{n})$, $countT = (\underbrace{0, \ldots, 0}_{n})$, $sumT = (\underbrace{0, \ldots, 0}_{n})$, $k = 1$.

Крок *k* ($k > 0$). На початку кроку встановлюємо $y^0 = z^{k-1}$, $r = 1$ і переходимо до першого етапу (кожен крок складається з декількох етапів).

На етапі r ($r > 0$) визначаємо індекс l_r. Покладаємо $t_r = n$ та розпочинаємо цикл пошуку індексу l_r. На кожному кроці циклу визначаємо значення l_r за правилом

$$l_r = \max \left\{ j \in \{s_x + 1, \ldots, t_r\} \mid y_j^{r-1} = 1;\ \sum_{\nu=1}^{j-1} c_\nu y_\nu^{r-1} + \sum_{\nu=j+1}^{n} c_\nu > f_0^{k-1} \right\}.$$

Якщо $l_r \le s_x$, то виходимо із циклу, оскільки індекс l_r має недопустиме значення. Нерівність $l_r \le s_x$ означає, що було порушено лексикографічні межі пошуку, тобто відбувся перехід за нижню межу лексикографічного інтервалу пошуку $\widetilde{x}$. У випадку $tabuX_{l_r} = 0$ припиняємо цикл, оскільки знайшли допустимий індекс l_r. Інакше ($tabuX_{l_r} > 0$) зменшуємо на 1 значення $tabuX_{l_r}$, покладаємо $t_r = l_r - 1$ та переходимо до наступного кроку циклу.

Якщо індекс l_r не знайдено, це означає, що за таких умов наближеного перегляду допустимих розв'язків множини X^D на обмеженому лексикографічному проміжку не знайдено розв'язку, кращого за значенням цільової функції, ніж z^{k-1}. Отже, припиняємо обчислення і як результат отримуємо найкращий із знайдених розв'язків, тобто z^{k-1}. У протилежному випадку маємо точку $w^r = (y_1^{r-1}, \ldots, y_{l_r-1}^{r-1}, 0, \underbrace{1, \ldots, 1}_{n-l_r})$.

Визначаємо допустимий розв'язок $y^r = \max^L \{x \in X^D \mid \widetilde{x} \le^L x \le^L w^r\}$, а $f_0^y = f_0(y^r)$ — відповідне йому значення цільової функції задачі. Вектор y^r отримуємо одним із ефективних алгоритмів пошуку лексикографічного максимуму множини, детально розглянутих у [14].

Змінюємо значення компонент масивів $countT$ та $sumT$ з індексом l_r: $countT_{l_r} = countT_{l_r} + 1$, $sumT_{l_r} = sumT_{l_r} + f_0^y$. Обчислюємо середнє арифметичне $avgT = \dfrac{sumT_{l_r}}{countT_{l_r}}$.

Якщо $avgT < f_{l_r}^{\lim} f_{lp}$, то покладаємо $tabuX_{l_r} = tabuMax$, тобто змінну з індексом l_r заборонено використовувати $tabuMax$ разів на наступних етапах алгоритму.

За умови $f_0^y \le f_0^{k-1}$ покладаємо $r = r + 1$ та переходимо до наступного етапу алгоритму. Для $f_0^y > f_0^{k-1}$ покладаємо $z^k = y^r$, $f_0^k = f_0^y$, $tabuX = (\underbrace{0, \ldots, 0}_{n})$.

Якщо $f_0^k > f_0^*$, то вектор z^k найкращий за значенням цільової функції з усіх знайдених раніше. Отже, припиняємо обчислення і як результат отримуємо розв'язок z^k. У протилежному випадку покладаємо $k = k + 1$ та переходимо до наступного кроку алгоритму.

Неважко показати, що скінченність алгоритму *APrLexMax* випливає з побудови на кожному кроці лексикографічно спадної послідовності розв'язків, яка обмежена знизу $\widetilde{x}$. На рис. 1 наведено конструктивну схему розглянутого алгоритму.

```
f_0^z ← LexMax(z, x̄, x̃); y ← z; f_0^y ← f_0^z;
countT ← 0̄; sum T ← 0̄; tabu X ← 0̄;
while (true){
int f_0^max ← f_0^y; int f_0^min ← f_0^y
int t ← n − 1;
  while (t > s_x){
    if (y_t = 1){
    y_t ← 0; f_0^min ← f_0^min − c_t;
      if (f_0^max > f_0^z){
        if (tabu X_t = 0) break;
        tabu X_t ← tabu X_t − 1;
      }
    } else {
    f_0^max ← f_0^max + c_t
    }
  t ← t − 1;
  }
if (t ≤ s_x) return z;
f_0^y ← f_0^min + LexMax(y, t);
countT_t ← countT_t + 1; sum T_t ← sum T_t + f_0^y; avgT ← sum T_t / count T_t;
if (avg T < f_t^lim * f_lp) tabuX_t ← tabu Max;
  if (f_0^y > f_0^z)
  z ← y; f_0^z ← f_0^y; tabu X ← 0̄;
  }
}
```

Рис. 1. Конструктивна схема алгоритму *APrLexMax*

ОБМЕЖЕНИЙ ЛЕКСИКОГРАФІЧНИЙ ПОШУК У БАГАТЬОХ ПОРЯДКАХ

У роботі [10] показано, що за вдалого вибору порядку або впорядкування змінних задачі оптимальний чи інший якісний розв'язок задачі може опинитися достатньо близько у лексикографічному сенсі до лексикографічного максимуму множини допустимих розв'язків X^D у цьому порядку. Тому знадобиться незначна кількість кроків алгоритму лексикографічного пошуку для отримання цього розв'язку. Лексикографічний максимум множини X^D у певному порядку для більшості задач булевого програмування може збігатися з оптимальним розв'язком цих задач.

Легко показати, що для задачі (1)–(3) існує такий лексикографічний порядок, що лексикографічний максимум множини X^D у ньому буде оптимальним розв'язком ББЗР. Дійсно, нехай x^* — оптимальний розв'язок булевої багатовимірної задачі про ранець (1)–(3). Упорядкуємо змінні цього розв'язку так, щоб на початку розташовувались змінні, значення яких дорівнюють 1, а потім — інші. У результаті отримаємо розв'язок $\bar{x}^* = (\underbrace{1, \ldots, 1}_{s}, \underbrace{0, \ldots, 0}_{n-s})$. Зауважимо, що за довільного впорядкування змінних розв'язку значення цільової функції задачі не

зміниться. Визначимо лексикографічний максимум множини допустимих розв'язків задачі $\bar{x}$ у новому порядку, розв'язуючи відповідні задачі скалярної максимізації послідовно, починаючи з першої координати. На кожному кроці k, $k = 1, 2, \ldots, n$, скалярної максимізації відшукують максимальне значення x_k для $x_j = \bar{x}_j$, $j = 1, \ldots, k-1$, $x_j = 0$, $j = k+1, \ldots, n$. Враховуючи, що $\bar{x}^*$ — допустимий розв'язок, допустимими будуть також розв'язки $\bar{x}^k = (\underbrace{1, \ldots, 1}_{k}, \underbrace{0, \ldots, 0}_{n-k})$, $k = 1, 2, \ldots, s$, тому $\bar{x}_k = \bar{x}_k^k = \bar{x}_k^*$, $k = 1, \ldots, s$. У розв'язку $\bar{x}^*$ цільова функція задачі набуває максимального значення. Крім того, $f_0(x_1, \ldots, x_{k-1}, 1, x_{k+1}, \ldots, x_n) > f_0(x_1, \ldots, x_{k-1}, 0, x_{k+1}, \ldots, x_n)$, $k = 1, 2, \ldots, n$. Якщо б для деякого $k > s$ виявилось, що $\bar{x}_k = 1$, то $f_0(\bar{x}^*) < f_0(\bar{x})$ і $\bar{x}^*$ не був би оптимальним розв'язком. Таким чином, $\bar{x}^* = \bar{x}$.

Наявні точні алгоритми лексикографічного пошуку створено для пошуку оптимального розв'язку в одному фіксованому порядку. Однак, як зазначено у [10, 11], цей процес може займати дуже тривалий час. Існування порядку, для якого оптимальний розв'язок ББЗР збігається з лексикографічним максимумом множини допустимих розв'язків, дозволяє визначити новий шлях до відшукання якісних розв'язків задачі (1)–(3). У детермінованих [10] алгоритмах лексикографічного пошуку рух завжди відбувається в одному напрямку лексикографічного спадання, починаючи з лексикографічного максимуму множини X^D в цьому порядку. Рух завершиться, досягнувши лексикографічного мінімуму множини X^D. Визначивши декілька порядків, здійснюватимемо рух на обмежених лексикографічних проміжках у цих порядках, які, ймовірно, можуть містити якісні розв'язки задачі. До того ж рух на таких інтервалах можна виконувати одночасно. Нижню межу лексикографічного проміжку визначають таким чином, щоб алгоритм лексикографічного пошуку на цьому інтервалі працював достатньо швидко і з задовільною точністю. Нехай x^{LexMax} — лексикографічний максимум множини X^D у поточному порядку. Упорядкуємо змінні цього розв'язку так, щоб на початку розташовувались змінні, значення яких дорівнюють 1, а потім — інші. У результаті отримаємо розв'язок $\bar{x}^{LexMax} = (\underbrace{1, \ldots, 1}_{s_x}, \underbrace{0, \ldots, 0}_{n-s_x})$. Задамо значення $l_x > 0$ та побудуємо вектор $\tilde{x} = (\underbrace{1, \ldots, 1}_{s_x - l_x}, \underbrace{0, \ldots, 0}_{l_x}, \underbrace{0, \ldots, 0}_{n-s_x})$. Зрозуміло, що $\tilde{x}$ є допустимим та лексикографічно меншим за вектор $\bar{x}^{LexMax}$. Отже, лексикографічний проміжок $\tilde{x} \le^L x \le^L \bar{x}^{LexMax}$ можна використати для відшукання кращих розв'язків одним з алгоритмів лексикографічного пошуку. Виявляється, що для значень $l_x > 4$ навіть алгоритм стохастичного лексикографічного пошуку не дає задовільних результатів, тобто його робота може тривати неприйнятно довго. За умови $l_x \le 4$ отримуємо дуже малий лексикографічний інтервал, що також негативно впливає на загальну ефективність роботи алгоритму. Прийнятні швидкість і точність роботи алгоритму для $l_x > 4$ забезпечує такий спосіб: випадковим чином вибираємо два значення: $s_x - l_x < i_1 < i_2 \le s_x$. Значення змінних розв'язку $\bar{x}^{LexMax}$ з координатами i_1 та i_2 покладаємо нульовими. У результаті отримаємо розв'язок $\tilde{x}^{(i_1, i_2)} = (\underbrace{1, \ldots, 1}_{s_x - l_x}, 1, \ldots, \underbrace{0}_{i_1}, \ldots, \underbrace{0}_{i_2}, \ldots, 1, \underbrace{0, \ldots, 0}_{n-s_x})$, який є допустимим, лексикографічно меншим за $\bar{x}^{LexMax}$, але лексикографічно більшим за $\tilde{x}$. На лексикографічному

інтервалі $\tilde{x}^{(i_1,i_2)} \le^L x \le^L \bar{x}^{(i_1,i_2)}$, де $\bar{x}^{(i_1,i_2)} = (\underbrace{1,\dots,1}_{s_x-l_x}, 1, 1, \dots, \underset{i_1}{0}, \dots, \underset{i_2}{0}, \dots, 1, \underbrace{1,\dots,1}_{n-s_x})$, алгоритмом стохастичного лексикографічного пошуку спробуємо знайти кращий розв'язок. Вибираємо наступну пару координат i_1 та i_2 і для неї також виконуємо пошук. Перебравши усі або певну кількість пар координат i_1 та i_2, пошук на лексикографічному інтервалі $\tilde{x} \le^L x \le^L \bar{x}^{LexMax}$ будемо вважати завершеним. За результатами експериментальних досліджень такий підхід виявився доволі ефективним для $20 \le l_x \le 40$. Після завершення пошуку на лексикографічному проміжку $\tilde{x} \le^L x \le^L \bar{x}^{LexMax}$, виходячи з отриманих результатів, вибираємо наступний лексикографічний порядок або впорядкування змінних задачі, будуємо новий лексикографічний інтервал $\tilde{x} \le^L x \le^L \bar{x}^{LexMax}$ і на ньому знову здійснюємо пошук за описаним вище способом. Зауважимо, що для різних значень пар координат i_1 та i_2 відповідні лексикографічні інтервали $\tilde{x}^{(i_1,i_2)} \le^L x \le^L \bar{x}^{(i_1,i_2)}$ не перетинаються. Тому з'являється можливість асинхронного аналізу таких інтервалів алгоритмом стохастичного лексикографічного пошуку, описаним у попередньому розділі.

На рис. 2 наведено конструктивну схему наближеного алгоритму лексикографічного пошуку $AParallelKnapsack2(p_{\max}, \bar{f}_0^{up}, \hat{f}_0^{up}, l_x)$, який дозволяє асинхронно виконувати роботу у різних лексикографічних порядках.

Значення $p_{\max}$ задає кількість гілок алгоритму, які виконуються паралельно алгоритмом наближеного лексикографічного пошуку *APrLexMax*, розглянутим у попередньому розділі; $\bar{f}_0^{up}$ та $\hat{f}_0^{up}$ — параметри алгоритму для фіксації хороших розв'язків та запобігання перегляду вже проаналізованих розв'язків задачі, до того ж $0 < \bar{f}_0^{up} < \hat{f}_0^{up} < 1$.

У цій схемі використано структури *Solution* та *BriefSolution.* Структура *Solution* містить дані про допустимий розв'язок задачі. Основні поля цієї структури такі: *Solution.Order* — порядок, за яким отримано розв'язок; *Solution.X* — допустимий розв'язок задачі в цьому порядку; *Solution.OriginalX* — розв'язок задачі на початковому впорядкуванні змінних; *Solution.F* — значення цільової функції розв'язку; *Solution.Ones* — кількість одиниць у розв'язку. Структуру *BriefSolution* використовують для швидкої ідентифікації розв'язку. У ній визначено два основні поля: *BriefSolution.F* та *BriefSolution.HashCode*: перше поле зберігає значення цільової функції розв'язку, друге — цілочислове значення, яке з високим ступенем імовірності ідентифікує цей розв'язок. Для певного розв'язку x значення ідентифікатора *HashCode* визначають як $HashCode = \sum_{j=1}^{n} hash_j * (x_j + 1)$, де $hash_j$, $j = 1, 2, \dots, n$, — випадкові цілочислові значення. Для більш зручного збереження результатів, отриманих після виконання однієї з паралельних гілок алгоритму, використовують структуру *TaskSolution.* Вона складається з трьох полів: *TaskSolution.Task* — номер паралельної гілки алгоритму, на якій отримано результат; *TaskSolution.F* — значення цільової функції розв'язку; *TaskSolution.Solution* — детальні відомості про розв'язок у вигляді структури *Solution.*

Робота алгоритму починається з визначення початкового порядку на основі оптимального розв'язку лінійної релаксації задачі та впорядкування змінних згідно з цим порядком (процедури *GetFirstOrder* та *ChangeOrder)*. Відшукується *LastSolution* — лексикографічний максимум множини X^D у порядку *Order* [10]. Змінні цього розв'язку впорядковують так: на початку розташовуються змінні, значення яких дорівнюють 1, а потім — інші (процедура *Reorder*). У результаті маємо

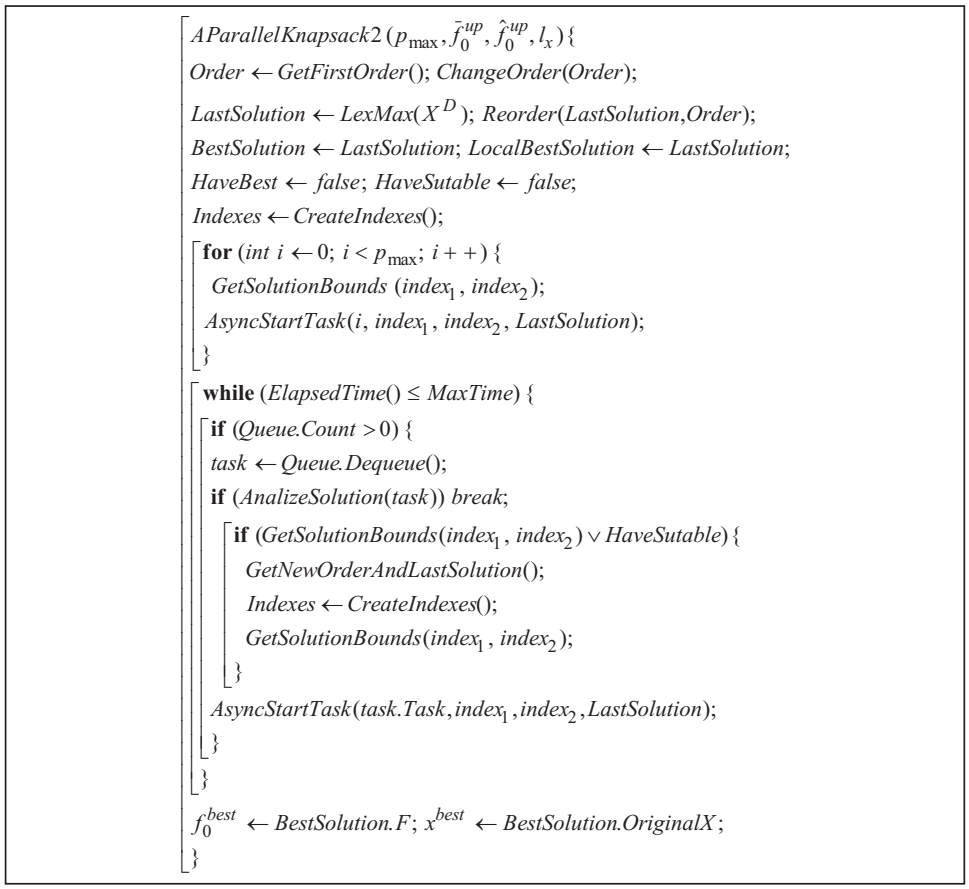

Рис. 2. Конструктивна схема алгоритму *AParallelKnapsack*2

$LastSolution.X = (\underbrace{1,\dots,1}_{s_x}, \underbrace{0,\dots,0}_{n-s_x})$. Запам'ятовуємо цей розв'язок як найкращий у *BestSolution* та *LocalBestSolution.* Надалі *BestSolution* містить найкращий з отриманих алгоритмом розв'язків, *LastSolution* — базовий розв'язок, на основі якого будують розв'язки вигляду $\tilde{x}^{(i_1,i_2)} = (\underbrace{1,\dots,1}_{s_x-l_x}, \underbrace{1,\dots,1,\underbrace{0}_{i_1},1,\dots,1,\underbrace{0}_{i_2},1,\dots,1}_{l_x}, \underbrace{0,\dots,0}_{n-s_x})$ та передають як початкові паралельним гілкам алгоритму, *LocalBestSolution* — найкращий з отриманих у результаті роботи паралельних гілок алгоритму розв'язків для базового розв'язку *LastSolution.* Значення *HaveBest* фіксує, чи отримано кращий розв'язок, *HaveSutable* — чи отримано хороший розв'язок. В алгоритмі визначається, що хороший розв'язок *solution* є допустимим розв'язком, який задовольняє умову $solution.F > \hat{f}_0^{up} * BestSolution.F$. Функція *CreateIndexes*() створює список *Indexes* усіх можливих пар індексів (i_1, i_2), $s_x - l_x < i_1 < i_2 \le s_x$, для побудови початкових розв'язків $\tilde{x}^{(i_1,i_2)}$ на основі базового розв'язку *LastSolution.* Випадковим чином $P_{\max}$ разів отримуємо чергову пару індексів (i_1, i_2) зі списку *Indexes* (процедура *GetSolutionBounds*), створюємо та асинхронно запускаємо обчислювальну гілку для паралельної роботи алгоритму $APrLexMax(X^D, LastSolution, i_1, i_2)$ (процедура *AsyncStartTask*). Після закінчення роботи чергової гілки алгоритму автоматично викликається процедура завершення, у якій результати локального пошуку фіксу-

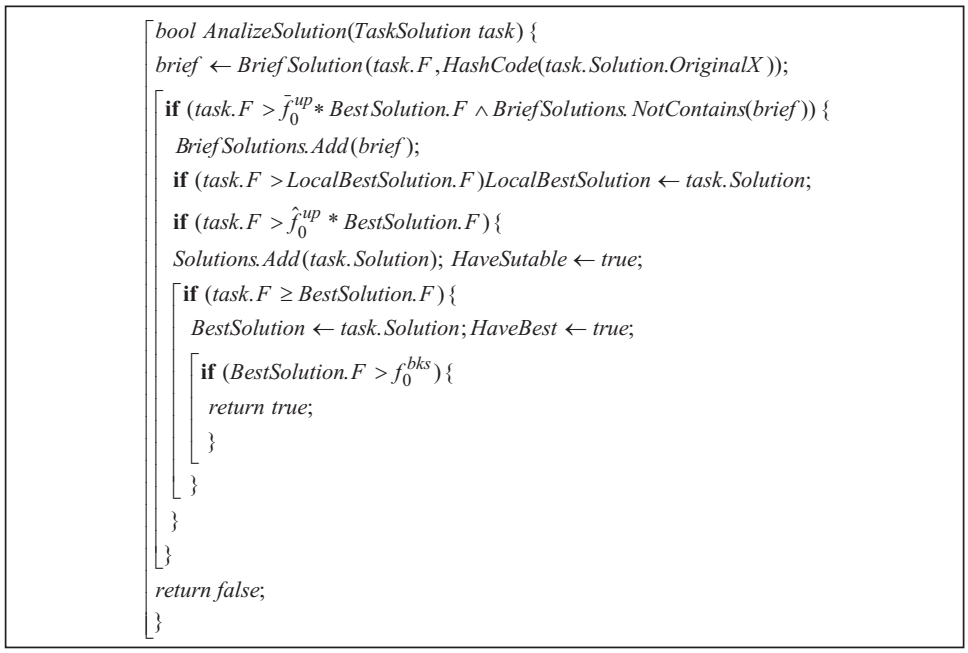

Рис. 3. Схема процедури аналізу розв'язку *AnalizeSolution*

ють у вигляді значення типу *TaskSolution.* Для подальшого аналізу в основному циклі алгоритму *AParallelKnapsack*2 це значення додають до черги *Queue.*

Основний цикл алгоритму продовжується до досягнення максимально можливого часу роботи *MaxTime* або визначення розв'язку, кращого за значенням цільової функції, ніж відомий на поточний момент рекордний розв'язок.

На кожному кроці циклу з черги *Queue*, якщо вона містить результати роботи паралельних гілок алгоритму, отримують значення *task* типу *TaskSolution* та передають для аналізу процедурі *AnalizeSolution.* Конструктивну схему процедури *AnalizeSolution* наведено на рис. 3.

Під час виконання процедури *AnalizeSolution* визначають, чи є сенс знайдений розв'язок запам'ятовувати для запобігання його повторного аналізу, і якщо варто ($task.F > \bar{f}_0^{up} * BestSolution.F$) і він не переглядався раніше (*BriefSolutions.NotContains*(*brief*)), то його ідентифікатор *brief* зберігають у списку *BriefSolutions.* У випадку, коли новий розв'язок кращий за *LocalBestSolution*, значення *LocalBestSolution* оновлюється. Якщо розв'язок хороший, то його записують у список хороших розв'язків *Solutions.* У випадку, коли розв'язок кращий за *BestSolution*, значення *BestSolution* оновлюється. Якщо значення цільової функції розв'язку більше за відомий рекорд f_0^{bks}, процедура *AnalizeSolution* повертає результат *true*, що означає завершення роботи алгоритму *AParallelKnapsack*2.

Якщо список пар індексів *Indexes* порожній або отримано хороший розв'язок, то відбувається перехід до іншого порядку та визначення нового базового розв'язку *LastSolution.* Ці дії виконує процедура *GetNewOrderAndLastSolution*, конструктивну схему якої наведено на рис. 4.

Якщо маємо хороший розв'язок, то його вибирають як новий базовий розв'язок *LastSolution.* Крім того, якщо розв'язок кращий, то зі списку хороших розв'язків *Solutions* видаляємо ті, які не є вже хорошими. Інакше на основі списку *Solutions* за функціями *GetNextBestSolution* та *GetNextOrder* визначаємо новий порядок *Order* та базовий розв'язок *LastSolution* відповідно. Зауважимо, що залежно від реалізації функцій *GetNextBestSolution* та *GetNextOrder* можна отримувати різні варіанти схем паралельного лексикографічного пошуку на обмежених лек-

```
void GetNewOrderAndLastSolution() {
  if (HaveSutable) {
    LastSolution ← LocalBestSolution;
    if (HaveBest)Solutions.RemoveAll(solution.F < f̂_0^up * BestSolution.F);
  }
  else {
    LastSolution ← GetNextBestSolution(Solutions);
    Order ← GetNextOrder(Solutions);
    ChangeOrder(Order);
  }
Reorder(LastSolution,Order);
LocalBestSolution ← LastSolution;
HaveBest ← false; HaveSutable ← false;
}
```

Рис. 4. Схема переходу до нового порядку пошуку

сикографічних проміжках. Після отримання порядку *Order* та базового розв'язку *LastSolution* змінюємо впорядкування змінних за процедурою *Reorder*, що дає змогу розпочати нову серію паралельних локальних пошуків. Після завершення роботи процедури *GetNewOrderAndLastSolution* створюємо новий список *Indexes* і маємо чергову пару індексів (i_1, i_2).

Якщо список пар індексів *Indexes* не порожній та отримано поганий розв'язок, також маємо пару індексів (i_1, i_2) зі списку *Indexes*.

Здійснивши аналіз розв'язку *task*.*Solution*, отриманого з гілки локального пошуку, номер якої *task*.*Task*, асинхронно по ній запускаємо наступний локальний пошук $AsyncStartTask(task.Task, i_1, i_2, LastSolution)$ з новими значеннями (i_1, i_2) та *LastSolution*. На цьому черговий крок роботи основного циклу алгоритму $AParallelKnapsack2$ завершено.

РЕЗУЛЬТАТИ ЧИСЛОВИХ ЕКСПЕРИМЕНТІВ

Для аналізу ефективності роботи алгоритму $AParallelKnapsack2(p_{\max}, \bar{f}_0^{up}, \hat{f}_0^{up}, l_x)$ паралельного лексикографічного пошуку якісного розв'язку задачі (1)–(3) використано два набори тестових задач. Перший — набір тестових задач про багатовимірний булевий ранець з відомої бібліотеки OR–Library (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html), запропонований Beasley [12]. Ці задачі розбито на групи відповідно до відношення ρ_x кількості ненульових компонент до загального числа компонент якісних розв'язків (у відсотках). Розв'язано тестові задачі, в яких кількість змінних n становила 250 та 500, а кількість обмежень m — 5, 10 та 30. Тридцять задач кожної розмірності умовно розбито на три групи по десять задач зі значеннями ρ_x — 25 % (задачі 00–09), 50 % (задачі 10–19) та 75 % (задачі 20–29) у кожній групі відповідно. Такі значення ρ_x зумовлені тим, що коефіцієнти обмежень задачі для цього тестового набору зв'язані співвідношеннями $\alpha = b_i / \sum_{j=1}^{n} a_{ij}$. Для $\rho_x = 25\%$ значення $\alpha = 0.25$, для $\rho_x = 50\%$ — $\alpha = 0.50$, для $\rho_x = 75\%$ — $\alpha = 0.75$. У табл. 1 та 2 представлено результати роботи наближеного паралельного алгоритму лексикографічного пошуку: f_0^{bks} та f_0^{best} — значення цільової функції задачі, отримані генетичним алгоритмом Chu–Beasley [3] та наближеним паралельним алгоритмом обмеженого лексикографічного пошуку. За аналогією з [5] у цих таблицях наведено результати розв'язання тих задач, для яких знайдено кращі, ніж у [3], розв'язки.

Таблиця 1. Результати розв'язання тестових задач з набору Beasley ($n = 250$)

Назва тестової задачі	m	ρ_x (%)	f_0^{bks}	f_0^{best}	Назва тестової задачі	m	ρ_x (%)	f_0^{bks}	f_0^{best}
5.250-05	5	25	60056	**60077**	30.250-01	30	25	58318	**58520**
5.250-12	5	50	108489	**108508**	30.250-03	30	25	56863	**56930**
5.250-16	5	50	109016	**109040**	30.250-05	30	25	57119	**57189**
5.250-18	5	50	109957	**109971**	30.250-07	30	25	56403	**56457**
5.250-19	5	50	107038	**107058**	30.250-10	30	50	107689	**107701**
5.250-20	5	75	149659	**149665**	30.250-12	30	50	106385	**106415**
5.250-21	5	75	155940	**155944**	30.250-13	30	50	106796	**106813**
10.250-01	10	25	58662	**58693**	30.250-14	30	50	107396	**107414**
10.250-07	10	25	58917	**58930**	30.250-15	30	50	107246	**107271**
10.250-08	10	25	59384	**59387**	30.250-16	30	50	106308	**106310**
10.250-09	10	25	59193	**59208**	30.250-17	30	50	103993	**103998**
10.250-10	10	50	110863	**110889**	30.250-20	30	75	150083	**150095**
10.250-11	10	50	108659	**108702**	30.250-22	30	75	152993	**153007**
10.250-13	10	50	110037	**110059**	30.250-23	30	75	153169	**153221**
10.250-14	10	50	108423	**108434**	30.250-25	30	75	148544	**148558**
10.250-20	10	75	151790	**151801**	30.250-26	30	75	147471	**147477**
10.250-24	10	75	151948	**151966**	30.250-27	30	75	152841	**152877**
10.250-27	10	75	153520	**153578**	30.250-28	30	75	149568	**149570**
30.250-00	30	25	56693	**56796**	30.250-29	30	75	149572	**149592**

Аналіз отриманих результатів свідчить, що наближений паралельний алгоритм лексикографічного пошуку достатньо ефективно працює для задач з кількістю змінних до 500 та довільним значенням ρ_x. У 38 з 90 розв'язаних тестових задач для $n = 250$ отримано кращі значення цільової функції, ніж відомі рекорди, знайдені генетичним алгоритмом. В інших задачах рекорди було повторено.

Для тестових задач, в яких кількість змінних дорівнює 500, найкращі результати отримано для задач з $\rho_x = 25$ та $\rho_x = 75$. Так, для значення $\rho_x = 25$ розв'язки з кращим значенням цільової функції знайдено у 18 задачах, для $\rho_x = 50$ — у 12 задачах, а для $\rho_x = 75$ — у 16 задачах. Загалом, з 90 задач розмірності $m \times 500$, $m = 5, 10, 30$, для 46 отримано кращі значення цільової функції, ніж відомі рекорди, знайдені генетичним алгоритмом. Залежність ефективності роботи наближеного паралельного алгоритму лексикографічного пошуку від значення ρ_x пов'язана з тим, що для $\rho_x = 25$ кількість хороших розв'язків задачі відносно невелика і тому їхній аналіз здійснюють досить швидко. Для задач з відношенням $\rho_x = 75$ лексикографічний інтервал $\tilde{x} \le^L x \le^L \bar{x}^{LexMax}$, на якому виконують локальний лексикографічний пошук у заданому порядку, відносно малий, що призводить до значного прискорення процесу пошуку на цьому проміжку. Зазначимо, що всі результати отримано за умови обмеження максимального часу роботи алгоритму до 1200 с.

Другий тестовий набір задач запропоновано Glover та Kochenberger (http://hces.bus.olemiss.edu/tools.html) [4]. Цей набір містить 11 задач, у яких кількість змінних варіюється від 100 до 2500, а обмежень — від 15 до 100.

У табл. 3 наведено результати розв'язання цих задач наближеним паралельним алгоритмом лексикографічного пошуку в порівнянні з іншими відомими результатами. Стовпці таблиці містять такі позначення: f_0^{GK} — найкраще значення цільової функції задачі, отримане Glover і Kochenberger [4]; f_0^{VH} — найкраще значення цільової функції задачі, отримане Vasquez і Jin-Kao Hao [6]; f_0^{best} — найкраще значення цільової функції задачі, отримане цим алгоритмом; f_0^{lp} — значення цільової функції задачі відповідної релаксаційної задачі лінійного програмування.

Таблица 2. Результати розв'язання тестових задач з набору Beasley ($n = 500$)

Назва тестової задачі	m	ρ_x (%)	f_0^{bks}	f_0^{best}	Назва тестової задачі	m	ρ_x (%)	f_0^{bks}	f_0^{best}
5.500-00	5	25	120130	**120148**	10.500-24	10	75	304344	**304350**
5.500-01	5	25	117837	**117854**	10.500-25	10	75	301730	**301745**
5.500-02	5	25	121109	**121123**	10.500-27	10	75	296437	**296441**
5.500-03	5	25	120798	**120804**	10.500-28	10	75	301313	**301327**
5.500-05	5	25	122007	**122009**	10.500-29	10	75	307014	**307033**
5.500-06	5	25	119113	**119127**	30.500-00	30	25	115868	**115950**
5.500-08	5	25	121575	**121586**	30.500-01	30	25	114667	**114701**
5.500-10	5	50	218422	**218426**	30.500-04	30	25	116353	**116355**
5.500-15	5	50	220514	**220530**	30.500-05	30	25	115604	**115623**
5.500-21	5	75	308077	**308083**	30.500-06	30	25	113952	**113959**
5.500-26	5	75	301322	**301329**	30.500-09	30	25	116947	**116981**
5.500-28	5	75	306430	**306454**	30.500-10	30	50	217995	**218032**
10.500-00	10	25	117726	**117809**	30.500-11	30	50	214534	**214626**
10.500-01	10	25	119139	**119160**	30.500-15	30	50	215762	**215782**
10.500-04	10	25	116434	**116462**	30.500-16	30	50	215772	**215777**
10.500-05	10	25	119454	**119504**	30.500-17	30	50	216336	**216429**
10.500-09	10	25	119125	**119146**	30.500-22	30	75	304995	**305062**
10.500-11	10	50	219022	**219032**	30.500-23	30	75	301935	**301945**
10.500-14	10	50	213809	**213827**	30.500-24	30	75	304404	**304413**
10.500-15	10	50	215013	**215062**	30.500-25	30	75	296894	**296962**
10.500-16	10	50	217896	**217931**	30.500-26	30	75	303233	**303277**
10.500-19	10	50	220833	**220872**	30.500-28	30	75	303057	**303092**
10.500-22	10	75	302354	**302395**	30.500-29	30	75	300460	**300473**

Таблица 3. Результати розв'язання тестових задач з набору Glover–Kochenberger

Назва тестової задачі	m	n	ρ_x (%)	f_0^{GK}	f_0^{VH}	f_0^{best}	f_0^{lp}
mk_gk_01	15	100	50	3766	3766	3766	3775.917
mk_gk_02	25	100	50	3958	3958	3958	3976.203
mk_gk_03	25	150	50	5650	5656	5656	5670.545
mk_gk_04	50	150	50	5764	5767	5767	5793.712
mk_gk_05	25	200	50	7557	7560	7560	7577.330
mk_gk_06	50	200	50	7672	7677	7677	7703.136
mk_gk_07	25	500	50	192215	19220	**19221**	19232.365
mk_gk_08	50	500	50	18801	18806	18806	18830.808
mk_gk_09	25	1500	50	58085	58087	**58089**	58100.455
mk_gk_10	50	1500	50	57292	**57295**	57294	57318.182
mk_gk_11	100	2500	50	95231	95237	**95238**	95277.491

Особливістю задач тестового набору є те, що рекорди f_0^{best} дуже близькі до значення цільової функції відповідної релаксаційної задачі лінійного програмування f_0^{lp}. Так, для 11-ї задачі, в якій кількість змінних дорівнює 2500, $\lfloor f_0^{lp} \rfloor - f_0^{best} = 39$. Крім того, всі хороші розв'язки задачі мають дуже близькі до рекордів значення цільової функції. Наприклад, значення цільової функції у по-

чатковому розв'язку для 11-ї задачі дорівнювало 95200 ± 5 для всіх спроб її розв'язання. Значення ρ_x у всіх задачах становить 50%. Усе це дає дуже щільне розміщення хороших розв'язків задачі в довільному околі локального максимуму. У таких умовах наближений паралельний алгоритм лексикографічного пошуку теж показав високу ефективність. Тільки для 10-ї задачі не вдалося отримати рекорд, знайдений гібридним алгоритмом [6]. У трьох найважчих задачах (7, 9, 11) отримано розв'язки з більшим значенням цільової функції, ніж у [6].

Зазначимо, що всі експериментальні досліди виконувались на комп'ютері з процесором Intel(R) Core(TM) i7-3770K, 3.50 GHz та 16 Gb оперативної пам'яті.

ВИСНОВКИ

У роботі запропоновано новий варіант лексикографічного пошуку якісного розв'язку задачі (1)–(3), який використовує наближений алгоритм лексикографічного пошуку для аналізу та виявлення кращих розв'язків на обмеженому лексикографічному проміжку. Оскільки такі інтервали будуються на основі різних впорядкувань змінних (лексикографічних порядків), лексикографічний рух вздовж цих проміжків призводить до перегляду різних послідовностей допустимих розв'язків задачі. У результаті виникає можливість паралельного аналізу декількох лексикографічних інтервалів. Виходячи з цього, такий пошук можна назвати паралельним обмеженим лексикографічним пошуком. У процесі експериментальних досліджень виявилося, що максимальна кількість паралельних гілок алгоритму залежить від структури конкретної задачі та величини лексикографічного проміжку, на якому здійснюють пошук. Чим менше хороших розв'язків містить лексикографічний інтервал, тим швидше буде здійснено його аналіз. Це, у свою чергу, призводить до збільшення максимально можливої кількості паралельних гілок алгоритму. Так, для задач з кількістю змінних до 500 найкращі результати досягалися за максимальної кількості гілок алгоритму, яка варіювалася від 64 до 80. Для задач з кількістю змінних від 500 і більше найкращою була кількість гілок від 100 до 120. Числові експерименти показали достатню ефективність запропонованого алгоритму як щодо швидкості роботи, так і якості отриманих розв'язків. У багатьох тестових задачах зафіксовано покращення відомих рекордів (для першого тестового набору з 180 розв'язаних задач у 84 отримано покращення, для другого набору з 11 задач — у трьох найскладніших). Різні способи вибору чергового лексикографічного порядку дають можливість побудови різних схем паралельного лексикографічного пошуку на обмежених лексикографічних проміжках.

Ідею паралельного обмеженого лексикографічного пошуку можна застосовувати для інших класів задач булевого програмування. Наприклад, запропонований алгоритм продемонстрував достатню ефективність для задачі про покриття скінченної множини. У подальших дослідженнях планується перевірити цей підхід на задачах, структура яких значно відрізняється від класу задач булевого лінійного програмування. Зокрема, це задача безумовної булевої квадратичної оптимізації (UBQP) та подібна їй за структурою задача про максимальний розріз графа (MAXCUT). Окремо слід виділити клас задач, у яких об'єктом пошуку є безпосередньо лексикографічний порядок або, що теж саме, перестановка перших n натуральних чисел. До цього класу належить, наприклад, квадратична задача про призначення (QAP).

СПИСОК ЛІТЕРАТУРИ

1. Сергиенко И.В., Шило В.П. Задачи дискретной оптимизации. Проблемы, методы решения, исследования. К.: Наук. думка, 2003. 264 с.
2. Fréville A. The multidimensional 0-1 knapsack problem: An overview. *European Journal of Operational Research*. 2004. Vol. 155. P. 1–21.
3. Chu P.C., Beasley J.E. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*. 1998. Vol. 4. P. 63–86.
4. Glover F., Kochenberger G.A. Critical event tabu search for multidimensional knapsack problems. *Metaheuristics*: *The Theory and Applications*. Kluwer Academic Publishers, 1996. P. 407–427.
5. Шило В.П. Решение многомерных задач о ранце методом глобального равновесного поиска. *Теория оптимальных решений*. Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2000. С. 10–13.

6. Vasquez M., Hao J.-K. A hybrid approach for the 0-1 multidimensional knapsack problem. *Proc. of the 17th Intern. Joint Conf. on Artificial Intelligence* (Seattle, 4–10 Aug. 2001). 2001. P. 328–333.
7. Vasquez M., Vimont Y. Improved results on the 0-1 multidimensional knapsack problem. *European Journal of Operational Research.* 2005. Vol. 165. P. 70–81.
8. Yuan Q., Yang Z.X. A weight-coded evolutionary algorithm for the multidimensional knapsack problem. *Advances in Pure Mathematics.* 2016. Vol. 6. P. 659–675.
9. Boyer V., Elkihel M., El Baz D. Heuristics for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*. 2009. Vol. 199. P. 658–664.
10. Чупов С.В. Новые подходы к решению задач дискретного программирования на основе лексикографического поиска. *Кибернетика и системный анализ*. 2016. Т. 52, № 4. С. 43–54.
11. Чупов С.В. Стохастичний алгоритм лексикографічного пошуку для задачі дискретного програмування. *Наук. вісник Ужгород. ун-ту. Сер. матем. і інформ*. 2016. Вип. 1 (28). С. 150–159.
12. Beasley J.E. OR-library; distributing test problems by electronic mail. *Journal of Operational Research Society*. 1990. Vol. 41. P. 1069–1072.
13. Чупов С.В. Алгоритм стохастичного лексикографічного пошуку оптимального розв'язку булевої задачі про ранець. *Математика. Інформаційні технології. Освіта: матеріали V Міжнар. наук.-практ. конф.* (Луцьк, 5–7 черв. 2016 р.). Луцьк: Вежа-Друк, 2016. С. 51, 52.
14. 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.

Надійшла до редакції 07.09.2017

С.В. Чупов

ПРИБЛИЖЕННЫЙ АЛГОРИТМ ЛЕКСИКОГРАФИЧЕСКОГО ПОИСКА ВО МНОГИХ ПОРЯДКАХ РЕШЕНИЯ МНОГОМЕРНОЙ БУЛЕВОЙ ЗАДАЧИ О РАНЦЕ

Аннотация. Предложена новая схема приближенного лексикографического поиска решения многомерной булевой задачи о ранце. Основная идея алгоритма состоит в постепенном определении лексикографического порядка (упорядочения переменных), в котором «качественные» решения задачи принадлежат прямому двустороннему лексикографическому ограничению, верхняя граница которого — лексикографический максимум множества допустимых решений задачи в этому порядке. Поскольку поиск «качественных» решений в каждом порядке осуществляется на ограниченном лексикографическом интервале, предлагаемый алгоритм назван ограниченным лексикографическим поиском. Качество работы приближенного метода ограниченного лексикографического поиска исследуется с помощью решения тестовых задач из известных наборов Beasley и F. Glover–G.A. Kochenberger.

Ключевые слова: лексикографический порядок, лексикографический максимум, многомерная булева задача о ранце, алгоритм лексикографического поиска.

S.V. Chupov

AN APPROXIMATE ALGORITHM FOR LEXICOGRAPHIC SEARCH IN MULTIPLE ORDERS FOR THE SOLUTION OF THE MULTIDIMENSIONAL BOOLEAN KNAPSACK PROBLEM

Abstract. A new scheme of approximate lexicographic search is proposed for the solution of the multidimensional boolean knapsack problem. The main idea of the algorithm is gradual definition of lexicographic order (ordering of variables) in which "qualitative" solutions of the problem belong to a direct two-sided lexicographic constraint whose upper bound is the lexicographic maximum of the set of feasible solutions of the problem in this order. Since the search for "qualitative" solutions in each order is carried out on a bounded lexicographic interval, the proposed algorithm is called a bounded lexicographic search. The quality of the approximate method of bounded lexicographic search is investigated by solving test problems from the well-known Beasley and Glover–Kochenberger sets.

Keywords: lexicographic order, lexicographic maximum, multidimensional boolean knapsack problem, lexicographic search algorithm.

Чупов Сергій Вікторович,
кандидат фіз.-мат. наук, доцент кафедри Вищого державного навчального закладу «Ужгородський національний університет», e-mail: sergey.chupov@gmail.com.