УДК 519.61 doi:10.15421/321612

О. М. Притоманова, В. В. Шаравара
Дніпропетровський національний університет імені Олеся Гончара

ОБ'ЄКТНО-ОРІЄНТОВАНИЙ ПІДХІД ДО ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АЛГОРИТМУ РОЗВ'ЯЗАННЯ ДЕЯКОГО КЛАСУ БАГАТОПРОДУКТОВИХ ЗАДАЧ ОПТИМАЛЬНОГО РОЗБИТТЯ МНОЖИН

Розглянуто застосування парадигми об'єктно-орієнтованого програмування в разі програмної реалізації алгоритму розв'язання багатопродуктової задачі оптимального розбиття множин із обмеженнями у вигляді рівностей і нерівностей за допомогою зведення до задачі недиференційовної оптимізації.

Рассмотрено применение парадигмы объектно-ориентированного программирования при программной реализации алгоритма решения многопродуктовой задачи оптимального разбиения множеств с ограничениями в форме равенств и неравенств путем сведения к задаче недифференцируемой оптимизации.

Use of object-oriented paradigm for implementing the algorithm of solving multiproduct optimal set partitioning problem with constraints in the form of equalities and inequalities is considered.

Ключові слова: задача оптимального розбиття множин, нескінченновимірна оптимізація, задача умовної оптимізації, недиференційовна оптимізація, r-алгоритм, об'єктно-орієнтоване програмування.

Вступ. У випадку програмної реалізації алгоритмів розв'язання математичних задач, зокрема і задач оптимального розбиття множин, як правило застосовують процедурний підхід. Його уже практично не застосовують у сучасному світі, йому на зміну прийшло об'єктно-орієнтоване програмування (ООП). ООП дозволяє відносно легко створювати програми таких розмірів, які неможливі в разі застосування процедурного підходу, у випадку застосування якого одержана програма була б настільки складною і заплутаною, що внесення будь-яких змін до неї стало б складним і ризикованим процесом. ООП дозволяє створювати програми, які легше читати, розуміти та розширювати за рахунок незначного зменшення ефективності роботи.

У даній статті застосовано об'єктно-орієнтований підхід до програмної реалізації алгоритму розв'язання неперервних багатопродуктових задач оптимального розбиття множин, який має переваги перед традиційним процедурним підходом.

Постановка задачі. Під неперервною багатопродуктовою задачею оптимального розбиття множин із n-вимірного евклідового простору E_n на підмножини із заданими положеннями центрів підмножин із обмеженнями у вигляді рівностей і нерівностей розумітимемо таку задачу [1].

Нехай Ω – обмежена, замкнута, вимірна за Лебегом множина в n-вимірному евклідовому просторі E_n. Необхідно знайти $N \times M$ вимірних за Лебегом таких підмножин $\Omega_1^1,...,\Omega_N^1,\Omega_1^2,...,\Omega_N^2,...,\Omega_1^M...,\Omega_N^M$ (можливо порожніх), щоб

$$mes\left(\Omega_i^j \cap \Omega_k^j\right) = 0, \quad i \neq k, \quad i,k = 1,...,N \quad j = 1,...,M, \tag{1}$$

$$\bigcup_{i=1}^{N} \Omega_i^j = \Omega, \quad j = 1,...,M, \tag{2}$$

$$\sum_{j=1}^{M} \int_{\Omega_i^j} \rho^j(x)dx = b_i, \quad i = 1,...,p, \tag{3}$$

$$\sum_{j=1}^{M} \int_{\Omega_i^j} \rho^j(x)dx \le b_i, \quad i = p+1,..,N \tag{4}$$

і при цьому, щоб функціонал (5) досягав мінімального значення

$$F\left(\Omega_1^1,...,\Omega_N^1,\Omega_1^2,...,\Omega_N^2,...,\Omega_1^M...,\Omega_N^M\right) = \sum_{j=1}^{M}\sum_{i=1}^{N} \int_{\Omega_i^j} (c^j(x,\ \tau_i) + a_i^j)\rho^j(x)dx. \tag{5}$$

Функції $c^j\left(x,\tau_i\right)$ – дійсні, обмежені, визначені на $\Omega \times \Omega$, вимірні за x за будь-якого фіксованого $\tau_i = \left(\tau_i^{(1)},...,\tau_i^{(n)}\right)$ із Ω для всіх $j = 1,...,M$.

Функції $\rho^j\left(x\right)$ – обмежені, вимірні на Ω для всіх $j = 1,...,M$.

$\tau_i = \left(\tau_i^{(1)},...,\tau_i^{(n)}\right)$ – задана точка підмножини Ω_i^j, одна і та сама для всіх $j = 1,...,M$, яку називають загальним центром підмножин $\Omega_i^1...,\Omega_i^M$.

$a_1^1,...,a_N^M$, $b_1,...,b_N$ – задані невід'ємні числа.

Для $b_1,...,b_N$ виконується умова розв'язності задачі

$$\int_{\Omega} \sum_{j=1}^{M} \rho^j(x)dx \le \sum_{i=1}^{N} b_i. \tag{6}$$

Метою дослідження є розробка програмного забезпечення для розв'язання багатопродуктових задач оптимального розбиття множин із фіксованими центрами й обмеженнями у вигляді рівностей і нерівностей, із застосуванням об'єктно-орієнтованого підходу під час програмування.

Метод розв'язання. Для розв'язання задачі зручно переформулювати її в термінах характеристичних функцій множин Ω_i^j:

$$\lambda_i^j(x) = \begin{cases} 1, & x \in \Omega_i^j \\ 0, & x \in \Omega \setminus \Omega_i^j \end{cases} \quad i = 1,...,N, \quad j = 1,...,M. \tag{7}$$

Тоді задачу можна буде переписати так.

Знайти вектор-функцію $\lambda_*(\cdot) \in \Gamma_1$ таку, що

$$I(\lambda_*(\cdot)) = \min_{\lambda \in \Gamma_1} \int_\Omega \sum_{j=1}^{M} \sum_{i=1}^{N} \left[c^j(x, \tau_i) + \alpha_i^j \right] \rho^j(x) \lambda_i^j(x) dx, \tag{8}$$

$$\Gamma_1 = \{\lambda(x) = \left(\lambda_1^1(x),...,\lambda_N^1(x); \lambda_1^M(x),...,\lambda_N^M(x) \right): \tag{9}$$

$$\lambda_i^j(x) = 0 \vee 1 \text{ майже всюди (м.в.) для } x \in \Omega,\ i = 1,...,N,\ j = 1,...,M, \tag{10}$$

$$\sum_{i=1}^{N} \lambda_i^j(x) = 1 \text{ м. в. для } x \in \Omega, \tag{11}$$

$$\int_\Omega \sum_{j=1}^{M} \rho^j(x) \lambda_i^j(x) dx = b_i,\ i = 1,...p, \tag{12}$$

$$\int_\Omega \sum_{j=1}^{M} \rho^j(x) \lambda_i^j(x) dx \le b_i,\ i = p+1,...,N\}. \tag{13}$$

Оптимальний розв'язок задачі визначають для $i = 1,...,N,\ \ j = 1,...,M$, та майже всіх $x \in \Omega$ таким чином [2]:

$$\lambda_{*i}(x) = \begin{cases} 1 & \text{для} \quad x \in \Omega_{*i}^{\ j} \\ 0 & \text{для} \quad x \in \Omega \setminus \Omega_{*i}^{\ j} \end{cases},$$

де

$$\Omega_{*i}^{\ j} = \{x \in \Omega : c^j(x, \tau_i) + a_i^j + \psi_i^* = \min_{k=1,...,N} [c^j(x, \tau_k) + a_k^j + \psi_k^*],$$

$$i \ne k \text{ м.в. для } x \in \Omega,\ j = 1,...,M\},$$

$\psi_1^*,...,\psi_N^*$ обирають оптимальний розв'язок двоїстої задачі у вигляді

$$G(\psi)=\int\limits_{\Omega}\sum_{j=1}^{M}\min_{k=1,\dots,N}\left[c^j\left(x,\tau_k\right)+a_k^j+\psi_k\right]\rho^j(x)dx-\sum_{i=1}\psi_i b_i\to\max \qquad (15)$$

за умов

$$\psi_i\geq 0,\ i=p+1,\dots,N\,. \qquad (16)$$

Від задачі умовної оптимізації (15) – (16) можна перейти до задачі безумовної оптимізації за допомогою введення до цільової функції негладкої штрафної функції множини $\{\psi_i\geq 0,\ i=p+1,\ p+2,\dots,N\}$

$$P(\psi)=G(\psi)-S\cdot\sum_{i=p+1}^{N}\max\left(0,-\psi_i\right)\to\max\,, \qquad (17)$$

де S – досить велике додатне число (значно більше за максимальний із множників Лагранжа).

Для функції (17) вектор узагальненого градієнта $g_P(\psi)=\left(g_P^{\psi_1}(\psi),\dots,g_P^{\psi_N}(\psi)\right)$ у точці $\psi=(\psi_1,\dots,\psi_N)$ можна обрати так:

$$g_P^{\psi_i}(\psi)=\int\limits_{\Omega}\sum_{j=1}^{M}\rho^j(x)\lambda_i^j(x)dx-b_i,\quad i=1,\dots,p\,, \qquad (18)$$

$$g_P^{\psi_i}(\psi)=\int\limits_{\Omega}\sum_{j=1}^{M}\rho^j(x)\lambda_i^j(x)dx-b_i+S\cdot\max\left(0,\operatorname{sign}(-\psi_i)\right),\quad i=1+p,\dots,N, \qquad (19)$$

де

$$\lambda_i^j\left(x\right)=\begin{cases}1, & \text{якщо } c^j\left(x,\tau_i\right)+a_i^j+\psi_i=\min\limits_{k=1,\dots,N}\left[c^j\left(x,\tau_k\right)+a_k^j+\psi_k\right],\\ 0 & \text{в іншому випадку.}\end{cases} \qquad (20)$$

Розв'язуватимемо задачу на координатній сітці. Нехай Π – такий паралелепіпед, що $\Omega\subset\Pi$. Покриємо Π прямокутною сіткою і покладемо $\rho^j(x)=0,\quad j=1,\dots,M$ для вузлів $x\in\Pi\setminus\Omega$.

Далі опишемо алгоритм розв'язку задачі на основі r-алгоритму у H-формі [3]:

1. Обрати початкове наближення $\psi^{(0)}$ і коефіцієнт розтягу простору $\lambda>1$. Покласти $H_0=I,\quad k=0$.

2. Обчислити значення $\lambda_i^j(x),\quad i=1,\dots,N,\quad j=1,\dots,M$ у вузлах сітки за формулами (20) у разі $\psi=\psi^{(k)}$.

3. Обчислити $g_P(\psi)$ за формулами (18) і (19) у випадку $\psi=\psi^{(k)}$.

4. Обчислити $\psi^{(k+1)}$ за формулою

$$\psi^{(k+1)} = \psi^{(k)} + h_k \frac{\mathrm{H}_k g_P(\psi^{(k)})}{\sqrt{(\mathrm{H}_k g_P(\psi^{(k)}),\ g_P(\psi^{(k)}))}}.$$

5. Обчислити значення $\lambda_i^j(x), \quad i = 1,...,N, \quad j = 1,...,M$ у вузлах сітки за формулами (20) за $\psi = \psi^{(k+1)}$.

6. Знайти $g_P(\psi)$ за формулами (18) і (19) у разі $\psi = \psi^{(k+1)}$.

7. Обчислити

$$\gamma_k = g_P(\psi^{(k+1)}) - g_P(\psi^{(k)}),$$

$$H_{k+1} = H_k + (1-\beta^2)\frac{(H_k\gamma^{(k)})(H_k\gamma^{(k)})^T}{\gamma^{(k)^T} H_k \gamma^{(k)}}, \quad \text{де} \quad \beta = \frac{1}{\lambda}.$$

8. Якщо тест зупинки виконано, то покласти $\psi^* = \psi^{(k+1)}$ і завершити роботу алгоритму, інакше покласти $k := k+1$ і перейти на крок 2.

У ході реалізації даного алгоритму як початкове наближення $\psi^{(0)}$ було обрано нульовий вектор, а коефіцієнт розтягу простору взято $\lambda = 2$.

Для розробки програмного забезпечення обрано мову програмування C# і середовище розробки Microsoft Visual Studio 2013.

Під час розробки було створено систему класів (рис. 1–2).

Подамо коротку характеристику класів, наведених на діаграмах.

1. MultiproductTaskSolver – основний клас програми, що розв'язує поставлену задачу.

Публічний інтерфейс класу включає властивості, що відповідають параметрам задачі:

- public int DimensionsCount { get; set; } – розмірність простору;
- public Tuple<double, double>[] AreaBounds { get; set; } – границі області для кожного виміру;
- public int[] GridSizes { get; set; } – кількість вузлів сітки для кожного виміру;
- public int ProductsCount { get; set; } – кількість продуктів;
- public Func<Vector, Vector, double>[] VariableCost { get; set; } – масив функцій $c^j(x,\tau)$;
- public Func<Vector, double>[] Density { get; set; } – масив функцій $\rho^j(x)$;
- public int CentersCount { get; set; } – кількість підмножин;

- public Vector[] Centers { get; set; } – масив векторів – центрів підмножин;
- public double[,] FixedCost { get; set; } – матриця a_i^j ;
- public MultiproductTaskConstraint[] Constraints { get; set; } – масив обмежень у вигляді рівностей і нерівностей.

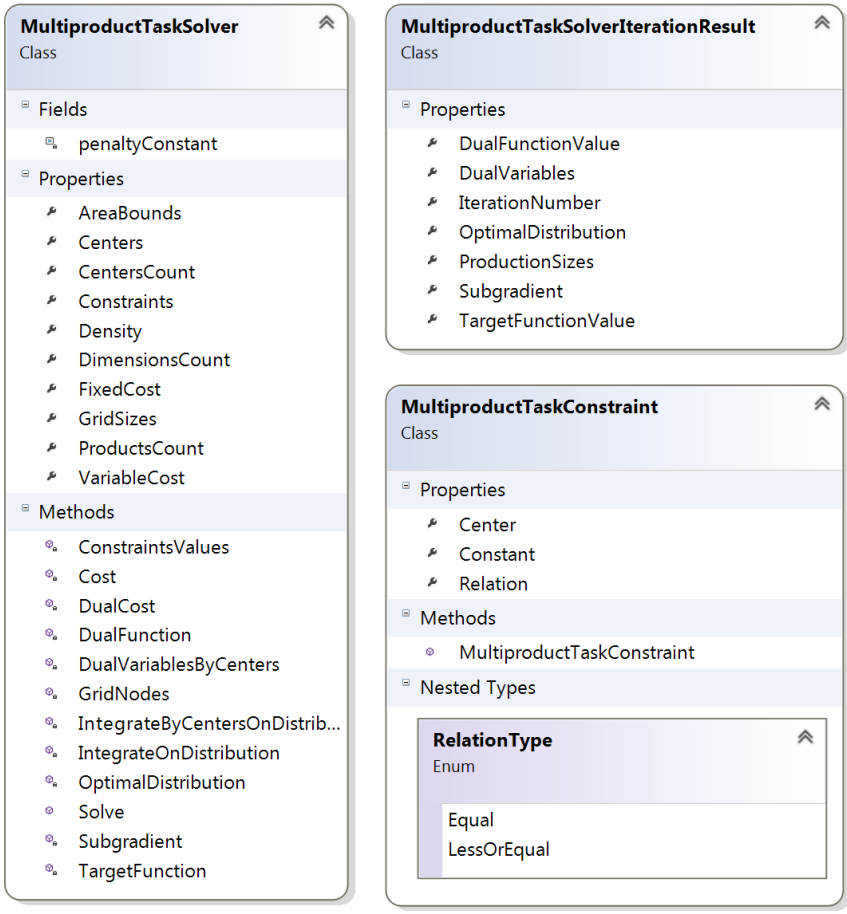


Рис. 1. Діаграма класів (початок)

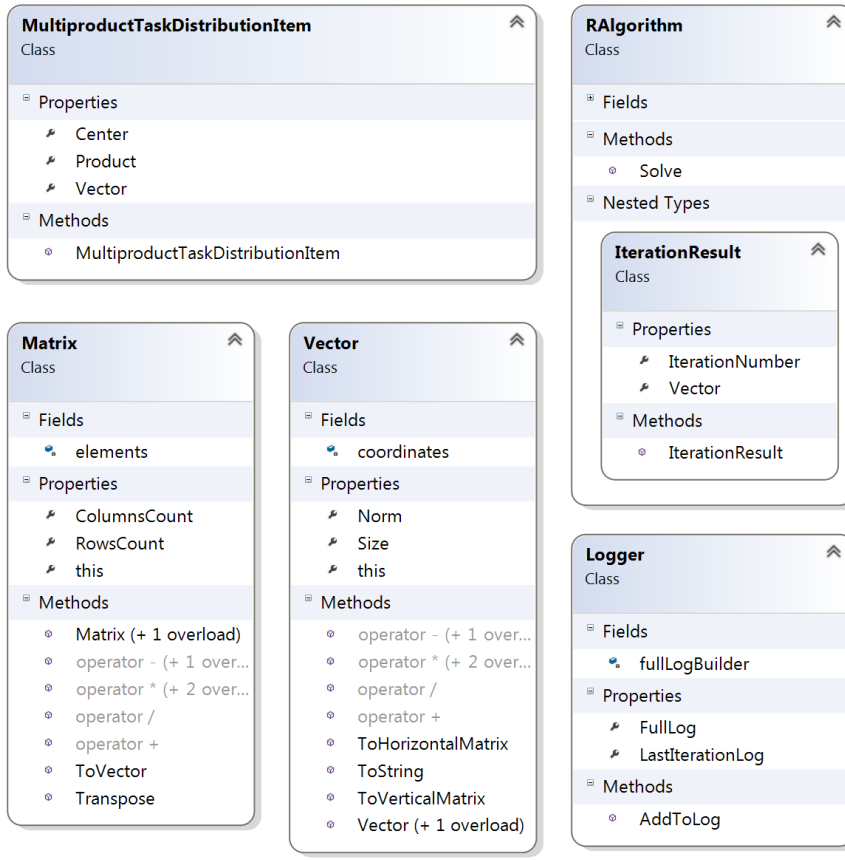


Рис. 1. Діаграма класів (закінчення)

Публічний інтерфейс також містить єдиний метод Solve, який має сигнатуру:

MultiproductTaskSolverIterationResult Solve(Action<MultiproductTaskSolverIterationResult> iterationDelegate).

У тілі цього методу запускають r-алгоритм за допомогою виклику функції Solve класу RAlgorithm із відповідними цільовою функцією та її субградієнтом, створеними на основі вище перелічених властивостей, а також після кожної ітерації r-алгоритму створюють об’єкт класу MultiproductTaskSolverIterationResult на основі одержаного після ітерації

об'єкта класу RAlgorithm.IterationResult і наведених властивостей, що містить інформацію про ітерацію алгоритму.

Єдиним параметром методу є функція, що приймає параметр типу MultiproductTaskSolverIterationResult. Ця функція буде викликана після кожної ітерації алгоритму, а як параметр – передано об'єкт відповідного класу, що містить інформацію про ітерацію алгоритму.

2. MultiproductTaskSolverIterationResult – клас, що зберігає інформацію про ітерацію алгоритму:

- public int IterationNumber { get; set; } – номер ітерації;
- public Vector DualVariables { get; set; } – вектор двоїстих змінних на поточній ітерації;
- public Vector Subgradient { get; set; } – вектор субградієнта на поточній ітерації;
- public double TargetFunctionValue { get; set; } – значення цільового функціонала початкової задачі на поточній ітерації;
- public double DualFunctionValue { get; set; } – значення функціонала двоїстої задачі на поточній ітерації;
- public Vector ProductionSizes { get; set; } – значення величин $\int_{\Omega} \sum_{j=1}^{M} \rho^j(x)\lambda_i^j(x)dx,\ i = 1,...,\ p$ на поточній ітерації;
- public List<MultiproductTaskDistributionItem> OptimalDistribution { get; set; } – розбиття на поточній ітерації.

3. MultiproductTaskConstraint – клас, що зберігає інформацію про обмеження:

- public int Center { get; set; } – номер центра, якому відповідає обмеження;
- public RelationType Relation { get; set; } – тип обмеження: Equal – рівність, LessOrEqual – нерівність;
- public double Constant { get; set; } – величина обмеження b_i.

4. MultiproductTaskDistributionItem – клас, що зберігає інформацію про вузол сітки та його приналежність до певної підмножини:

- public Vector Vector { get; set; } – координата вузла сітки;
- public int Product { get; set; } – номер продукту;
- public int Center { get; set; } – номер центра, якому відповідає вузол у розбитті за вказаним вище продуктом.

5. RAlgorithm – клас, що реалізує r-алгоритм Шора з постійним кроковим множником. Публічний інтерфейс включає єдиний метод Solve, який має таку сигнатуру:

IterationResult Solve(Vector x0, Func<Vector, double> function, Func<Vector, Vector> subgradientFunction, Action<IterationResult> iterationDelegate).

Перший параметр – функція, яку необхідно максимізувати r-алгоритмом.

Другий параметр – функція, за допомогою якої можна одержати субградієнт цільової функції у довільній точці.

Третій параметр – функція, що приймає параметр типу RAlgorithm.IterationResult. Ця функція буде викликана після кожної ітерації r-алгоритму, а як параметр – передано об'єкт відповідного класу, що містить інформацію про ітерацію r-алгоритму: номер ітерації та поточне наближення.

6. RAlgorithm.IterationResult – клас, що зберігає інформацію про ітерацію r-алгоритму:

- public int IterationNumber { get; set; } – номер ітерації;
- public Vector Vector { get; set; } – поточне наближення.

7. Matrix – клас, який моделює матрицю.

Даний клас містить перевантажені оператори, які відповідають математичним операціям над матрицями (додавання та множення матриць, множення і ділення матриці на число), а також містить метод Transpose, що повертає транспоновану матрицю.

8. Vector – клас, який моделює вектор.

Даний клас містить перевантажені оператори, які відповідають математичним операціям над векторами (додавання векторів, множення вектора на число, скалярний добуток векторів), а також містить властивість Norm, що повертає норму вектора.

9. Logger – клас, що зберігає текстовий лог, у якому містяться результати кожної ітерації алгоритму. Має властивості для одержання повного логу та логу останньої ітерації та метод AddToLog, за допомогою якого здійснюють додавання ітерації до логу.

Аналіз одержаних результатів. Застосування об'єктно-орієнтованого підходу дозволило досягти таких позитивних результатів:

1. Легкість читання програми

Перевантаження операторів для матриць і векторів спрощує сприймання коду порівняно з великою кількістю функцій із назвами sum, add, subtract, multiply, product тощо, які були б у випадку застосування процедурного підходу.

Застосування складних користувацьких типів даних, які об'єднують пов'язані дані, також спрощує їх передачу до інших функцій, що також полегшує сприйняття коду. У випадку процедурного підходу замість од-

ного об'єкта необхідно було б передавати 3–7 окремих параметрів, що надзвичайно незручно.

2. Роздільність коду – код програми природним чином розділений на логічні взаємопов'язані частини (код кожного класу міститься в окремому файлі), що полегшує сприйняття, пошук та налагодження.

3. Легкість внесення змін. Досягається за рахунок роздільності коду та незалежності частин одна від одної.

Роботу алгоритму було перевірено на модельних задачах, результати – порівняно з результатами, одержаними у [1].

Модельна задача 1. Задача розбиття області на зони обслуговування п'яти підприємств, що виробляють продукцію двох видів із обмеженнями із урахуванням обсягів виробництва.

Множина споживачів має вигляд $\Omega = \{(x, y) : 0 \le x \le 5, \quad 0 \le y \le 10\}$.

Координати пунктів виробництва:

$\tau_1 = (1;9,5),\ \tau_2 = (2;5),\ \tau_3 = (3;4),\ \tau_4 = (4;9),\ \tau_5 = (4;2)$.

Функції, що описують вартість транспортування одиниці продукції *j*-го виду із *i*-го пункту виробництва до пункту споживання із координатами (x, y):

$$c^j(x, y, \tau_i) = \sqrt{(x - \tau_i^{(1)})^2 + (y - \tau_i^{(2)})^2},\ i = 1,2,...,5,\ j = 1,2.$$

Функція попиту на продукцію: $\rho^j(x, y) \equiv 1,\ j = 1,2$.

Вартість виробництва одиниці продукції *j*-го виду на *i*-му підприємстві:

$$a_1^1 = 1,\ a_2^1 = 100,\ a_3^1 = 1,\ a_4^1 = 100,\ a_5^1 = 100,$$
$$a_1^2 = 100,\ a_2^2 = 1,\ a_3^2 = 100,\ a_4^2 = 10,\ a_5^2 = 1.$$

Обмеження на обсяги виробництва:

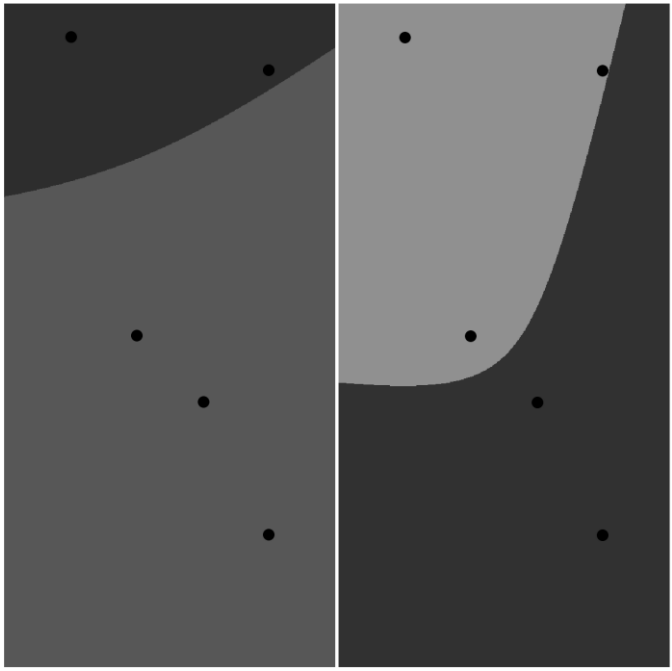
$$\sum_{j=1}^{2} \iint_{\Omega_i^j} \rho^j(x, y)dxdy = b_i,\ i = 1,2, \quad b_1 = 10, \quad b_2 = 20,$$

$$\sum_{j=1}^{2} \iint_{\Omega_i^j} \rho^j(x, y)dxdy \le b_i,\ i = 3,4,5 \quad b_3 = 500, \quad b_4 = 500, \quad b_5 = 900.$$

Необхідно розбити Ω на Ω_i^j так, щоб мінімізувати функціонал сумарних витрат на виробництво продукції та доставку її споживачам:

$$F\left(\Omega_1^1,...,\Omega_5^2\right) = \sum_{j=1}^{2} \sum_{i=1}^{5} \iint_{\Omega_i^j} (c^j(x,\ y, \tau_i) + a_i^j)\rho^j(x, y)dxdy.$$

На рис. 2 зображено результат роботи програми для сітки 500×1000. На рис. 3 – оптимальне розбиття, отримане у [1] для такої ж сітки.

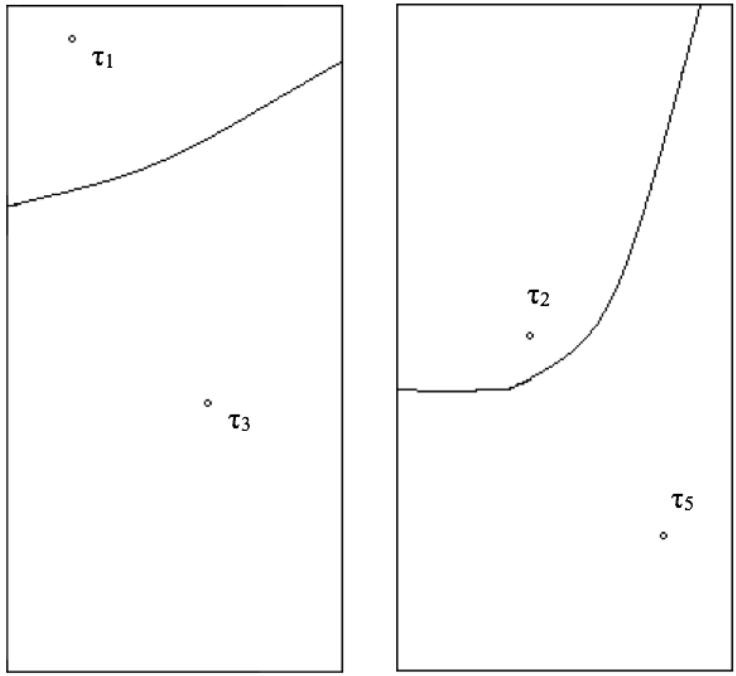


```
Ітерація 101:
Двоїсті змінні: (1,70333559; 2,49048048; 0,00000007; 0,00000002; 0,00000003)
Субградієнт: (0,00450000; 0,00780000; -460,00450000; -500,00000000; -870,007806
Значення цільового функціоналу: 361,612567653189
Значення функціоналу двоїстої задачі: 361,639590651959
Об'єми виробництва: (10,00450000; 20,00780000; 39,99550000; 0,00000000; 29,9921
```

Рис. 2. Результат роботи програми для модельної задачі 1 із розмірами сітки 500×1000

Нижче (табл. 1) наведено числові результати, одержані для різних розмірів сіток, і результати, одержані у [1]. Як критерій зупинки всюди було застосовано умову

$$\left\| \psi^{(k)} - \psi^{(k-1)} \right\| \le 10^{-6}.$$



а *б*

Рис. 3. Оптимальне розбиття за 1-м продуктом (*а*) і 2-м продуктом (*б*) для модельної задачі 1, одержане у [1]

Таблиця 1

Порівняльна таблиця результатів розв'язку модельної задачі 1

Параметр	Сітка 100×200	Сітка 200×400	Сітка 500×1000	Результати, одержані у [1] (сітка 500×1000)
Кількість ітерацій	102	89	101	59
Мінімальне значення прямого функціонала	361,63	361,73	361,61	361,64
Максимальне значення двоїстого функціонала	361,63	361,64	361,64	—

Модельна задача 2. Задача розбиття області на зони обслуговування дев'яти підприємств, що виробляють продукцію трьох видів із обмеженнями із урахуванням обсягів виробництва.

Множина споживачів має вигляд $\Omega = \{(x, y) : 0 \le x \le 6, \quad 0 \le y \le 20\}$.

Координати пунктів виробництва:

$$\tau_1 = (0{,}2;0{,}1),\ \tau_2 = (1{,}6;1{,}3),\ \tau_3 = (2{,}9;2{,}1),\ \tau_4 = (4{,}4;5{,}7),\ \tau_5 = (5{,}1;10{,}0)$$
$$\tau_6 = (5{,}6;11{,}5),\ \tau_7 = (1{,}0;12{,}9),\ \tau_8 = (1{,}5;13{,}9),\ \tau_9 = (3{,}5;19{,}0).$$

Функції, що описують вартість транспортування одиниці продукції j-го виду із i-го пункту виробництва до пункту споживання із координатами (x, y):

$$c^j(x, y, \tau_i) = \sqrt{(x - \tau_i^{(1)})^2 + (y - \tau_i^{(2)})^2},\ i = 1,2,...,9,\ j = 1,2,3.$$

Функція попиту на продукцію:

$$\rho^j(x, y) \equiv 1,\ j = 1,2,3.$$

Вартість виробництва одиниці продукції j-го виду на i-му підприємстві:

$$a_1^1 = 0,\ a_2^1 = 100,\ a_3^1 = 1{,}00\ a_4^1 = 100,\ a_5^1 = 100,\ a_6^1 = 100,\ a_7^1 = 1{,}00\ a_8^1 = 100,\ a_9^1 = 0,$$
$$a_1^2 = 0,\ a_2^2 = 100,\ a_3^2 = 100,\ a_4^2 = 100,\ a_5^2 = 0\ a_6^2 = 100,\ a_7^2 = 100\ a_8^2 = 0,\ a_9^2 = 100,$$
$$a_1^3 = 0,\ a_2^3 = 0,\ a_3^3 = 100\ a_4^3 = 100,\ a_5^3 = 100,\ a_6^3 = 0,\ a_7^3 = 1{,}00\ a_8^3 = 100,\ a_9^3 = 100.$$

Обмеження із урахуванням обсягів виробництва:

$$\sum_{j=1}^{3} \iint_{\Omega_1^j} \rho^j(x, y)dxdy = 200, \quad \sum_{j=1}^{3} \iint_{\Omega_2^j} \rho^j(x, y)dxdy = 50,$$

$$\sum_{j=1}^{3} \iint_{\Omega_i^j} \rho^j(x, y)dxdy \le 20, \quad i = 3,4,6,7,8,$$

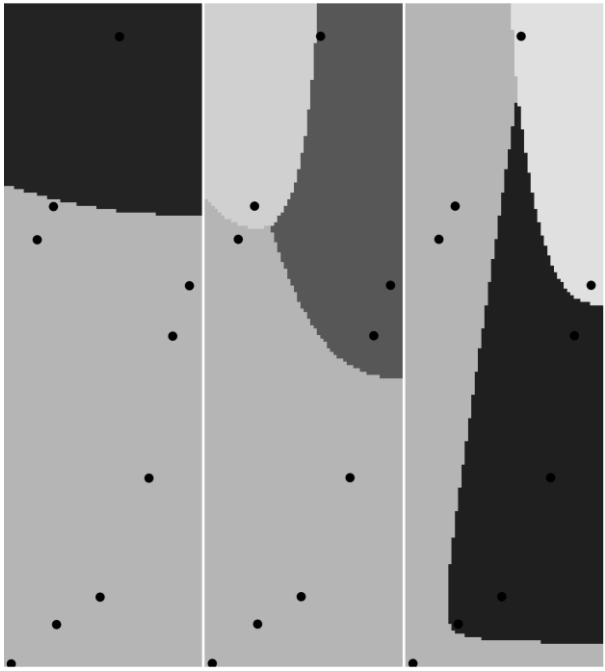
$$\sum_{j=1}^{3} \iint_{\Omega_5^j} \rho^j(x, y)dxdy \le 60, \quad \sum_{j=1}^{3} \iint_{\Omega_9^j} \rho^j(x, y)dxdy \le 120.$$

Необхідно розбити Ω на Ω_i^j так, щоб мінімізувати функціонал сумарних витрат на виробництво продукції та доставку її споживачам:

$$F\left(\Omega_1^1, ..., \Omega_9^3\right) = \sum_{j=1}^{3} \sum_{i=1}^{9} \iint_{\Omega_i^j} (c^j(x,\ y, \tau_i) + a_i^j)\rho^j(x, y)dxdy.$$

Нижче (рис. 4) зображено результат роботи програми для сітки 60×200. На рис. 5 наведено оптимальне розбиття, одержане у [1] для такої ж сітки. Як критерій зупинки всюди було застосовано умову

$$\left\|\psi^{(k)} - \psi^{(k-1)}\right\| \le 10^{-4}.$$



```
Ітерація 105:
Двоїсті змінні: (-8,78114635; -7,38697578; 0,00000113; 0,00000113; 0,00
Субградієнт: (0,07000000; -0,06000000; -20,00000000; -20,00000000; -26,
Значення цільового функціоналу: 2380,34355058204
Значення функціоналу двоїстої задачі: 2380,10948203947
Об'єми виробництва: (200,07000000; 49,94000000; 0,00000000; 0,00000000;
```

Рис. 4. Результат роботи програми для модельної задачі 2 із розмірами сітки 60×200

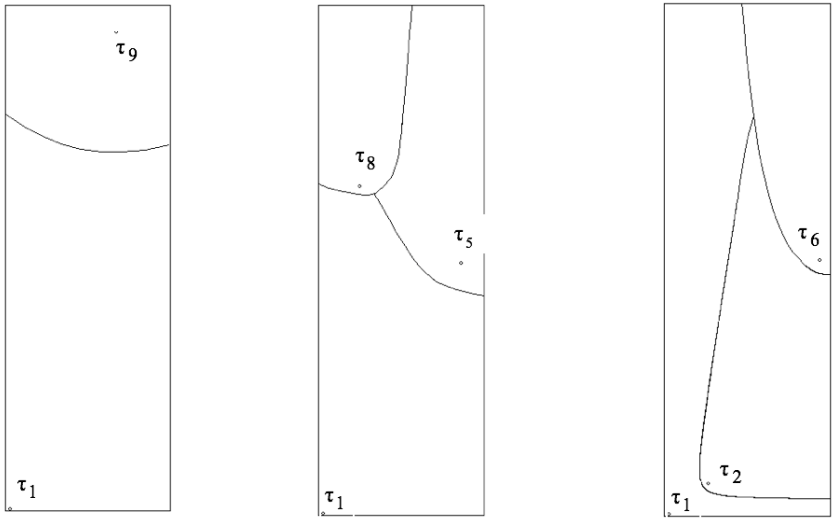


Рис. 5. Оптимальне розбиття по 1-му продукту (ліворуч), 2-му продукту (посередині) і 3-му продукту (праворуч) для модельної задачі 2, отримане у [1]

Нижче (табл. 2) наведено числові результати, одержані для різних розмірів сіток, і результати, одержані у [1].

Таблиця 2

Порівняльна таблиця результатів розв'язку модельної задачі 2

Параметр	Сітка 15×50	Сітка 30×100	Сітка 60×200	Результати, отримані у [1] (сітка 60×200)
Кількість ітерацій	110	117	105	59
Мінімальне значення прямого функціоналу	2379,35	2380,03	2380,34	2371,33
Максимальне значення двоїстого функціоналу	2379,67	2379,99	2380,11	2379,6

У ході порівняння результатів було зроблено такі загальні висновки:

1) результати порівняння підтверджують коректність роботи розробленого програмного забезпечення;

2) результати, одержані за допомогою розробленого програмного забезпечення, дають менше відхилення значень прямого і двоїстого функці-

оналів, що свідчить про те, що розроблене програмне забезпечення якісніше задовольняє обмеження задачі;

3) розроблене програмне забезпечення дозволяє одержати результат за більшу кількість ітерацій порівняно з результатами, одержаними у [1], що можна пояснити тим, що в розробленому програмному забезпеченні застосовано r-алгоритм із постійним кроковим множником, що має повільнішу збіжність, ніж r-алгоритми із прогресивнішими методами вибору крокового множника.

Висновки. У ході даного дослідження розроблено програмне забезпечення, що дозволяє користувачу одержувати розв'язки багатопродуктових задач оптимального розбиття множин із фіксованими центрами та обмеженнями у вигляді рівностей і нерівностей у зручній формі із детальною інформацією про кожну ітерацію алгоритму розв'язання задачі. Під час розробки програмного забезпечення застосовано об'єктно-орієнтований підхід, що відрізняє роботу від інших напрацювань у даній галузі. Розроблене програмне забезпечення було протестовано, а результати, одержані із його допомогою, – порівняно з результатами інших авторів, таким чином було підтверджено правильність функціонування цього програмного забезпечення.

Бібліографічні посилання

1. **Киселева, Е.М.** Непрерывные задачи оптимального разбиения множеств [Текст]: монография / Е.М. Киселева, Н.З. Шор. – К.: Наук. думка, 2005. – 564 с.
2. **Киселева, Е.М.** Модели и методы решения непрерывных задач оптимального разбиения множеств [Текст]: линейные, нелинейные, динамические задачи: монография / Е.М. Киселева, Л.С. Коряшкина. – К.: Наук. думка, 2013. – 606 с.
3. **Киселева, Е.М.** Непрерывные задачи оптимального разбиения множеств и r-алгоритмы [Текст]: монография / Е.М. Киселева, Л.С. Коряшкина. – К.: Наук. думка, 2015. –400 с.
4. **Troelsen, A.** Pro C# and the .NET 4.5 Framework, Sixth Edition [Text]/ A. Troelsen. – NY: Springer Science+Business Media, 2012. – 1487 c.

Надійшла до редколегії 16.09.2016