УДК 519.8

[15]О.М. Кісельова, П.В. Сьомчина
Дніпропетровський національний університет імені Олеся Гончара

НЕПЕРЕРВНА ЗАДАЧА ОПТИМАЛЬНОГО РОЗБИТТЯ МНОЖИНИ В УМОВАХ НЕВИЗНАЧЕНОСТІ

Представлена неперервна задача оптимального розбиття множини без обмежень при фіксованих центрах в умовах невизначеності. В якості невизначеності розглядається нечітке розбиття. Реалізовано алгоритм розв'язання задачі, наведено приклад.

Представлена непрерывная задача оптимального разбиения множества без ограничений с фиксированными центрами в условиях неопределенности. В качестве неопределенности рассматривается нечеткое разбиение. Реализовано алгоритм решения задачи, приведен пример.

The continuous set partitioning problem without constrains with fixed centers in uncertainity. As uncertainity the fuzzy partition is considered. The solving problem algorithm is cited.

Ключові слова: оптимальне розбиття множин, нечітке розбиття, невизначеність.

Введення. Вимога знаходження однозначного (чіткого) розбиття елементів множини Ω з E_N може виявитися досить грубою і жорсткою при вирішенні задач з погано або слабо структурованою вихідною інформацією, тобто задач, в яких невизначеність має нечітко -ймовірнісну природу. Ослаблення цієї вимоги здійснюється за рахунок введення в розгляд нечітких підмножин множини і відповідних їм функцій приналежності, що приймають значення з відрізка [0,1] [1, 4, 8].

Постановка задачі. Нехай Ω з E_N – обмежена, вимірна за Лебегом, опукла множина. Через $\Re_\Omega^N$ позначимо клас всіх можливих нечітких розбиттів $\Re_\Omega^N$ чіткої множини Ω на N нечітких підмножин. Введемо на мно-

жині можливих нечітких розбиттів цільовий функціонал $(F: \Re_{\Omega}^{N1} \quad R)$ у вигляді:

$$F(\Omega_1,\ldots,\Omega)=_N \quad \sum_{k=1}^{N} \int_{\Omega_k} (c(x,\tau_k)+a_k)\rho(x)dx$$

де функції $c(x, \tau_i)$ – задані, дійсні, обмежені, визначені на $\Omega \times \Omega$, вимірні по x при будь якому фіксованому $\tau_i = (\tau_i^{(1)},\ldots, \tau_i^{(n)})$ з Ω для всіх i = 1 ,..., N; ρ (x) – задана, обмежена, вимірна на Ω функція; $a_1,\ldots, a_n$ – задані невід'ємні числа.

Під неперервною однопродуктовою задачею оптимального нечіткого розбиття множини з n мірного евклідового простору E_N на нечіткі підмножини із заданим положенням «центрів» підмножин без обмежень будемо розуміти таку задачу.

Нехай заданий фіксований вектор $\tau=(\tau_1\ldots,\tau_N), \quad \tau_i \in \Omega \forall \quad i=\overline{1,N}$, який будемо інтерпретувати як вектор центрів N нечітких підмножин деякого нечіткого розбиття з $\Re_{\Omega}^{N}$. Знайти таке нечітке розбиття $\Re(\Omega) = \{\Omega_i \subseteq \Omega, \quad \forall i=\overline{1,N}\}$ множини Ω з безлічі розбиттів $\Re_{\Omega}^{N}$, яке є рішенням наступної задачі оптимального нечіткого розбиття:

$$F(\Omega_1,\ldots, \Omega_N)=\sum_{k=1}^{N} \int_{\Omega_k} (c(x,\tau_k)+a_k)\rho(x)dx \to \min_{(\Omega_1,\ldots,\Omega_N)\in \Re_{\Omega}^{N}}$$

Тут під «мінімізацією» будемо розуміти вибір нечіткого розбиття, якому відповідає в деякому сенсі найкраще нечітке значення цільового функціоналу.

Для того, щоб мати можливість ідентифікувати нечітке розбиття $(\Omega_1,\ldots, \Omega_N)$ множини Ω, потрібно знати вектор функцію приналежності виду:

$$\mu(x)=(\mu_1(x),\ldots,\mu_N(x)), \quad x \in \Omega$$

Переформулюємо задачу в термінах функцій приналежності.

Знайти вектор-функцію $\mu^*(x) = (\mu_1^*(x),\ldots,\mu_N^*(x))$ таку, що:

$$I(\mu_1(x),\ldots,\mu_N(x))=\int_{\Omega} \sum_{k=1}^{N} (\mu_k(x))^m (c(x,\tau_k)+a_k)\rho(x)dx \to \min_{\mu(x)\in M}$$

,

де

$$M = \{\mu(x) \quad (\mu_1(x), \quad ,\mu \quad (x)):$$
$$0 \le \mu_k(x) \quad \le 1 \quad \forall x \in \Omega, \ k=1,...,N,$$
$$\sum_{k=1}^{N} \mu_k(x) \quad = 1 \quad \forall x \in \Omega\}$$

m – параметр, який називається експоненціальним вагою.

Для інтерпретації отриманих результатів, а саме віднесення кожного вузла сітки до якої -небудь підмножині, або до нечіткого кордону, введемо наступне допоміжне поняття ступеня недовіри СН.

Ступінь недовіри $CH \in [0;1]$ – це мінімальне значення функції приналежності деякої нечіткої множини, при якому дана точка може бути з упевненістю віднесена до цієї нечіткої множини, іншими словами, при якому ми вважаємо дану точку належить даній множині. Цей показник можна також інтерпретувати як мінімальну ступінь достовірності факту «дана точка х належить даній підмножині», достатню для його ухвалення. Ступінь недовіри – це наші вимоги щодо чіткості розбиття. Чим нижче вона буде, тим більш нечітким буде розбиття, тим ширше будуть області нечіткого кордону. Очевидно, що при СН = 0 і отримуємо чітке розбиття, а при СН = 1 – максимально нечітке розбиття, при якому до певного підмножині буде віднесено лише його центр і точки ядра, якщо такі з'являться [5,6].

Нижче представлений алгоритм розв'язання пропонованої задачі [2, 3].

Результати роботи. Тестування ефективності алгоритма і роботи програмного комплексу, який було розроблено у середовищі Matlab [7], відбувалося шляхом порівняння з відомими даними.

Наведені результати чисельних експериментів по нечіткому розбиттю одиничного квадрату з E_2 з евклідовою метрикою за допомогою даного алгоритму з сіткою 50 х 50.

На рисунках, наведених нижче, центр підмножини, заданий початковими даними, позначений «+».

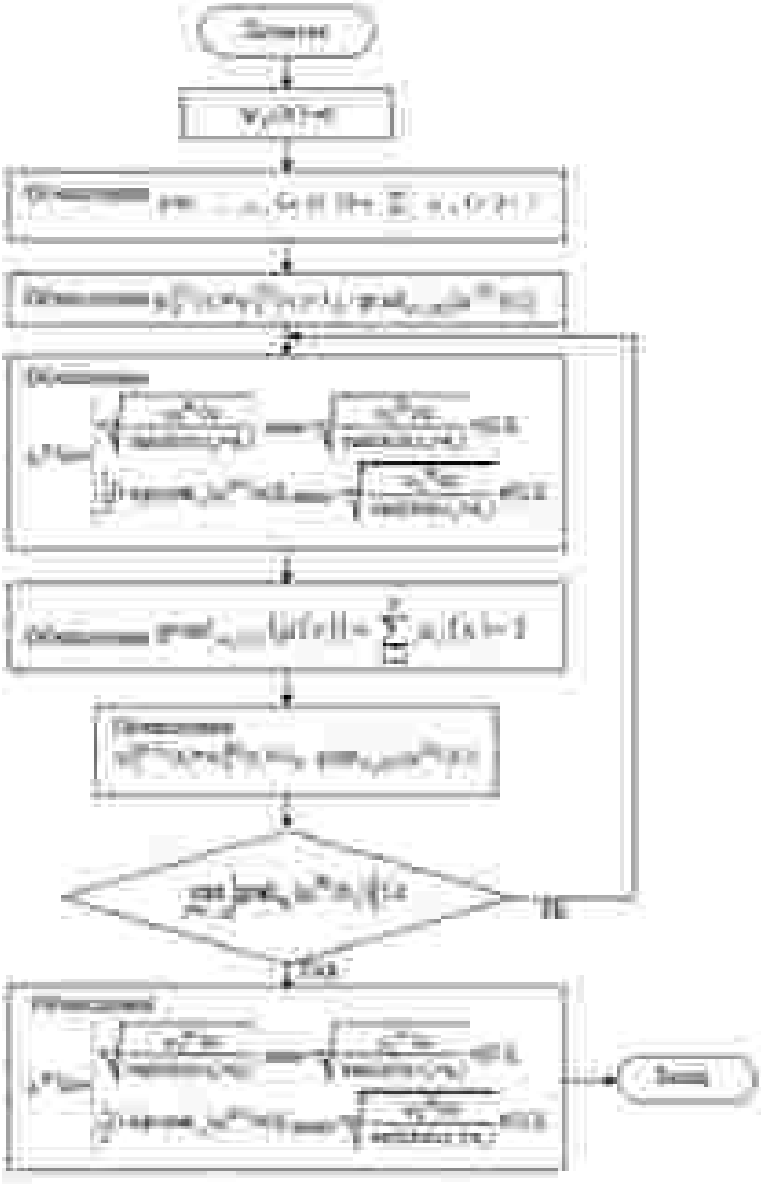


Рис. 1. Алгоритм розв'язання неперервної нечіткої задачі оптимального розбиття множини

Приклад 1. Розбиття одиничного квадрата на дві нечітких підмножини при рівних між собою a_i, i = 1, 2.

Таблиця 1.

Початкові дані для розбиття на дві підмножини

№ підмножини	Центри		a_i	Точність
	x	y		
1	0,25	0,5	0	0,001
2	0,75	0,5	0	

Таблиця 2.

Результати розбиття на дві підмножини

Значення $\max_{j=1,\dots,q} \left\| \operatorname{grad} \psi_0^j(\mu^{(k)}(x)) \right\|$	Значення функціонала	Кількість ітерацій
0,0010	0,1884	967

Графічна ілюстрація:

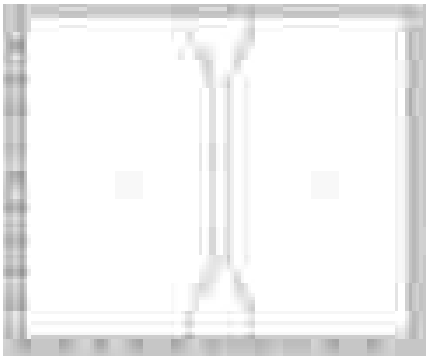


Рис. 2. Нечітке розбиття на дві підмножини (CH=0,55)

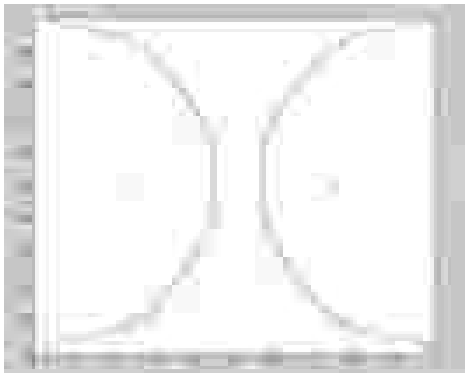


Рис. 3. Нечітке розбиття на дві підмножини (CH=0,65)

Як видно з рисунків 2, 3, при підвищенні ступеня недовіри рамки нечіткого кордону розширюються, форма чітких областей наближається до кола. Це можна пояснити логічно: чим ближче вузол до центру, тим з бі-

льшою ймовірністю його можна віднести до даної підмножини. Ця думка підтверджується тим, що знайдені значення функцій приналежності дійсно збільшуються в міру наближення до центру.

Приклад 2. Розбиття одиничного квадрата на два нечітких підмножини при нерівних між собою a_i, i = 1, 2.

Таблиця 3.

Початкові дані для розбиття на дві підмножини

№ підмножини	Центри		a_i	Точність
	x	y		
1	0,25	0,5		0
2	0,75	0,5	,2	0 0,001

Таблиця 4.

Результати розбиття на дві підмножини

Значення $\max_{j=1,\ldots,q} \left\| \operatorname{grad} \psi_{j_0}(\mu^{(k)}(x)) \right\|$	Значення функціонала	Кількість ітерацій
0,00010	0,2349	1211

Графічна ілюстрація:

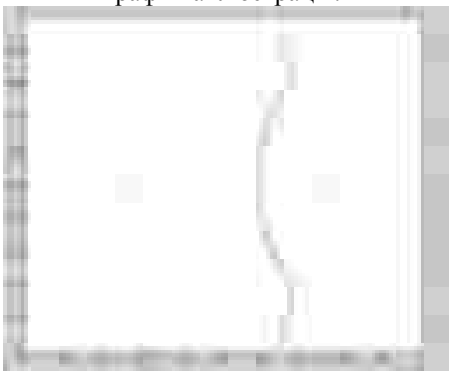


Рис. 4. Нечітке розбиття на дві підмножини (CH=0,55)

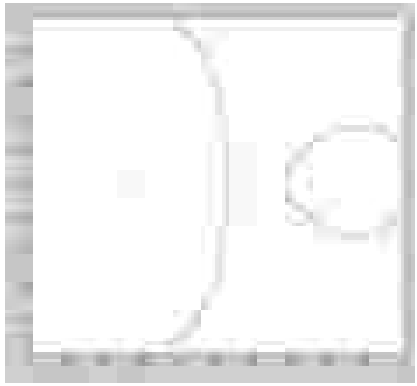


Рис. 5. Нечітке розбиття на дві підмножини (CH=0,65)

Результати, отримані для чіткого варіанту розбиття, показують, що в цьому випадку межа між підмножинами має вигляд гіперболи. Як бачимо з результатів, на нашому прикладі це положення підтверджується і в нечіткому варіанті. Цікавими є видозміни форми областей підмножин. Тепер на їх конфігурацію впливає дві криві: коло та гіпербола одночасно.

Приклад 3. Розбиття одиничного квадрата на три нечіткі підмножини при рівних між собою a_i, i = 1, 2,3.

Таблиця 5.

Початкові дані для розбиття на три підмножини

№ підмножини	Центри		a_i	Точність
	x	y		
1	0,25	0,5	0	0,001
2	0,75	0,5	0	
3	0,5	0,75	0	

Таблиця 6.

Результати розбиття на три підмножини

Значення $\max_{j=1,\dots,q} \left\| \operatorname{grad} \psi_{j_0}(\mu^{(k)}(x)) \right\|$	Значення функціонала	Кількість ітерацій
0,0010	0,1244	693

Графічна ілюстрація:

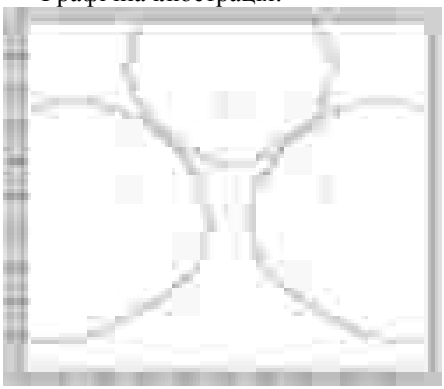


Рис. 6. Нечітке розбиття на три підмножини (CH=0,45)

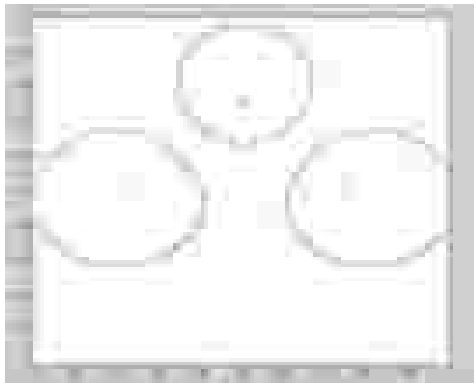


Рис. 7. Нечітке розбиття на три підмножини (CH=0,65)

Приклад 4. Розбиття одиничного квадрата на чотири нечіткі підмножини при рівних між собою a_i, i = 1, 2, 3, 4.

Таблиця 7.

Початкові дані для розбиття на чотири підмножини

№ підмножини	Центри		a_i	Точність
	x	y		
1	0.25	0.25	0	0,001
2	0.75	0.25	0	
3	0.25	0.75	0	
4	0.75	0.75	0	

Таблиця 8.

Результати розбиття начотири підмножини

Значення $\max_{j=1,\dots,q} \left\| \operatorname{grad} \psi_{i_0}(\mu^{(k)}(x)) \right\|$	Значення функціонала	Кількість ітерацій
0.0010	0.0819	495

Графічна ілюстрація:

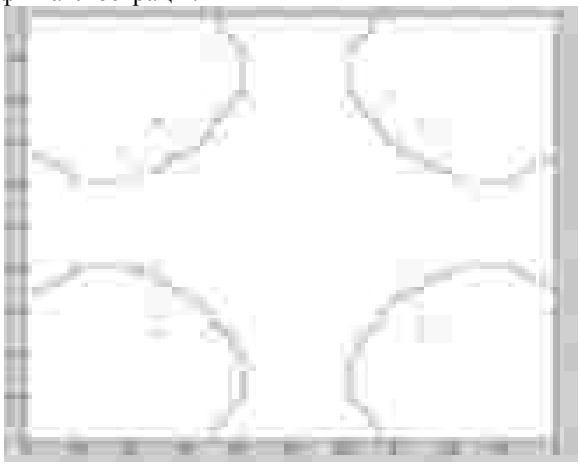


Рис. 8. Нечітке розбиття на чотири підмножини (CH=0,4)

Висновки. Реалізовано алгоритм рішення неперервної нечіткої задачі оптимального розбиття множини без обмежень при фіксованих центрах.

Для різних значень коефіцієнтів a_i проведені обчислювальні експерименти.

При підвищенні ступеня недовіри рамки нечіткого кордону розширюються, форма чітких областей наближається до кола.

Результати, отримані для чіткого варіанту розбиття, показують, що в цьому випадку межа між підмножинами має вигляд гіперболи. Як бачимо з результатів, на прикладі 2 це положення підтверджується і в нечіткому варіанті.

Розбиття, представлене на рис. 4 (низький ступінь недовіри), повторює результат, отриманий для випадку чіткого розбиття. Відзначимо, що це досить вагомий результат, адже ні метод рішення, ні сам функціонал, який є нелінійним, не повторюють ідейно задання і методів для чіткого розбиття. При підвищенні ступеня недовіри області підмножин деформуються.

При досить низькому ступеню недовіри, першими в область нечіткого кордону потрапили точки, що знаходяться між всіма множинами відразу.

Цікавим є факт зменшення числа ітерацій при збільшенні кількості підмножин.

Що стосується модифікації моделі, інтерес представляє задача нечіткого розбиття множини без обмежень з відшукання координат ценрів підмножин.

Бібліографічні посилання

1. **Бочарников В.П.** Fuzzy-технология: Математические основы. Практика моделирования в экономике. / В.П. Бочарников – СПб. РАИ. – 2001. – 328 с.
2. **Васильев Ф.П.** Лекции по методам решения экстремальных задач. / Ф.П. Васильев – М.: МГУ. – 1974. – 376 с.
3. **Васильев Ф.П.** Методы решения экстремальных задач. / Ф.П. Васильев – М., 1981. – 400с.
4. **Заде Л.А.** Понятие лингвистической переменной и его применение к принятию приближённых решений. / Л.А. Заде – М., 1976. – 167 с.
5. **Канторович Л.В.** Функциональный анализ. / Л.В. Канторович, Г.П. Акилов – М., 1977. – 742с.
6. **Киселева Е.М.** Математические методы и алгоритмы решения непрерывных задач оптимального разбиения множеств и их приложения: Автореф. дис….д-ра физ.-мат.наук: 01.01.09 / Е.М. Киселева – К. – 1991. – 33 с.
7. **Леоненков А.В.** Нечёткое моделирование в среде MATLAB и fuzzy TECH. / А.В. Леоненков – СПб., 2003. – 736 с.
8. **Орловский С.А.** Проблемы принятия решений при нечёткой исходной информации. / С.А. Орловский – М., 1981. – 206 с.

Надійшла до редколегії 07.06.11

УДК 519.8

[16]И.В. Козин
Запорожский национальный университет

МЕРЫ СИММЕТРИИ И ЗАДАЧИ ОПТИМИЗАЦИИ

Розглядається проблема оцінювання ступеня симетрії множини. Показано, що у простіших випадках задача відшукання множин з мінімальною або максимальною мірою симетрії зводиться до задачі цілочисельного лінійного програмування.Запропоновано найпростіші методи пошуку розв'язків задач о симетричних підмножинах.

Рассматривается проблема оценивания степени симметрии множества. Показано, что в простейших случаях задача отыскания множеств с минимальной или максимальной мерой симметрии сводится к задаче целочисленного линейного программирования. Предложены простейшие методы поиска решений задач о симметричных подмножествах.

The problem of estimation of set symmetry is considered. It was shown that in the simplest cases the problem of finding the sets with minimal or maximal measure of symmetry reduces to the integer linear programming problem. The simplest methods of finding solutions of symmetry subsets problems are proposed.

Ключевые слова: множество, симметрия, орбита, мера симметрии.

Введение. В ряде задач принятия решений приходится сталкиваться с неформальными критериями. Достаточно часто ЛПР, наряду с обычными числовыми критериями, накладывает на принимаемое решение еще неформальные ограничения, которые описываются словами «красивое», «симметричное», и т.д. С таким подходом часто приходится сталкиваться в прикладных задачах размещения, задачах покрытия. Иногда критерии симметрии очень тесно связаны с оптимальным решением задачи. Достаточно вспомнить такие классические оптимизационные задачи, как изопериметрическая задача, задача о теле максимального объема при заданной площади поверхности. В книге Саати [1] присутствует замечательное выражение «оптимизировать – значит симметризировать». То, что неформально принято считать «красивым» решением, означает инвариантность

решения относительно той или иной группы симметрии. Таким образом, решение задачи по критерию «красоты» сводится к отысканию решения, инвариантного относительно действия той или иной группы преобразований.

Условие инвариантности относительно той или иной группы симметрии может значительно упростить поиск оптимального решения [2]. Однако, как правило, идет речь не о «строгой» симметрии, а об определенном приближении к симметричному решению, поиски своеобразного компромисса между оптимумом и симметричностью решения. В этом случае задача может значительно усложниться и потребовать разработки принципиально новых методик и алгоритмов.

Постановка проблемы. Рассмотрим конечное множество X. Преобразованием множества X называется любое взаимнооднозначное отображение $F: X \to X$ этого множества на себя. Группа S всех преобразований множества X изоморфна симметрической группе подстановок S_n, где n – число элементов множества X. Всякую подгруппу A группы S будем называть группой симметрий множества X.

Определение 1. Для любого элемента $x \in X$, орбитой этого элемента при действии группы симметрии S называется множество $O_A(x) = \{y : y = g(x), g \in A\}$. Орбитой подмножества $Y \subseteq X$ будем называть объединение орбит всех его элементов и обозначать $O_A(Y)$. Имеет место очевидное утверждение

Теорема 1. Орбиты любых двух элементов множества X либо не пересекаются, либо совпадают.

Кроме того, справедлива следующая

Теорема 2. Для любого подмножества $Y \subseteq X$ и для любой группы симметрии $A \subseteq S$ имеет место включение $Y \subseteq O_S(Y)$.

Доказательство. Тождественной отображение является элементом любой группы симметрии. Отсюда вытекает утверждение теоремы. ■ [1)]

Подмножество $Y \subseteq X$ называется инвариантным (или вполне симметричным) относительно действия группы $A \subseteq S$, если $\forall g \in A \quad gY(\ ) = Y$.

Отметим очевидное определяющее свойство инвариантных множеств. Множество $Y \subseteq X$ инвариантно относительно действия группы A в том и

только в том случае, когда оно совпадает со своей орбитой, то есть $Y = O_A(Y)$.

Если подмножество не является вполне симметричным, то возникает вопрос: каким образом можно оценить его отклонение от симметрии. Другими словами, какова мера симметрии произвольного подмножества множества X.

Меры симметрии подмножеств. *Определение 2*. Внешней *мерой симметрии* $\mu_A(Y)$ непустого подмножества $Y \subseteq X$, будем называть отношение числа элементов этого множества к числу элементов его орбиты, то есть $\mu_A(Y) = \frac{|Y|}{|O_A(Y)|}$. Для пустого подмножества будем полагать по определению $\mu_A(\varnothing) = 1$.

Очевидно, что для любого подмножества $Y \subseteq X$ справедливо неравенство $\frac{|Y|}{|X|} \le \mu_A(Y) \le 1$. Инвариантные относительно действия группы симметрии A множества – это множества с максимальной мерой симметрии, то есть множества, для которых $\mu_A(Y) = 1$.

)Здесь и далее знаком ■ будет обозначаться конец доказательства

Определение 3. Вполне асимметричным по отношению к группе симметрии A будем называть такое подмножество $Y \subseteq X$, каждый элемент $y \in Y$ которого обладает следующим свойством: $Y \cap O_A(y) = \{y\}$. То есть для любого элемента $y \in Y$ и любого преобразования $g \in A$, отличного от тождественного, $g(y) \notin Y$.

Определение 4. Будем называть *ядром симметрии* или S_A-ядром множества Y любое его максимальное по включению вполне асимметричное подмножество.

Теорема 3. Пусть Y – произвольное подмножество множества X, A – подгруппа S. Мощности всех S_A-ядер множества Y одинаковы.

Доказательство. Так как орбиты любых двух элементов множества X или не пересекаются, или совпадают, то множество X оказывается разби-

тым на классы эквивалентности. Два элемента эквивалентны (принадлежат одному классу) в том и только в том случае, когда один из них может быть получен из другого с помощью преобразования из группы A. Ядро симметрии содержит ровно по одному элементу из каждого класса эквивалентности. Следовательно, его мощность равна количеству классов эквивалентности. ■

Возможны и другие способы описания меры симметрии. Естественное требование к мере симметрии – следующее. Для инвариантных относительно действия группы S подмножеств мера симметрии должна быть максимальной.

Пусть на конечном множестве X действует группа преобразований A. Для каждого подмножества $Y \subseteq X$ определим множество инвариантных элементов следующим образом:

$$I_A(Y) = \{x \in Y \mid \forall g \in A \; gx \in Y\}$$

*Определение 5. В*нутренней мерой симметрии непустого подмножества $Y \subseteq X$ относительно действия группы A будем называть число

$$\nu_A(Y) = \frac{|I_A(Y)|}{|Y|}$$

Для пустого множества по определению будем полагать $\nu_A(\varnothing) = 1$.

Для любого подмножества $Y \subseteq X$ очевидны следующие неравенства:

$$0 \le \nu_A(Y) \le 1,$$

Причем, для инвариантных относительно действия группы A подмножеств мера симметрии равна 1. Очевидна следующая

Теорема 4. Непустое подмножество $Y \subseteq X$ является вполне антисимметричным относительно нетривиальной группы преобразований в том и только в том случае, когда его внутренняя мера симметрии равна 0.

Простейшие экстремальные задачи о симметричных подмножествах

Простейшая оптимизационная задача с критерием симметрии формулируется следующим образом: для заданного множества X и его группы симметрии A найти множество заданной мощности m с максимальной внешней мерой симметрии.

Как уже отмечалось ранее, множество X разбивается на попарно непересекающиеся классы, каждый из которых представляет собой орбиту некоторого элемента из X. Перенумеруем эти классы и далее будем полагать $X=O_1\cup O_2\cup\ldots\cup O_s$. Для каждого класса определим величину $k_{ii}=|O_i|,\ i=1,2,\ldots,s$ - мощность этого класса. С каждым множеством $Y\subseteq X$ свяжем бинарный вектор $y=(y_1,y_2,\ldots,y_s)$, где $y_i=\begin{cases}1, \text{ если } O_{iA}\subseteq O(Y)\\ 0, \text{ если } O_{iA}\not\subset O(Y)\end{cases}$.

При заданном значении m задача отыскания множества Y мощности m с максимальной внешней мерой симметрии сводится к решению следующей оптимизационной задачи:

$$\begin{aligned} &k_1\cdot y_1+k_2\cdot y_2+\ldots+k_{ss}\cdot y \ \to \min\\ &k_1\cdot y_1+k_2\cdot y_2+\ldots+k_{ss}\cdot y \ \geq m,\\ &y_i\in\{0,1\},i=1,2,\ldots,s \end{aligned} \qquad (1)$$

Задача (1) является частным случаем классической задачи о ранце [3,4]. Как следствие получаем, что задача о максимальном симметричном подмножестве NP-полна.

Быстрый эвристический алгоритм решения этой задачи выглядит следующим образом. Упорядочиваем и перенумеровываем числа k_i, $i=1,2,\ldots,s$ по возрастанию и выбираем минимальный номер t, для которого $k_1+k_2+\ldots+k_t\geq m$. В качестве решения берем множество $Y=O_1\cup O_2\cup\ldots\cup O_{t-1}\cup B$, где множество $B\subseteq O_t$ и содержит произвольные $m-(k_1+k_2+\ldots+k_{t-1})$ элементов множества O_t.

Следующая задача – задача отыскания подмножества заданной мощности m с минимальной внешней мерой симметрии имеет полиномиальную трудоемкость. Алгоритм поиска оптимального решения задачи следующий:

Если $m \geq s$, то выбираем по одному элементу из каждого из s классов эквивалентности, а остальные $m-s$ выбираются произвольно. Мера симметрии полученного множества из m элементов будет равна $\frac{m}{n}$. Если $m<s$, то упорядочим классы $O_1, O_2, \ldots, O_s$ по убыванию их мощности. Выберем по одному элементу из первых m классов. Это и будет требуемое решение с минимальной мерой симметрии.

Для внутренней меры симметрии рассмотрим простейшую оптимизационную задачу с критерием симметрии: для заданного множества X и его группы симметрии A найти подмножество $Y \subseteq X$ заданной мощности m с максимальной мерой симметрии $\nu_A(Y)$.

Как и прежде разобьем множество X на классы $\{O_i\}_{i=1}^{s}$, каждый из которых представляет собой орбиту некоторого элемента из X. Перенумеруем эти классы и будем полагать $X = O_1 \cup O_2 \cup \ldots \cup O_s$. Для каждого класса определим величину $k_i = |O_i|$ - мощность этого класса. С каждым множеством $Y \subseteq X$ свяжем бинарный вектор $y=(y_1, y_2, \ldots, y_s)$, где $y_i = \begin{cases} 1, & \text{если } O_{iA} \subseteq O(Y) \\ 0, & \text{если } O_{iA} \not\subset O(Y) \end{cases}$.

При заданном значении m задача отыскания множества Y мощности m с максимальной внутренней мерой симметрии сводится к решению следующей оптимизационной задачи:

$$\begin{aligned} & k_1 \cdot y_1 + k_2 \cdot y_2 + \ldots + k_s \cdot y \rightarrow \max \\ & k_1 \cdot y_1 + k_2 \cdot y_2 + \ldots + k_s \cdot y \leq m, \\ & y_i \in \{0,1\}, \quad i = 1,2,\ldots,s \end{aligned} \qquad (2)$$

Эта задача относится к числу NP – полных задач.

Заключение. В работе показано, что задачи, связанные с определением меры симметрии в простейших случаях могут быть сведены к известным оптимизационным задачам. Даже в простейших случаях задачи такого вида могут иметь достаточно высокую трудоемкость. Более сложные

задачи появляются, когда на множестве X задана структура, определенная некоторым отношением [5]. Задача поиска симметричных подмножеств в этом случае состоит в отыскании максимально симметричных (или асимметричных) подмножеств, на которых сохранена заданная структура.

Библиографические ссылки

1. **Саати Т.** Принятие решений. Метод анализа иерархий / Т. Саати. – М. : Радио и связь, 1993. – 278 с.
2. **Миллер У.** Симметрия и разделение переменных / У. Миллер. – М. : Мир, 1981. – 342 с.
3. **Сачков В. Н.** Введение в комбинаторные методы дискретной математики / В. Н. Сачков. – М. : Наука, 1982. – 384 с.
4. **Сигал И. Ч.** Введение в прикладное дискретное программирование / И. Ч. Сигал, А. П. Иванова. – М. : Физматлит, 2002. – 240 с.
5. **И. В. Козин.** Фрагментарный алгоритм для задачи симметричного размещения. // Радиоэлектроника, информатика, управление, №1, 2005. С.76–83.

Надійшла до редколегії 15.06.11

УДК 519.1

[1]Л. М. Колєчкіна, О.А. Родіонова
ВНЗ Укоопспілки «Полтавський університет економіки і торгівлі»

ПІДХІД ДО РОЗВ'ЯЗУВАННЯ ЕКСТРЕМАЛЬНИХ ЗАДАЧ НА КОМБІНАТОРНИХ КОНФІГУРАЦІЯХ

Розглядається екстремальна задача на комбінаторній конфігурації розміщень. Описується метод локалізації значення цільової функції на основі теорії графів з урахуванням структури множини розміщень. Розглянуто приклад реалізації методу та описані параметри числових експериментів.

Рассматривается экстремальная задача на комбинаторной конфигурации размещений. Описывается метод локализации значения целевой функции на основании теории графов с учетом структуры множества размещений. Рассмотрено пример реализации метода и описаны параметры числовых экспериментов.

An extreme task is examined on combinatorial configuration of permutation. The method of objective function's value localization is described on the basis of the graphs theory taking into account the structure combinatorial set. The example of method's realization is considered and the numerical experiments' parameters are described.

Ключові слова: комбінаторна конфігурація, екстремальні задачі на комбінаторних конфігураціях, граф многогранника розміщень, метод локалізації значення функції.

Вступ. Екстремальні задачі є досить поширеними в останній час, що зумовлюється можливістю їх використання для моделювання прикладних задач та проблем. Цільова функція визначається на певній комбінаторній конфігурації, властивості якої відповідають умовам поставленої задачі. Оскільки спектр застосування моделей комбінаторної оптимізації зростає, то й самі задачі ускладнюються, адже не всі задачі на комбінаторних конфігураціях є достатньо вивченими та дослідженими. Дана ситуація підштовхує до пошуку нових методів та модифікації вже існуючих з метою оптимізації процесу обробки даних. Одним зі шляхів розвитку методів розв'язування екстремальних задач є застосування теорії графів.

Багато типів комбінаторних задач за допомогою теорії графів дуже добре описуються. Однак перевагою є не лише наочність графічного представлення, а й можливість одержання нових підходів до розв'язування. Слід зауважити, що проблеми оцінки числа ітерацій та ефективності алгоритмів симплексного типу в задачах лінійного програмування мають тісний зв'язок з метричними характеристиками графів.

Дана стаття є продовженням досліджень [1–3] і дає можливість розв'язати складнішу задачу локалізації значень лінійної функції, що полягає в знаходженні множини елементів конфігурації, при яких це значення досягається. Слід зазначити, що дана задача є під задачею для загальної екстремальної задачі при умові багатокритеріальності та додаткових обмеженнях.

Постановка задачі. Введемо поняття комбінаторної конфігурації з метою подальшого викладу матеріалу. Дамо визначення поняттю конфігурація. Нехай задані множини $X = \{1,2,...,m\}$, $Y = \{y_1, y_2,..., y_n\}$, і нехай на Y заданий строгий лінійний порядок: $Y = y_1 < y_2 < ... < y_n$. Відображення $\varphi : X \to Y$, що задовольняє деякий комплекс обмежень Λ, будемо називати конфігурацією. Комплекс обмежень Λ, що задовольняє відображення φ, визначає деякий клас конфігурації, який відповідає умовам комбінаторних конструкцій в задачі [3].

Екстремальна комбінаторна задача в загальному вигляді формулюється наступним чином: маємо множину з n елементів, на ній задається комбінаторна конфігурація $A = \{\sigma_i\} = \{a_1, a_2,..., a_k\}$. На множині A задається функція $f(x)$. Необхідно знайти екстремум функції $f(x)$ та елементи множини A, в яких цей екстремум досягається. Розглянемо загальну схему алгоритму розв'язування задачі локалізації значення функції на довільній комбінаторній конфігурації.

Суть алгоритму полягає у поступовому заглибленні до структури графа, розбитого на підграфи, причому відкидаються ті підграфи, які не можуть містити розв'язку. Такий підхід значно скорочує кількість варіантів перебору та оптимізує процес розв'язування задачі.

Алгоритм локалізації значення лінійної функції на комбінаторній конфігурації розміщень

Крок 1. Визначаємо кількість вершин графа комбінаторної конфігурації [1]. Для розміщень – $\frac{k!}{(k-n)!}$.

Крок 2. Визначаємо кількість крайніх точок у загальному графі кожного з підграфів.

Крок 3. Обчислюємо максимальне та мінімальне значення заданої цільової функції в крайніх точках підграфів загального графа:

$$f(x)_{\max} = c_1 x_{k-n+1} + c_2 x_{k-n+2} + \ldots + c_{n-1} x_{k-1} + c_n x_k;$$
$$f(x)_{\min} = c_n x_1 + c_{n-1} x_2 + \ldots + c_2 x_{n-k-1} + c_1 x_{n-k}$$

Крок 4. Будуємо структурний граф, що відображає лише крайні вершини підграфа.

Крок 5. Визначаємо значення цільової функції в початковій та кінцевій вершинах підграфів.

Крок 6. Визначаємо множину структурних підграфів, для яких задане значення цільової функції може міститись між його крайніми точками.

Крок 7. Формуємо множину точок -елементів конфігурації, для яких $f(x^*) = y$ [1-3]. Якщо всі точки знайдено (тобто серед множини підграфів немає таких, для яких би виконувалась умова кроку 6), то задача розв'язана. Виводимо елементи конфігурації. Інакше – переходимо на крок 8.

Крок 8. У підграфі, визначеному на кроці 6, фіксуємо останню координату в точці – вершині підграфа. Здійснюємо перехід на крок 1.

Описаний вище алгоритм реалізовано програмно для конфігурації розміщень. Розглянемо числовий приклад його роботи.

Приклад 1.

Розглянемо наступну комбінаторну задачу.

Дано: а) функція $f(x) = \sum_{i=1}^{k} c_i x_i$, де $k = 5$, тоді $f(x) = 2x_1 + 5x_2 + 3x_3 + 6x_4 + 4x_5$;

б) визначене значення функції $f(x^*) = 109$;

в) елементи множини розміщень (1 2 3 4 5 6 7).

Знайти: точки – елементи конфігурації розміщень, в яких досягається задане значення цільової функції.

Розв'язання: для впорядкування елементів цільової функції визначимо відображення π : $c = (2,5,3,6,4)$, $\varepsilon_{i^0} = (2,3,4,5,6)$, $\varepsilon_{i^0} = c_{\pi^{-1}}$

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 4 & 2 & 5 & 3 \end{pmatrix}, \quad \pi^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 5 & 2 & 4 \end{pmatrix}.$$

Будуємо граф, який містить $\frac{7!}{(7-5)!} = 2520$ вершин. Визначимо мінімальне та максимальне значення цільової функції:

$$\max := 2 \cdot 3 + 3 \cdot 4 + 4 \cdot 5 + 5 \cdot 6 + 6 \cdot 7 = 110,$$

$$\min := 2 \cdot 5 + 3 \cdot 4 + 4 \cdot 3 + 5 \cdot 2 + 6 \cdot 1 = 50.$$

Згідно алгоритму, визначимо крайні точки підграфів та знайдемо значення цільової функції в них.

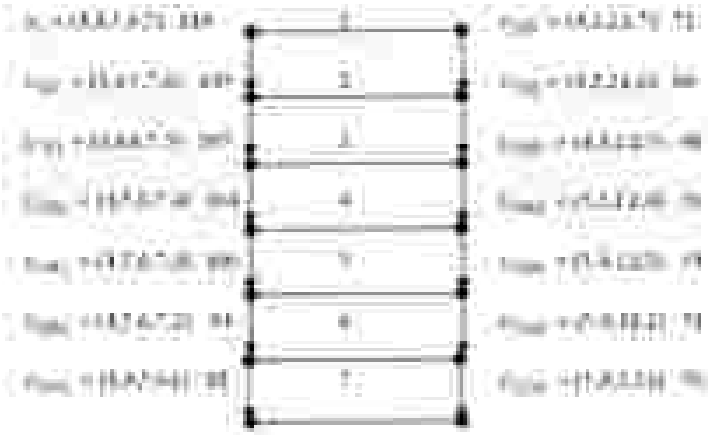


Рис. 1. Структурний граф многогранника розміщень

Відкидаємо п'ять нижніх рівнів підграфа, оскільки в них не може бути досягнуто встановлене значення цільової функції, що рівне 109. Розглянемо наступний підграф, зафіксувавши останню координату $x_5 = 6$, що належить усім вершинам підграфа (таблиця 1).

Таблиця 1

Значення функції у крайніх точках підграфів II рівня

x_1	x_2	x_3	x_4	$f(x)$	x_1	x_2	x_3	x_4	$f(x)$
3	4	5	7	**109**	3	2	1	7	**87**
3	4	7	5	107	3	2	1	5	77
3	5	7	4	105	3	2	1	4	72
4	5	7	3	102	4	2	1	3	69
4	5	7	2	97	4	3	1	2	67
4	5	7	1	92	4	3	2	1	66

Тепер зафіксуємо значення наступної координати $x_4 = 7$ та проведемо обчислення для підграфа наступного рівня, де знаходяться вершини (таблиця 2).

Таблиця 2

Значення функції у крайніх точках підграфів III рівня

x_1	x_2	x_3	$f(x)$	x_1	x_2	x_3	$f(x)$
3	4	5	**109**	2	1	5	**98**
3	5	4	108	2	1	4	94
4	5	3	106	2	1	3	90
4	5	2	102	3	1	2	88
4	5	1	98	3	2	1	87

Зафіксуємо значення $x_3 = 5$ та проведемо відповідні розрахунки. Аналогічно визначимо значення координат на наступних рівнях, одержимо $x_2 = 4$.

Таким чином, з урахуванням відображення функції, одержуємо $x^* = (3,7,4,6,5)$, для якої $f(x^*) = 109$.

Провівши аналогічні розрахунки для першого підграфа (рис. 1), одержимо наступні точки, що будуть розв'язками задачі.

Таблиця 3

Координати вершин, що є розв'язками задачі

x_1	x_2	x_3	x_4	x_5	$f(x)$
3	7	4	6	5	109
4	6	3	7	5	109
3	6	5	7	4	109
3	5	4	7	6	109

Отже, одержано чотири точки, що задовольняють умову задачі, значно скоротивши кількість варіантів перебору.

Таблиця 4

Результати числових експериментів

№	Розмірність функції n	Кількість елементів множини k	Кількість підграфів	Кількість точок	Час	Значення функції	Кількість розвязків
1	5	7	34	117	0,66	107	9
2	5	7	5	18	0,0001	110	1
3	5	7	15	54	0,22	51	4
4	5	7	57	181	1,15	105	16
5	5	6	120	274	2,19	79	26
6	5	6	166	366	2,02	74	20
7	5	6	194	413	3,57	69	34
8	5	6	33	83	0,55	53	8
9	4	6	64	178	1,26	53	17
10	4	6	10	31	0,1	67	3
11	4	6	29	87	0,49	62	6
12	4	6	54	154	1,05	57	12
13	4	7	32	124	0,66	76	7

Продовження таблиці 4

№	Розмірність функції n	Кількість елементів множини k	Кількість підграфів	Кількість точок	Час	Значення функції	Кількість розвязків
14	4	7	85	317	1,98	66	19
15	4	7	108	399	2,59	61	32
16	4	7	32	124	0,66	36	7
17	3	6	6	20	0,0001	46	2
18	3	6	14	48	0,22	40	5
19	3	7	28	123	0,66	35	11
20	3	7	15	65	0,27	45	5
21	3	7	25	103	0,55	41	10
22	3	7	9	39	0,11	19	3

Вище вказаний алгоритм програмно реалізований та проведені числові експерименти. Зазначимо, що розрахунки показали ефективність даного алгоритму для розглядуваного класу задач. Числові експерименти об'єднаємо і представимо у таблиці 4.

Аналізуючи дані таблиці 4, можна побудувати наступну графічну залежність часу роботи програми від кількості розв'язків задачі (рис. 2).

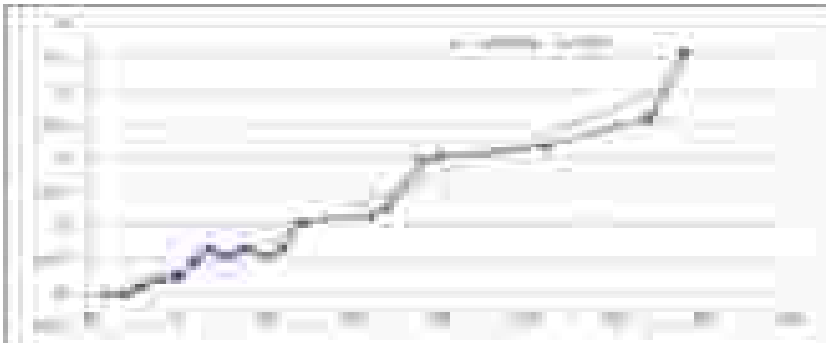


Рис. 2. Графік залежності часу роботи програми від кількості розв'язків

Висновки. Сформульована задача є актуальною у світлі теоретичних та прикладних досліджень та використовується як допоміжний алгоритм у загальному методі розв'язування екстремальних задач при умові багатокритеріальності. Застосування алгоритму локалізації значення функції на комбінаторних конфігураціях дає можливість знайти шлях розв'язування поставленої задачі. Подальші дослідження плануються у сфері розробки загального алгоритму на різних комбінаторних конфігураціях.

Бібліографічні посилання

1. **Донец Г. А.** Локализация значения линейной функции заданной на перестановках / Г. А. Донец, Л. Н. Колечкина // Радиоэлектроника и информатика. – 2009. – № 1. – С. 76-81.
2. **Колечкина Л. Н.** Обоснование структурированного метода локализации значения линейной функции, заданной на комбинаторной конфигурации перестановок. / Л. Н. Колечкина // Динамические системы. – 2009. – Вып. 27 – С. 67–80.
3. **Сачков В.Н.** Комбинаторные методы дискретной математики. / В.Н. Сачков – М., 1977. – 320 с.

Надійшла до редколегії 11.11.10

УДК 519.246.8

[2]Н.В. Комарова
Кафедра статистики и теории вероятностей ДНУ им. О. Гончара

НЕКОТОРЫЕ ИНСТРУМЕНТЫ НЕЛИНЕЙНОЙ ДИНАМИКИ В АНАЛИЗЕ И ПРОГНОЗИРОВАНИИ ВРЕМЕННЫХ РЯДОВ

Розглядається метод найближчих сусідів для прогнозування часових рядів, що виник у нелінійній динаміці, приводиться порівняльний аналіз його застосування до прогнозу цін на метали та металопродукцію.

Рассматривается метод ближайших соседей для прогнозирования временных рядов, возникший в нелинейной динамике, приводится сравнительный анализ результатов его применения для прогноза цен на металлы и металлопродукцию.

In the article the nearest neighbor method is considered, which appeared within non-linear dynamics and is used for time series forecasting; the results of the method application to metal and steel product prices forecasting are analyzed in comparison with the results of other methods.

Ключевые слова: нелинейная динамика, метод ближайших соседей, прогнозирование, временные ряды.

Вступление. Анализ и прогнозирование временных рядов – важная задача во многих областях исследований. Основными инструментами ее решения являются статистические методы, среди которых наиболее широкое распространение получили методы авторегрессии и скользящего среднего. Однако их применение для прогноза многих временных рядов, описывающих сложные системы, не дает удовлетворительных результатов.

Новый подход предложила нелинейная динамика, изучающая структуру и свойства эволюционных процессов в нелинейных динамических системах.

Постановка задачи. Нестационарные временные ряды встречаются довольно часто. Известно, что классические методы авторегрессии, скользящего среднего и индексов сезонности часто не являются адекватным инструментом их прогнозирования. Поэтому возникает задача поиска ме-

тода прогнозирования, который можно было бы использовать для широкого класса нестационарных временных рядов. Решению этой задачи посвящена настоящая работа.

Подход к решению. *Инструменты нелинейной динамики для анализа временных рядов.* В основе подхода нелинейной динамики лежит теорема Такенса о размерности вложения. Она дает возможность «восстанавливать» сложную открытую динамическую систему по нескольким ее существенным показателям, называемым параметрами порядка. Теорема Такенса подводит строгую математическую основу под идеи нелинейной авторегрессии.

Применение этих подходов оказалось результативным к исследованию рядов, описывающих изменение показателей открытых нелинейных систем, в частности, рыночной цены продукции.

Фазовые траектории. Один из основных методов нелинейной динамики состоит в построении и анализе фазовых траекторий временного ряда. Фазовая траектория временного ряда $\{x_i,\ i=1,...,N\}$ – это последовательность точек с координатами $(x_i, x_{i+\tau},..., x_{i+(m-1)\tau})$, $i=1,..., N-(m-1)\tau;\ \ m, \tau << N$ в пространстве R^m. При $m \le 3$ можно графически изобразить фазовую траекторию ряда, что удобно для ее анализа.

На рис. 1 и рис. 2 приведены соответственно график ряда значений цен на стальной листовой прокат и фазовая траектория ряда (при $m=2, \tau=1$). Резкому росту, а затем падению цены на листовой прокат (рис. 1) соответствует самая большая петля на фазовой траектории (рис. 2), которая отражает поведение цены в период с февраля 2008 г по январь 2009 г – период «разбухания» рынка и последующего падения и кризиса. На фазовой траектории отчетливо видно, что, сделав петлю в кризисный год, значение показателя оказалось близким к значению траектории 2005 г.

Характерные петли появляются на фазовых траекториях цен и на другие виды промышленной продукции и сырья в периоды кризиса. Этот факт иллюстрируется рисунком 4, на котором приведена фазовая траектория мировой цены на нефть в период 1950–2009 гг (годовые значения показателя). Одна петля приходится на годы 1979–1985, включающие период войны между Ираком и Ираном. В последние годы также наблюдается быстрое изменение цен, что на фазовой траектории отражается в виде части новой петли.

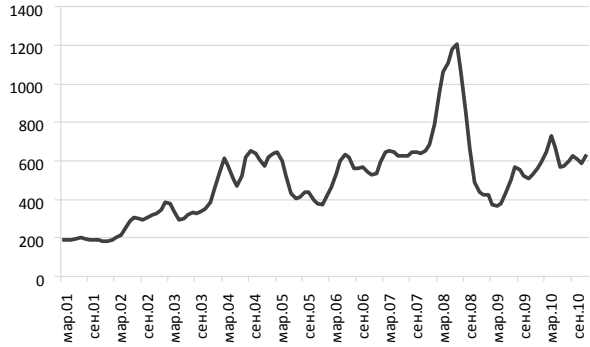


Рис. 1. Динамика цены на г/к рулон, очищенной от инфляции, $/т; источник: Металл_Курьер

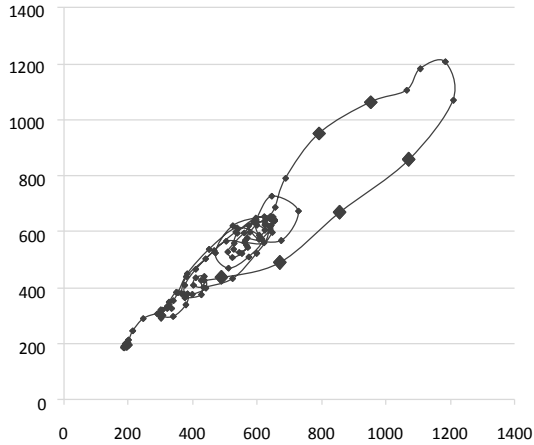


Рис. 2. Фазовая траектория цены на г/к рулон, очищенной от инфляции, $/т

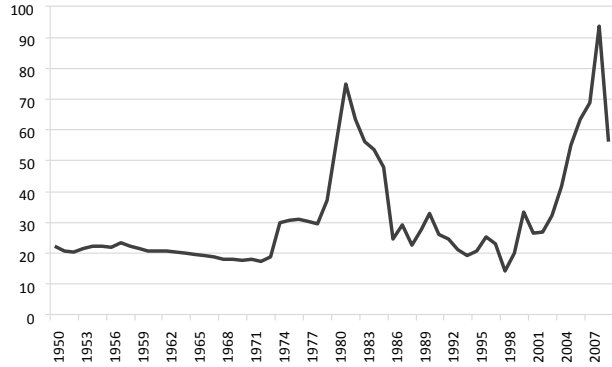


Рис. 3. Динамика цены на нефть, очищенной от инфляции, $/т

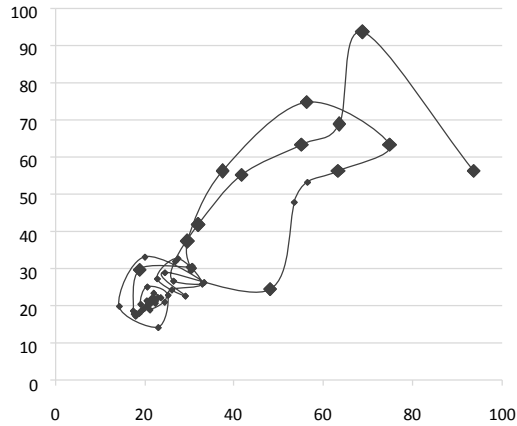


Рис. 4. Фазовая траектория цены на нефть, очищенной от инфляции, 1950-2009 гг, $/т

Русла и джокеры. В фазовом пространстве существуют области, где для описания динамики системы требуется меньшее количество переменных, чем в общем случае или чем для полного описания. Такие области характеризуются детерминированным и «медленным» движением точки фазового пространства и называются областями русел, а правила функционирования системы в них – руслами. Когда система находится в области русла, возможно, получить достаточно точный прогноз ее динамики на

некоторый период. Области же, которые характеризуются хаотическим и «быстрым» движением точки фазового пространства, зачастую управля - емым вероятностными законами, именуются областями джокеров. А правила (обычно вероятностные), по которым функционирует система, попадая в такие области – джокерами [1]. Поведение системы в области джокера отличается сложностью и разнообразием [2]. Поэтому для получения адекватных результатов прогнозирования необходимо выделять области русел и джокеров, однако это довольно непростая задача.

Отметим, что в кризисные периоды развития рынка (например, рис. 2 и рис. 4) скорость движения точки фазового пространства существенно возрастает, т. е. расстояния между точками фазовой траектории за один и тот же промежуток времени резко увеличиваются. На рис. 2 и рис. 4 жирным выделены точки с максимальной скоростью движения. Все они приходятся на периоды кризисов.

Прогнозирование временных рядов. Метод ближайших соседей. Фазовые траектории можно использовать в прогнозировании временных рядов. Самым распространенным методом является метод ближайших соседей. Его простейший вариант – метод одного ближайшего соседа нулевого порядка. Для построения прогноза показателя в момент времени M, $M \le N$, находят точку фазовой траектории, ближайшую к точке фазовой траектории с координатами $(x_{M-1-(m-1)\tau},...,x_{M-1})$. Пусть ближайшей к точке $(x_{M-1-(m-1)\tau},...,x_{M-1})$ является точка $(x_{k-1-(m-1)\tau},...,x_{k-1})$. Следующей за ней точкой на фазовой траектории будет $(x_{k-(m-1)\tau},...,x_k)$. В качестве прогноза x_M значения временного ряда в момент M принимается значение x_k.

В случае использования двух и более ближайших соседей прогноз обычно рассчитывается как среднее или средневзвешенное значение прогнозов по каждому из ближайших соседей. Важно отметить, что оптимальный выбор параметров m и τ, а также количество ближайших соседей – является отдельной задачей. На практике , не худшим вариантом оказывается подбор этих параметров. При неудачном выборе параметров может возникнуть так называемый эффект ложных ближайших соседей, то есть получение таких точек фазовой траектории, расстояние до которых от $(x_{M-1-(m-1)\tau},...,x_{M-1})$ мало, однако прогноз по ним существенно отличается от реальных значений.

В качестве меры близости между точками в фазовом пространстве чаще всего используют евклидово расстояние. При прогнозировании рыночной цены по методу ближайших соседей с евклидовым расстоянием необходимо очищать цену от инфляции. Это приводит значения цены за весь рассматриваемый период к одному уровню и делает их сравнимыми.

Прогнозирование цен на металлы. Описанный подход прогнозирования временных рядов применен для прогноза цен на различные виды металлов. Прогноз строился на 6 месяцев – июль–декабрь 2010 г. При этом использовался прямой метод ближайших соседей, когда прогноз строится независимо для всех месяцев как среднее соответствующих значений для ближайших соседей. Как было показано в [4], прямой метод является оптимальным по точности и устойчивости прогноза по сравнению с итеративными методами, предполагающими последовательное построение прогноза на один шаг с добавлением результата прогноза к исходным данным.

Наряду с прогнозами, построенными по методу ближайших соседей с использованием евклидового расстояния (БСЕ), приведены прогнозы, построенные по методу авторегрессии и скользящего среднего (АРСС) и методу индексов сезонности (ИС). В случае метода ближайших соседей строились средние значения прогнозов для m=6,…,18 и количества ближайших соседей от 1 до 5.

На рис. 5 приведены исторические значения очищенных от инфляции цен на стальной г/к рулон, алюминий и цинк. Далее по ним строились прогнозы на 6 месяцев (с июля по декабрь 2010 г).

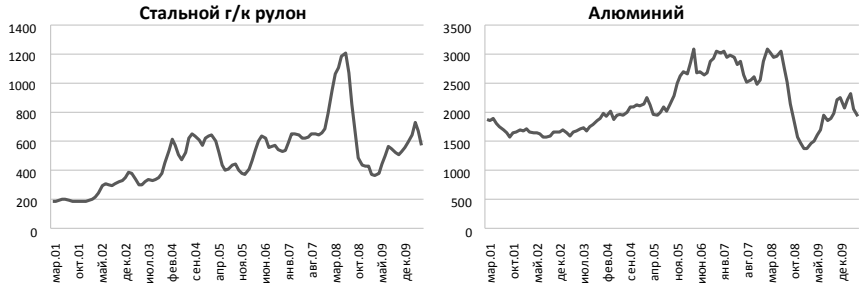


Рис. 5. Динамика цен на различные виды металлов (очищенных от инфляции), март 2001 – июнь 2010, $/т

Рис. 5 – Продолжение. Динамика цен на различные виды металлов (очищенных от инфляции), март 2001 – июнь 2010, $/т

Полученные результаты проиллюстрированы на приведенных ниже графиках.

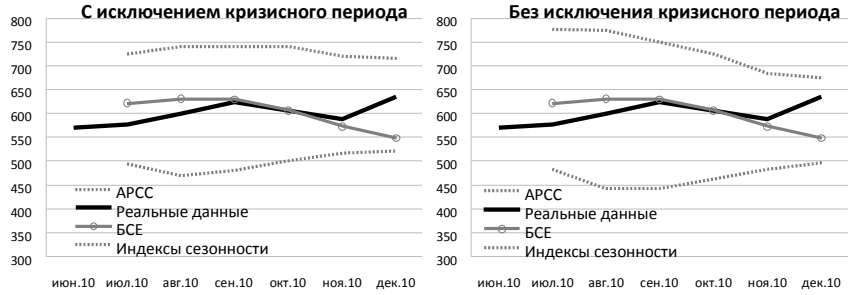


Рис. 6. Результаты прогноза цены на г/к рулон, очищенной от инфляции, на июль-декабрь 2010 г с помощью разных моделей и реальные значения цены

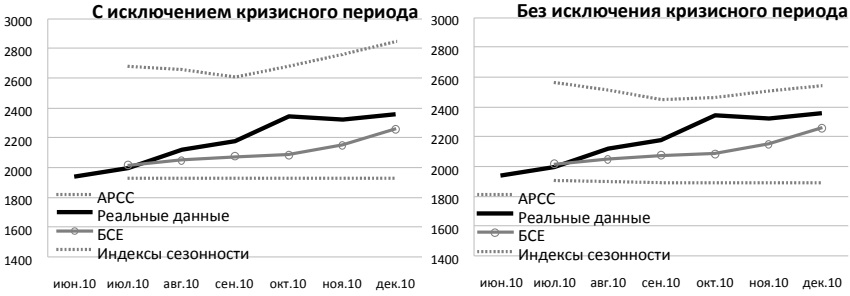


Рис. 7. Результаты прогноза цены на алюминий, очищенной от инфляции, на июль-декабрь 2010 г с помощью разных моделей и реальные значения цены

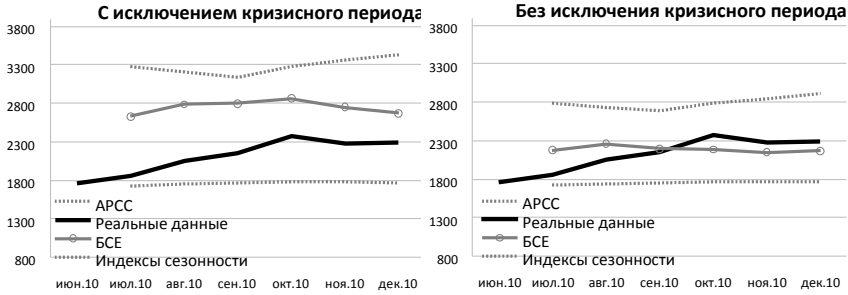


Рис. 8. Результаты прогноза цены на цинк, очищенной от инфляции, на июль-декабрь 2010 г с помощью разных моделей и реальные значения цены

В таблице приведены результаты прогнозирования цен, а именно среднее значение погрешности, рассчитанное как среднее значение относительного отклонения прогноза от реальных данных по всем шести месяцам, и среднее абсолютного значения погрешности – усредненное по шести месяцам абсолютное значение относительного отклонения.

Таблица

Погрешности прогнозов, полученных различными методами

Показатель	Метод	С исключением кризисного периода		Без исключения кризисного периода	
	Вид погрешности	Δ	|Δ|	Δ	|Δ|
Цена на г/к рулон	БСЕ	**0%**	**5%**	**0%**	**5%**
	ИС	21%	21%	21%	21%
	АРСС	-18%	18%	-23%	23%
Цена на алюминий	БСЕ	**-5%**	**5%**	**-5%**	**5%**
	ИС	22%	22%	13%	13%
	АРСС	-13%	13%	-14%	14%
Цена на цинк	БСЕ	28%	28%	**2%**	8%
	ИС	52%	52%	30%	30%
	АРСС	-18%	18%	-19%	19%

Δ – среднее значение погрешности

Из таблицы видно, что для прогнозирования цен на стальной рулон, алюминий и цинк (в случае «без исключения кризисного периода») на июль –декабрь 2010 г наилучшие результаты дал метод ближайших соседей.

Выводы. В статье показаны преимущества метода ближайших соседей при прогнозировании нестационарных временных рядов на примерах нестационарных рядов цен на металлы и металлопродукцию.

Библиографические ссылки

1. **Зульпукаров М..Г.М.** Метод русел и джокеров на примере исследования системы Розенцвейга–Макартура. / М..Г.М. Зульпукаров, Г.Г. Малинецкий, А.В. Подлазов // Институт прикладной математики им. М.В. Келдыша РАН (препринт) – 2006. – № 21.
2. **Лоскутов А.Ю.** Временные ряды: анализ и прогноз. / А.Ю. Лоскутов, О.Л. Котляров, Д.И. Журавлев // Математика. Компьютер. Образование. Сб. тр. XI междунар. конф. – Ижевск, 2004. – С. 9-46.
3. **Малинецкий Г.Г.** Современные проблемы нелинейной динамики. / Г.Г. Малинецкий, А.Б. Потапов. – М., 2000.
4. **Малинецкий Г.Г.** Русла и джокеры: о новых методах прогноза поведения сложных систем. / Г.Г. Малинецкий, А.Б. Потапов. // ИПМ им. М.В.Келдыша РАН (препринт). – 1998. – № 32.

Надійшла до редколегії 31.05.11

УДК 621.3

[3]А.Я. Кулик, Я.А. Кулик
Вінницький національний технічний університет

ІДЕНТИФІКАЦІЯ ЦИФРОВИХ СИГНАЛІВ У СИСТЕМІ ПЕРЕДАВАННЯ ІНФОРМАЦІЇ

На підставі розробленої математичної моделі проведено аналіз похибки ідентифікації прийнятого цифрового сигналу для системи передавання інформації. Показано, що ідентифікацію можна здійснювати за одним значенням на відміну від вимог класичних алгоритмів.

На основе разработанной математической модели проведен анализ ошибки идентификации принятого цифрового сигнала для системы передачи информации. Показано, что идентификацию можно осуществлять по одному значению в отличие от требований классических алгоритмов.

On the basis of developed mathematical model the signal identification error for transmission system is analized. It was shown that identification can be done using a single value unlike classical algorithms' requirements.

Ключові слова: система передавання інформації, сигнал, ідентифікація, математичне сподівання, дисперсія.

Вступ. Для побудови комп'ютерної системи передавання інформації необхідно приділити увагу методу модуляції, відповідно до якого формуються сигнали, що призначаються для перенесення інформації. У сучасних системах використовується аналіз сигналів, як за допомогою синусоїдних або косинусоїдних функцій [1], так і функцій інших базисів (Хартлі, Уолша тощо) [2 – 6]. Для асинхронних систем на етапах очікування часове розташування $x(t-\tau)$ стартового сигналу чи синхросимволу є невідомим параметром. Сам процес пошуку початкових сигналів незалежно від режиму роботи центрального процесора приймача (програмного чи переривань [7]) вимагає достатньо короткого циклу опитування (високої роздільної здатності за часом), що обов'язково потрібно враховувати на стадії проектування системи в цілому.

Постановка задачі. Метою оброблення сигналів у системах обміну інформацією є розв'язання двох задач: визначення вірогідних параметрів сигналу, спотвореного дією завад або впливом середовища, яким передається сигнал, та оцінка змін параметрів сигналу під впливом цього середовища [8]. Оскільки процес передавання даних є багатоетапним, то на кожному з них можуть виникати як систематична, так і випадкова складові похибки. Визначення і оцінювання цих складових стає основною задачею при виділенні інформативного сигналу з шуму. Таким чином, оброблення прийнятих даних являє собою складну комплексну задачу, яка вимагає для її розв'язання залучення різноманітних методів та засобів.

Метод розв'язування. Умови роботи сучасних комп'ютерних засобів передавання інформації такі, що в каналі зв'язку крім флуктуаційних шумів $\xi(t)$ і мультиплікативної завади $v(t)$ наявний імпульсний потік

$$\chi(t) = \sum_{j=0}^{L-1} f_j(t - \Delta T) \qquad (1)$$

з регулярною чи хаотичною структурою. Таким чином інформативний сигнал $x(t)$ підпадає під вплив комбінованої завади і на вході приймача буде мати вигляд

$$\hat{x}(t) = \nu(t) \cdot x(t) + \chi(t) + \xi(t) \ . \qquad (2)$$

Причиною виникнення імпульсного потоку $\chi(t)$ можуть бути як зовнішні імпульсні завади, так і неінформативні імпульси (за рахунок перегонів та луна_сигналів), які можуть формуватися самою схемою передавача.

Під час приймання інформації найбільшу складність викликають імпульсні завади, вигляд яких нагадує інформативний сигнал – послідовність прямокутних імпульсів шпаруватістю 2 (меандр) і амплітудою, розподіленою за нормальним законом з математичним сподіванням m_ξ та дисперсією D_ξ. Енергетичний спектр такого коливання, усереднений за часом, описується співвідношенням [9]

$$G_\xi(\omega) = \frac{1}{\pi} \cdot \tau \cdot D_\xi \cdot \left(\frac{\sin \frac{\omega\tau}{2}}{\frac{\omega\tau}{2}} \right)^2 . \qquad (3)$$

Потужність завади, нормована за смугою частоти,

$$G_{\xi.н}(\omega) = \frac{1}{\pi} \cdot \tau \cdot D\xi , \tag{4}$$

або, з урахуванням того, що для флуктуаційної завади $\sigma_ф^2 = 2\pi \cdot G_{\xi.н}(\omega)$,

$$D\xi_{.н} = \frac{\sigma_ф^2}{2\tau} , \tag{5}$$

де $\sigma_ф$ – спектральна щільність флуктуаційної завади.

Якщо врахувати, що миттєві значення завади розподілені нормально, то імовірність спотворення одного елементарного імпульсу тривалістю τ для моменту часу t визначається

$$p_\tau(t) = \frac{1}{\sigma_\xi \sqrt{2\pi}} \int_{-\infty}^{-\hat{x}(t)} e^{-\frac{z^2}{2D\xi}} dz . \tag{6}$$

З урахуванням визначених раніше умов передавання даних, спотворенням є така дія завади, яка призводить до інвертування сигналу. Враховуючи, що $D\xi >> |\hat{x}(t)|$, з (6) можна отримати [10]

$$p_\tau(t) = \frac{1}{2} - \frac{1}{\sqrt{2\pi}} \cdot \frac{\hat{x}(t)}{\sigma_\xi} = \frac{1}{2} - \Delta p_\tau(t) . \tag{7}$$

Оскільки форма сигналу $\hat{x}(t)$ за рахунок впливу завад відрізняється від прямокутної, то можна визначити тривалість еквівалентного прямокутного імпульсу такої самої амплітуди U_0, імовірність помилкового приймання якого дорівнює аналогічній імовірності прийнятого інформативного сигналу $\hat{x}(t)$. На інтервалі часу ідентифікації T гіпотетично може виникнути

m імпульсів завади. Ділянка інформаційного посилання ΔT після кореляційного аналізу буде вважатися спотвореною якщо більше ніж $m/2$ імпульсів завади викликатимуть інвертування інформативного сигналу. Імовірність цього $p_{\Delta T}$ визначається

$$p_{\Delta T} = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{-m\cdot\Delta p_\tau} e^{-\frac{z^2}{2\sigma^2}} dz , \qquad (8)$$

де $\sigma \approx \sqrt{m}/2$,

або

$$p_{\Delta T} = \frac{1}{2} - \frac{1}{2}\Phi\left(\frac{m\cdot\Delta p_\tau}{\sqrt{2}\cdot\sigma}\right). \qquad (9)$$

Імовірність $p_{\Delta T} = const$ за умови $m\cdot\Delta p_\tau/\sqrt{2}\sigma = const$. З урахуванням умови спотворення

$$\frac{m\cdot\Delta p_\tau}{\sqrt{2}\sigma} = \frac{1}{\sqrt{\pi}}\cdot\frac{\hat{x}(t)}{\sigma_\xi}\sqrt{m} , \qquad (10)$$

виходячи з чого $\hat{x}(t)\cdot\sqrt{m} = const$, і

$$\hat{u}_c\cdot\sqrt{m_0} = \hat{u}_i\cdot\hat{x}(t_i)\cdot\sqrt{m_i} \qquad (11)$$

або

$$m_0 = \hat{x}^2(t_i)\cdot m_i \ . \qquad (12)$$

Вираз (12) визначає кількість імпульсів $m_0 = \Delta T_e' / \tau_\xi$, які спотворюють з імовірністю $p_{\Delta T}$ ділянку реального сигналу $T/n < \Delta T = T \cdot m_0 / n < T$ амплітудою U_0, з урахуванням того, що $m_i = \Delta T / \tau_\xi$ характеризує кількість імпульсів завади, які спотворюють з такою самою імовірністю ділянку реального сигналу $T \cdot m_i / n$ в момент часу t_i. При цьому ΔT визначає проміжок реального часу, а $\Delta T_e'$ – ділянку умовного часу, в якому діє еквівалентний сигнал (рис. 1). Тоді

$$\frac{\Delta T_e'}{\tau} = \hat{x}^2(t)\frac{\Delta T}{\tau} \tag{13}$$

або

$$T_e = \int_0^T \hat{x}^2(t)dt \,. \tag{14}$$

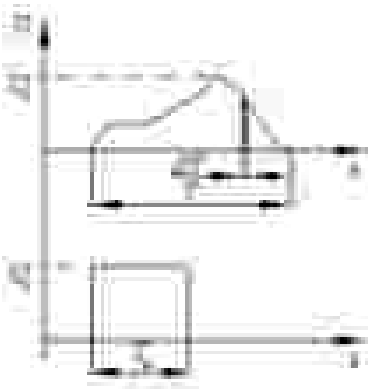


Рис. 1. Прийнятий та еквівалентний сигнали

Тобто, замість сигналу $\hat{x}(t)$, форма якого внаслідок впливу завад відрізняється від прямокутної, сформовано прямокутний імпульс (рис. 1), який за характеристиками приймання еквівалентний початковому і має таку саму амплітуду $U_e = v \cdot U_{пер}$.

Аналогічно сформульованій для реального сигналу умові, спотворення посилання визначається його інвертуванням під впливом завади на інтервалі протягом більше половини тривалості або на відрахунках кількістю більше половини від загальної. Імовірність спотворення еквівалентного сигналу $p_{x.e}$ дорівнює імовірності p_x спотворення реального сигналу

$$p_x = p_{x.e} = \frac{1}{2} - \frac{1}{2}\Phi\left(\frac{n_e \cdot \Delta p_\tau}{\sqrt{2} \cdot m}\right), \tag{15}$$

або

$$p_x = \frac{1}{2} - \frac{1}{2}\Phi\left(\sqrt{2} \cdot \Delta p_\tau \cdot \sqrt{n_e}\right). \tag{16}$$

Тоді

$$n_e = \frac{T_e}{\tau} = \frac{1}{\tau}\int_0^T \hat{x}^2(t)dt = \frac{E_{np}}{\tau \cdot \nu \cdot U_0}. \tag{17}$$

З урахуванням

$$\Delta p_\tau = \frac{1}{\sqrt{\pi}} \cdot \frac{U_{np} \cdot \sqrt{\tau}}{\sigma_{\xi.\phi}}; \tag{18}$$

$$p_x = \frac{1}{2} - \frac{1}{2}\Phi\left(\sqrt{\frac{2}{\pi}} \cdot \frac{E}{\sigma}\right). \tag{19}$$

Але для біполярних двійкових сигналів потенційна завадозахищеність визначається імовірністю помилки

$$p_{x.BI} = \frac{1}{2} - \frac{1}{2}\Phi\left(\frac{E}{\sigma}\right). \tag{20}$$

Оброблення миттєвих значень згідно з принципами побудови оптимального приймача Котельникова вимагає безкінцевої кількості відрахунків для відновлення прямокутного імпульсу. Ідентифікація імпульсу за принципом визначення одного середнього значення знижує завадозахищеність в $\sqrt{\pi/2}$ разів. Різниця в імовірностях для цих випадків ілюструється на рис. 2.

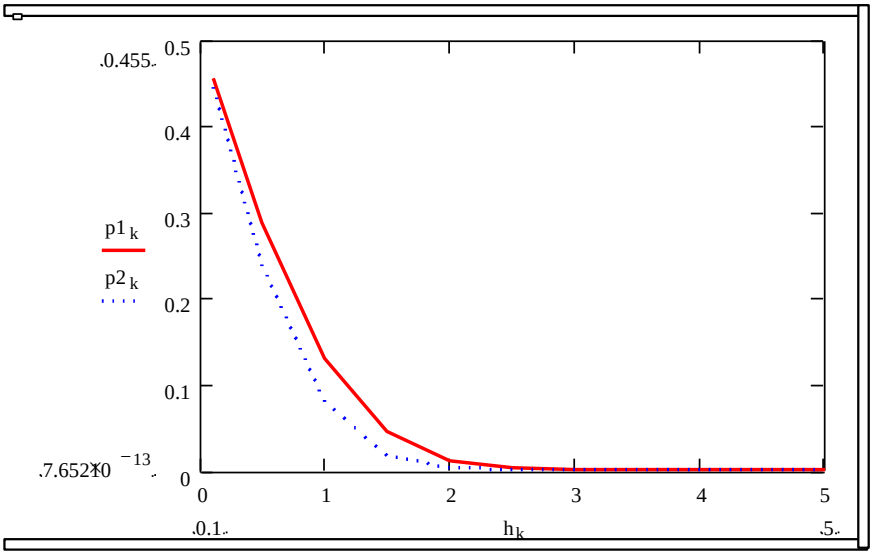


Рис. 2. Імовірності помилок при прийманні інформації:
p_1 – для випадку ідентифікації за одним значенням;
p_2 – для випадку ідентифікації сигналу за теоремою Котельникова

Отримані графіки показують порівняно незначний вплив на завадозахищеність, але побудова приймача при цьому значно спрощується. Виходячи з цього, можна визначити умови фільтрації сигналів під час приймання інформації.

Висновки. Враховуючи, що форма сигналу, який надходить до приймача з каналу зв'язку, змінюється за рахунок впливу завад, показано адекватність ідентифікації інформативного сигналу за рядом значень згідно з теоремою Котельникова та ідентифікації за одним зареєстрованим значенням. Показано різницю в імовірності помилки при цих двох методах ідентифікації, яка для різних співвідношень *сигнал / шум* не перевищує 15%.

Бібліографічні посилання

1. **Кузьмин И.В.** Основы теории информации и кодирования: [учебн. для студ. высш. уч. зав.] / И.В. Кузьмин, В.А. Кедрус. – К., 1986 – 238 с.

2. **Залманзон Л.А.** Преобразования Фурье, Уолша, Хаара и их применение в управлении, связи и других областях / Л.А. Залманзон. – М., 1989. – 496 с.

3. Хааро-подобные системы сигналов [Електрон . ресурс] / В.С. Выхованец. – Режим доступу: http://www.autex.spb.ru

4. **Брейсуэлл Р.** Преобразование Хартли. Теория и приложения: пер. с англ. / Р. Брейсуэлл. – М., 1990. – 175 с.

5. **Хармут Х.Ф.** Передача информации ортогональными функциями: пер. с англ. / Х.Ф. Хармут. – М., 1975. – 272 с.

6. Лекции по цифровой обработке сигналов [Електронний ресурс] / Е.Л. Столов. – Режим доступу: http://www.kcn.ru/tat-ru/universitet/infres/stolov

7. **Кулик А.Я.** Використання інтерфейсних мікросхем при проектуванні мікропроцесорних засобів автоматики: навч. посіб. [для студ. вищ. навч. закл.] / А.Я. Кулик. – Вінниця, 1999. – 129 с.

8. Повышение достоверности первичной обработки результатов измерений: тр. 5-й Междунар. конф. «Цифровая обработка сигналов и её применение (DSPA-2003)» [Електронний ресурс] / В.И. Марчук. – С.-Пб., – 2003 – Режим доступу: http://www.autex.spb.ru

9. **Харкевич А.А.** Борьба с помехами. / А.А. Харкевич – М., 1965. – 276 с.

10. **Прокис Дж.** Цифровая связь: пер. с англ. / Дж. Прокис. – М., 2000. – 800 с.

Надійшла до редколегії 15.01.11

УДК 519.814

[4]О.В. Рычка
Донецкий национальный технический университет

РАЗРАБОТКА И АНАЛИЗ МЕТОДА ПОВЫШЕНИЯ ТОЧНОСТИ ПРОГНОЗНЫХ РЕГРЕССИОННЫХ МОДЕЛЕЙ И ЕГО МОДИФИКАЦИЙ

Запропоновано метод та його модифікації, що дозволяють підвищити якість прогнозних регресійних моделей. Метод базується на виключенні частини статистики, яка виходить за межі заданої області. Проведено аналіз ефективності методу та його модифікацій. Надано рекомендації до використання розробленого методу.

Предложен метод и его модификация, позволяющие повысить качество регрессионных моделей. Метод основан на исключении части статистики, которая выходит за пределы заданной области. Проведен анализ эффективности метода и его модификацией. Даны рекомендации по использованию предложенного метода.

The method and its modification, permitting to improve accuracy of regression models, is offered in the article. Method is based on the exception of part of statistics which are outside the scope of area. The effiency analisys of method and its modifications is conducted. Advices on the use of the offered method are given.

Ключевые слова: аномальные данные, прогнозная модель, повышение точности, метод

Введение. В настоящее время ни одна сфера деятельности человека не может обойтись без прогнозов. Прогнозирование позволяет подготавливать и принимать обоснованные решения, обеспечивающие эффективность управления, предупреждает возможные сбои и срывы в работе. Чем выше точность прогнозов, тем более весомым будет их вклад в развитие системы. Однако встречаются данные, которые представляют собой аномальные выбросы. Если не учитывать наличие таких данных, это может привести к значительному ухудшению качества прогноза.

На сегодняшний момент существует большое количество методов, позволяющих найти и устранить аномальные наблюдения. Для линейных

регрессионных моделей к таким методам относятся – метод Прескотта - Лунда, Эктона, Титьена -Мура-Бекмана[1;2]. Несмотря на свою простоту, эти методы имеют ряд недостатков:

методы применяются последовательно к каждому из n измерений, что при большом объеме выборки увеличивает их трудоемкость;

для использования данных методов требуется нормальное распределение случайных величин невязок;

методы не применяются при объеме статистики менее 30;

при отбрасывании данных, вследствие изменения положения регрессионного уравнения, возникает смещение прогнозного значения.

Последнего недостатка лишен метод Кука. Согласно ему при отбрасывании находятся уравнивающие наблюдения, которые стабилизируют параметры нового регрессионного уравнения. Т. о. величина смещения минимизируется, однако данный метод содержит сложные вычисления и большое число элементарных операций, а при отбрасывании более чем одного наблюдения при компьютерной обработке данных можно получить результаты, резко снижающие величину R^2. При исключении исходных статистических данных требуется большое число элементарных операций, т. к. количество возможных вариантов отбрасывания составляет $C_n^2 + C_n^3 + C_n^{n/2}$ [3].

Постановка проблемы. В связи с отсутствием в настоящее время достаточно простого и точного метода, устраняющего выявленные в ходе исследований недостатки, есть необходимость в разработке нового метода. Поэтому, целью исследований является разработка и оценка эффективности метода, позволяющего повысить точность регрессионных моделей. Основная особенность данного метода заключается в простоте реализации и возможности использования в современных компьютерных технологиях.

Сущность предлагаемого метода заключается в том, что среди исходных статистических данных находятся измерения, которые не попадают в заданную область, после чего они исключаются из выборки. Эти измерения представляют собой аномальные выбросы или не достаточно весомые наблюдения, которые существенно ухудшают качество прогноза. Отличие модификаций состоит в параметрах задаваемой области.

Вначале по всем исходным статистическим данным, находятся коэффициенты линейного уравнения $\hat{Y} = A \cdot x + B$. После этого определяется СКО невязок

$$\sigma_e = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n-2}}. \qquad (1)$$

Далее необходимо найти коэффициенты перпендикуляра. Уравнение будет иметь вид: $\hat{Y}_p = A_p \cdot x + B_p$. СКО невязок, в данном случае определяется по формуле 2

$$\sigma_{p_e} = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_{p_i})^2}{n-2}}. \qquad (2)$$

После этого строится прямоугольник со сторонами $2\ k\sigma_e$ и $2k\sigma_{pe}$ (рис.1), где k – коэффициент (обычно $0{,}6 \le k < 2$), соответствующий вероятности попадания в заданную область. Вероятность попадания в область рассчитывается по формуле

$$P_0=1-2[1-P(k)], \qquad (3)$$

где $P(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{k} e^{-t^2/2} dt$.

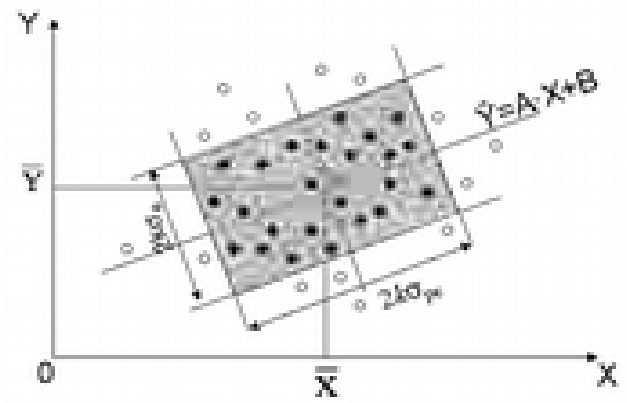


Рис. 1. Предложенный метод

Наблюдения, выходящие за рамки, построенного прямоугольника, отбрасываются, а по оставшимся данным находятся параметры нового регрессионного уравнения, после чего рассчитывается новое значение R^2 по формуле 4

$$R^2 = \frac{\sum(\hat{Y}_i - \overline{Y})^2}{\sum(Y_i - \overline{Y})^2}, \qquad (4)$$

где $\overline{Y} = \frac{\sum_{i=1}^{k} Y_i}{k}$ – математическое ожидание случайной величины Y_i.

При использовании данного метода рекомендуется отбрасывать не более 30–35% исходной статистики. При таком отбрасывании рост коэффициента детерминации R^2, как правило, составляет не менее 20 % от исходного значения R^2 (при R^2 от 0.6 до 0.75). Предложенный метод не нарушает исходную картину положения данных, т. к. наблюдения по x и по y, отбрасываются с одинаковой вероятностью.

В некоторых практических ситуациях достаточным будет применить одну из модификаций предложенного метода. Модификации метода заключаются в том, что строится не прямоугольная область, а определенный «коридор». В первом случае он представляет собой две параллельные линии, расстояние между которыми составляет $2k\sigma_e$ (рис.2). Здесь критерием остановки будет достижение коэффициента детерминации R^2 величины 0.9. Данная модификация применяется, когда $\sigma_y^2 > \sigma_x^2$. Во втором случае линии параллельны построенному перпендикуляру. Расстояние между ними равно $2k\sigma_{pe}$ (рис.3). При использовании данной модификации следует отбрасывать не более 15 % наблюдений.

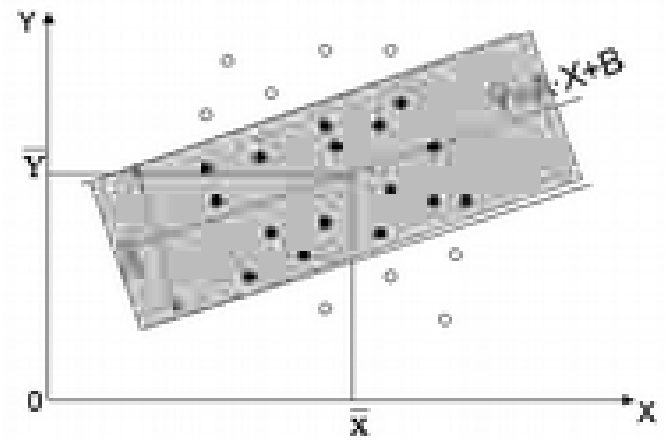


Рис. 2. Первая модификация метода

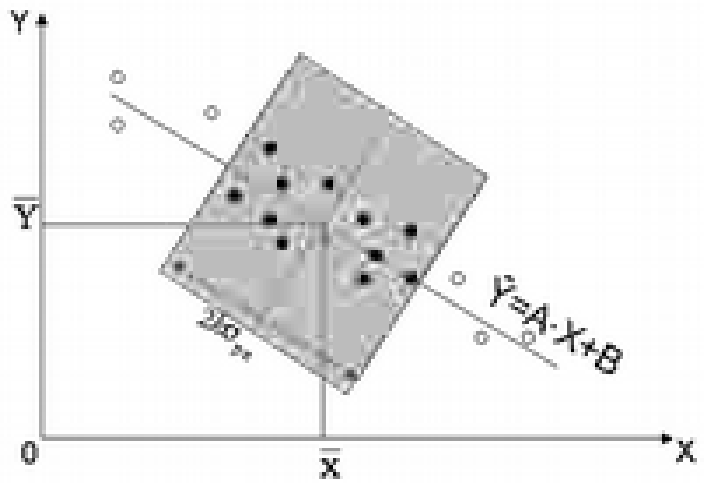


Рис. 3. Вторая модификация метода

Следует отметить, что вероятность попадания в прямоугольник равна произведению вероятности попадания в «коридор» для первой модификации на вероятность попадания для второй, при этом вероятности равны друг другу.

Анализ эффективности предложенного метода и его модификаций. Основными критериями оценки эффективности нового метода являются:

коэффициент детерминации R^2;

величина доверительного интервала;

величина смещения результата прогноза;

количество элементарных операций ЭВМ, необходимых для реализации метода.

С помощью первых двух критериев оценивается качество линейной регрессионной модели. Третий критерий является показателем трудоемкости.

При определении доверительный интервалом прогноза квантили Стьюдента $t(P_{дов}; k=n-2)$ не использовались. Это связано с тем, что закон распределения рассматриваемой статистики невязок отличен от нормального. Исключение части невязок еще более усугубляет эту проблему. Поэтому использовались доверительные интервалы свободные от закона распределения случайных величин невязок [4].

Пусть e_1, e_2, e_3, …, e_i порядковые статистики и $e_1<e_2<e_3 \ldots e_{i-1}< e_i$. При этом доверительный интервал (e_1, e_i) симметричен относительно своего медианного значения e_k и определяется доверительной вероятностью:

$$P_{дов} = I_{1/2}(1,i) - I_{1/2}(i,1),$$
$$I_x(a,b) = 1 - I_{1-x}(b,a),\ (0 \le x \le 1),$$
$$I_x(a,b) = \frac{1}{B(a,b)} \int_0^x t^{a-1} \cdot (1-t)^{b-1} dt,$$
$$B(z,\omega) = \frac{\Gamma(z) \cdot \Gamma(\omega)}{\Gamma(z+\omega)} \quad [5],$$

$\Gamma(n+1) = 1 \cdot 2 \cdot 3 \ldots (n-1) \cdot n = n!$ для целых значений аргумента,
где: $I_x(a,b)$ – неполная Бета -функция; $B(z, \omega)$ – Бета-функция; $\Gamma(n+1)$ – Гамма-функция.

Для вычисления доверительного интервала подбирались пары 1, i; 2, i-1; 3, i-2 и т. д. симметрично расположенные относительно медианного значения e_k, обеспечивающие заданное значение $P_{дов}$. Найденная пара представляет собой размах доверительного интервала. Естественно, что из-за дискретного характера порядковой статистики точное значение величины доверительного интервала таким способом найти не удается, однако с этим приходится мириться.

Анализ метода и его модификаций по предложенным критериям, проводился на различных наборах наблюдений. Для наглядного отражения результатов использования предложенного метода и его модификаций в данной статье, рассматриваются два варианта выборок с различными исходными значениями коэффициента детерминации R^2.

Первая выборка представляет зависимость стоимости перевозок у от объема продаж х (табл.1). Исходное значение коэффициента детерминации R^2 равно 0.71.

Таблица 1

Исходные данные первой выборки

x	301	328	353	372	386	389	401	408
y	52.46	72.3	54	62.98	52.95	53.7	63.7	58.99
x	415	444	446	457	458	463	484	491
y	66.8	59.7	71.66	72.81	68.44	69.33	70.77	79.38
x	503	512	517	527	535	547	596	623
y	74.39	85.58	82.03	94.44	70.84	89.18	93.24	90.5

Вторая выборка отражает зависимость процентной доли учеников, успешно прошедших тестирование от средней посещаемости уроков. Исходная величина R^2 составляет 0.6

Таблица 2

Исходные данные третьей выборки

x	88.67	91.93	92.08	92.21	92.22	92.63	92.8	92.87
y	26	38	60	42	49	58	53	40
x	93.05	93.06	93.22	93.39	93.67	93.71	93.8	93.93
y	43	48	22	59	45	58	70	60
x	93.95	93.96	94.03	94.04	94.16	94.2	94.3	94.36
y	59	59	67	44	71	49	46	78
x	94.4	94.4	94.42	94.5	94.54	94.66	94.7	94.72
y	65	69	68	60	66	69	66	62
x	94.76	94.83	94.93	94.97	95.56	95.66	95.7	95.76
y	67	55	38	58	84	88	79	84
x	95.76	95.77	95.93	96.07	96.13	96.33	96.8	
y	74	72	78	82	92	83	93	

Результаты применения предложенного метода, а также его модификаций представлены в таблицах 3–6. В таблицах 3 и 4, содержатся зависимости величин R^2 и доверительного интервала от количества отбрасываемых точек в процентах для первой и второй выборки соответственно. Количество отбрасываемых точек соответствует вероятности непопадания в заданную область. Величины смещения прогнозного значения $Y_{прогн}$, рассчитанные в процентах, полученные в результате использовании предложенного метода, представлены в таблице 5 . В таблице 6 приведена информация о числе элементарных операций, необходимых при использовании метода.

Таблица 3

Значения R^2 и величина доверительного интервала для первой выборки

Процент отброшенных точек	1-я модиф. R^2	2-я модиф. R^2	Метод, R^2	1-я модиф. ДИ,%	2-я модиф. ДИ, %	Метод. ДИ, %
0	0.71	0.71	0.71	20.42	20.42064	20.42064
10	0.837	0.58	0.799	12.945	21.02419	17.61326
15	0.87	0.7572	0.8145	11.07	17.78211	13.29402
20	0.89	0.7572	0.7622	9.68	17.78211	13.22544
30	0.9115	0.731	0.8387	8.33	17.83178	11.42163
35	0.9226	0.7176	0.871	7.71	18.07439	8.831413
40	0.9226	0.8273	0.8424	7.71	9.209786	8.571503
50	0.94	0.685	0.781	5.5	9.806398	7.858299

Т. о. из таблицы видно, что коэффициент детерминации R^2, при использовании предложенного метода после отбрасывания 35 % исходных данных вырос с 0.71 до 0.87, величина доверительного интервала уменьшилась в 2.3 раза. Использование первой модификации также дало хорошие результаты − рост R^2 до рекомендуемого значения и уменьшение доверительного интервала в 2.5 раза. В данном примере вторая модификация также дает результаты, однако не такие значительные, как метод и первая модификация.

Таблица 4

Значения R^2 и величина доверительного интервала для второй выборки

Процент отброшенных точек	1-я модиф. R^2	Метод, R^2	1-я модиф. ДИ,%	Метод. ДИ, %
0	0.602	0.6	39.02	39.02
10	0.729	0.68	24.83	28.1
20	0.786	0.67	21.5	25.03
25	0.79	0.685	19.73	25.01
30	0.85	0.74	18.24	19.71
35	0.897	0.687	14.33	19.61
40	0.917	0.594	11.43	19.32
50	0.919	0.716	8.984	14.77

В данном случае при использовании нового метода коэффициент детерминации вырос на 23 % от исходного значения равного 0,6 и составил 0,74, величина доверительного интервала при этом уменьшилась в 2,1 раза. При использовании первой модификации, здесь также как и в предыдущем примере достигается рекомендуемое значение R^2 равное 0.9. Однако, в данном случае, использование второй модификации значительно снизило величину коэффициента детерминации (в таблице это не отражено). Это связано со спецификой исходных данных.

Таблица 5

Значения величины смещения Δ

Процент отброшенных точек	1-я выборка Δ,%	2-я выборка Δ, %
10	0.906846	1.562204
15	2.263314	0.997816
20	2.355117	0.21496
30	1.225094	0.50746
35	0.536757	0.125088
40	0.173853	0.048557
50	0.167467	2.575949

Таблица 6

Количество элементарных операций

Количество отброшенных точек, %	1-я выборка			2-я выборка		
	1-я модиф.	2-я модиф.	Метод	1-я модиф.	2-я модиф.	Метод
0	219	219	219	426	426	426
5	639	646	843	1247	1254	1654
15	613	620	809	1208	1215	1620
20	600	607	792	1195	1150	1603
30	574	581	775	1104	1098	1535
40	561	542	724	1039	1020	1433
50	535	503	690	1000	968	1331

Применение предложенного метода дало следующие результаты:

коэффициент детерминации R^2 растет и достигает рекомендуемого значения при вероятности попадания в заданную область равную 65 %;

величина доверительного интервала уменьшается до 2.5 раза;

величина смещения прогнозного значения $Y_{\text{прогн}}$ не превышает 2.5 %;

число элементарных операций увеличилось в 3.5 раза и составило 750 и 1535 при исходном объеме выборки 24 и 47 значений соответственно, но оно является меньшим, чем при применении известных методов.

Т. о. можно сказать, что применение предложенного метода и его первой модификации дало хорошие результаты.

В рассматриваемых примерах использование второй модификации метода показало результат хуже, чем при применении метода или его первой модификации. Однако существуют ситуации, когда вторая модификация дает хорошие результаты.

Пусть дана следующая выборка (табл.7).

Таблица 7

Исходные данные

x	2	4	6	8	10	30
y	7	11	15	19	23	45

Значение R^2, полученное по исходной выборке равно 0,977, а рассчитанное после отбрасывания 10 % данных по второй модификации составляет 1.

Отсюда можно сделать вывод, что данная модификация применима для ситуаций, когда есть ярко выраженные аномальные значения переменной.

Выводы. В результате проведенных исследований можно сделать следующие выводы:

1. Применение разработанного метода повышения качества прогнозных регрессионных моделей приводит к росту коэффициента детерминации R^2.
2. Величины доверительного интервала снижается.
3. Смещение прогнозной величины $Y_{прогн}$ является незначительным.
4. При использовании предложенного метода рекомендуется исключать не более 30–35% наблюдений. Рост коэффициента детерминации R^2 при этом, как правило, составляет не менее 20 % от исходного значения R^2 (при R^2 от 0.6 до 0.75).
5. Для первой модификации критерием остановки является увеличение R^2 до 0.9.
6. Применяя вторую модификацию, следует отбрасывать не более 15 % данных для того, чтобы не произошло искажения исходного вида представления независимой переменной x.
7. Исходное значение коэффициента детерминации R^2 не должно выходить за пределы интервала от 0.6 до 0.8.
8. Предложенный метод и его модификации легко реализуются в современных компьютерных технологиях, т. к. не содержат большого числа элементарных операций и не требуют постоянного контроля со стороны пользователя.

Библиографические ссылки

1. **Кобзарь А.И.** Прикладная математическая статистика. Для инженеров и научных работников. / А.И. Кобзарь − М., 2006. − 816 с.
2. **Дрейпер Н.Р., Смит Г.** Прикладной регрессионный анализ. 3-е изд.: Пер. с англ. / Н.Р. Дрейпер, Г. Смит − М., 2007. − 912 с.
3. **Смирнов А.В., Рычка О.В.** Метод повышения качества прогнозных регрессионных моделей./ А.В. Смирнов, О.В. Рычка // Наукові праці Донецького національного технічного університету. Серія «Інформатика, кібернетика та обчислювальна техніка». Вип. 12(165), 2010. − С.141-147.
4. **Ллойд Э**. Справочник по прикладной статистике. В 2-х т. Т.1: Пер. с англ./ Под ред. Э. Ллойда, У. Ледермана, Ю.Н. Тюрина. − М., 1989. − 510 с.
5. **Абрамовиц М.** Справочник по специальным функциям с формулами, графиками и математическими таблицами./ Под. ред. М.Абрамовица и Н. Стигана: пер. с англ. под ред. В.А. Диткина и Л.И. Кармазиной. − М., 1979. − 830 с
6. **Айвазян С.А.** Прикладная статистика: Основы моделирования и первичная обработка данных. Справочн. изд./ С.А. Айвазян, И.С. Енюков, Л.Д. Мещалкин. − М., 1983. − 471 с.

Надійшла до редколегії 10.11.10

УДК 519.876.5

[5]І.І. Скрильник
Полтавський національний технічний університет імені Юрія Кондратюка

ЗАСТОСУВАННЯ ТЕОРІЇ n-МІРНИХ ПАТТЕРНІВ ДО ОПТИМІЗАЦІЇ ПОРТФЕЛЯ ЦІННИХ ПАПЕРІВ

Розглядається проблема застосування теорії n-мірних паттернів до оптимізації портфеля цінних паперів у випадку декількох інвесторів. Побудовані n-мірні паттерни дають уяву про варіанти розподілу капіталовкладень та можливість знаходження оптимального розв'язку. Побудова паттерна здійснюється на основі пофарбування відповідної графової моделі.

Рассматривается проблема использования теории n-мерных паттернов в оптимизации портфеля ценных бумаг в случае нескольких инвесторов. Построенные n-мерные паттерны дают представление о вариантах распределения капиталовложений и возможность нахождения оптимального решения. Построение паттерна осуществляется на основе раскраски соответствующей графовой модели.

The article is devoted to the problem of utilization of the theory of n-dimensional patterns to resolve the problem of optimization of capital repartition for several stakeholders. The obtained patterns provide information about possible variants of the capital repartition and indicate the optimal solution. The design of n-dimensional pattern is based on the adequate graph colouring problem solution.

Ключові слова: n -мірний паттерн, графова модель, частка капіталу, ризик, ефективність цінних паперів, коваріація, математичне сподівання.

Вступ. Досліджуючи ряд задач, виявлено, що їх можна розв'язувати за певними шаблонами, так званими паттернами. Це дозволяє розробникам стандартизувати розв'язання багатьох класів практичних задач.

Основоположником теорії паттернів вважається Ульф Гренадер, який намагався побудувати теорію логічних шаблонів для моделювання образів. Теорія паттернів ефективно використовується в психології, мінералогії, біохімії, органічній хімії, командному розробленні комп'ютерних додатків, об'єктно-орієнтованому програмуванні [1]. Теорію паттернів можна також застосувати до розв'язання економічних задач, наприклад, управління портфелем цінних паперів, задач логістики, планування та ін.

Для розв'язання таких класів задач дослідниками використовувалися різні методи оптимізації. Фундаментальні результати у цьому напрямку

належать Д. Конолі, Д. Абрамсону, М. Рандалу [2; 3]. Але такі підходи не завжди дають уяву про множину розв'язків. Тому виникла потреба створювати моделі шаблонів, які б були універсальними структурами і охоплювали усі варіанти розв'язків задачі , а також несли інформацію про зв'язки між об'єктами.

У статті розглянута задача формування оптимального портфеля цінних паперів у випадку наявності декількох інвесторів. Дана задача розв'язана за допомогою побудови відповідного паттерна на основі пофарбування його еквівалентної теоретико-графової моделі та композиційної операції між вектором оптимального розв'язку $x^*_{опт}$ та вектором кольорів E, тобто $\Re(E)\ x^*_{опт}\ E$.

Постановка задачі. На фінансовому ринку обертається значна кількість цінних паперів: державні цінні папери, акції приватних фірм, векселі тощо. Такі цінні папери мають різний ступінь ризику й ефективність. Папери із меншим ступенем ризику, як правило, приносять низький прибуток. Високоприбуткові папери є більш ризикованими.

Розглянемо оптимізацію капіталовкладень у цінні папери за наявності n рівноправних інвесторів. У цьому випадку кожний учасник має свій власний капітал $XW_1, XW_2, \ldots, XW_n$, який вони хочуть розподілити певним чином для формування індивідуальних портфелів та спільного портфеля цінних паперів. Нехай на ринку цінних паперів обертається також q видів паперів, що мають ефективності $E_1, E_2, \ldots, E_q$. Кожний з учасників може сформувати свій власний портфель незалежно від інших. Задача полягає у зниженні значення максимального ризику інвесторів.

Метод розв'язування. Це можна зробити завдяки формуванню спільного портфеля шляхом об'єднання часток їх капіталів [4]. Оптимізація спільного портфеля відбувається за рахунок спільних коштів, вона передбачає купівлю та розподіл цінних паперів між учасниками. Хоча формально існує $n+1$ портфель, у реальності придбані папери зберігаються у n учасників, тобто реально існує n портфелів. Ризик спільного портфеля обчислюється за формулами:

$$\sigma_{\Sigma} = \frac{\bar{\sigma}_p}{\sqrt{q}}; \qquad (1)$$

$$\bar{\sigma}_p = \max_j \sigma_{pj};\ j = 1, n, \qquad (2)$$

де σ_{Σ} – ризик спільного портфеля, тобто ризик для кожного з учасників; $\bar{\sigma}_p$ – максимальний ризик портфелів учасників; σ_{pj} – значення ризику для j-го портфеля. У даному випадку склад початкового портфеля не на-

кладає ризик на всі інші портфелі, оскільки формування спільного портфеля оптимізує індивідуальні портфелі учасників.

З'ясуємо, яким чином відбувається оптимізація портфелів кожного з учасників. Нехай кожний із учасників виділяє певну частку свого капіталу для формування спільного портфеля, тоді капітал кожного з інвесторів буде дорівнювати

$$XW_i = w_{vpi} + W_i; 1 \le i \le n, \tag{3}$$

де w_{vpi} – частка капіталу і-го інвестора, котру він планує вкласти до спільного портфеля; W_i – капітал, що і-ий інвестор планує вкласти у власний портфель, який буде для кожного учасника розподілений по-різному

$$W_i w = \sum_{k=1}^{q} x_{i,k}, \tag{4}$$

де q – загальна кількість цінність паперів, що обертаються на ринку цінних паперів.

Капітал, виділений кожним учасником для формування спільного портфеля, у цьому разі дорівнює

$$\sum_{i=1}^{n} w_{vpi} W_i = max. \tag{5}$$

Графова модель $G(X,W)$, котра відображає такий розподіл капіталу між учасниками, матиме вигляд, зображений на рис. 1.

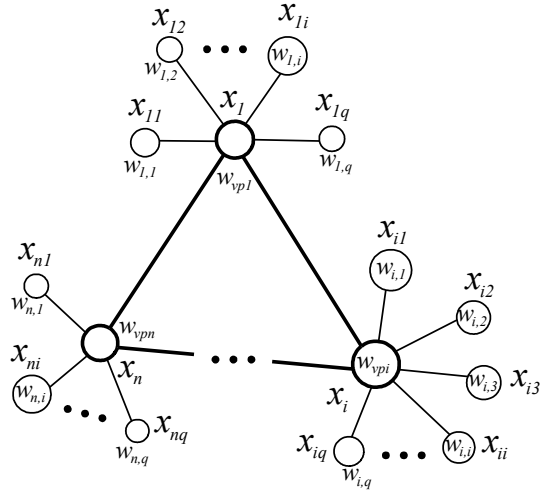


Рис. 1. Графова модель $G(X,W)$ формування спільного портфеля

Таким чином, задача формування спільного портфеля відповідає моделі (6) – (9), (1).

$$V_p = \sum_{i=1}^{N} w_i x_{i,j} \cdot V_{ij} \cdot \left(w_i x_{i,j}\right)^T \to min; \tag{6}$$

$$\sum_{j=1}^{q}\sum_{i=1}^{N} x_{i,j} \cdot w_i = W_{max}; 0 \le w_i \le W_{max}; \tag{7}$$

$$\sum_{j=1}^{q}\sum_{i=1}^{N} m_i x_{i,j} \cdot w_i = m_p; \tag{8}$$

$$\begin{aligned} &x_{i,j} + x_{k,j} \le 1; x_{i,j} \in \{0,1\}, x_{k,j} \in \{0,1\}, \\ &1 \le (i,k) \le N; 1 \le j \le q. \end{aligned} \tag{9}$$

Тут V_p є дисперсією портфеля, що визначає в кінцевому випадку його ризик; $x_{i,j} \in \{0,1\}$ є бінарною змінною (вершина графа), значення якої дорівнює 1, якщо $x_{i,j}$ пофарбована у колір E_j та 0 в іншому разі; w_i – це вага вершини в графі (частка капіталу); W_{max} – загальний капітал; m_i – відоме математичне очікування ефективності; m_p – бажане математичне очікування ефективності портфеля, що задається інвестором. Коваріаційна матриця V_{ij} обчислюється за формулою:

$$V_{ij} = cov(E_i, E_j). \tag{10}$$

Щоб мати уяву про варіанти розподілу капіталовкладень між учасниками спільного портфеля та визначити оптимальний варіант, побудуємо n-мірний паттерн. Проблема побудови паттерна висвітлювалася раніше [5; 6], зокрема при розробленні алгоритмів побудови паттернів із накладеними обмеженнями. Автором дано визначення n-мірного паттерна для розв'язування цілого класу задач дискретної оптимізації.

Означення 1. n-мірний паттерн Π – це структура виду $\Pi(\Lambda, K, F, \Phi)$, де $\Lambda = (\lambda_{1,1}, \lambda_{1,2}, ..., \lambda_{n,n})$ – масив $n \times n$ елементів (об'єктів); $K = \{K_1, K_2, ..., K_r\}$ – множина типів (властивостей); F – сюр'єктивне відображення $F: \Lambda \to K$, яке розбиває множину елементів Λ на n класів за даними типами; $\Phi = \{\Phi_1, \Phi_2, \Phi_3, ..., \Phi_m\}$ – множина

ін'єктивних відображень $\Phi_{jj}:,K \to \Lambda \quad j = \overline{1,m},\ \Lambda\Lambda\subset$, що $\overset{m}{\underset{j\mp}{}} \Phi_j : K \to \Lambda$ і виконуються умови:

1. $Im\Phi\Phi==\overset{r}{\underset{i\mp}{}}{}_j(K_i)\ \ j\ \ \overline{1,m}$;

2. $\forall j,j'',j \neq j\ /Im\ \Phi\Phi\neq Im\ {}_{j'}$.

Означення 2. n-мірний паттерн $\Pi\Lambda(K,\ ,F,\Phi)$ називається гомогенним, якщо $\forall j,j'',j \neq j\ /Im\ \Phi\Phi\ \ Im\ {}_{j'} = \varnothing$.

Означення 3. n-мірний паттерн називається гетерогенним, якщо $\exists\neq j,j'',j\quad j\ /\ Im\Phi\Phi\ \ Im\ {}_{j'} \neq \varnothing$.

Щоб побудувати паттерн, необхідно і достатньо:

1. Визначити $F:\ \Lambda K\to$ (умова необхідності).
2. Задати набір ін'єктивних відображень (функцій) $\Phi\Phi\{\ {}_j\}$, де $\overset{m}{\underset{j\mp}{}} \Phi_j : K \to \Lambda$ (умова достатності).

Отже, для побудови паттерна потрібно розробити математичну модель, в якій розв'язуються дві проблеми:

1. Пофарбування зв'язаного графа $G = (V,E\ )$ з заданою кількістю вершин $Vn= \ {}^2$, (з урахуванням або не врахуванням обмежень на пофарбування). У результаті отримаємо оптимальний розв'язок $x^*_{опт}$;
2. Визначення Φ за допомогою композиційної операції $x^*_{опт} =\Re\ (C)$, де $\Re(C)$ – пофарбування графа, задане матрицею, C – набір кольорів.

Для розв'язання поставленої задачі потрібно пофарбувати граф $Gx(,w\ )$ у кольори з множини E та визначити матрицю $\Re\{E)\ \ x^*_{опт}\ \ E$. Математична модель описується (6 – 9), (1). При цьому умова перевірки правильності побудови паттерна у випадку формування спільного портфеля набуває вигляду

$$\sum_{j=1}^{nn}\sqrt{\sum_{i\ 1} w_i V_{\ ij} \cdot w_j} \le \sigma_\Sigma . \qquad (11)$$

Задачу пофарбування зв'язаних графів ($1, 6 - 9$) можна розв'язати за допомогою методів оптимізації, а також вона ефективно розв'язується при застосуванні генетичного алгоритму $[7 - 11]$.

Приклад розв'язування. Нехай на ринку обертається 5 видів цінних паперів q_1, q_2, q_3, q_4, q_5, що мають за чотири періоди випадкову ефективність E_1, E_2, E_3, E_4, E_5 (табл. 1). Три інвестори сформували власні портфелі цінних паперів незалежно один від одного. Так портфель першого інвестора складають цінні папери 1 -го, 3 -го, 4 -го видів, другого – 2-го, 3-го, 5-го видів, а третього – 1-го, 3-го, 4-го, 5-го видів.

Для формування спільного портфеля кожний із учасників може виділити певну частку від власного капіталу: перший – не більше 0,2, другий – не більше 0,2, третій – не більше 0,1. На спільні кошти передбачено купівлю та розподіл цінних паперів між учасниками, але таких паперів, що вже складають їх власні портфелі.

Побудувати паттерн, перевірити правильність його побудови. Визначити найменший ризик спільного портфеля та частки капіталу, які виділить кожний інвестор на придбання певних видів паперів для його формування при такому ризику.

Таблиця 1

Таблиця ефективності цінних паперів за чотири періоди

Види паперів	Випадкова ефективність	1 період	2 період	3 період	4 період	Середнє значення ефективності	Відхилення від середнього значення
q_1	E_1	1,10	1,30	0,90	1,20	1,13	0,125
q_2	E_2	0,78	0,94	1,10	1,30	1,03	0,170
q_3	E_3	0,96	0,89	1,12	1,30	1,07	0,1425
q_4	E_4	0,86	0,83	1,12	1,15	0,99	0,145
q_5	E_5	1,20	1,30	1,60	1,40	1,38	0,125

Отже, розподіл капіталу між учасниками можна представити у вигляді графової моделі $G(x,w)$. Спільний портфель, сформований трьома інвес-

торами, є графом K_3, причому $K_3 \subset G(X,W)$. Визначимо найменший ризик спільного портфеля для кожного з учасників. Моделлю власних портфелів кожного з учасників є підграфи графа $G(X,W)$, тобто $G_1' \subset G(X,W)$, $G_2' \subset G(X,W)$, $G_3' \subset G(X,W)$. Вони являють собою структури у вигляді зірок, що знаходяться в кожній вершині графа K_3. Для першого інвестора – це підграф G_1', що складається з трьох вершин, з відповідними видами цінних паперів q_1, q_3, q_4 (табл. 2).

Таблиця 2

Таблиця ефективності цінних паперів першого інвестора

Види паперів	Випадкова ефективність	1 період	2 період	3 період	4 період	Середнє значення ефективності	Відхилення від середнього значення
q_1	E_1	1,10	1,30	0,90	1,20	1,13	0,125
q_3	E_3	0,96	0,89	1,12	1,30	1,07	0,1425
q_4	E_4	0,86	0,83	1,12	1,15	0,99	0,145

Коваріаційна матриця для портфеля цінних паперів першого інвестора наведена в таблиці 3.

Таблиця 3

Коваріаційна матриця цінних паперів першого інвестора

V_{ij}	E_1	E_3	E_4
E_1	0,021875	− 0,00569	− 0,0105
E_3	− 0,00569	0,024969	0,0216
E_4	− 0,0105	0,0216	0,02125

Для другого інвестора – це підграф G_2', що складається з трьох вершин, з відповідними видами цінних паперів q_2, q_3, q_5 (табл. 4).

Таблиця 4

Таблиця ефективності цінних паперів другого інвестора

Види паперів	Випадкова ефективність	1 період	2 період	3 період	4 період	Середнє значення ефективності	Відхилення від середнього значення
q_2	E_2	0,78	0,94	1,10	1,30	1,03	0,17
q_3	E_3	0,96	0,89	1,12	1,30	1,07	0,1425
q_5	E_5	1,20	1,30	1,60	1,40	1,38	0,125

Коваріаційна матриця цінних паперів обчислена за допомогою (10) і наведена в таблиці 5.

Таблиця 5

Коваріаційна матриця цінних паперів другого інвестора

V_{ij}	E_2	E_3	E_5
E_2	0,0371	0,027325	0,01825
E_3	0,027325	0,024969	0,012438
E_5	0,01825	0,012438	0,021875

Для третього інвестора – це підграф $G_3^{'}$, що складається з чотирьох вершин, з відповідними видами цінних паперів q_1, q_3, q_4, q_5 (табл. 6).

Таблиця 6

Таблиця ефективності цінних паперів третього інвестора

Види паперів	Випадкова ефективність	1 період	2 період	3 період	4 період	Середнє значення ефективності	Відхилення від середнього значення
q_1	E_1	1,10	1,30	0,90	1,20	1,13	0,125
q_3	E_3	0,96	0,89	1,12	1,30	1,07	0,1425
q_4	E_4	0,86	0,83	1,12	1,15	0,99	0,145
q_5	E_5	1,20	1,30	1,60	140	1,38	0,125

Коваріаційна матриця для третього портфеля наведена в таблиці 7.

Таблиця 7

Коваріаційна матриця цінних паперів третього інвестора

V_{ij}	E_1	E_3	E_4	E_5
E_1	0,021875	− 0,00569	− 0,0105	− 0,01438
E_3	− 0,00569	0,024969	0,0216	0,012438
E_4	− 0,0105	0,0216	0,02125	0,017
E_5	− 0,01438	0,012438	0,017	0,021875

Оскільки перший інвестор вклав свої кошти у 3 види цінних паперів, то пофарбування графової структури, що відповідає його портфелю визначається вектором $x_1^* = (100010001\quad)$. Пофарбування графової структури, що відповідає другому портфелю – вектором $x_2^* = (100010001\quad)$, графової структури, що відповідає третьому портфелю – вектором $x_3^* = (1000010000100001\quad)$.

Визначимо частки капіталу, вкладені в кожний вид цінних паперів кожним із трьох інвесторів відповідно їх власних портфелів. Для цього використаємо рівняння (6), (7), (9). У результаті маємо, що перший учасник повинен вкласти капітал у цінні папери власного портфеля в таких частках: $w_1 = 0{,}3442$, $w_2 = 0{,}3016$, $w_3 = 0{,}3542$. При цьому коваріація портфеля першого інвестора дорівнює $V_1 = 0{,}0075$, а ризик дорівнює $\sigma_{\overline{1}}\ \ 0{,}0868$. Другий учасник повинен вкласти капітал у цінні папери власного портфеля в таких частках: $w_1 = 0{,}2391$, $w_2 = 0{,}3553$, $w_3 = 0{,}4056$. Коваріація портфеля другого інвестора дорівнює $V_2 = 0{,}0089$, а ризик дорівнює $\sigma_{\overline{2}}\ \ 0{,}0942$. Третій учасник – в частках $w_1 = 0{,}2561$, $w_2 = 0{,}2242$, $w_3 = 0{,}2635$, $w_4 = 0{,}2563$. Коваріація портфеля третього інвестора дорівнює $V_3 = 0{,}0056$, а ризик дорівнює $\sigma_{\overline{3}}\ \ 0{,}0748$.

Відповідно до цього розв'язку визначаємо ризик спільного портфеля цінних паперів $\sigma_{\overline{\Sigma}} = \dfrac{max(\sigma\varphi,\ _2, \sigma_3)}{\sqrt{5}}\ \ 0{,}0421$, коваріація спільного портфеля дорівнює $V_{\Sigma\Sigma}\sigma = \overline{\ }^2\ \ 0{,}0018$.

Знаючи найменший ризик спільного портфеля, потрібно визначити який цінний папір із 5 -и видів буде придбаний кожним учасником для формування спільного портфеля (при цьому враховується склад кожного власного портфеля) і яку частку свого капіталу вони для цього виділять. Граф K_3 потрібно пофарбувати у кольори з множини фарб $\{E_1E,E_2E,E_3 \quad _4 \quad _5\}$.

Оскільки спільний портфель формується з п'яти видів цінних паперів, то коваріаційна матриця, обчислена за допомогою (10), має наступний вид (табл. 8).

Таблиця 8

Коваріаційна матриця цінних паперів для спільного портфеля

V_{ij}	E_1	E_2	E_3	E_4	E_5
E_1	0,021875	− 0,00125	− 0,00569	− 0,0105	−0,01438
E_2	− 0,00125	0,0371	0,027325	0,0248	0,01825
E_3	− 0,00569	0,027325	0,024969	0,0216	0,012438
E_4	− 0,0105	0,0248	0,0216	0,02125	0,017
E_5	− 0,01438	0,01825	0,012438	0,017	0,021875

Тоді задача знаходження оптимального портфеля полягає у пофарбуванні графа K_3 у кольори з множини фарб $\{E_1E,E_2E,E_3 \quad _4 \quad _5\}$ так, щоб $VV_p = \quad \Sigma$. Мовою 0 -1 програмування задача побудови гетерогенного патерна формулюється таким чином:

$$V_p = \sum_{i\mp}^{N} w_i x_{i,j} \cdot V_{ij} \cdot \left(w_i x_{i,j}\right)^T = V_\Sigma \tag{12}$$

$$\begin{aligned} &x_{i,j} + x_{k,j} \le 1; x_{i,j} \in \{0,1\}, x_{k,j} \in \{0,1\}, \\ &1 \le (i,k) \le N; 1 \le j \le q. \end{aligned} \tag{13}$$

$$x_{1,4} = \ ; \ x_{1,5} = \ ; \ x_{2,2} = \ ; \ x_{2,4} = \ ; \ x_{3,5} = \ . \tag{14}$$

Отриманий розв'язок задачі за допомогою рівнянь (12), (13), (14) дорівнює $x^*_{опт} = (000100000100100 \quad)$, тобто, при найменшому ризику спільного портфеля перший учасник вкладе свою частку капіталу у 4 -ий вид цінних паперів, другий – у 5 -й, а третій – у 3 -й. Маємо перший варіант формування спільного портфеля $(453 \quad)$. Відповідно на закупівлю цих

паперів будуть виділені такі частки капіталів кожного інвестора: $w_1 = 0{,}0929$, $w_2 = 0{,}2545$, $w_3 = 0{,}0835$.

Розглянемо другий варіант формування спільного портфеля (354). У цьому разі $x^*_{опт} = (001000000100010)$, тобто перший учасник придбає для спільного портфеля 3-й вид цінних паперів, другий – 5-й вид, а третій – 4-й вид. Визначимо частки капіталів інвесторів, вкладених у цінні папери за допомогою рівнянь (6), (7), (9). Отримуємо розв'язок: $w_1 = 0{,}1832$, $w_2 = 0{,}1504$, $w_3 = 0{,}1442$.

І нарешті, маємо третій варіант формування спільного портфеля (435). У цьому разі $x^*_{опт} = (000100010000001)$, тобто перший учасник придбає для спільного портфеля 4-й вид, другий – 3-й вид, а третій – 5-й вид цінних паперів. Визначимо частки капіталів інвесторів, вкладених у цінні папери, за допомогою рівнянь (6), (7), (9). Отримуємо розв'язок: $w_1 = 0{,}1442$, $w_2 = 0{,}1832$, $w_3 = 0{,}1504$. У результаті побудовано паттерн

$$\Pi = \begin{pmatrix} 453 \\ 354 \\ 4\ \ 3\ \ 5 \end{pmatrix}.$$

Отриманий гетерогенний паттерн можна представити у вигляді таблиці (табл. 9).

На основі даної таблиці та відомостей про частки капіталу кожного із учасників, вкладених у спільний портфель, маємо таблицю 10.

Таблиця 9

Варіанти спільного портфеля

Учасники	Перший	Другий	Третій	Ризик
Варіант 1	4	5	3	0,042
Варіант 2	3	5	4	0,042
Варіант 3	4	3	5	0,042

Таблиця 10

Розподіл капіталовкладень

Учасники	Перший	Другий	Третій
Варіант 1	0,09	0,25	0,08
Варіант 2	0,18	0,15	0,14
Варіант 3	0,14	0,18	0,15

Оскільки для формування спільного портфеля перший учасник може виділити не більше 0,2 частки від власного капіталу, другий – не більше 0,2, третій – не більше 0,1, то другий варіант розподілу капіталовкладень є найбільш прийнятним.

Аналіз одержаних результатів. Перевіримо правильність побудови гетерогенного партерна Π у випадку такого формування спільного портфеля, виходячи з умови (11). Оскільки спільний портфель сформований з паперів виду q_3, q_5, q_4, що мають відповідно ефективність E_3, E_5, E_4, а значить коваріаційна матриця цінних паперів, обчислена за допомогою (10), матиме вид (табл. 11).

Таблиця 11

Коваріаційна матриця цінних паперів спільного портфеля

V_{ij}	E_3	E_5	E_4
E_3	0,025	0,012	0,022
E_5	0,012	0,022	0,017
E_4	0,022	0,017	0,021

Обчислення за формулою (11) представлені у таблиці 12. Сума ризиків портфелів інвесторів дорівнює $\sum_{j=1}^{nn}\sqrt{\sum_{i=1} w_i V_{ij} \cdot w_j} = 0{,}0379$. Оскільки ризик спільного портфеля дорівнює $\sigma_\Sigma = 0{,}0421$, то умова (11) виконується, а значить гетерогенний паттерн побудовано правильно. Дана задача має три варіанти розв'язку. Оптимальним є другий варіант.

Таблиця 12

Перевірка побудови гетерогенного паттерна

$w_i V_{ij} w_j$	w_3	w_5	w_4	$\sum_{i=1}^{n} w_i V_{ij} w_j$	$\sqrt{\sum_{i=1}^{n} w_i V_{ij} w_j}$
w_3	$2{,}7E^{-5}$	$4{,}5E^{-5}$	$4{,}2E^{-5}$	0,000114	0,0106964
w_5	$4{,}5E^{-5}$	$7{,}62E^{-5}$	$7{,}05E^{-5}$	0,000192	0,0138605
w_4	$4{,}2E^{-5}$	$7{,}05E^{-5}$	$6{,}51E^{-5}$	0,000178	0,013324

На рис. 2 зображено графову модель сформованого оптимального портфеля, що відповідає другому варіанту розв'язку побудованого гетерогенного паттерна.

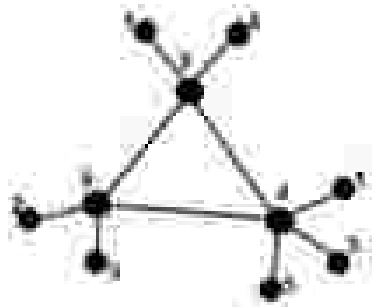


Рис. 2. Графова модель сформованого спільного портфеля

Слід відмітити, що графова структура, наведена на рисунку 2, є моделлю n-мірного гетерогенного паттерна Π, який дає уяву про множину розв'язків задачі, а також несе у собі інформацію про зв'язки між об'єктами. Така модель є умовною, так як у ній зв'язки відображують усні чи письмові домовленості між утримувачами часток капіталу $w_1, w_2, \ldots, w_5$.

Висновки. Задача формування й оптимізації спільного інвестиційного портфеля цінних паперів розв'язана за допомогою нового підходу з використанням побудови n-мірного паттерна на основі пофарбування відповідної теоретико-графової моделі зі зваженими вершинами. Розв'язок задачі пофарбування зваженого n-дольного графа дає такий розподіл цінних паперів між частками капіталу, при якому ризик портфеля є мінімальним.

При вирішенні поставленої задачі в роботі застосовуються: методи 0-1 програмування та квадратичної оптимізації для побудови математичної моделі n-мірного паттерна; евристичний метод пофарбування теоретико-графової моделі; об'єктно-орієнтований метод до програмної реалізації алгоритмів пофарбування графів.

Результати, що характеризують наукову новизну дослідження, полягають у наступному: набули подальшого розвитку математичні моделі побудови n-мірних паттернів; уперше для побудови n-мірних паттернів запропоновано комбінований метод розв'язування задачі оптимізації квадратичної функції з лінійними обмеженнями та метод 0-1 програмування.

Створені комп'ютерні програми «Pattern Designer Toolbox», «Latin Square Designer», «Pattern». Отримані алгоритми можуть бути використані у сферах управління, планування ресурсів, соціології, логістиці, а також при проектуванні систем автоматизованого управління підприємствами, що сприятиме підвищенню ефективності документообігу, ведення бухгалтерського обліку, планування сівозміни у сільському господарстві.

Результати дисертаційної роботи впроваджено в регіональній програмі Полтавського національного технічного університету імені Юрія Кондратюка «Теоретичні основи формування господарського механізму регіонів України» та у навчальному процесі на кафедрі економічної кібернетики при викладанні дисциплін «Теорія графів», «Автоматизовані системи управління», «Економічна кібернетика», «Математичне моделювання».

Автор вважає перспективним застосування розробленого підходу для розв'язання оптимізаційних економічних задач.

Бібліографічні посилання

1. **Гренандер У.** Лекции по теории образов / У. Гренандер; пер. с англ. И. Гуревича. – Т.3: Регулярные структуры. – М., 1983. – 430 с.
2. **Connolly D.** General Purpose Simulated Annealing // The Journal of the Operational Research Society: Mathematical Programming in Honour of Ailsa Land. – 1992. – V. 43, № 5. – P. 495 – 505.
3. **Abramson D.** A Simulated Annealing Code for General Integer Linear Programs / D. Abramson, M. Randall // Annals of Operations Research. – 1999. – V. 86, P. 3 – 24.
4. **Скрильник І.І.** Управління портфелем цінних паперів у випадку декількох інвесторів / І.І. Скрильник // Стратегические вопросы мировой науки — 2009 : V міжнар. наук.-практ. конф., 7-15 лютого 2009 р. : (Математичні методи в економіці): тези доп. – Przemysl: Nauka i studia, 2009. – С. 65–68.
5. **Скрильник І.І.** Алгоритм побудови n-мірних паттернів із накладеними обмеженнями / І.І. Скрильник // Вісник Запорізького національного університету : (Фізико-математичні науки. Біологічні науки) : наукові статті. — 2006. – № 1. – С. 110–116.
6. **Скрильник І.І.** Проблема побудови n-мірних паттернів / І.І. Скрильник // Вісник Східноукраїнського національного університету імені Володимира Даля : наукові статті. – 2009 – Ч. 2, № 1(131). – С. 245–252.
7. **Mertz P.** Genetic algorithms for binary quadratic programming / P. Mertz, B. Freisleben // In Proceedings of the 1999 International Genetic and Evolutionary Computational Conference (GECCO'99), Morgan Kaufmann. – 1999.
8. **Вороновский Г.К.** Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности / Г.К. Вороновский, К.В. Махотило, С.М. Петрашов, С.А. Сергеев] – Х., 1997. – С. 107–112.
9. **Boros E.** Pseudo-Boolean optimization / E. Boros, P. Hammer // Discrete Applied Mathematics. – 2002. – № 123 (1–3).

10. **Скрильник І.І.** Розв'язання NP-складних задач як пофарбування теоретико-графових моделей за допомогою генетичного алгоритму / І.І. Скрильник // Вісник Тернопільського державного університету : (Фізико-математичні науки) : наукові статті. – 2007. – № 2. – С. 166–176.
11. **Скрильник І.І.** Пофарбування графів за допомогою генетичного алгоритму / І.І. Скрильник // Вісник Тернопільського державного університету : (Математичне моделювання. Математика. Фізика) : наукові статті. – 2010. – № 1. – С. 194–203.

Надійшла до редколегії 22.10.10

УДК 519.5

[6]Е.А. Татаринов
Институт прикладной математики и механики НАНУ, г. Донецк

ВОССТАНОВЛЕНИЕ ГРАФА ПРИ ПОМОЩИ КАМНЕЙ

Анот

Анот

Анот

Ключевые слова:

Введение. Рассматривается задача распознавания конечного, неориентированного графа, без петель и кратных ребер при помощи агента, который перемещается по нему и изменяет метки на вершинах и ребрах графа. В [1] агент расставляет метки на концах ребер графа (инциденторах ребер) и использует D красок, D – максимальная степень вершины в графе и $D \leq n-1$. В [2–5] предложены алгоритмы восстановления графа, в которых агент метит вершины и инциденторы, при этом использует две краски и $n/2$ камней. Предлагаемый алгоритм является модификацией «Базового алгоритма» предложенного в [3]. **Целью данной работы** было построение алгоритма восстановления графа, для которого $T^*(n) = O(n)$. Предложен новый алгоритм восстановления неизвестного графа, где оптимизируется количество шагов и различных камней, которые используются при проведении эксперимента по восстановлению графа. Уменьшение количества шагов происходит за счет использования агентом дополнительных ресурсов (камней и счетчиков). Данный алгоритм основан на методе восстановления графа при помощи построения на его вершинах неявной нумерации [4].

Необходимые определения и понятия. Рассматриваются конечные, неориентированные графы без петель и кратных ребер. Все неопределяемые понятия общеизвестны и их можно найти, например, в [6–8]. Пусть $G = (V, E)$ – граф, у которого V – множество вершин, E – множество

ребер, $E \subseteq V \times V$, мощности n и m соответственно. Ясно что $m \le \frac{n(n-1)}{2}$. Тройку $((u, v), v)$ будем называть инцидентором («точкой прикосновения») ребра (u, v) и вершины v. Множество таких троек обозначим I. Множество $L = V \cup E \cup I$ назовем множеством элементов графа G. Краской будем называть ресурс, который имеется у агента в неограниченном количестве, а камнем – ресурс, имеющийся у агента в единичном экземпляре. Краски и камни образуют множество меток – M, которые могут быть использованы при раскраске элементов графа G. Если элемент графа метится некоторой краской, то предыдущая – стирается и вместо нее наносится новая. Если элемент графа метят камнем, то существующая на нем краска не уничтожается, а становится «невидимой» до тех пор, пока камень будет находиться на элементе. Функцией раскраски графа G назовем отображение $\mu : L \to M$, где $M = \{w, b, r, 1, .., t\}$, w интерпретируется как белый цвет (краска), r – красный, b – черный, а множество чисел $1, .., t$ интерпретируется как множество попарно различных камней.

Последовательность $u_1, u_2, .., u_k$ попарно смежных вершин называется путем длины k в графе G. При $u_1 = u_k$ этот путь называется циклом. Окрестностью $O(v)$ вершины v будем называть множество элементов графа, состоящее из вершины v, всех вершин u смежных с v, всех ребер (v, u) и всех инцидентов $((v, u), v), ((v, u), u)$. Будем обозначать $O_k(v)$ – k - окрестностью вершины v. $O_k(v)$ будет содержать в себе объединение всех окрестностей вершин графа u, в которые существует простой путь из вершины v длины не более чем k. Через χ обозначим цикломатическое число графа, равное $m - n + 1$. Изоморфизмом графа G и графа H назовем такую биекцию $\varphi : V_G \to V_H$, что $(u, v) \in E_G$ точно тогда, когда $(\varphi(u), \varphi(v)) \in E_H$. Изоморфные графы равны с точностью до обозначения вершин и раскраски их элементов.

Мобильный агент A характеризуется следующими свойствами. Он передвигается по графу из вершины v в вершину u по ребру (v, u). При этом он может изменить окраску вершин v, u, ребра (v, u), инцидентов

$((v,u),v),((v,u),v)$ метками из множества M. Находясь в вершине v агент A воспринимает («видит») метки всех элементов окрестности $O(v)$ и на этом основании определяет по какому ребру (v,u) он будет перемещаться и как будет перекрашивать элементы из $O(v)$. Агент A обладает конечной, неограниченно растущей внутренней памятью, в которой фиксируется результат функционирования агента на каждом шаге, и, кроме того, постепенно выстраивается представление графа G, в начале неизвестного агенту, списками ребер и вершин. Вершину, где агент находится в данный момент, будем называть текущей. Будем говорить, что агент «зациклился», если он попал в вершину, откуда он может попасть только в ранее посещенные вершины. Помеченным путем будем назвать простой путь, образованный последовательностью вершин помеченных камнями в порядке их посещения. Таким образом, длина помеченного пути не более чем n.

Нумерацией $F(V) \to N$ вершин графа G называется инъективное отображение множества его вершин во множество N натуральных чисел. Номер вершины v в нумерации F обозначается $F(v)$. M-нумерацией вершин графа называется нумерация вершин графа, соответствующая порядку их обхода при поиске в глубину [6]. Древесными ребрами называются [6] ребра, порождающие дерево поиска при обходе графа в глубину, остальные ребра – обратными.

Через $T(n)$ обозначим временную сложностью выполнения алгоритма, $S(n)$ – емкостную сложность, а через $T^*(n)$ и $T_*(n)$, соответственно, верхнюю и нижнюю оценку временной сложности.

Постановка задачи. Пусть задан класс K графов: конечных, неориентированных, без петель и кратных ребер, – все элементы которых окрашены краской w. Задан агент, обладающий свойствами описанными в пункте 2. Требуется разработать такой алгоритм функционирования (т. е. передвижения по графу и раскраски его элементов) агента A, что он, будучи помещен в произвольную вершину произвольного неизвестного агенту графа $G(K)$, через конечное число шагов построит граф H, изоморфный G, т. е. восстановит G.

Алгоритм восстановления графа. Стратегия восстановления. Алгоритм основан на методе обхода графа в глубину. Стратегия обхода гра-

фа в глубину хорошо известна [7–8]. Она изменена с учетом того, что агенту граф изначально неизвестен, агент может воспринимать локальную информацию о графе и агент строит на вершинах графа нумерацию в порядке посещения вершин (M-нумерация).

Реализовывать нумерацию агент будет неявно, при помощи набора камней. При этом для каждого камня i существует счетчик количества непомеченных ребер, которые есть в окрестности текущей вершины, перед тем как агент установит в ней камень i. Счетчик уменьшается на единицу, когда агент «видит» в окрестности текущей вершины вершину помеченную камнем i. Когда счетчик обнулится, камень станет свободным и агент перейдет в вершину, снимет с нее камень i, вершину пометит краской b и вернется обратно. Агент использует для пометки свободный, на данный момент, камень. Камень считается свободным, если его счетчик равен нулю. Свободный на данный момент камень – свободный камень с наименьшим номером. Камень i ассоциируется с M-номером, неявно присвоенным вершине в момент установки в ней камня i. Такая ассоциация сохраняется до тех пор, пока камень не станет свободным.

В процессе обхода графа в глубину агент строит неявное дерево поиска в глубину, относительно которого ребра графа разбиваются на два множества: древесные и обратные. Древесные ребра проходятся два раза – в прямом и обратном направлении. При первом прохождении по древесному ребру оно восстанавливается в графе G и один из его инцидентов метится краской r. После перехода по ребру агент устанавливает в новой вершине свободный камень и восстанавливает все обратные ребра текущей вершины. Для восстановления обратных ребер агенту не требуется переходить по ним поскольку, находясь в текущей вершине помеченной камнем i, он увидит камень j, установленный в вершине, с которой смежно восстанавливаемое обратное ребро. Зная i и j, агент может однозначно определить ассоциируемые с ними неявные M-номера и восстановить обратное ребро. При установке в вершине камня она добавляется в помеченный путь, при пометке ее краской b – удаляется из помеченного пути.

Алгоритм останавливается, когда помеченный путь будет пустым, а все вершины помечены краской b.

Алгоритм. *Вход:* граф G неизвестный агенту, все элементы графа G окрашены краской w, агент помещен в произвольную вершину s.

Выход: все элементы графа G окрашены краской b, агент находится в вершине s; список вершин V_H и ребер E_H графа H, изоморфного G.

Данные: v – текущая вершина, *Сч* – счетчик числа посещенных вершин графа G, V_H и E_H множества вершин и ребер графа H соответственно, список $k(1)..k(t)$, в котором на i-ом месте находится номер вершины в графе H, соответствующей вершине помеченной камнем i в графе G, p – длина списка и совпадает с количеством камней у агента; список $l(1)..l(t)$, в котором на i-ом месте находится число не помеченных вершин видимых из вершины помеченной камнем i в момент установки в вершине камня; j – счетчик элементов списка используется для перебора элементов списка; переменная *peble* – используется для хранения номера найденного свободного камня и для временного хранения номера камня текущей вершины.

Изначально списки $k(1)..k(t)$ и $l(1)..l(t)$ заполнены нулями, а множества V_H и E_H пустые.

1. $Сч := 1$;
2. $V_H := \{1\}$;
3. *ПОКРАСИТЬ* (v);
4. *if* в $O(v)$ есть u с $\mu(u) = w$ *then* *ВПЕРЕД* (v); *ИССЛЕДОВАТЬ* (v); *goto* 4;
5. *if* в $O(v)$ есть u с $\mu(v,(u,v)) = r$ *then* *НАЗАД* (v); *goto* 4;
6. Печать V_H и E_H;

В строках 1 – 3 происходит инициализация алгоритма. Начальная вершина метится камнем 1, ей неявно присваивается номер 1 и она добавляется в помеченный путь. Для камня 1 заполняется счетчик количества непомеченных вершин видимых из вершины v, процедура *ПОКРАСИТЬ* (v). Формируется начальное значение множества вершин графа H.

В строках 4 – 5 происходит анализ окрестности $O(v)$ и выбор одного из двух вариантов передвижения. Строка 4, если в окрестности $O(v)$ есть не помеченная вершина, то агент движется вперед и устанавливает в вершине камень и восстанавливает обратные ребра с не помеченными инцидентами, процедуры *ВПЕРЕД* (v) и *НАЗАД* (v). Строка 5, если в окрестности вершины нет не помеченных вершин, но есть ребро, ближний ин-

цидентор которого помечен краской r, агент совершает отход по нему назад и закрашивает краской b уже исследованную вершину и древесное ребро, процедура *НАЗАД* (v). Если же нет такого ребра, то алгоритм завершит работу и печатаются множества V_H и E_H. Заметим, что в строках $4-5$, после выполнения процедур, происходит переход к началу выполнения строки 4 (*goto* 4).

Рассмотрим подробнее процедуры *ВПЕРЕД* (v), *ПОКРАСИТЬ* (v), *ИССЛЕДОВАТЬ* (v), *НАЗАД* (v).

ВПЕРЕД (v)

1. Перейти по ребру (v,u) у которого $\mu(u,v)=$;
2. $\mu(u,(v,u))=r$;
3. $C_4:=C_4+1$;
4. $E_H:=E_H\cup\{k(\mu(v)),C_4\}$;
5. *ПОКРАСИТЬ* (v);

В строке 1 агент выбирает ребро для перехода в не помеченную вершину. В строке 2 метит инцидентор ребра, по которому он перешел, краской r. Увеличивает счетчик посещенных вершин графа G и восстанавливает обратное ребро (строки $3-4$). В строке 5 агент устанавливает в текущей вершине свободный камень (выполняется процедура *ПОКРАСИТЬ* (v) для текущей вершины).

ПОКРАСИТЬ (v)

1. *peble* := 1;
2. *for* j := 1 *to* *p* *step* 1 *do*
3. *if* $l(j)\neq 0$ *then* *peble* := j; *break*;
4. *if* *peble* = 1 *then* «Ошибка! Не хватило камней»; Конец;
5. $\mu(v):=$ *peble*
6. $k(\mu(v)):=C_4$;
7. *foreach* $u\in O(v)$ *do*
8. *if* $\mu(u)=w$ *then* $l(\mu(v)):=l(\mu(v))+$;

В строке переменная *peble* инициализируется заведомо не существующим номером камня. В строках 2 – 3 выполняется цикл, который перебирает элементы списка $l(1)..l(t)$ для поиска свободного камня. В строке 3 проверяется : равен ли нулю соответствующий элемент списка. Если равен, то в переменную *peble* записывается номер свободного камня и завершается цикл. В строке 4 проверяется : найден ли свободный камень, если он не найден , то выдается сообщение «Ошибка! Не хватило камней» и агент завершает свою работу. Если же свободный камень найден *peble*, то агент устанавливает в текущей вершине свободный камень, строка 5. В строке 6 в элемент номер $\mu(v)$ списка $k(1)..k(t)$ записывается номер вершины графа H, соответствующей вершине помеченной камнем $\mu(v)$ в графе G. В строках 7 – 8 агент анализирует вершины из $O(v)$ и подсчитывает количество не помеченных вершин, которое он запишет в счетчик камня *peble*.

ИССЛЕДОВАТЬ (v)

1. *foreach* $u \in O(v)$ è u *vdo*
2. *if* $\mu(u) = w$ è $(v,u) \in E$ *wthendobegin*
3. $l(\mu(u)) := l(\mu(u)) + 1$;
4. $E_H := E_H \cup \{k(u), k(v)\}$;
5. $\mu(v,u) := b$
6. *if* $l(\mu(u)) = 0$ *thenbegin*
7. *peble* $:= \mu(v)$;
8. агент переходит по ребру (u,v);
9. снимает камень с вершины;
10. $\mu(u) := b$;
11. вернуться в вершину помеченную камнем *peble*;
12. *end*;
13. *end*;
14. *if* $l(\mu(v)) = 0$ *thenbegin*
15. снять с вершины v камень;

16. $\mu(v,u) = b$;
17. *end*;

В строке 1 агент анализирует все вершины из $O(v)$. Если вершина не помечена w и инцидентор $\mu(v,(v,u))$ (строка 2), то агент уменьшает на единицу счетчик не помеченных вершин для соответствующего камня (строка 3). Добавляет в E_H ребро, которое соединяет вершины помеченные камнями $\mu(u)$ и $\mu(v)$, используя неявную нумерацию. В строке 5 инцидентор $\mu(v,(v,u))$ метится краской b. Если счетчик не помеченных вершин для камня $\mu(u)$ стал равен нулю (строка 6), то агент запоминает номер текущей вершины в переменную *peble* (строка 7), переходит по ребру в вершину u, снимает камень и метит ее краской b (строки 8–9), после чего возвращается в текущую вершину, помеченную камнем *peble*. В строке 14 проверяется счетчик не помеченных вершин видимых из вершины v, если он равен нулю, то агент снимает с вершины камень и метит ее краской b (строки 15–16).

НАЗАД (v)

1. прейти по ребру (u,v) у которого $\mu(v,(v,u)) = r$;
2. $\mu(v,(v,u)) := b$;

В строке 1 агент выбирает ребро (u,v), ближний инцидентор которого помечен r и переходит по нему. При переходе по нему он метит инцидентор краской b (строка 2).

Сходимость и сложность алгоритма восстановления графа. Для начальной вершины графа G агент выполняет только процедуру *ПОКРАСИТЬ* (v), однако он добавляет во множество V_H вершину и тем самым присваивает начальной вершине неявный номер 1. После этого агент выполняет процедуру *ВПЕРЕД* (v) и *ИССЛЕДОВАТЬ* (v) для всех остальных вершин отличных от начальной вершины. Если движение вперед возможно, то агент переходит в новую вершину и метит ее свободным камнем, при этом в графе H агент создает новую вершину. Таким образом, в процессе блуждания агент строит на вершинах графа G неявную нумерацию, соответствующую явной нумерации вершин в графе H. Таким обра-

зом, осуществляется отображение $\varphi: V_G \to V_H$, при этом равенство $\varphi(v_0) = v$ устанавливается тогда, когда агент устанавливает камень в вершине v и сохраняется до тех пор, пока агент не снимет камень с вершины v, предварительно не посетив все не помеченные ребра из $O(v)$. Такая нумерация будет биекцией, поскольку в связном, неориентированном графе G все вершины достижимы из начальной вершины, т. е. агент в каждой из них установит свободный камень.

Из описания алгоритма следует, что агент посетит все вершины графа, и в каждой из них будет установлен свободный камень, и, тем самым, будет установлена неявная нумерация вершин графа G. Из каждой вершины агент выполнит процедуру *ВПЕРЕД* (v), если это возможно, восстановив при этом древесное ребро (u, v) графа G, и при этом неявно нумерует вершину u так, что ему однозначно соответствует ребро $(\varphi(v), \varphi(u))$ в графе H. Далее агент выполняет процедуру *ИССЛЕДОВАТЬ* (v) для вершины u, восстанавливая при этом обратные ребра при помощи неявной нумерации вершин графа G, при этом каждому обратному ребру (u, v) графа G будет однозначно соответствовать ребро $(\varphi(u), \varphi(v))$ в графе H. Поскольку алгоритм основан на стратегии обхода (поиска) графа в глубину, значит, все ребра графа либо древесные, либо обратные. Это следствие теоремы 23.9 [8, с. 452], поэтому агент восстановит все ребра графа H.

Таким образом, φ будет изоморфизмом графов G и H. Из проведенных выше рассуждений следует справедливость следующей теоремы.

Теорема 1. *Выполняя алгоритм восстановления графа, агент восстановит любой граф G, с точностью до изоморфизма.*

Подсчитаем временную и емкостную сложность алгоритма восстановления графа в равномерной шкале [7], а также количество различных красок и камней, используемых агентом.

Будем предполагать, что за одну единицу времени агент может выполнить все или часть следующих действий: 1) установить или снять камень; 2) изменить метку на вершине и / или инциденторе; 3) найти свободный камень; 4) заполнить счетчик не посещенных вершин для заданного камня; 5) анализировать метки на элементах 1 – окрестности текущей вершины; 6) добавлять вершины и ребра во множества V_H и E_H. Предполага-

ется, что переход по ребру из вершины в вершину выполняется за время много большее, чем все остальные действия. Поэтому подсчитаем, сколько переходов по ребрам сделает агент при выполнении алгоритма восстановления графа. При выполнении процедур *ВПЕРЕД* (v) и *НАЗАД* (v) агент выполняет один переход по ребру. При помощи этих двух процедур агент организовывает проход по неявному дереву поиска в глубину, тогда общее время выполнения этих двух процедур потребует $2(n-)$ шага. При выполнении процедуры *ИССЛЕДОВАТЬ* (v), агент выполняет по два перехода по ребру : агент снимает с вершины камень и метит ее краской b. Поскольку пометка вершины краской b выполняется только один раз для каждой вершины, то общее время выполнения процедуры *ИССЛЕДОВАТЬ* (v) потребует $2(n-)$ шага, для начальной вершины процедура не выполняется.

Следовательно, $T(n)=4(n)$ шага и будет величиной $O(n)$.

В процессе выполнения алгоритма агент использует списки $k(1)..k(p)$ и $c(1)..c(p)$, длины p, две локальные переменные j, *peble*, и одну глобальную *Сч*. Агент строит множество вершин V_H и ребер E_H, для хранения которых потребуется n и $2m$ ячеек памяти соответственно. Значит, $S(n)=2p+n+3+n+2m$ ячеек памяти и будет величиной $O(m+n)$.

При выполнении алгоритма агент использует две краски b и r, для пометки вершин используются камни. Камни могут быть использованы повторно, поэтому их количество будет зависеть от длины помеченного пути. При выполнении процедуры *ВПЕРЕД* (v) длина помеченного пути увеличивается, а при выполнении процедура *НАЗАД* (v) – уменьшается. Следовательно, длина красного пути может быть оценена сверху максимальной высотой дерева при поиске в глубину, обозначим эту величину h_t. В общем случае эта величина соизмерима с n. Однако повторное использование камней позволит получить оценки лучшие чем n. Из проведенных выше рассуждений следует справедливость следующей теоремы.

Теорема 2. *Агент восстановит любой граф G, с точностью до изоморфизма, за время $O(n)$, при этом будет использовано $O(m+n)$ ячеек памяти, две краски и h_t камней.*

Рассмотрим частные случаи видов графов, для восстановления которых агент использует конечное число камней.

Графом вида «Линия» будем назвать цепочку вершин последовательно соединенных друг с другом.

Лемма 1. Для восстановления графа вида «Линия» агент использует не более трех различных камней.

Доказательство. Рассмотрим худший случай, когда агент попадает в вершину удаленную от концов «Линии». Тогда, согласно алгоритму, в ней он установит камень 1, и перейдет в соседнюю вершину, и установит в ней камень 2, после чего перейдет в соседнюю вершину и пометит ее камнем 3, и при этом камень 2 станет свободным, а вершину агент пометит краской b. После перехода в соседнюю не помеченную вершину агент пометит ее камнем 2, при этом камень 3 станет свободным, а вершину агент пометит b. И так далее, чередуя использование камней 2 и 3, агент дойдет по конца «Линии», а потом вернется в начальную вершину, и так же, чередуя использование камней 2 и 3, дойдет до другого конца «Линии» и тем самым восстановит граф с использованием только трех красок.

Графом вида «Простой цикл» будем называть граф вида «Линия», у которого первая и последняя вершины совпадают.

Лемма 2. Для восстановления графа вида «Простой цикл» агент использует не более трех камней.

Доказательство. Рассмотрим, как будет использовать агент в данном случае камни. В начальной вершине он установит камень 1, и перейдет в соседнюю и пометит ее камнем 2, после чего перейдет в соседнюю не помеченную вершину и пометит ее камнем 3. При этом камень 2 станет свободным, а вершину агент пометит b. После перехода в соседнюю не помеченную вершину агент пометит ее 2, при этом камень 3 станет свободным, а вершину агент пометит краской b. Там образом, чередуя использование камней, агент дойдет до начальной вершины и вернется обратно по пройденному пути и тем самым восстановит граф.

При добавлении к графу вида «Простой цикл» вершины в центр и соединения ее с f вершинами колеса получится граф вида «Колесо с f спицами».

Лемма 3. Для восстановления графа вида «Колесо с f спицами» агенту потребуется не более $f+2$ камней.

Доказательство. Рассмотрим случай, когда агент будет проходить по вершинам колеса, не переходя по спицам. Этот случай заведомо хуже других, поскольку камни установлены в вершинах, которым инциденты спицы, не будут сразу же освобождаться, как это было описано в лемме 2.

Следовательно, будет f не свободных камней. Как и в лемме 2 агенту потребуется, два камня, чередуя использование которых, агент сможет пройти по вершинам колеса. Таким образом, всего потребуется $f+2$ камня.

Рассмотрим граф вида треугольник, будем обозначать верхнюю его вершину c, левую нижнюю его вершину a, b – правую нижнюю вершину. Пусть есть q треугольников, отождествим в одну вершину c все вершины $c_1, \ldots, c_q$, полученный граф будем называть «q-мельницей».

Лемма 4. Для восстановления графа вида «q-мельница» агенту потребуется не более четырех камней.

Доказательство. Рассмотрим два вариант начального положения агента: агент помещается в вершину c и агент помещается в вершину a_i. Случай, когда агент помещается в вершину b_i аналогичен случаю, когда агент помещается в вершину a_i.

Рассмотрим случай, когда агент помещается в вершину c, в этом случае агент установит в ней камень номер 1, после чего перейдет в вершину b_j (a_j) и установит в ней камень 2, после чего переходит в вершину a_j (b_j) и установит в ней камень 3. В результате чего камень 2 станет свободным, агент пометит вершину краской b. После чего он прейдет вершину c и, чередуя использование камней 1, 3, выполнит аналогичные действия для оставшихся вершин b_i (a_i). Таким образом, будет использовано три камня.

Рассмотрим случай, когда агент помещается в вершину a_i, в этом случае агент установит в ней камень 1. После чего перейдет в вершину c установит в ней камень 2, и перейдет в вершину a_j (b_j), где $ij \neq$, и установит в ней камень 3. После чего он прейдет в вершину b_j (a_j) и установит в ней камень 4. В результате чего камни 3 и 4 станут свободными, а агент пометит вершины краской b. Таким образом, будет использовано не более четырех красок.

Модификации алгоритма. Поскольку в прикладных задачах камни могут быть дорогостоящим или ограниченным ресурсом, то естественным образом возникает задача уменьшения количества используемых агентом камней. Ниже будут рассмотрены модификации алгоритма восстановле-

ния графа, позволяющие уменьшить, количество камней, используемых при восстановлении графа. Их сокращение происходит за счет проведения дополнительного анализа меток элементов из области восприятия агента и / или расширения области восприятия агента.

Модификация 1. Уменьшение количества используемых агентом камней можно добиться путем модифицирования процедур *ПОКРАСИТЬ* (v) и *ИССЛЕДОВАТЬ* (v). Поскольку агент, приходя в вершину , сразу же устанавливает в ней свободный камень, не анализируя, нужно ли вообще ставить в ней камень. Можно выделить два вида вершин для которых нет необходимости использовать камни: 1) вершины из которых не видно непомеченных; 2) вершины из которых видно только одну не помеченную вершину.

Первые не нужно метить камнем, поскольку это вновь добавленная вершина и ей соответствует вершина номер *Сч* в графе H, поэтому восстановление обратных ребер можно провести без использования камня на вершине.

Вторые метить не следует, поскольку из этой вершины, назовем ее v_1, агент может попасть только в одну не помеченную вершину, назовем ее v_2. Таким образом, он посещает последнюю не помеченную вершину , которую было видно из v_1. Таким образом, если камень был бы установлен в v_1 он сразу же и освободился, а ребро , по которому агент перешел в v_2, было бы восстановлено как древесное.

Рассмотрим, как изменится процедура алгоритма восстановления графа для данной модификации. В процедуре *ВПЕРЕД* (v) будет убрана строка 5, а процедуры *ИССЛЕДОВАТЬ* (v) и *ПОКРАСИТЬ* (v) изменятся следующим образом.

ИССЛЕДОВАТЬ (v)

1. *foreach* $u \in O($) и $u \neq v$ *do*
2. *if* $\mu(u) \neq w$ и $\mu(v,(vu)) = w$ *then do begin*
3. $l(\mu(u)) := l($ $(u))$;
4. ; $E_H := E_H \cup \{k(u), Сч\}$
5. $\mu(v,(uv)) := b$

6. *if* $(\mu(u))=0$ *then begin*
7. $peble := \mu(v)$;
8. агент переходит по ребру (u,v);
9. снимает камень с вершины;
10. $\mu(u) := b$;
11. вернуться в вершину помеченную камнем $peble$;
12. *end*;
13. *end*;
14. *if* $(\mu(v))=0$ *then begin*
15. снять с вершины v камень;
16. $\mu(v) := b$;
17. *end*;
18. *ПОКРАСИТЬ* (v);

В строке 4 вместо $k(\mu())$ используется значение счетчика вершин *Сч* и в строке 18 добавлен вызов процедуры *ПОКРАСИТЬ* (v).

ПОКРАСИТЬ (v)

1. $peble := -1$;
2. *for* $j = 1$ *to* p *step* 1 *do*
3. *if* $l(j) \neq 0$ *then* $peble := j$; *break*;
4. *if* $peble = -1$ *then* «Ошибка! Не хватило камней»; Конец;
5. $j := 0$;
6. *for each* $u \in O(v)$ и $u \neq v$ *do*
7. *if* $\mu(u) = w$ *then* $j := j + 1$;
8. *if* $j \mu 2$ *then* (v) b;
9. *else begin*
10. $\mu(v) := peble$
11. $l(\mu(v)) := j$;
12. *end*;

До строки 4 процедура выполняется так же , как и в алгоритме восстановления. В строке 5 сбрасывается счетчик j перед подсчетом количества не помеченных вершин видимых из вершины v. В строках 6 - 7 агент анализирует 1 – окрестность вершины v. Если он видит не помеченную вершину, то увеличивает на единицу значение j. Далее анализируется значение j. Если $j < 2$, то из вершины v, либо не видно не помеченных вершин, либо видна только. В любом из этих случаев вершина не метится камнем, и она закрашивается краской b (строка 8). В противном случае – вершине присваивается свободный камень $peble$, а счетчику не помеченных вершин вершины v присваивается значение j (строки 10–11).

В алгоритм, модифицированный таким образом , добавляется дополнительный анализ меток на элементах графов из области восприятия. Такой анализ не требует проведения дополнительных сложных вычислений, поэтому они не приведут к существенному усложнению реализации модификации 1. Однако, это позволяет уменьшить количество камней используемых для восстановления некоторых видов графов. Ясно, что, выполняя модификацию 1, агент будет использовать не большее количество камней, чем алгоритм восстановления графа. Приведем ниже несколько лемм, иллюстрирующих случаи, когда агент будет использовать меньшее количество камней, чем в основном алгоритме.

Лемма 5. Любая висячая вершина графа может быть восстановлена без использования камней.

Доказательство. Действительно, поскольку есть только одно ребро (xz) в графе G, где z висячая вершина, по которому можно попасть в висячую вершину, то агент сначала посетит вершину x, установит в ней камень, и только потом попадет в вершину z, но из нее не будет видно непомеченных вершин. Следовательно, не будет использован камень.

Лемма 6. Для восстановления графа вида «Линия» агент, выполняя модификацию 1 алгоритма восстановления графа, использует один камень.

Доказательство. Воспользуемся рассуждениями леммы 1 заметив, что вершины, которые агент метит камнями 2 и 3 будут вершинами, из которых видно только одну не помеченную вершину. Следовательно, агенту не потребуются камни для их пометки, а только один камень для пометки начальной вершины.

Лемма 7. Для восстановления графа вида «Простой цикл» , агент, выполняя модификацию 1 алгоритма восстановления графа, использует один камень.

Доказательство. Воспользуемся рассуждениями леммы 1, заметив, что вершины, которые агент метит камнями 2 и 3 будут вершинами, из которых видно только одну не помеченную вершину. Следовательно, агенту не потребуются камни для их пометки, а только один камень для пометки начальной вершины.

Модификация 2. Уменьшение количества использованных камней можно добиться путем увеличения области восприятия агента, т. е. метить камнями все вершины из $O_k(v)$, где v – текущая вершина, выполняя для них процедуры *ПОКРАСИТЬ* (v) и *ИССЛЕДОВАТЬ* (v). При этом камни на вершинах, которые не будут принадлежать границе k-окрестности, будут свободными.

Алгоритм восстановления графа модифицируется следующим образом. Вместо пометки и исследования текущей вершины, агент помечает свободными камнями и исследует все вершины из $O_k(v)$, где v – текущая вершина. Затем выбирает из $O_k(v)$ вершину u, помеченную камнем, и переходит в нее, отмечая по пути инциденторы ребер краской r. Агент снова помечает и исследует вершины из $O_k(u)$. Если нет вершины для прохода вперед, то агент отступает по ребрам, инциденторы которых помечены краской r, при этом инцидентор красится краской b.

Рассмотрим подробно случай при $k=1$. Изменению будет подвергнута только основная часть алгоритма и процедуры *ВПЕРЕД* (v), в ней изменится условие выбора новой текущей вершины и не будет восстановления древесного ребра, поскольку агент не переходит в не помеченную вершину. Процедуры: *НАЗАД* (v), *ПОКРАСИТЬ* (v), *ИССЛЕДОВАТЬ* (v), не изменяются, а используются, в зависимости от того была модификация 3 применена к алгоритму восстановления или к одной из его модификаций.

1. $Сч:=1;$
2. $V_H := \{1\};$
3. *ИССЛЕДОВАТЬ* (v);
4. *ПОКРАСИТЬ* (v);
5. *if* и $O_k(v)$ есть u с $\mu(uv)=$ *thenbegin*
6. $Сч:=Сч+1;$
7. *ПОКРАСИТЬ* (u);

8. *ИССЛЕДОВАТЬ* (u);
9. *goto* 5;
10. *end*;
11. *if* в $O(v)$ есть u с $\mu(uv)\neq$ и $\mu(ub)\neq$ *then* *ВПЕРЕД* (v); *goto* 5;
12. *if* в $O(v)$ есть u с $\mu(v,(uv))=r$ *then* *НАЗАД* (v); *goto* 5;
13. Печать V_H и E_H;
14. Конец;

Заметим, то процедура *ПОКРАСИТЬ* (v) выполняется только в том случае, если она не выполняется в процедуре *ИССЛЕДОВАТЬ* (v).

ВПЕРЕД (v)

1. Перейти по ребру (v,u), у которого $\mu(uv)\neq$ и $\mu(ub)\neq$;
2. $\mu(u,(vu))=r$;
3. $Сч:=Сч+1$

Рассмотрим некоторые частные случаи видов графов, которые восстанавливаются агентом, выполняющим модификацию 2 алгоритма восстановления графа, при $k=1$, с использованием константного числа камней.

Лемма 8. Для восстановления графа вида «Линия» агент, выполняя модификацию 2 при $k=1$ алгоритма восстановления графа, использует не более трех камней.

Доказательство. Рассмотрим худший случай, когда агент попадает в вершину удаленную от концов «Линии». Тогда, согласно алгоритму, в ней он установит камень 1 и перейдет в правую вершину $1-$ окрестности, и установит в ней камень 2, после чего перейдет в левую вершину из $1-$ окрестности и установит в ней камень 3. В результате этого камень 1 станет свободным, и вершину агент пометит краской b. Далее агент переходит в вершину помеченную камнем 2 (3). Перейдя в правую (левую) вершину из ее $1-$ окрестности, установит в ней камень 1. В результате этого камень 2 (3) станет свободным, а вершину агент пометит краской b. И так далее, чередуя использование камней 2 (3) и 1, агент дойдет по конца линии, а потом вернется в начальную вершину, и так же, чередуя использование камней 3 (2) и 1, дойдет до другого конца линии и тем самым восстановит граф с использованием только трех красок.

Граф, полученный добавлением к каждой вершине «Линии» двух висячих вершин, будем назвать «Древовидным расширением линии». Для удобства будем предполагать, что одна присоединенная висячая вершина расположена выше лини, а вторая – ниже. Так же будем обозначать, вершины линии v_i, соответствующие висячие вершины с верху t_i, соответствующие висячие вершины снизу b_i, где $1 \le i \le n$ а n – количество вершин в исходной линии.

Лемма 9. Для восстановления «Древовидного расширения линии» агент, выполняя модификацию 2 , при $k = 1$, алгоритма восстановления графа, использует не более четырех камней.

Доказательство. Случай, когда агент попадает в вершину t_i (b_i), где и $2 \le i \le n-1$, не увеличивает количества камней, поскольку агент установит камень 1 в вершине t_i (b_i), а в вершине v_i он установит камень 2. В результате этого камень 1 станет свободным, а вершину агент пометит краской b и перейдет в вершину v_i. Таким образом, далее агент будет выполнять действия так же как, если бы он начал в вершине v_i и при этом у него будет использован только один камень.

Рассмотрим случай, когда агент попадает в некоторую вершину v_i, где и $2 \le i \le n-1$. Согласно алгоритму, агент установит в ней камень 1. Затем перейдет в вершину v_{i+1} и установит в ней камень 2. Затем агент перейдет в вершину (v_{i-1}) и установит в ней камень 3. После этого агент прейдет в висячую вершину t_i (b_i) и установит в ней камень 4, который сразу же станет свободным, а вершину агент пометит краской b. После этого агент перейдет в висячую вершину b_i (t_i) и установит в ней камень 4, который сразу же станет свободным, а вершину агент пометит краской b. В результате этого камень номер 1 станет свободным, агент пометит вершину краской b. После этого агент прейдет в вершину v_{i+1} (v_{i-1}). Затем агент перейдет в вершину v_{i+2} (v_{i-2}) и установит в ней камень 3 (2). После чего он перейдет в висячую вершину t_{i+1} (b_{i+1}) и установит в ней камень 4, который сразу же станет свободным, а вершину агент пометит краской b. После чего он перейдет в висячую вершину b_{i+1} (t_{i+1}) и установит в ней камень 4, который сразу же станет свободным, а вершину агент пометит краской b. В результате этого камень 3 (2) станет свобод-

ным, а вершину агент пометит краской b. После этого агент перейдет в вершину v_{i+2} (v_{i-2}). Затем агент перейдет в вершину v_{i+3} (v_{i-3}) и установит в ней камень 1. После чего он перейдет в висячую вершину t_{i+2} (b_{i+2}) и установит в ней камень 4, который сразу же станет свободным, а вершину агент пометит краской b. После чего он перейдет в висячую вершину b_{i+2} (t_{i+2}) и установит в ней камень номер 4, который сразу же станет свободным, а вершину агент пометит краской b. В результате этого камень номер 1 станет свободным, а агент покрасит вершину краской b. И так далее, чередуя использование красок 3 (2) и 1, агент дойдет до конца линии, вернется в начальную вершину и, чередуя использование красок 2 (3) и 1, агент дойдет до второго конца линии.

Таким образом, граф будет восстановлен с использованием четырех красок.

Для построения частных случаев видов графов рассмотрим графы, состоящие из трех вершин соединенных в виде треугольника, стоящего на основании. Будем обозначать верхнюю вершину b, левую нижнюю – a, правую нижнюю – c.

Будем называть «Треугольной линией» граф, образованный из треугольников, которые соединены следующим образом, вершина c_i отождествляется с вершиной a_{i+1}, где $i=\overline{1,..,1}$. Вершины, полученные в результате соединения, будем обозначать ca_i, где $i=\overline{0,..,1}$.

Лемма 10. Для восстановления графа вида «Треугольная линия» агент, выполняя модификацию 2, при $k=1$, алгоритма восстановления графа, использует не более четырех камней.

Доказательство. Рассмотрим случай, когда агент попадает в вершину ca_i, удаленную от начала и конца ленты. В этом случае агент согласно алгоритму, установит в ней камень номер 1, после чего он переходит в вершину ca_{i+1} (ca_{i-1}) и устанавливает в ней камень 2 (3). После этого агент переходит в вершину ca_{i-1} (ca_{i+1}) и устанавливает в ней камень 3 (2). После чего агент переходит в вершину b_i (b_{i+1}) и устанавливает в ней камень 4, который сразу же освобождается, а агент метит вершину краской b. После чего агент переходит в вершину b_{i+1} (b_i) и устанавливает в ней камень номер 4, который сразу же освобождается, а агент метит вер-

шину краской b. В результате этого освобождается камень номер 1, а агент метит вершину краской b.

Легко видеть, что другие варианты посещения $1-$ окрестности начальной вершины не приведут к увеличению количества используемых красок.

Далее агент переходит в вершину ca_{i-1} (ca_{i+1}). После чего он переходит в вершину ca_{i-2} (b_{i-2}) и устанавливает в ней камень 1. Поле чего агент переходит в вершину b_{i-2} (ca_{i-2}) и устанавливает в ней камень 4. В результате чего камень номер 3 станет свободным, агент пометит вершину краской b. И так далее, чередуя использование камней 1, 3, 4, агент дойдет до конца, линии вернется в начальную вершину и так же, чередуя использование камней 1, 2, 3, дойдет до другого конца ленты.

Случай, когда агент изначально помещается в вершину b_i, не увеличит количества используемых красок. Поскольку в этом случае агент, согласно алгоритму, установит в ней камень 1. Затем он перейдет в вершину ca_{i-1} (ca_i) установит в ней камень 2. Затем агент перейдет в вершину ca_i (ca_{i-1}) и установит в ней камень 3. В результате этого камень 1 станет свободным, а агент пометит вершину краской b. После чего он перейдет в вершину ca_{i-1} (ca_i) и при этом у него будет два занятых камня. Такое состояние соответствует одному из способов посещения вершин из $1-$ окрестности, если бы начальной вершиной была вершина ca_i.

Граф, полученный из треугольной ленты путем добавления ребер соединяющих вершины c_i и c_{i+1}, где $i=\overline{1,..,1}$ будем называть «Трапеция из треугольников».

Лемма 11. Для восстановления графа вида «Трапеция из треугольников» агент, выполняя модификацию 2, при $k=1$, алгоритма восстановления графа, использует не более шести камней.

Доказательство. Воспользуемся рассуждениями из леммы 10. Поскольку вершины c_i и c_{i+1} соединены ребрами, то камни установленные в них агентом не станут сразу же свободными. Следовательно, потребуется два камня для установки в них (камни 4 и 5). Поскольку агент, перейдя в вершину ca_{i+1} (ca_{i-1}), может начать посещение вершин из $1-$ окрестности ca_{i+1} (ca_{i-1}), с вершины ca_{i+2} (ca_{i-2}), в которой он установит сво-

бодный камень 1. Для пометки вершины b_{i+1} (b_{i-1}) агенту потребуется дополнительный камень 6. После этого камни 3 и 5 станут свободными. Чередуя использование камней 1, 3, 5, агент дойдет до одного конца ленты, после чего вернется в начальную вершину. Чередуя использование камней 1, 2, 3, агент дойдет до другого конца ленты. Следовательно, потребуется не более шести камней.

Рассмотрим графы вида квадраты, вершины, которого будем обозначать так: левая нижняя a, левая верхняя – b, правя верхняя – c, правая нижняя d.

«Ромбовидной лентой» будем называть граф образованный из квадратов следующим образом, вершина c_i и a_{i+1} отождествляются, где $i = 1,..., q-1$ а q – количество квадратов. Вершину, полученную в результате этого, будем обозначать ca_i.

Лемма 12. Для восстановления графа вида «Ромбовидная лента» агент, выполняя модификацию 2 , при $k = 1$, алгоритма восстановления графа, использует не более пяти камней.

Доказательство. Возможно два начальных положения агента – вершина ca_i и b_i, случай, когда агент помещается в вершину d_i, аналогичен случаю помещения агента в вершину b_i.

Рассмотрим случай, когда агент помещается в вершину ca_i, удаленную от начала и конца ленты. Согласно алгоритму агент установит в ней камень 1. После чего перейдет в вершину b_i (d_i) и установит в ней камень 2. После чего перейдет в вершину d_i (b_i) и установит в ней камень 3. После чего перейдет в вершину b_{i+1} (d_{i+1}) и установит в ней камень 4. После чего перейдет в вершину d_{i+1} (b_{i+1}) и усиановит в ней камень 5. В результате этого камень 1 станет свободным, а агент пометит вершину краской b. Далее агент переходит в вершину b_i или d_i (b_{i+1} или d_{i+1}). После чего агент переходит в вершину ca_{i-1} (ca_{i+1}) и устанавливает в ней камень 1. В результате этого становятся свободными камни 2 и 3 (4 и 5), а вершины агент метит краской b. И так далее, чередуя использование камней 1, 2, 3, (1, 4, 5) , агент дойдет до конца ленты, и вернется обратно в начальную вершину, а из нее, чередуя использование камней 1,2, 3, дойдет до другого конца ленты. Следовательно, будет использовано не более пяти различных камней.

Рассмотрим случай, когда агент помещается в вершину b_i, удаленную от начала и конца ленты. В этом случае, согласно алгоритму, агент установит в ней камень 1. После чего перейдет в вершину ca_i (ca_{i+1}) и пометит ее камнем 2. После чего перейдет в вершину ca_{i+1} (ca_i) и пометит ее камнем 3. В результате этого камень 1 станет свободным, а агент пометит вершину краской b. После чего агент переходит в вершину ca_i (ca_{i+1}). После этого агент устанавливает камни 1 и 4 в вершинах b_{i-1} и d_{i-1} (b_{i+1} и d_{i+1}). После чего устанавливает камень 5 в вершине d_i, который сразу же становится сводным, а агент метит вершину краской b. В результате этого камень 2 (3) станет свободным, а агент пометит вершину краской b. После чего агент перейдет в вершину b_{i-1} или d_{i-1} (b_{i+1} или d_{i+1}) из нее перейдет в вершину ca_{i-1} (ca_{i+2}), в которой он установит свободный камень 2 (3). В результате этого камни 1 и 4 станут свободными, а вершины агент пометит краской b. После чего агент перейдет в вершину ca_{i-1} (ca_{i+2}). При этом агент использовал 5 камней 3 из которых свободны. Дальнейшее выполнение агентом восстановление графа аналогично случаю помещения агента в вершину ca_i.

Модификация 3. В предложенном алгоритме восстановления графа и его модификациях агенту необходимо анализировать 1 – окрестность вершины. Поскольку при достаточно больших n, степень вершины может быть достаточно большой, и анализ ее окрестности может быть очень трудоемким процессом. Поэтому имеет смысл рассмотреть графы из более узкого класса K_D^p. Класс K_D^p содержит графы, степень вершин которых ограниченна константой D (ординарные вершины) [5], а так же p вершин, степень которых больше D (неординарные вершины) [5]. При этом предполагается, что каждое ребро графа инцидентно хотя бы одной ординарной вершине и удаление двух неординарных вершин не нарушает связности графа. В этом случае для анализа 1 – окрестности ординарной вершины агенту потребуется анализировать не более D вершин. 1 – окрестность неординарных вершин агент анализировать не будет. Кроме того, элементы списка $l(1)..l(l)$ будут содержать число не большее D. Неординарная вершина метится камнем i, а $li() := \infty$, что будет означать, камень забираться не будет, поскольку агент не анализировал 1 – окрест-

ность вершины и не может однозначно определить были ли посещены все не помеченные смежные с неординарной вершиной вершины.

В алгоритме восстановления графа модификации 1, перед выполнением основного алгоритма, необходимо выполнить обход в глубину с использованием красок b и r, для того чтобы расставить камни на неординарных вершинах, при этом счетчик вершин увеличивается только при посещении неординарной вершины. После чего агент проходит по вершинам графа и снимает все свои метки и выполняет основную часть алгоритма, при этом счетчик *Сч* не инициализируется. Такая модификация алгоритма не ухудшит временную сложность, поскольку будет выполнено два дополнительных обхода в глубину неизвестного графа.

В модификации 2 агенту в процедуре *ИССЛЕДОВАТЬ* (v) после поиска свободного камня необходимо проанализировать степень вершины, и если она больше чем D, пометить вершину свободным камнем i, а $li() := \infty$.

Лемма 13. Для восстановления графа G агенту потребуется на p камней больше, чем для восстановления графа полученного из G удалением из него неординарных вершин.

Доказательство. Поскольку неординарные вершины метятся камнями, которые агент потом не снимает для повторного использования, то потребуется p камней помимо тех, которые будут использоваться при восстановлении ординарных вершин.

Выводы. Предложен новый полиномиальный алгоритм восстановления графа, который использует $O(n)$ шагов, $O(n)$ ячеек памяти, две краски b и r, а так же h_t камней, $h_t \leq$. Предложенный алгоритм, так же как и алгоритм [2], имеет линейную сложность, но использует не h_t красок, а h_t камней. Кроме того, предложенный алгоритм использует 2 различные краски как и алгоритмы [3–5], но в отличие от последних алгоритмов, имеющих нелинейную сложность, за счет использования дополнительных камней, имеет линейную сложность. Модификации предложенного алгоритма позволяют уменьшить количество используемых камней для некоторых видов графов. Найдены виды графов, для восстановления которых агент будет использовать константное число красок.

Библиографические ссылки

1. **Dudek G., Jenkin M.** Computational principles of mobile robotic – Cambridge Univ. Press. 2000 – 280 p.
2. **Грунский И.С.** О распознавании графов конечным автоматом. / И.С. Грунский, М.Ю. Тихончев //Вестник Донецкого университета. Серия А. Естественные науки – 2001, – вып.2. – С. 351–356.
3. **Грунский И. С.** Распознавание конечного графа блуждающим по нему агентом. / И. С. Грунский, Е. А. Татаринов // Вестник Донецкого университета. Серия А. Естественные науки –, 2009, вып. 1. – с. 492 – 497.
4. **Татаринов Е. А.** M – нумерация, как метод распознавания графов. / Е. А. Татаринов //Збірник наукових праць Питання прикладної математики математичного моделювання – 2010, С. 260 – 272.
5. Труды ИПММ моя статья этого года
6. **Касьянов В. Н.** Графы в программировании: обработка, визуализация и применение. / В. Н. Касьянов, В. А. Евстигнеев – СПБ., 2003. – 1104 с.
7. **Ахо А.** Построение и анализ вычислительных алгоритмов. / Ахо А., Дж. Хопкрофт, Дж. Ульман – М., 1979. – 536 с.
8. **Кормен Т.** Алгоритмы: построение и анализ./ Т. Кормен, Ч. Лейзерсон, Р. Ривест – М., 2001. – 960 с.

Надійшла до редколегії 06.06.11

УДК 519.6

[7]**В. А. Турчина, Є. В. Козаченко**
Дніпропетровський національний університет імені Олеся Гончара

ВИКОРИСТАННЯ НЕЧІТКОГО ЛОГІЧНОГО ВИСНОВКУ ПРИ ОЦІНЮВАННІ ЯКОСТІ ІНТЕРФЕЙСУ КОРИСТУВАЧА

Створено метод для оцінювання якості інерфейсу користувача, оснований на психологічних та фізіологічних факторах, що впливають на сприйняття візуальної інформації людиною, який видає висновок за допомогою нечіткої моделі логічного висновку.

Создан метод для оценки качества пользовательского интерфейса, основаный на психологических и физиологических факторах, влияющих на восприятие визуальной информации человеком, и выдающий выводы при помощи нечёткой модели логического вывода.

There was developed a method for assessment of quality of user interface, which is based on psyhological and phisiological factors that affect on humans' visual perception. The method gives output produced by the fuzzy-logic productional model.

Ключові слова: оцінка якості інтерфейсу, нечіткий логічний висновок.

Постановка проблеми. Сьогодні комп'ютерна техніка все більше занурюється у життя не тільки науковців та бізнесменів, а і звичайних людей. Тому якість комп'ютерних інтерфейсів є дуже важливим фактором, який впливає на успішність використання комп'ютерної техніки та якість отриманого результату.

Починаючи з 80-х років минулого сторіччя науковці намагаються деяким чином стандартизувати інтерфейси, ввести деякі показники їхньої якості та за допомогою цих показників мати змогу оцінювати інтерфейси, і розробляти їх зручними та корисними людині [1].

Саме з 80-х років і до сьогодні не вдалося сформулювати формальної математичної моделі якості інтерфейсу. Більшість методів оцінки якості інтерфейсів нині базуються на ГОСТ-ах 1980-х рр. або на суб'єктивних відчуттях оцінювачів, або на знаннях експертів, що входять до складу експертних груп з оцінки якості інтерфейсів користувачів. Крім того,

сьогодні процедура оцінки якості інтерфейсу коштує досить дорого (від $1000 за оцінку одного інтерфейсу), при тому вартість підвищується, якщо при оцінці було виявлено помилку, і інтерфейс потребує повторної оцінки, після її виправлення.

Отже, проблема розробки автоматичного методу оцінки якості інтерфейсу користувача є дуже актуальною сьогодні.

Відразу зауважимо, що метод може оцінювати виключно візуальні інтерфейси, тобто ті, що взаємодіють із користувачем шляхом візуальних образів, які користувач сприймає за допомогою очей. Оцінка аудіальних та тактильних інтерфейсів потребує окремого дослідження та розробки окремих блоків аналізу для методу.

У ході дослідження виявлені декілька факторів, що у першу чергу впливають на сприйняття інтерфейсу користувачем. Вони поділяються на психологічні та фізіологічні. До них належать такі, що впливають на психо_емоціональний стан людини, під час споглядання зображення інтерфейсу. До таких факторів можна віднести кольори зображення (вони впливають на емоціональне ставлення споглядача на зображення у перші секунди взаємодії), загальну геометрію та пропорційність зображення інтерфейсу (наявність на зображенні об'єктів із пропорціями золотого перетину тощо). До фізіологічних факторів, що вливають на сприйняття зображення людиною відносять наявність людських облич та образів на зображенні, кегль шрифту, кількість складно _ламаних ліній та об'єктів на зображенні. Оскільки основним носієм інформації у комп'ютерному інтерфейсі є текст, то окремим фактором слід виділити його читабельність, особливо тих блоків, що розташовані безпосередньо поряд з обличчями людей, оскільки на ці блоки тексту користувач зверне увагу у першу чергу.

Метод розв'язання. Зважаючи на наведені вище фактори, які впливають на сприйняття візуальної інформації людиною, а значить і є показниками якості інтерфейсу користувача, розроблений метод проводить аналіз зображення інтерфейсу з метою виявлення областей зображення із наявними психологічними та фізіологічними факторами.

Отже, до областей, які підлягають локалізації та аналізу належать:

- людські обличчя та образи;
- яскраві блоки;
- блоки із комплементарними кольорами (коли один із кольорів блоку знаходиться з іншого боку колірного кола);
- блоки із текстом (для подальшого аналізу читабельності).

Крім того, метод збирає інформацію про загальний колір зображення, для аналізу психологічного ставлення спостерігача до інтерфейсу.

Після отримання всіх необхідних даних стосовно зображення, їх данні необхідно проаналізувати та консолідувати для того, щоб отримати стислий висновок, щодо якості інтерфейсу. Очевидно, що дані, які метод отримує під час аналізу зображення, мають різну природу, і , взагалі кажучи, погано формалізовані, тому логічним є використання апарату нечіткого логічного висновку для обробки результатів аналізу зображення та видачі обґрунтованого висновку.

Для того, щоб обробити результати аналізу, за допомогою апарату нечіткої логіки необхідно виділити лінгвістичні змінні, побудувати для них характеристичні функції, скласти базу правил та сформувати функцію дефазифікації.

Складність полягає в тому, що кількість лінгвістичних змінних може змінюватись залежно від зображення інтерфейсу, так, наприклад на зображенні може бути декілька окремих блоків з текстом, 3 обличчя і жодного об'єкта із складною геометрією (об'єкти, які мають складно - ламані контури).

Тому, для простоти розділимо гіпотетичні лінгвістичні змінні на класи:

- Обличчя;
- Блок тексту;
- Кольори;
- Інше.

До класу **Обличчя** будуть належати такі змінні:

- РОЗМІР_ОБЛИЧЧЯ (значення, які змінна може набувати: велике, середнє і маленьке);
- ПОЗИЦІЯ_ОБЛИЧЧЯ (згори, всередині, знизу).

- До класу **Блок тексту** будуть належати змінні:
- ЧИТАБЕЛЬНІСТЬ_ТЕСТУ_У_БЛОЦІ (добра, середня, погана);
- НАЯВНІСТЬ_ОБЛИЧЧЯ_ПОБЛИЗУ_БЛОКУ_ТЕКСТУ (поблизу є обличчя, поблизу немає обличчя);
- ЯСКРАВІСТЬ_КОЛЬОРУ_ФОНУ_ЗОБРАЖЕННЯ (яскравий, тьмяний).

До класу **Кольори** мають належати 2 змінні:

ЗАГАЛЬНИЙ_КОЛІР_ЗОРАЖЕННЯ (активний, пасивний та нейтральний);

НАЯВНІСТЬ_БЛОКІВ_ІЗ_КОМПЛЕМЕНТАРНИМИ_КОЛЬОРАМИ (такі блоки є, таких блоків немає).

До класу **Інше** поки що відносяться змінні:

- КОЛІР_У_ЗОЛОТОМУ_ПЕРЕТИНІ (активний, пасивний та нейтральний);
- НАЯВНІСТЬ_СКЛАДНО-ЛАМАНИХ_ОБ'ЄКТІВ (такі об'єкти є, таких об'єктів немає).

Але до цього класу можна додати необхідні змінні, зважаючи на спеціальну спрямованість інтерфейсу, та вимоги до нього, що випливають з його спеціалізації.

Зважаючи на те, що кількість змінних -результатів аналізу зображення заздалегідь невідома, то раціональним буде створювати списки змінних, що належать до однакових класів. Тобто в результаті первинної обробки зображення ми маємо отримати низку змінних, таких як:

tbw1, tbh1, tbr1, tbw2, tbh2, tbr2 і так далі, де змінні розшифровуються так: tb – блок тексту (text block), w – ширина (width), h – висота (height), r – читабельність (readability).

Таким чином можуть бути створені списки змінних за кожним із локалізованих і проаналізованих блоків тексту, облич тощо. Така ускладненість із змінними необхідна, оскільки на зображенні може бути локалізована необмежена кількість блоків тексту, та інших шуканих об'єктів.

Визначивши класи лінгвістичних змінних можемо створити для них характеристичні функції. Не будемо розглядати характеристичні функції для всіх змінних, оберемо наразі найбільш впливові на якість інтерфейсу. Вважається, що більшість корисної інформації в інтерфейсі користувача несе текст, тому розглянемо характеристичні функції для таких змінних як:

ЧИТАБЕЛЬНІСТЬ_ТЕСТУ_У_БЛОЦІ та
НАЯВНІСТЬ_ОБЛИЧЧЯ_ПОБЛИЗУ_БЛОКУ_ТЕКСТУ.

Читабельність тексту залежить від дисперсії кольорових показників фону, на якому зображено текст, та кольору самого фону. Логічно, що, чим більша різниця між кольорами, тим краща читабельність. Тому зручно в якості характеристичної функції для змінної ЧИТАБЕЛЬНІСТЬ_ТЕСТУ_ У_БЛОЦІ обрати гаусову характеристично функцію, вигляду:

$$\begin{cases} \mu G(x) = 1, (x \le 0); \\ \mu G(x) = \exp\left(-\dfrac{x^2}{\sigma}\right), (0 < x \le 0.25); \\ \mu G(x) = \exp\left(-\dfrac{(x-0.5)^2}{\sigma}\right), (0.25 < x \le 0.75); \\ \mu G(x) = \exp\left(-\dfrac{(x-1)^2}{\sigma}\right), (0.75 < x \le 1). \end{cases}$$

де σ – коефіцієнт «крутості» зростання функції.

Для даної змінної (ЧИТАБЕЛЬНІСТЬ_ТЕСТУ_У_БЛОЦІ), враховуючи можливі значення цієї змінної, графік функції приналежності матиме вигляд, зображений на рис. 1.

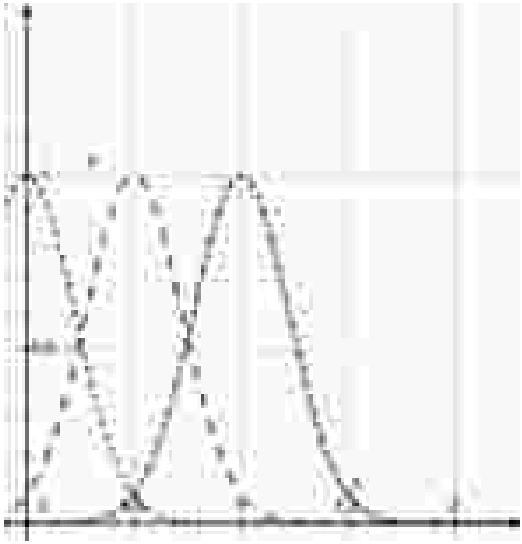


Рис. 1. Графік функції приналежності для змінної ЧИТАБЕЛЬНІСТЬ_ТЕСТУ_ У_БЛОЦІ (σ=0.1)

Параметр σ можна «підлаштовувати» залежно від вимог до читабельності тексту у інтерфейсі. Чим менша потреба у читабельності тексту, тим більший параметр σ можна брати (при збільшенні σ графік кожного із значень функції приналежності буде «розтягуватись» горизон-

тально, тим самим зменшуючи природні границі між блоками і зменшуючи впливовість читабельності тексту на якість інтерфейсу).

Для змінної НАЯВНІСТЬ_ОБЛИЧЧЯ_ПОБЛИЗУ_БЛОКУ_ТЕКСТУ, яка приймає лише 2 нечітких терми (є обличчя поблизу з текстом і немає) гаусова функція приналежності не підходить. Цей тип змінних набуватиме своїх «порогових» термів тим швидше, чим менша Евклідова відстань від обличчя до блоку з текстом. Тому найкраще цю лінгвістичну змінну охарактеризує звичайна трикутна функція:

$$\begin{cases} \mu T(x) = 1 - \left(\dfrac{a-x}{a}\right), (0 < x < a); \\ \mu T(x) = 1 - \left(\dfrac{x-a}{a}\right), (a < x < 1); \\ \mu T(x) = 0, (x \le 0, x \ge 1). \end{cases}$$

де a – параметр, який вказує на умовну середину між першим і другим нечітким термом. У даному випадку a=0.5. [2]

Графік характеристичної функції для змінної НАЯВНІСТЬ_ ОБЛИЧЧЯ_ПОБЛИЗУ_БЛОКУ_ТЕКСТУ матиме вигляд як на рис. 2. [3]

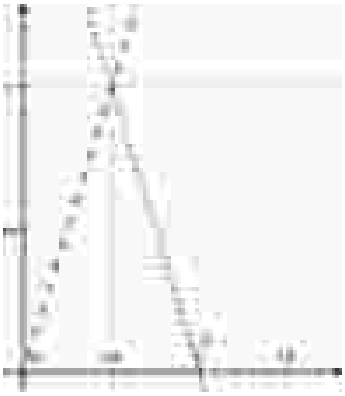


Рис. 2. Графік трикутної функції приналежності для змінно
НАЯВНІСТЬ_ОБЛИЧЧЯ_ПОБЛИЗУ_БЛОКУ_ТЕКСТУ
із параметром a=0.5

Із аналогічних міркувань будуються характеристичні функції і для інших змінних.

При побудові бази правил також необхідно враховувати те, що змінних може бути необмежена кількість. Тому при розробці програмної реалізації нечіткого логічного висновку мною запропоновано враховувати шаблони правил. Для кожного із класів змінних, таких як **Блок тексту**, наприклад, складається шаблонна база правил. У кожному із правил замість змінних ставляться їх загальні назви, які на початку роботи алгоритму нечіткого логічного висновку замінюються конкретними змінними, отриманими у ході аналізу зображення. Після обробки всіх шаблонів програма створює динамічну базу правил, в яких присутні і враховані всі змінні, і проставлені всі необхідні зв'язки. Таким чином база правил, яку створено в ході розробки, жорстко не прив'язується до змінних, і автоматично масштабується під їх наявність (деяких змінних може і не бути, наприклад, якщо на зображенні інтерфейсу не має облич) і кількість.

Кожне правило у своїй частині «ТО» встановлює конкретне нечітке значення у одне із 2_х можливих змінних: «добре» чи «погано».

Після обробки динамічно згенерованої бази правил ми маємо набір значень у змінних «добре» та «погано», серед яких, за алгоритмом нечіткого логічного висновку, обираємо максимальні та дефазифікуємо.

Висновки. У результаті проведеної роботи розроблено метод оцінки якості інтерфейсу, який базується на основних психологічних та фізіологічних факторах, що впливають на сприйняття візуальної інформації людиною. За алгоритмом методу розроблено програмний продукт, який проводить необхідну графічну та аналітичну обробку зображення, та за описаною вище схемою проводить нечіткий логічний висновок спираючись на результати, отримані під час обробки зображення інтерфейсу.

У подальшій роботі планується збільшити кількість факторів, які враховуються під час роботи методу, та покращити базу правил для нечіткого логічного висновку.

Бібліографічні посилання

1. **Пономарев И.А.** Методы оценки качества пользовательского интерфейса / И.А. Пономарев.

2. **Ободан Н. І.** Моделі та методи систем штучного інтелекту. / **Н. І. Ободан**, В. А. Громов – Д., 2008. – 119 с.

3. **Рыжов А. П.** Элементы теории нечетких множеств и ее приложений. / А.П. Рыжков – М., 2003. — 81с.

Надійшла до редколегії 06.07.11

УДК 519.8

[8]В.А. Турчина, Н.К. Федоренко
Дніпропетровський національний університет імені Олеся Гончара

АЛГОРИТМИ ПОБУДОВ УСІХ ПАРАЛЕЛЬНИХ УПОРЯДКУВАНЬ ЗАДАНОЇ ДОВЖИНИ

Розглядається задача побудови всіх паралельних упорядкувань заданої довжини для заданого орграфу. Пропонується алгоритм її розв'язання, що базується на аналізі зв'язків в графі.

Рассматривается задача построения для данного орграфа всех параллельных упорядочений заданной длины. Предлагается алгоритм ее решения, базирующийся на анализе связей в графе.

The article includes the description of algorithms of building all schedules of given length for given directed acyclic graph. The algorithm of its solving is based on the analyses of graph-structure is provided.

Ключові слова: паралельне упорядкування, дискретна оптимізація, графи.

Вступ. Задачі побудови оптимальних розкладів виконання завдань, для яких задані часткові обмеження на їх порядок, досліджуються різними науковими школами [1; 2]. Оскільки моделлю непротирічних відношень порядку виступають орграфи, то ці задачі можна формулювати як оптимізаційні задачі на графах [3].

З теоретичної точки зору представляє інтерес задача побудови всіх паралельних упорядкувань заданої довжини, розв'язком якої є множина всіх допустимих розв'язків задачі побудови упорядкування даної довжини.

На базі цієї множини можуть формуватися різні підмножини, як тестові бази для перевірки точності алгоритмів та множини всіх оптимальних паралельних упорядкувань мінімальної ширини. Крім того, отримані упорядкування можна використовувати для уточнення оцінки ширини упорядкування заданої довжини.

Постановка задачі. Дано: ациклічний граф $G = \{V, U\} (|V| = n)$,

натуральне число $l (l \le l \le n)$, де l – число на одиницю більше довжини критичного шляху в орграфі.

Необхідно: побудувати множину Θ всіх допустимих паралельних упорядкувань S вершин орграфу , таких, що довжина цих паралельних упорядкувань $l(S) \le l$ та ширина упорядкувань $h(S)$ – довільна.

Методи розв'язання. При аналізі паралельних упорядкувань у задачах, що пов'язані з оцінками довжини та ширини, та при розробці деяких алгоритмів часто використовують, як базові, такі паралельні упорядкування довжини l: $\underline{S}(\overline{S})$ – паралельне упорядкування, в якому кожна вершина займає крайнє ліве (праве) допустиме місце. Позначимо через $\underline{\mu}_i(\overline{\mu}_i)$ – номер місця, яке вершина v_i займає в упорядкуванні $\underline{S}(\overline{S})$. Тоді $\mu_i = [\underline{\mu}_i, \mu_i + 1, \cdots, \overline{\mu}_i - 1, \overline{\mu}_i]$ – множина допустимих місць, які вершина v_i може займати в будь-якому упорядкуванні довжини l.

Паралельне упорядкування n вершин орграфу G можна інтерпретувати, як комбінацію n чисел, кожне з яких належить проміжку $[1, l]$ та відповідає номеру місця в упорядкуванні, яке займає вершина.

Оцінимо зверху загальну кількість упорядкувань довжини l для заданого орграфу G (зрозуміло, що їх буде не менше одного). Ця величина не перевищує $\prod_{i=1}^{n} |\mu_i|$. Але для графів, що задають реальні процеси, ця величина є меншою, адже якщо, наприклад, $\mu_i = [\underline{\mu}_i, \cdots, \overline{\mu}_i]$ та $\mu_j = [\underline{\mu}_i + 1, \cdots, \overline{\mu}_i + 1]$, з вершини v_i іде дуга в v_j, та вершина v_i в упорядкуванні S^* розташована на деякому місці $\mu_i^* > \underline{\mu}_i$, то множина місць, на яких може бути розташована вершина v_j, буде $\mu_j = [\mu_i^* + 1, \overline{\mu}_i + 1]$.

Ідея алгоритму побудови всіх паралельних упорядкувань така: будуємо $\prod_{i=1}^{n} |\mu_i|$ комбінацій з n чисел, кожне з яких відповідає номеру місця, яке може займати в упорядкуванні певна вершина орграфу. Після цього видаляємо такі комбінації, в яких номер місця вершини є не більшим за номер місця, яке займає її попередник. Усі інші комбінації інтерпретуємо як до-

пустимі паралельні упорядкування. Алгоритм реалізує схему перебору і має експоненційну складність.

Алгоритм побудови всіх упорядкувань довжини l

1. Для даного орграфа G будуємо упорядкування $\underline{S}$ та $\overline{S}$ [3].
2. Для кожної вершини v_i $(i=\overline{1,n})$ графа G визначаємо величини $\underline{\mu}_i$ та $\overline{\mu}_i$ і будуємо множину допустимих місць $\mu_i=[\underline{\mu}_i,\mu_i+1,\cdots,\overline{\mu}_i-1,\overline{\mu}_i]$.
3. Будуємо всі можливі комбінації $M_k=\{m_{k1},\cdots,m_{kn}\}$ елементів множин μ_i $(m_{ki}\in\mu_i)$. Усього таких комбінацій: $\prod_{i=1}^{n}|\mu_i|$.
4. Якщо в графі G існує ребро із вершини v_i до вершини v_j, то видаляємо із розгляду ті комбінації M_k, для яких $m_{ki}\geq m_{kj}$.
5. Кожній комбінації M_k, що залишилася, ставимо у відповідність упорядкування S_k, при чому вершина v_i займає в цьому упорядкуванні місце m_{ki}.

Розглянемо роботу алгоритму на прикладі.

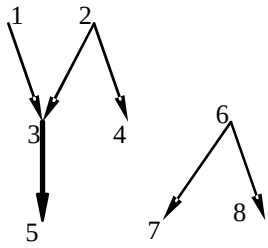


Рис.1 Орграф G

Приклад

Дано орграф G (рис.1). Спеціальні паралельні упорядкування

$$\underline{S}=\begin{Bmatrix}1&3&5\\2&4&\\6&7&\\&8&\end{Bmatrix}\quad \overline{S}=\begin{Bmatrix}1&3&5\\2&6&7\\&&8\\&&4\end{Bmatrix}.$$

Довжина упорядкувань $l=3$. Множини допустимих місць для кожної вершини:

$$\mu_1=\{1\},\mu_2=\{1\},\mu_3=\{2\},\mu_4=\{2,3\},\mu_5=\{3\},\mu_6=\{1,2\},\mu_7=\{2,3\},\mu_8=\{2,3\}.$$

Можливі комбінації наведені в табл. 1 (комбінації, в яких порушується порядок слідування вершин позначені сірим).

Можливі упорядкування довжини 3 (множина Θ):

$$S_1=\begin{Bmatrix}1&3&5\\2&4&\\6&7&\\&8&\end{Bmatrix},\ S_2=\begin{Bmatrix}1&3&5\\2&4&8\\6&7&\\&&\end{Bmatrix},\ S_3=\begin{Bmatrix}1&3&5\\2&4&7\\6&8&\\&&\end{Bmatrix},\ S_4=\begin{Bmatrix}1&3&5\\2&4&8\\6&&7\\&&\end{Bmatrix},$$

$$S_8=\begin{Bmatrix}1&3&5\\2&4&7\\&6&8\\&&\end{Bmatrix},\ S_9=\begin{Bmatrix}1&3&5\\2&7&4\\6&8&\\&&\end{Bmatrix},\ S_{10}=\begin{Bmatrix}1&3&5\\2&7&4\\6&&8\\&&\end{Bmatrix},\ S_{11}=\begin{Bmatrix}1&3&5\\2&8&4\\6&&7\\&&\end{Bmatrix},$$

$$S_{12}=\begin{Bmatrix}1&3&5\\2&&4\\6&&8\\&&7\end{Bmatrix},\ S_{16}=\begin{Bmatrix}1&3&5\\2&6&7\\&&8\\&&4\end{Bmatrix}.$$

Таблиця 1

M_k	m_{k1}	m_{k2}	m_{k3}	m_{k4}	m_{k5}	m_{k6}	m_{k7}	m_{k8}	S_i
1	1	1	2	2	3	1	2	2	S_1
2	1	1	2	2	3	1	2	3	S_2
3	1	1	2	2	3	1	3	2	S_3
4	1	1	2	2	3	1	3	3	S_4
5	1	1	2	2	3	2	2	2	
6	1	1	2	2	3	2	2	3	
7	1	1	2	2	3	2	3	2	
8	1	1	2	2	3	2	3	3	S_8
9	1	1	2	3	3	1	2	2	S_9
10	1	1	2	3	3	1	2	3	S_{10}
11	1	1	2	3	3	1	3	2	S_{11}
12	1	1	2	3	3	1	3	3	S_{12}
13	1	1	2	3	3	2	2	2	
14	1	1	2	3	3	2	2	3	
15	1	1	2	3	3	2	3	2	
16	1	1	2	3	3	2	3	3	S_{16}

Як бачимо, $|\Theta| = 10$. Аналізуючи упорядкування, що входять до множини Θ, можна сказати, що для даного графа і $l = 3$ ширина упорядкування $h(S) \geq 3$ і множина Θ містить усі сім розв'язків задачі $S(G,3,h)$ для даного графа.

Проаналізуємо як між собою пов'язані кількість дуг у графі та кількість можливих паралельних упорядкувань заданої дожини для цього графа.

Зрозуміло, що чим більш зв'язним є граф, тим менше для нього $|\Theta|$ при фіксованому l. Для графа, в якому $|U| = \varnothing$, $|\Theta| = 1$ при $l = l$, а для $l = n$ - $|\Theta| = n!$.

Позначимо через $\tilde{S}$ таке паралельне упорядкування, що $\tilde{S}[i] = \underline{S}[i] \cap \overline{S}[i]$. Це паралельне упорядкування включає до себе вершини, що належать критичному шляху і можуть займати лише одне місце в упорядкуванні довжини l. Зрозуміло, що ці вершини ніяк не впливатимуть на кількість можливих упорядкувань довжини l для заданого орграфа. На цю величину впливатимуть лише вершини з упорядкування $\hat{S} : \hat{S}[i] = \overline{S}[i] \setminus \tilde{S}[i]$.

А отже загальна кількість можливих паралельних упорядкувань для орграфа складатиме не $\prod_{i=1}^{n} |\mu_i|$, а $\prod_{\substack{j \\ v_j \in \hat{S}}} |\mu_j|$. Зауважимо, що ці числа, взагалі кажучи, співпадають, адже для вершин із упорядкування $\tilde{S}$ (вершин критичного шляху) $\underline{\mu}_i = \overline{\mu}_i$ і $|\mu_i| = 1$. Кількість комбінацій, що будується на третьому кроці алгоритму не скоротиться, але скоротиться кількість перевірок, що має бути зроблена на четвертому кроці. Адже, вершини, що належать критичному шляху, завжди матимуть допустимі номери місць, а тому перевірку для них робити не треба. Крім того, якщо в графі існують вершини, що не належать критичному шляху і при цьому зв'язані лише з вершинами критичного шляху (це можуть бути лише вершини, в які ідуть дуги з вершин критичного шляху), то такі вершини також завжди матимуть лише допустимі номера місць. Дійсно, якщо в графі G існує ребро із вершини v_i (вершина критичного шляху) до вершини v_j (вершина, що не належить критичному шляху) і вершина v_i займає в упорядкуванні S

місце $\underline{\mu}_i$, то вершина v_j займатиме в цьому упорядкуванні місце $\underline{\mu}_i + 1$, а в упорядкуванні $\overline{S}$ місце $\underline{\mu}_i < \overline{\mu}_j$. Тобто не буде існувати комбінацій M_k, для яких $m_{ki} \geq m_{kj}$.

Таким чином, крок 4 алгоритму можна переформулювати так: якщо в графі G існує ребро із вершини v_i, що не належить критичному шляху, в вершину v_j, то видаляємо із розгляду ті комбінації M_k, для яких $m_{ki} \geq m_{kj}$.

Розглянемо, як зміниться алгоритм побудови всіх упорядкувань заданої довжини у випадку, коли $l > l$. Від задачі $S(G, l, h)$ можна перейти до задачі $S(G, l, h)$, додаючи до орграфа, що розглядається, ланцюг із l вершин.

У такому випадку всі вершини вихідного орграфа потрапляють до множини $\hat{S}$, а отже впливатимуть на кількість можливих паралельних упорядкувань заданої довжини. Велична $\underline{\mu}_i$ для всіх вершин не зміниться, а от величина $\overline{\mu}_i$ збільшиться на $l - l$, і множина допустимих місць для кожної вершини буде $\mu_i = \left[\underline{\mu}_i, \overline{\mu}_i + l - l\right]$ отже загальна кількість можливих паралельних упорядкувань стане не більше за $\prod_{i=1}^{n} (|\mu_i| + (l - l))$. Сам алгоритм побудови всіх паралельних упорядкувань при цьому не зміниться.

Наприклад, для орграфа, розглянутого вище, при збільшенні довжини на одиницю ($l = 4$) кількість комбінацій M, які мають бути побудовані і перевірені на допустимість номерів місць вершин, збільшиться із 16 до 1296. І навіть після видалення серед них тих, в яких вершини наступники розташовані не пізніше вершин попередників, їх загальна кількість складатиме понад 800.

Висновки. Запропонований у статті алгоритм дозволяє побудувати всі паралельні упорядкування заданої довжини. Основною перевагою цього алгоритму є загальна простота реалізації та прозорість. Він дозволяє побудувати всі паралельні упорядкування із мінімальною кількістю додаткових дій та обчислень.

За допомогою запропонованого алгоритму можна на ЕОМ будувати всі паралельні упорядкування для заданих графів для подальшого аналізу роботи алгоритмів або оцінки параметрів паралельних упорядкувань.

Бібліографічні посилання

1. Танаев В. С. Теория расписаний. Одностадийные системы. / В. С. Танаев, В.С. Гордон, Я. М. Шафранский — М., 1984. — 384 с.

2. Коффман Э.Г. Теория расписаний и вычислительные машины; пер. с англ . / Э.Г. Кофман – М., 1984. — 336 с.

3. Бурдюк В.Я. Алгоритмы параллельного упорядочения: учебное пособие. / В.Я. Бурдюк , В.А. Турчина – Д., 1985. – 84 с.

Надійшла до редколегії 15.06.11

УДК 519.85: 519.816

[9]С.А. Ус
Национальный горный университет

ЗАДАЧА ОПТИМАЛЬНОГО РАЗБИЕНИЯ МНОЖЕСТВ С МНОЖЕСТВЕННОЗНАЧНЫМ ЦЕЛЕВЫМ ФУНКЦИОНАЛОМ

Сформульвано нову постановку задачі оптимального розбиття множин (ОРМ) із множиннозначним цільовим функціоналом та запропонований підхід до її розв'язання, на основі зведення до звичайної задачі ОРМ із застосуванням критеріїв прийняття рішень в умовах невизначеності.

Сформулирована новая постановка задачи оптимального разбиения множеств (ОРМ) с множественнозначным целевым функционалом и предложен подход к ее решению, основаный на сведении к обычной задаче ОРМ, используя критерии принятия решений в условиях неопределенности.

The new statement of set patitioning problem with multivalue objective functionalis formulated, and the approach solving that is proposed. It is based on a reduction mentioned problem to the usual set patitioning problem using the criteria of decision making under uncertainty.

Ключевые слова: оптимальное разбиение множеств, неполная информация, критерий принятия решений, неопределенность.

Введение. Многие известные и практически важные задачи, возникающие в производственной и научной деятельности человека, могут быть представлены как задача разбиения некоторого множества на его непересекающиеся подмножества. К числу таких задач относятся, например, задачи классификации и идентификации объектов, задачи размещения производства и определения зон обслуживания каждым предприятием, задачи орошения и др. Методы решения задач разбиения множеств в дискретной постановке широко исследуются уже на протяжении столетия. Задачи оптимального разбиения континуального множества (ОРМ) гораздо менее исследованы ввиду их сложности. В [1] представлены результаты исследования этой проблемы в детерминированной постановке, а также в условиях неопределенности. В частности, исследованы стохастические задачи ОРМ и некоторые задачи оптимального нечеткого разбиения. Зада-

чи оптимального разбиения в условиях неопределенности исследовались также в [2;3;5].

Постановка задачи. Рассмотрим линейную задачу ОРМ с размещением центров подмножестви ограничениями в следующей постановке [1].

Пусть потребитель некоторой однородной продукции распределен в области Ω. Конечное число N производителей, расположенных в изолированных точках τ_i, $i=\overline{1,N}$ области Ω, образуют систему точек $\tau_1,\tau_2,\cdots,\tau_N$, причем координаты некоторых из них, или даже всех могут быть заранее неизвестны. Известен спрос $\rho(x)$ на продукцию в каждой точке области Ω, а также стоимость доставки продукции $c_i(x,\tau_i)$, $i=\overline{1,N}$ - из пункта производства τ_i в пункт потребления x. Предполагается, что прибыль производителя зависит только от транспортных расходов. Мощность i-го производителя определяется суммарным спросом обслуживаемых потребителей и не должна превышать заданных объемов b_i, $i=\overline{1,N}$. Необходимо разбить область Ω на зоны обслуживания каждым из производителей, т. е. на подмножества Ω_1, Ω_2, …, Ω_N, так, чтобы суммарные затраты на доставку продукции были минимальны.

Однако, при решении реальных задач, точно определить стоимость доставки невозможно, поскольку она зависит от множества факторов, многие из которых имеют вероятностный или нечеткий характер и не могут быть точно определены. Примерами таких факторов являются: погодные условия, качество дорог, техническое состояние транспорта, квалификация персонала, экономические условия и др. Кроме того, сложным представляется описание функциональной зависимости стоимости доставки от этих факторов, а выбор одного фиксированного значения огрубляет модель. Но, в большинстве случаев, можно указать некоторое множество возможных функций стоимости, соответствующие определенным состояниям внешней среды. В результате будет получена задача такого вида:

Пусть потребитель некоторой однородной продукции распределен в области Ω. Конечное число N производителей, расположенных в изолированных точках τ_i, $i=\overline{1,N}$ области Ω, образуют систему точек $\tau_1,\tau_2,\cdots,\tau_N$, причем координаты некоторых из них, или даже всех могут быть заранее неизвестны. Известен спрос $\rho(x)$ на продукцию в каждой точке области Ω, а также множество возможных значений стоимости

доставки продукции $c_i(x,\tau_i)=\left\{c_i^1(x,\tau_i), c_i^2(x,\tau_i),\cdots c_i^{M_i}(x,\tau_i)\right\}$ $i=\overline{1,N}$ – из пункта производства τ_i в пункт потребления x. Предполагается, что прибыль производителя зависит только от транспортных расходов. Мощность i-го производителя определяется суммарным спросом обслуживаемых потребителей и не должна превышать заданных объемов b_i, $i=\overline{1,N}$. Необходимо разбить область Ω на зоны обслуживания каждым из производителей, т. е. на подмножества Ω_1, Ω_2, …, Ω_N, так, чтобы суммарные затраты на доставку продукции были минимальны.

Поставленной задаче соответствует следующая математическая модель.

Пусть Ω – замкнутое ограниченное измеримое по Лебегу множество евклидова пространства E^n. Необходимо разбить его на N измеримых по Лебегу подмножеств $\Omega_1, \Omega_2, \cdots, \Omega_N$, и разместить центры подмножеств $\tau_1,\tau_2,...,\tau_N$ в области Ω так, чтобы функционал

$$F(\Omega_1,\Omega_2,...,\Omega_N,\tau_1,\tau_2,\cdots,\tau_N)=\sum_{i=1}^{N}\int_{\Omega_i} c_i(x,\tau_i)\rho(x)dx \qquad (1)$$

достигал минимального значения при ограничениях:

$$\int_{\Omega_i}\rho(x)dx \le b_i, \quad i=\overline{1,N} \qquad (2)$$

$$\text{mes}\,(\Omega_i \cap \Omega_j)=0,\ i\neq j,\ i,j=\overline{1,N}, \qquad (3)$$

$$\bigcup_{i=1}^{N}\Omega_i=\Omega. \qquad (4)$$

Здесь $c_i(x,\tau_i)=\left\{c_i^1(x,\tau_i), c_i^2(x,\tau_i),\cdots c_i^{M_i}(x,\tau_i)\right\}$ $i=\overline{1,N}$ – множество возможных функций стоимости; $\rho(x)$ действительная неотрицательная интегрируемая функция, заданная на Ω; b_i, $i=\overline{1,N}$ –

заданные действительные положительные числа, удовлетворяющие условию разрешимости задачи

$$\sum_{i=1}^{N} b_i \geq S \ , S = \int_{\Omega} \rho(x) dx \ .$$

Метод решения. Полученная задача является задачей бесконечномерного программирования с множественнозначной функцией стоимости и к ее решению возможно несколько подходов.

Очевидно, что говорить о минимизации в обычном понимании, в этой задаче не имеет смысла, поскольку значением функционала также будет не число, а некоторое множество чисел, поэтому нужно оговорить, что будем понимать под решением задачи.

Обозначим, согласно [1], $P(\Omega, N)$ – множество всех возможных разбиений задачи (1) – (4):

$$P(\Omega, N) = \{(\Omega_1, \Omega_2, ..., \Omega_N, \tau_1, \tau_2, \cdots, \tau_N): \ mes(\Omega_i \cap \Omega_j) = 0, \ i \neq j, \ i, j = \overline{1, N};$$

$$\bigcup_{i=1}^{N} \Omega_i = \Omega; \ \int_{\Omega_i} \rho(x) dx \leq b_i, \ i = \overline{1, N}\},$$

$R = (\Omega_1, \Omega_2, ..., \Omega_N, \tau_1, \tau_2, \cdots, \tau_N)$ – некоторое возможное разбиение множества Ω.

Учитывая ситуацию, решением задачи можно считать:

– совокупность $P^*(\Omega)$ возможных разбиений множества Ω, удовлетворяющих условиям:

$$P*(\Omega) = \{R(\Omega) \mid \exists R', \ F(R(\Omega)) \leq F(R'(\Omega)) \text{для всех } c_i^j(x, \tau_i) \in C$$

и $F(R(\Omega)) < F(R'(\Omega))$ для $c_i^l(x, \tau_i) \in C\}$.

Такое определение решения исходит из интерпретации задачи (1) – (4), как задачи многокритериальной оптимизации, и соответствует множеству Парето_оптимальных решений.

– разбиение множества Ω на подмножества, доставляющее оптимальное в некотором смысле значение функционалу (1).

Оптимальность будем понимать как достижение одного из эффективных решений многокритериальной задачи и (или) достижение функционалом некоторого значения, удовлетворяющего определенным требованиям и которое будет не хуже других решений с точки зрения предпочтений, индуцированных условиями задачи и требованиями ЛПР.

– некоторое нечеткое разбиение множества Ω на подмножества (определение нечеткого разбиения дано в [3]).

В данной работе будут рассмотрены подходы к решению задачи, обеспечивающие решение типа 2. А именно, под решением задачи оптимального разбиения с множественнозначной функцией стоимости будем понимать разбиение, которое доставляет оптимальное (в указанном выше смысле) значение целевому функционалу (1).

Учитывая, что функция стоимости $c_i(x,\tau_i),\ i=\overline{1,N}$, для каждого фиксированного x и τ_i представляет собой некоторый набор значений, полученная задача может быть интерпретирована как задача многокритериальной оптимизации, множество критериев которой формируется как совокупность функционалов, полученных при подстановке в исходный функционал одного из возможных наборов функций стоимости либо как задача выбора (если предполагать, что среди всего множества функций стоимости актуальным будет лишь один представитель этого множества).

Один из подходов, применяемых к решению таких задач, состоит в сведении исходной задачи с множественнозначной целевой функцией к обычной задаче математического программирования, путем преобразования исходного множества целевых функций в единственную компромиссную целевую функцию. Наиболее простым способом такого преобразования является выбор одного представителя из всего множества целевых функций. Очевидно, что методы выбора этого представителя различны, в зависимости от имеющейся исходной информации.

Прежде всего, заметим, что в условиях полной неопределенности, т. е. когда ЛПР ничего не известно о возможных предпочтениях на множестве стоимостей, то можно выбирать любое из этих значений, например:

– минимальное значение

$$c_i(x,\tau_i)=\min_j c_i^j(x,\tau_i),\qquad i=\overline{1,N},\ x\in\Omega. \tag{5}$$

Такой выбор соответствует «оптимистической позиции», т.к. для расчета принимается минимальная цена;

– максимальное значение

$$c_i(x,\tau_i)=\max_{\theta_j} c_i^j(x,\tau_i), \qquad i=\overline{1,N},\ x\in\Omega. \tag{6}$$

Такой выбор соответствует «пессимистической позиции», поскольку ожидаемой считается максимальная цена для всех пунктов производства;

– любое (случайное) из возможных значений стоимости.

Однако такой подход имеет существенный недостаток. Выбирая одно из возможных значений функции стоимости, мы , тем самым отказываемся от всех других значений, т. е. не учитывается множественность исходных данных.

Предположим, что ЛПР обладает некоторой информацией о возможных состояниях среды и соответствующих им функциях стоимости:

состояние среды	функция стоимости			
θ_1	$c_1^1(x,\tau_i)$	$c_2^1(x,\tau_i)$	…	$c_N^1(x,\tau_i)$
θ_2	$c_1^2(x,\tau_i)$	$c_2^2(x,\tau_i)$	…	$c_N^2(x,\tau_i)$
…	…	…	…	…
θ_M	$c_1^M(x,\tau_i)$	$c_2^M(x,\tau_i)$	…	$c_N^M(x,\tau_i)$

Тогда, в какой -то мере учесть все множество функций стоимости позволяет выбор компромиссной функции на основе принципа недостаточного основания, который состоит в том, что если нет основания считать какое-либо состояние среды более вероятным, чем любое другое состояние среды, то их априорные вероятности нужно считать равными. Т. е. в качестве компромиссной функции выбирается среднее арифметическое всех возможных функций

$$c_i(x,\tau_i)=\frac{1}{M}\sum_{j=1}^{M} c_i^j(x,\tau_i), \qquad i=\overline{1,N}, x\in\Omega\ . \tag{7}$$

Другой вариант выбора в этой ситуации – выбор компромиссной функции в соответствии с критерием Гурвица:

$$c_i(x,\tau_i)=\alpha_i \min_{\theta_j} c_i^j(x,\tau_i)+(1-\alpha_i)\max_{\theta_j} c_i^j(x,\tau_i), \quad 0\le\alpha_i\le 1, \quad i=\overline{1,N},\ x\in\Omega. \tag{8}$$

Коэффициенты $\alpha_i,\ i=\overline{1,N}$ позволяют учитывать уровень оптимизма - пессимизма, в соответствии с убежденностью ЛПР.

Если ЛПР обладает информацией относительно вероятностей ситуаций принятия решений, то возможно использовать подход, основанный на критериях первой информационной ситуации. А именно:

– выбор в качестве возможной функции стоимости ее математического ожидания

$$c_i(x,\tau_i)=\sum_{j=1}^{M} p_j c_i^j(x,\tau_i), \quad i=\overline{1,N},\ x\in\Omega, \tag{9}$$

– выбор значения, соответствующего наиболее вероятному состоянию среды

$$c_i(x,\tau_i)=c_i^{j^*}(x,\tau_i), \text{ где } p_{j^*}=\max_{\theta_j} p_j \quad i=\overline{1,N},\ x\in\Omega. \tag{10}$$

Возможно использования и других критериев [4].

В частности, если ЛПР не известны априорные вероятности состояний среды, но известно отношение порядка, заданное на этих состояниях, возможно использование критериев третьей информационной ситуации [4], в частности, оценок Фишборна.

После того, как выбраны компромиссные функции стоимости, исходная задача бесконечномерного математического программирования с множественнозначной функцией стоимости преобразуется в стандартную задачу ОРМ, для которой имеются хорошо разработанные методы и алгоритмы решения [1].

Сформулируем алгоритм решения задачи (1) – (4).

1. Учитывая имеющуюся информацию относительно ситуации принятия решения выбираем один из критериев (5) – (10).
2. Используя выбранный критерий преобразуем исходное множество целевых функций в единственную компромиссную функцию. Тем самым исходная множественнозначная задача сводится к обычной задаче ОРМ.

3. Решаем полученную задачу оптимального разбиения множеств, используя метод ОРМ, описаный в [1].

Модельная задача и анализ полученых результатов. Потребитель некоторой однородной продукции, производимой тремя предприятиями, непрерывно распределен в области $\Omega = \{(x, y) \in R^2 | 0 \le x \le 10, 0 \le y \le 10\}$. Заданы начальные координаты расположения предприятий $\tau_i = (\tau_i^x, \tau_i^y) = (0,0),\ i = 1,2,3$. Известен спрос $\rho(x, y)$ на продукцию в каждой точке (x, y) области Ω, для простоты полагаем $\rho(x, y) = 1$ для всех $(x, y) \in \Omega$; а также множество возможных значений стоимости доставки продукции $c_i(x, \tau_i) = \{c_i^1(x, \tau_i), c_i^2(x, \tau_i), \cdots c_i^{M_i}(x, \tau_i)\}$ $i = \overline{1,3}$ – из пункта производства τ_i в пункт потребления x, в зависимости от состояния среды θ_j, $j = 1,5$: $c_i^j(x, \tau_i) = \gamma_i^j \sqrt{\alpha_i^j (x - \tau_i^x)^2 + \beta_i^j (y - \tau_i^y)^2}$.

Состояние среды	функция стоимости доставки								
	предприятие 1			предприятие 2			предприятие 3		
	α_1^j	β_1^j	γ_1^j	α_2^j	β_2^j	γ_2^j	α_3^j	β_3^j	γ_3^j
θ_1	1	1	1	1	1	1	1	1	1
θ_2	0,5	1	1	1	1	2	1	1	2
θ_3	1	1,5	2	0,5	0,5	0,5	1	1	1
θ_4	0,5	1	2	1,5	3	2	2	3	1
θ_5	1	1	1	2	2	1	1	1	4

Предполагается, что прибыль производителя зависит только от транспортных расходов.

Мощность i-го производителя определяется суммарным спросом обслуживаемых потребителей и не должна превышать заданных объемов b_i, $i=\overline{1,3}$: $b_1=25$, $b_2=60$, $b_3=50$. Необходимо разбить область Ω на зоны обслуживания каждым из производителей, т. е. на подмножества Ω_1, Ω_2, Ω_3, так, чтобы суммарные затраты на доставку продукции были минимальны.

При использовании критерия Байеса (9) был получен следующий результат:

Координаты центров:

τ_1: (7.16, 2.03);

τ_2: (2.56, 5.16);

τ_3: (7.42, 7.52).

Оптимальные значения мощности производителей: 25.00; 47.50; 27.50.

Значение компромиссного целевого функционала: 485.99.

Оптимальное разбиение представлено на рис. 1.

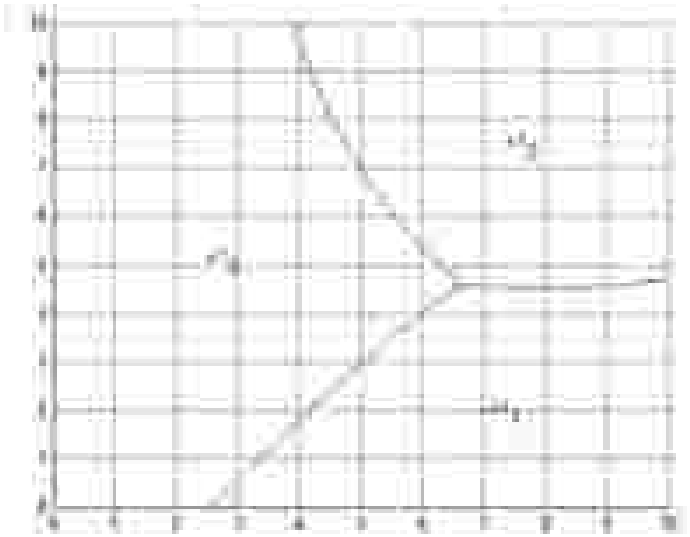


Рис.1. Оптимальное разбиение, полученное с использованием критерия Байеса

Используя критерий Гурвица с константой 0,7 были получены следующие результаты:

Координаты центров:

τ_1: (2.76, 2.15);

τ_2: (5.02, 7.14);

τ_3: (7.99, 2.22).

Оптимальные значения мощности производителей: 24.97; 56.67; 18.36.

Значение компромиссного целевого функционала: 402.90.

Оптимальное разбиение представлено на рис. 2.

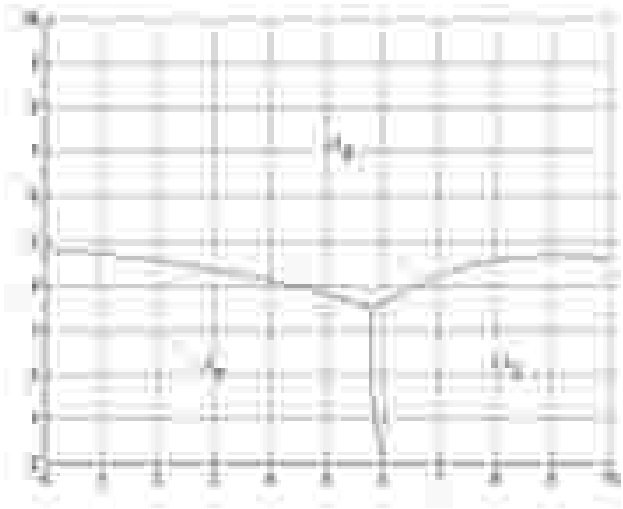


Рис. 2. Оптимальное разбиение, полученное при использовании критерия Гурвица с константой 0.7

Выводы. Использование различных критериев для перехода от множественнозначной задачи к скалярной, дает различные компромиссные функции стоимости и , соответственно, различные решения исходной задачи. Поэтому, следующей задачей является проверка эффективности полученных решений.

Обобщением рассмотренной задачи будут бесконечномерные задачи размещения в нечетких условиях. Некоторые математические модели таких задач были сформулированы в [5].

Библиографические ссылки

1. **Киселева Е.М.** Непрерывные задачи оптимального разбиения множеств: теория, алгоритмы, приложения: Монография / Е.М. Киселева, Н.З. Шор – К., 2005. – 564 с.
2. **Кісельова О.М.** Метод розв'язання задачі нечіткого розбиття множин / О.М. Кісельова, О.Ю.Лебідь// Питання прикладної математики і математичного моделювання: збірник наукових праць. – 2010. – С.139–145
3. **Кісельова О.М.** Методи розв'язання задач нечіткого розбиття множин із нечіткими параметрами у функціоналі / О.М. Кісельова, О.Ю.Лебідь// Питання прикладної математики і математичного моделювання: збірник наукових праць.– 2007. – С.115–123.
4. **Трухаев Р.И.** Модели принятия решений в условиях неопределенности / Р.И. Трухаев – М., 1981.
5. **Ус С.А.** О моделях оптимального разбиения множеств в условиях неопределенности / С.А. Ус // Питання прикладної математики і математичного моделювання: збірник наукових праць.– 2010, – С. 320–326.

Надійшла до редколегії 15.05.11

УДК 519.8

[10]А.Д. Фирсов
Днепропетровский национальный университет имени Олеся Гончара

АЛГОРИТМ ОБХОДА ОБЪЕКТОВ НА ПЛОСКОЙ КАРТЕ

Розглядається проблема пошуку оптимальної траєкторії обходу об'єктів на плоскій карті. Запропоновано алгоритм перебору траєкторій та їх відбору. Ефективність алгоритму дозволяє використовувати його в реальних умовах.

Рассматривается проблема поиска оптимальной траектории обхода объектов на плоской карте. Предлагается алгоритм перебора траекторий и их отбора. Эффективность алгоритма позволяет его использовать в реальных задачах.

The problem of finding the optimal path transversal of objects on a flat map is considered. The algoritm of path sorting and its selection is proposed. The algorithm effectiveness allows its use in real applications.

Ключевые слова: траектория, обход, робот, граф.

Внедрение технологий автоматического управления, требует построения математических моделей, формализующих прикладные проблемы и дающих возможность найти оптимальные решения прикладных задач. В частности, актуальным направлением является построение информационных систем в области транспорта и инфраструктуры, которые, в свою очередь, состоят из высокоуровневых управляющих систем, оптимизирующих затраты и принимающих решения, а также бортовых систем, анализирующих местность и действующих в соответствии с внешними инструкциями, при этом имеющими возможность работы в автономном режиме, требующем частичной реализации высокоуровневой функциональности.

Одной из актуальных задач, которые регулярно решаются при построении оптимальных траекторий, является задача построения оптимального маршрута между двумя точками, где расположены непроходимые препятствия, имеющие фиксированные размеры, меньшие чем линейные размеры карты, на которой определяется путь [1;3]. Данная задача может решаться как бортовой, так и управляющей системой. Регулярное решение

данной задачи подразумевает ограничение на время счета, а соответственно на требование к решающему алгоритму. На рис.1. приведен пример карты с объектами и оптимальным маршрутом.

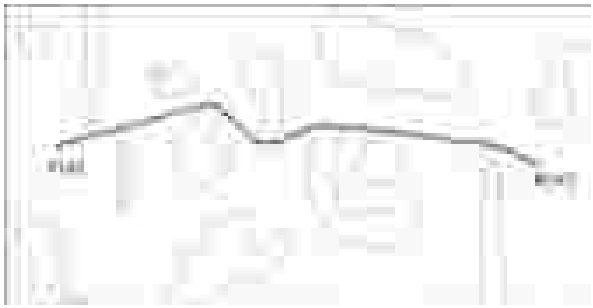


Рис.1. Оптимальная траектория движения между двумя точками X1 и X2 на ограниченной карте с тринадцатью препятствиями

Проблема поиска оптимальной траектории может рассматриваться на разных уровнях абстракции. Это определение точного пути с учетом геометрии (формы границ) объектов обхода, либо учет только предельных координат этих объектов, что приведет к построению модели, позволяющей получать приближенное решение при построении маршрута.

Назовем предельными координатами объекта обхода две точки с координатами (xl_i, yl_i), (xr_i, yr_i) $i=1,\ldots,n$, (n − число объектов), которые определяются как координаты пересечения границы объекта и наиболее удаленной параллельной прямой слева и справа для линии, проведенной через точки, между которыми ведется поиск оптимального пути (Рис.2). В случае, когда параллельная прямая и граница области пересекаются в нескольких точках то в качестве предельной координаты может выступать любая из них. Для определенности будем выбирать ближайшую к ($X1,Y1$).

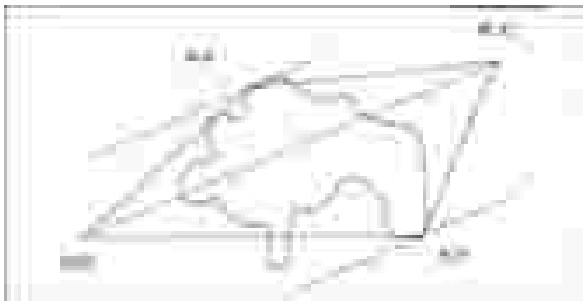


Рис.2. Объект лежащий между точками ($X1,Y1$),($X2,Y2$) и его предельные координаты

Такое определение предельных координат позволяет перейти от рассмотрения сложных границ объектов к приближению их отрезками, соединяющими точки (xl_i, yl_i), (xr_i, yr_i) $i=1,\ldots,n$. Тогда траектория будет вычисляться как сумма длин отрезков , соединяющих последовательно предельные точки отрезков между точкой ($X1,Y1$) и предельной точкой, точкой ($X2,Y2$) и предельной точкой.

Смоделировать траектории обхода объектов , замененных на соответствующие отрезки возможно при помощи $k+2$ дольного графа G, где k-число отрезков, каждый из которых соответствует двухвершинной доле. Вершины графа соединим взвешенным ребром, если между предельными координатами объекта проходит возможная траектория. Вес ребра определим как расстояние между соответствующими точками на карте. В результате получим граф как показано на рис.3.

а) б)

Рис.3. Справа приведен пример карты, в которой объекты заменены на отрезки, определены и соединены возможными траекториями предельные точки. Слева граф G, моделирующий набор траекторий между $X1$ и $X2$. Веса ребер в графе равны длинам кратчайших путей между соответствующими предельными точками на карте

Задача поиска кратчайшего пути между двумя вершинами в графе широко известна, в данном случае – веса положительны, поэтому найти решение можно при помощи алгоритма Дейкстры [2].

В связи с тем, что объекты обхода имеют произвольную форму, приближение при помощи отрезков может давать результаты, отличающиеся от точных в несколько раз. Пример приведен на рис.4.

Рис.4. Толстая линия показывает приближенную траекторию, реальное расстояние определяется путем обхода объекта из точки $X1$ в $X2$

Таким образом, для применения алгоритма Дейкстры возникает проблема точного определения весов дуг в графе G, построенном по описанной выше схеме. Что, в свою очередь, означает необходимость определения кратчайших допустимых траекторий обхода для объектов.

Допустимой траекторией между объектами назовем отрезок на прямой, соединяющий точку на границе объекта А и точку на границе объекта В так, что данный отрезок не проходит ни через одну точку принадлежащую границе любого из объектов на карте.

Касательной допустимой траекторией между объектами А и В назовем отрезок, лежащий на прямой, являющейся касательной к границе каждого объекта. Для двух объектов таких касательных может быть четыре.

Траекторией обхода объекта А назовем кратчайший путь между двумя точками X1 и X2, лежащими на его границе, такой, что ни одна из точек этого пути не лежит внутри объекта А.

Допустимой траекторией между точками X1 и X2 в случае, если точки отрезка прямой, соединяющей эти точки, лежат внутри объекта А, будем считать путь, составленный из отрезка касательной, проходящей из точки X1 к объекту А (до точки x1), отрезка касательной, проходящей из точки X2 к объекту А (до точки x2) и траектории обхода объекта А. Таких траекторий может быть четыре.

Заметим, что точки $X1$ и $X2$ могут лежать на границах объектов (в точках пересечения границы и касательной допустимой траектории). Тогда допустимой траекторией между точками $X1$ и $X2$ будет последовательность – касательная для объекта (если он лежит на пути из $X1$ в $X2$), траектории обхода этого объекта, касательная между текущим объектом и следующим (если он лежит на пути из конечной точки траектории обхода), траектории обхода следующего объекта и т. д., заканчивая касательной, проходящей через точку $X2$.

Замкнутым контуром объекта назовем траекторию обхода объекта, начинающуюся и заканчивающуюся в одной точке, лежащей на его границе, при этом одна из координат этой точки принимает экстремальное значение среди всех координат точек границы объекта.

Точку замкнутого контура объекта назовем крайне левой и обозначим W_i (i-номер объекта), если ее координата по оси x минимальна.

Точку замкнутого контура объекта назовем крайне правой и обозначим O_i (i-номер объекта), если ее координата по оси x максимальна.

Точку замкнутого контура объекта назовем крайне верхней и обозначим N_i (i-номер объекта), если ее координата по оси y максимальна.

Точку замкнутого контура объекта назовем крайне нижней и обозначим Z_i (i-номер объекта), если ее координата по оси y минимальна.

Так как в контуре может быть несколько точек, удовлетворяющих определению, то для однозначности выберем из них ту, у которой вторая координата минимальна.

Введем понятие направления для допустимой траектории между точками X1 и X2, которое используем при представлении объектов отрезками.

Зафиксируем точку X1($x1$, $y1$), тогда точка X2($x2$, $y2$) может располагаться относительно X1 в двух областях, задаваемых условиями:

1) $|x1-x2|<|y1-y2|$;
2) $|x1-x2|\geq|y1-y2|$.

Направленной вверх будем считать траекторию, проходящую из точки X1 в X2, если выполняется первое условие, направленной влево, если выполняется второе.

Построим граф G, согласно введенным определениям:

1. Проверим направление траектории.
2. Для каждого из объектов на карте определим крайние точки W,O, N,Z.
3. Пронумеруем объекты. Для определенности пронумеруем слева направо согласно возрастанию координаты x точки W каждого из объектов, если у объектов эти координаты совпадают, то перенумеруем согласно возрастанию координаты y.
4. Проложим все допустимые траектории между всеми крайними точками, точками X1, X2 и крайними точками, точками в которых начинаются и заканчиваются траектории обхода объекта при заданном направлении обхода.
5. Выберем в качестве вершин графа крайние точки объектов в соответствии с направлением траектории между начальной и конечной точка-

ми (если такая траектория существует) и точки в которых начинаются и заканчиваются траектории обхода объекта.

6. Определим расстояния вдоль траекторий , полученных в п.4 между всеми выбранными точками.
7. Удалим путь между двумя точками, если существует более короткий путь между ними. Удалим путь , заканчивающийся в крайней точке, начальной или конечной точке обхода объекта.
8. Соединим ребрами вершины в том случае, если они лежат на одной траектории.
9. Присвоим веса ребрам согласно расстояний, полученных в п.6.
10. Удалим, за исключением одного, ребра, соединяющие две вершины, в случае если они имеют одинаковые веса.

Проиллюстрируем графически ситуацию с траекториями, которая может сложиться при некотором размещении объектов на карте и выполнении пункта 4 алгоритма.

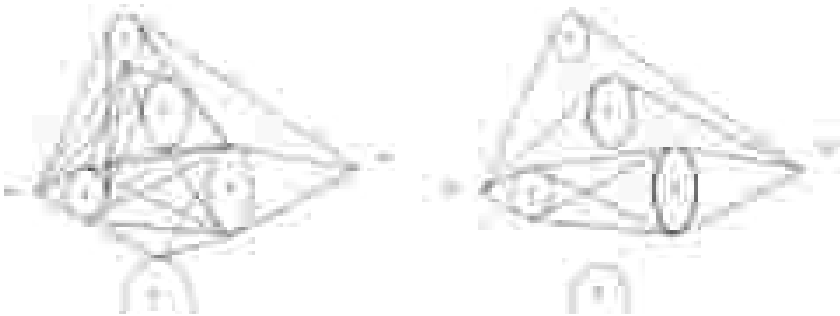


Рис.5. Траектории обхода пяти объектов, согласно реализации пункта 4 алгоритма (слева) и траектории обхода после реализации пункта 8 (справа)

В результате выполнения пункта 4 алгоритма получаем все возможные пути обхода объектов через их крайние точки и точки начала и конца траектории обхода объектов. Очевидно, что среди построенных траекторий есть оптимальная. Также очевидно, что число построенных траекторий избыточно, но в данном случае частичное удаление траекторий , соединяющих две точки и имеющих большую длину среди возможных , требует дополнительной проверки, которую можно делать либо на этапе построения для каждой траектории, либо отдельной процедурой сравнения после

их построения, как предлагается в данном случае. Сложность проверки одинакова, но шаги алгоритма могут быть реализованы независимо.

Рассмотрим также на примере реализацию пункта 5 алгоритма. Выберем объект номер 1 , приведенный на рис.5 после удаления избыточных ребер.

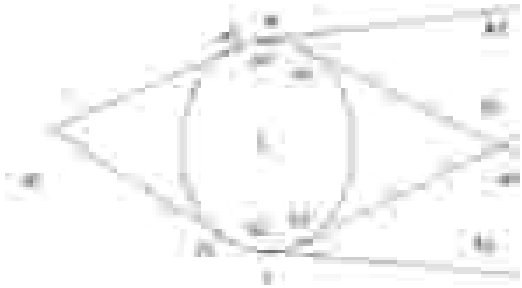


Рис.6. Увеличенный фрагмент рис.5. с размеченными точками начала и конца траекторий обхода и предельными точками. Для удобства описания окончания траекторий помечены

На рисунке 5 приведен объект эллипсообразной формы, который необходимо обойти по определенным на шаге 5 траекториями (одна из них является оптимальной). В общем случае объект может иметь произвольную форму, но даже в случае эллипса возникает несколько вариантов траекторий его обхода, на рис.6. это: a –a2 траектории X1–A2, a1–a3 траектории X1–A3, $b1-b2$ траектории X1– B2, $b1-b3$ траектории X1– B3. Соответственно точки $a2,a3,b2,b3$ будут вершинами строящегося графа G (точки $a1$, $b1$ в данном случае могут быть исключены).

Далее реализуя пункты 7 и 8 получим граф G для набора объектов приведенных на рис.5.

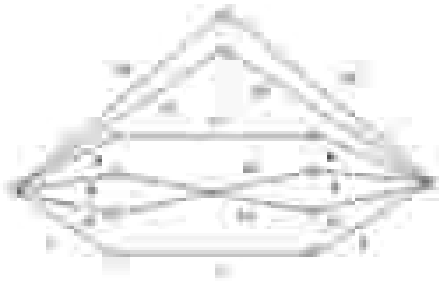


Рис.7. Граф расстояний между крайними точками обхода объектов, полученный после применения алгоритма

Далее для такого графа применяем алгоритм Дейкстры и получаем траектории минимальной длины.

Сложность предложенного алгоритма определяется процедурой построения касательных к объектам, процедурой построения траектории обхода для каждого из объектов, процедурой попарного сравнения длин участков траекторий и дальше непосредственно алгоритмом Дейкстры для графа G. Первые две линейно зависят от числа объектов, третья не превосходит k^2, где k – число траекторий между объектами, алгоритм Дейкстры не превосходит n^2, где n – число вершин в графе. Следовательно, оценкой сверху будет величина k^2+n^2.

Таким образом, предложенный алгоритм является полиномиальным и позволяет не только точно, но и эффективно в смысле числа операций решить задачу поиска кратчайшей траектории между двумя точками, расположенными на плоской карте с препятствиями.

Библиографические сылки

1. **Терещенко В.М.** Підхід до пошуку оптимального шляху між двома точками на множині перешкод. / Терещенко В.М., Янчик Д., Пустовойтов Д., Чернишов Е. // Штучний інтелект. N4. – 2010. С.297–303.
2. **Томас Х. Кормен** Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е изд. / Кормен Томас Х., Лейзерсон Чарльз И., Ривест Рональд Л., Клиффорд Штайн — М., 2006. — 1296 с.
3. **Bryan Stout.** Smart Move: Intelligent Path-Finding / Bryan Stout // Game Developer Magazine, October 1996. – P. 28–35.

Надійшла до редколегії 21.02.11

УДК 621.372

[11]Д.А. Цымбал
Новгородский государственный университет им. Ярослава Мудрого

АЛГОРИТМ РАСПОЗНАВАНИЯ РАЗМЫТЫХ ИЗОБРАЖЕНИЙ БАРКОДОВ UPC-A

Розглядається практична реалізація системи розпізнавання баркодів за допомогою камери мобільного телефону. Пропонується розвязок проблеми розпізнавання в умовах обмежень на апаратні можливості. Використовуються стандартні методи обробки зображень, застосовані до спеціальних шаблонів.

Рассматривается практическая реализация системы распознавания баркодов при помощи камеры мобильного телефона. Предлагается решение проблемы распознававания в условиях ограничений на аппаратные возможности. Используются стандартные методы обработки изображений, примененные к специальным шаблонам.

The practical implementation of bar c odes' system recognition using mobile phone camera is considered. The solution of recognition problem with constraints of hardware possibilities is proposed. It was used the standard methods of image processing applied to special templates.

Ключевые слова: баркод, шаблон, распознавание, мобильная платформа.

Введение. Современные мобильные технологии качественно меняют наши представления о взаимодействии между людьми, объектами и информационными потоками. Пользователь, владеющий смартфоном со скоростным доступом в Интернет, может как пользоваться внешним данными, так и сам создавать данные, а с учетом имеющейся бортовой вычислительной мощности, выполнять полноценный анализ данных на лету. При этом функции, которые ранее были доступны только на ПК, фактически извлекаются из кармана, и что, немаловажно, предоставляются в дружелюбной среде.

Целью данной работы является построение алгоритма, позволяющего распознавать изображения баркодов (штрих-коды) UPC-A нанесенных на продукты при помощи мобильного устройства iPhone 3G.

Код UPC-A состоит из 12 цифр. Последняя цифра содержит контрольную сумму, вычисленную по первым 11 цифрам. Каждая цифра кодируется 7-битной последовательностью, содержащей две группы нулей и две

группы единиц. Левая и правая части кодируются каждая своим кодом (правый код является обратным по отношению к левому). Единица кодируется чёрным цветом, ноль – белым. Также баркод содержит сторожевые паттерны, которые размечают границы блоков: левый и правый образцы представляют собой трёхбитный код 101, центральный образец – пятибитный 01010. Общая ширина всего кода равна 95 битам (12 x7=84 информационных бита и 3+3+5=11 бит в паттернах).

Рис. 1. Пример изображения UPC_A кода

Постановка проблемы. Требуется разработать программный продукт, позволяющий пользователю получить в цифровом виде баркод, изначально получаемый как изображение камеры iPhone. При этом необходимо находить изображение самого баркода.

Алгоритм распознавания должен быть построен в расчёте на работу на устройствах, не обеспечивающих чёткость получаемого изображения.

Устройство iPhone 3G не обладает автофокусировкой (сфокусирован на даль), поэтому получаемые с его камеры изображения, находящиеся на расстоянии 10_15 см, оказываются размытыми.

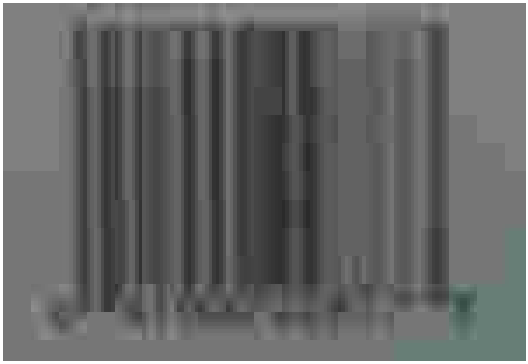


Рис. 2. Расфокусированное изображение UPC-A кода на iPhone 3G

Кроме того, разрешение изображения, получаемого на iPhone 3G в реальном времени (при видеосъёмке), является весьма низким. Так, одна цифра кодовой последовательности (7-битный блок кода) соответствует примерно 14–16 точкам изображения.

Поэтому задача заключается в распознавании числовой последовательности на основе размытого изображения баркода с низким разрешением за приемлемое время.

Схема обработки изображения. Если полагать изображение идеально чётким, то задача распознавания состоит из следующих этапов [1;4]:

- первичное обнаружение границ баркода;
- подготовка исходных данных (выделение одномерного сигнала);
- определение размера одного кодового бита;
- выделение 12-ти блоков, содержащих информационные биты (12 цифр);
- нахождение наиболее соответствующего кода для каждого из 12 блоков;
- проверка контрольной суммы.

Данная последовательность шагов фактически выполняется любым лазерным сканером. В случае камеры телефона, необходимо работать с изображением, которое сформировано из пикселей, размер которых сравним с информационно значимыми элементами изображения.

Сдвиг границ изображения в одном направлении на 0.5 точки приводит к тому, что происходит смещение границ всех значащих битов – следовательно, отдельный бит кода может быть получен с ошибкой 25 % своего размера (при $\mathrm{Bitlen} = 2$), что составляет серьёзную погрешность.

В таких условиях, нахождение кодовых последовательностей, наиболее соответствующих исходным данным, естественно производить методом аппроксимации реального сигнала эталонными данными.

Рис. 3. Аппроксимация исходного сигнала (0) значениями 0, 6 и 9

Это означает, что нужно построить эталонные сигналы для всех кодовых комбинаций (в UPC-A их по 10 для каждой цифры, 0..9), и выбрать тот вариант, который наиболее близок к реальному.

В случае, когда баркод при сканировании находится не в фокусе, изображение получается размытым (нерезким). В этом случае сигнал можно представить следующим образом [4]

$$U = Blur(U_{ид.}) + n,$$

где $Blur(U)$ − функция расфокусировки, $U_{ид.}$ − идеально резкий сигнал, n − шум.

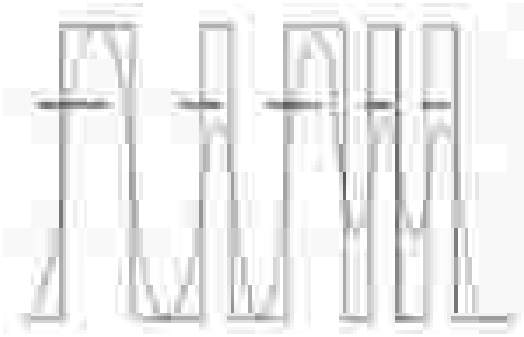


Рис. 4. Идеальный и нерезкий сигналы

Восстановление резкости такого изображения должно опираться на знание о природе размытия и представляет собой сложную вычислительную задачу.

В то же время, количество корректных комбинаций значений кода ограничено 10 для каждого блока.

Поэтому рациональным подходом является применение метода аппроксимации эталонным сигналом, дополненного размывающим фильтром. То есть, реальные данные в каждом блоке сравниваются поочерёдно с десятью эталонами, пропущенными через фильтр.

Для эмуляции размытия используется гауссово (нормальное) распределение – как наиболее естественный вид распределения случайных величин. Экспериментально проверено, что реальные данные оказываются весьма близко к эталону, полученному гауссовым размытием идеальных данных.

Дискретный фильтр Гаусса применяется к каждой точке одномерного сигнала (значение сигнала в точке обусловлено значениями точек в некоторой окрестности; значения накапливаются с соответствующими весами)

$$x'_i = \sum_{k=-r}^{r} \left(x_{i+k} \cdot G_{m,\sigma}(k) \right),$$

где $G_{m,\sigma}(x)$ – веса для фильтра, r – радиус окна фильтра.

Веса вычисляются следующим образом:

$$G_{m,\sigma}(x) = \begin{cases} 0, & x < -r \\ \Gamma_{m,\sigma}(x+0.5), & x = -r \\ \Gamma_{m,\sigma}(x+0.5) - \Gamma_{m,\sigma}(x-0.5), & -r < x < r \\ 1 - \Gamma_{m,\sigma}(x-0.5), & x = r \\ 0, & x > r \end{cases}$$

$\Gamma_{m,\sigma}(x)$ — функция нормального распределения.

Для создания образцов сходных с получаемыми с камеры изображениями были эмпирически подобраны оптимальные параметры фильтра Гаусса. Ширина окна фильтра равна 7 ($r=3$), $m=0$, $\sigma=2$.

Обнаружение границ. Локализовать баркод на изображении можно на основе краевых пикселей полученных как значения изменения градиента яркости. Для этого используется оператор Собеля с применением горизонтальной и вертикальной матриц преобразования [2].

По результатам применения оператора получается первая оценка границ баркода – значения start и end (координата первого крайнего левого пикселя с максимальной яркостью и координата крайне правого пикселя с максимальной яркостью).

Уточнение границы через поиск сторожевых паттернов. Метод Собеля даёт приблизительное положение границ. Поскольку границы баркода на расфокусированном изображении получаются нерезкими, ошибка в нахождении границ может достигать нескольких точек.

Уточним положение границ, применив метод уточняющей аппроксимации для поиска наилучшего положения сторожевых паттернов.

Для этого строится битовая последовательность, представляющая собой сторожевой паттерн и пустое («белое») поле значительной ширины (чтобы не перепутать сторожевой паттерн с такой же кодовой подпоследовательностью – 0101, – следующей за ним). Т. е. для левого паттерна происходит подбор наилучшей позиции для кода 000101, а для правого – 101000. В качестве критерия наилучшего положения границы служит ми-

нимум ошибки при сравнении реальных данных с построенным эталоном паттерна.

Вычисление границ блоков. Ширина бита . Как уже было сказано ранее, баркод UPC-A состоит из 95 информационных битов. Таким образом, ширина бита вычисляется как

Bitlen = (end − start) / 95.0, где start – начальная позиция баркода на изображении, end – конечная.

Ширина левого и правого образцов - side_pattern_size = (Bitlen * 3), а центрального – center_pattern_size = (Bitlen * 5).

Ширина блока данных - block_size = (Bitlen * 7).

Таким образом, шесть левых блоков имеют следующие границы:

Block_start[i] = left_start + i * block_size,

Block_stop[i] = Block_start[i] + block_size,

где left_start = start + side_pattern_size, i=0..5.

Шесть правых блоков имеют границы:

Block_start[i] = right_start + i * block_size,

Block_stop[i] = Block_start[i] + block_size,

где right_start = start + side_pattern_size + center_pattern_size + (6 * block_size) = start + (Bitlen * 50), i = 0..5.

Построение и размытие исходных сигналов . После того, как вычислены границы блоков, для каждого из них производится аппроксимация сигнала эталонными значениями.

Предварительно вычисляются оценочные амплитуды сигнала: максимальная и минимальная. Максимум и минимум выбираются, исходя из характеристик сигнала.

Для этого отбирается несколько значений (в текущей реализации – 3) для максимальной амплитуды сигнала: абсолютный максимум (255); значение 90 % от наблюдаемого максимума; максимальное значение сигнала за выбросом 10 % (т. е. отбраковываются 10 % наибольших значений, и среди оставшихся берётся максимум). Далее выбирается то из значений, которое даёт меньшую погрешность при аппроксимации сторожевых паттернов.

Затем для каждого из 12 блоков производится ряд итераций сравнения с эталонным сигналом (по числу возможных кодовых комбинаций, для UPC-A это всегда 10 вариантов).

Каждая итерация состоит из следующих шагов:

1. Построение очередного эталонного сигнала, масштабируемого по ширине блока.
2. Размытие эталонного сигнала фильтром Гаусса.

3. Расчёт ошибки для текущей аппроксимации относительно реальных данных.

На первом этапе эталонная кодовая последовательность (7 бит) масштабируется по ширине блока и нормируется по диапазону амплитуд реального сигнала

$$curr'_j[x] = Min + Delta * code_j\left[\frac{(x - Block\_start[i]) * 7}{block\_size}\right],$$

$$Delta = Max - Min,$$

где Max – максимальная амплитуда, Min – минимальная амплитуда сигнала; $code_j$ – 7-битная кодовая комбинация для цифры j, $x \in [Block\_start[i]; Block\_stop[i]]$.

На втором этапе фильтр Гаусса применяется к полученным эталонным данным в диапазоне [Block_start[i]; Block_stop[i]]

$$curr_j[x] = \sum_{k=-r}^{r} \left(curr'_j[x+k] * G_{m,\sigma}(k)\right),$$

$$x \in [Block\_start[i]; Block\_stop[i]],$$

где $curr'$ – масштабированный эталонный сигнал, нормированный по амплитудам.

На третьем этапе вычисляется ошибка данной аппроксимации, опираясь на критерий наилучшего варианта, изложенный далее.

Критерий поиска наилучшего варианта. Наилучшим вариантом для i-того блока считается вариант, дающий минимальное значение средневзвешенной квадратичной ошибки на интервале (Block_start[i], Block_stop[i]), т. е. Best_code[i] = k,

$$diff[i,k] = \min_{j}(diff[i,j]), \quad j = 0..9$$

$$diff[i,j] = \sum_{x=Block\_start[i]}^{Block\_stop[i]} \left(data[x] - curr_j[x]\right)^2$$

Таким образом, на i-том блоке вычисляется ошибка $diff[i,j]$ для всех вариантов значения j (цифр от 0 до 9). В качестве наилучшего варианта выбирается значение j, для которого ошибка минимальна.

Уточнение границ через итеративную аппроксимацию. Уточнение положения границ баркода повышает качество распознавания и уменьшает количество ошибок.

Алгоритм повторно уточняет границы баркода, производя сдвиг границ в радиусе 2 точек. Т. е. сдвиг составляет от –2 до 2 относительно текущего положения с шагом 0.5.

Для каждого из положений границы выполняется аппроксимация баркода идеальным сигналом: при поиске левой границы аппроксимируются первые шесть кодовых блоков, при поиске правой – вторые шесть (правая часть).

Границы сдвигаются поочерёдно: сначала производится поиск наилучшего положения для левой границы, затем – для правой. После того, как лучшее положение левой границы найдено, оно устанавливается в качестве текущего, и поиск правой границы производится с учётом найденной левой.

После каждого сдвига границ происходит оценка качества аппроксимации. При сдвиге левой границы оценка выполняется по левой группе данных (первые 6 цифр), при сдвиге правой – по правой группе (цифры с 7 по 12).

Затем выполняется финальная итерация аппроксимации баркода, выполняемая для найденных наилучших границ. Целью является получение наиболее достоверного приближения, от которого в дальнейшем отталкивается эвристика для поиска корректного баркода.

Оценка наилучшей аппроксимации. Как сказано выше, оценка аппроксимации при уточнении каждой из границ производится отдельно.

Критерий для левой границы выглядит следующим образом:

Если $\mathrm{BestCode}[i] = k_i$,

где

$$diff[i,k_i] = \min_j(diff[i,j]), \quad j = 0..9, \quad i = 1..6,$$

то наилучшим называется такое смещение границы δ_i (т.е. $start_i = start + \delta_i$) что

$$\forall \delta \in [-r..r]: \max_i(diff_{\delta_i}[i,k_i]) \le \max_i(diff_\delta[i,k_i])$$

Т. е., не существует другого допустимого сдвига в пределах [– r, r], для которого максимальная ошибка в блоках опорного баркода превышает такую же ошибку при сдвиге границы на δ_i.

Соответственно, для правой границы критерий аналогичен:

BestCode[i] = k_i,

где $diff[i,k_i] == \min_j(diff[i,j]),\ j = 0..9,\ i = 7..12,$

то наилучшим называется такое смещение правой границы δ_i (т.е. $end_i = end + \delta_i$), что

$$\forall \delta \in [-r..r]: \max_i(diff_{\delta_i}[i,k_i]) \le \max_i(diff_{\delta}[i,k_i]).$$

Исправление ошибок. Баркод считается прочитанным верно, если контрольная сумма сходится, и при этом значение ошибки для каждой из цифр последовательности не превышает определённого предела:

$$diff[i,k_i] \le Delta * C,$$
$$diff[i,k_i] == \min_j(diff[i,j]), \quad j = 0..9,$$
$$i = 1..12$$

где $Delta = Max - Min$, C – константа, примерно равная 1, подбираемая экспериментально (текущее значение 1.15).

Проверка на размер ошибки обеспечивает фильтрацию изображений, на которых был обнаружен «ложный» баркод или баркод с ошибками в границах – в этом случае средние ошибки в кодовых блоках будут достаточно велики.

Проверка контрольной суммы. При считывании кода правильность контрольной суммы проверяется следующим образом:

1. Суммируются все нечётные (при нумерации с 1) цифры и умножаются на 3.
2. Суммируются все четные цифры, включая контрольную цифру (последнюю).
3. Две эти суммы складываются по модулю 10.

Если в результате получился ноль, то контрольная сумма сошлась, и баркод является корректным.

В случае если контрольная сумма не сошлась, предпринимается попытка достичь корректной суммы путём поиска среди ближайших правдоподобных вариантов, которые хранятся в таблице со средневзвешенными квадратичными ошибками для допустимых кодовых комбинаций в блоках.

Пусть найден наилучший вариант приближения (т.е. такой, который даёт минимальную суммарную ошибку). Этому варианту соответствует таблица средних ошибок, где каждому допустимому значению на каждой

(i-той) позиции в коде соответствует вычисленное значение средневзвешенной квадратичной ошибки *diff[i,j]*.

Для каждого допустимого варианта из таблицы вычисляются относительные отклонения от лучшего текущего варианта:

$$\mathbf{mindiff[i]} = \min_j(diff[i,j]), \qquad j = 0..9$$

$$ratio[i,j] = \frac{(diff[i,j] - \mathbf{mindiff[i]})}{\mathbf{mindiff[i]}}, \qquad i = 1..12, \quad j = 0..9$$

Таким образом, ratio[i,j]>=0 для любого *j*, ratio[i,j]==0 для варианта с минимальной ошибкой, и чем выше ошибка – тем выше относительная ошибка.

Таблица позволяет накапливать результаты аппроксимации по нескольким изображениям (2 и более). В этом случае ratio[i,j], полученное для аппроксимации текущего реального сигнала, добавляется к сумме значений, накопленных на предыдущих итерациях.

По минимальным значениям ratio[i,j] строится первичный вариант баркода. Т. е. формируется такой цифровой код $\{c_1,..,c_{12}\}$, для которого

$$ratio[i,c_i] == \min_j(ratio[i,j]), \qquad j = 0..9$$

Если полученный код успешно прошёл проверку контрольной суммы, этот код является искомым результатом, и работа алгоритма распознавания на этом прекращается.

В случае, когда данный код не прошёл проверку контрольной суммы, производится поиск ближайшей корректной последовательности.

Поиск ближайшей корректной последовательности . Если в результате обработки результатов аппроксимации был получен код $\{ c_1,..,c_{12}\}$, не прошедший проверку контрольной суммы, это означает, что в коде присутствуют ошибочные значения.

В случае небольших ошибок (т. е. когда истинное и ошибочное значения имели близкие значения средней ошибки, и ratio[i,c_i]~=ratio[i,c_i'], где c_i' – корректное значение), можно компенсировать возникающую ошибку.

Для этого необходимо найти такой код $\{ c_1',..,c_{12}'\}$, что для него сходится контрольная сумма, и этот код является ближайшим к $\{ c_1,..,c_{12}\}$ корректным кодом. То есть,

$$\exists [c_1' \ldots c_{12}']$$
$$- \text{корр.}, \forall \text{корр.} [c_1'' \ldots c_{12}'']: \sum_{i=1}^{12} ratio[i, c_i'] \leq \sum_{i=1}^{12} ratio[i, c_i''].$$

В таком случае, результатом алгоритма является код $\{c_1', \ldots, c_{12}'\}$.

Таким образом, задача сводится к поиску такой корректировки опорного кода, которая обладает верной контрольной суммой и минимальным совокупным отклонением.

Если при этом суммарное относительное отклонение достаточно велико, может получиться, что корректный код сильно отклоняется от реальных данных. Поэтому требуется установить порог, выше которого результат игнорируется, и результат распознавания считается неудовлетворительным.

Данный порог устанавливается путём калибровки алгоритма экспериментально и зависит от числа изображений, по которым накапливаются данные для эвристики.

Выводы. Результатом работы стал программный продукт, основанный на описанном выше алгоритме, который решает поставленную проблему. При этом на действия пользователя накладываются ограничения на размер вводимого изображения и конечно на исходное качество самого баркода (деформации, загрязнение).

Библиографические ссылки

1. **Васильев В.И.** Распознающие системы / В. И. Васильев. – К., 1983.
2. **Воробель Р.А.** Цифровая обработка изображений на основе теории контрастности: дис… доктора техн. наук: 05.13.06. – Львов, 1999. – 369 с.
3. **Фисенко В.Т.** Компьютерная обработка и распознавание изображений: учеб. пос. /В.Т. Фисенко, Т.Ю. Фисенко.
4. **Голд Б.** Цифровая обработка сигналов /Б. Голд, Ч. Рэйдер; пер. с англ. под ред. А.М. Трахтмана. – М., 1973. – 368 с.

Надійшла до редколегії 14.02.11

УДК 681.3.012:621.1

[12]Г.Г. Швачич[1], С.О. Чернецький[2], О.Г. Холод[3]
[1]Національна металургійна академія України,
[2]Дніпропетровський національний університет імені Олеся Гончара,
[3]Дніпропетровський університет економіки та права

ДЕЯКІ АСПЕКТИ КЛАСТЕРНИХ ТЕХНОЛОГІЙ МОДЕЛЮВАННЯ ЗАДАЧ МОНТЕ–КАРЛО

Розглядаються|розглядують| кластерні технології моделювання задач Монте_Карло. Показано, що лише|тільки| застосування|застосування| сучасних суперпродуктивних систем дозволило по_новому реалізувати| механізм відповідних розподілених обчислень|підрахунків|. Приводяться|наводять| схеми обчислень|підрахунків|, які забезпечують збільшення продуктивності і швидкодії|підрахунків|. Ефективність запропонованого підходу ілюструється порівняльним аналізом розв'язку деякого класу |низки|задач|задач|.

Рассматриваются кластерные технологии моделирования задач Монте_Карло. Показано, что только применение современных суперпродуктивных систем позволило по_новому реализовать механизм соответствующих распределенных вычислений. Приводятся схемы вычислений, которые обеспечивают увеличение продуктивности и скорости. Эффективность предложенного подхода иллюстрируется сравнительным анализом решения некоторого класса задач.

Clustering technologies of Monte-Carlo problems' modeling are considered. It is shown that using of modern superproduction systems only allowed to newly implementing the mechanism of relevant distribution calculations. We present computational scheme which provides productivity and speed increasing. The effectiveness of the proposed approach is illustrated by a comparative analysis of a certain class problems solution.

Ключові слова: метод Монте_Карло, кластер, паралельні обчислення, крайова задача.

Вступ. Серед різноманіття числових методів у математичних розрахунках останнім часом можна говорити про відродження таких методів, як методи Монте_Карло [1 – 4]. Ця назва об'єднує групу числових методів, основаних на одержанні|отриманні| великого числа реалізацій

стохастичного|самодифузія| процесу, який формується таким чином, щоб його ймовірнісні характеристики збігалися з|із| аналогічними величинами розв'язуваної|рішати| задачі|задачі|. Методи Монте_Карло використовуються для розв'язування задач|задач| у галузях фізики, математики, економіки, оптимізації, теорії керування та ін. |ймовірністьВітчизняні праці про методи Монте_Карло з'явились|появлялися| у 1955 – 1956 роках. З того часу написано багато наукових праць, що описують застосування цього методу [5 – 10]. Навіть поверховий перегляд назв робіт дозволяє зробити висновок|висновок| про застосовуваність методу Монте_Карло для розв'язку прикладних задач|задач| у багатьох галузях науки і техніки.

Відмітною рисою методів Монте _Карло вважається|з'являється| їх експериментальний характер. Для певності будемо називати методом Монте_Карло процедуру, що включає використання прийомів статистичної вибірки для наближеного розв'язку тієї чи іншої математичної або фізичної задачі|задачі|.

Методи Монте_Карло суттєво впливали|виявляв| і|суттєвий| впливають на розвиток методів обчислювальної математики, а при розв'язку певного класу задач|задач| успішно поєднуються|сполучається| з|із| іншими обчислювальними методами і доповнюють їх. Їх застосування|застосування| виправдане, в першу чергу, до тих задач|задачах|, які допускають теоретико_ймовірнісний| опис. Це пояснюєтьс|тлумачить|я як природністю одержанн|отримання|я відповіді |із|з певною заданою ймовірніст|ймовірністю|ю в задача|задачах|х, що мають|із| імовірнісний зміст|змістом|, так і суттєви|суттєвим|м спрощенням процедури розв'язку|вирішення| . На даний час|нині| існують статистичні методи розв'язування|вирішення| для деяких класів диференціальних рівнянь у частинних похідних, інтегральних рівнянь, задач|задач| про власні значення і системи лінійних алгебраїчних рівнянь.

Серед інших обчислювальних методів метод Монте _Карло виділяється своєю простотою і універсальністю. Повільна збіжність є істотним|суттєвим| недоліком|нестачею| методу, проте|однак| в даній статті|розділі| будуть описані його обчислювальні модифікації, які забезпечують високий порядок|лад| збіжності при застосуванні певних припущень. Правда, обчислювальна процедура при цьому ускладнюється і цим наближається до інших процедур обчислювальної математики. Збіжність методу Монте_Карло – це збіжність за ймовірністю|ймовірності|. Цю обставину навряд чи слід відносити до його недоліків|нестач|, бо ймовірнісні методи великою мірою виправдовують себе в практичних додатках. Що ж до задач|задач|, які мають ймовірнісний опис, то при їх розв'язуванні збіжність за ймовірністю|ймовірності| є навіть до певної міри природною.

Нарешті зауважимо|помітимо|, що точність обчислень|підрахунків| методу дуже залежить від якості застосовуваного генератора псевдовипадкових чисел, а швидкість обчислень|підрахунків| визначається функцією, що описує аналізований процес і, звичайно ж, продуктивністю самого «обчислювача». На сьогодні тактова частота сучасних процесорів вже перейшла межу ГГц, а обсяг|обсяг| оперативної пам'яті персональних комп'ютерів вимірюється в гігабайтах. Зважаючи на те, що відповідний клас задач|задач| буде оброблятися на персональному обчислювальному кластері, продуктивність «обчислювача» не є більше стримуючим чинником|фактором| (а скоріше,|скоріш за все| визначальним) для застосування|застосування| чутливих до потужності «обчислювача» числових алгоритмів при розв'язуванні багатовимірних|багатомірних| задач|задач|. Практичну ілюстрацію механізму роботи методу і деякі принципові особливості його реалізації будуть розглядатися|розглядуватимемо| на фоні|на фоні| розв'язку типових задач|задачі| теплофізики.

Постановка задачі. Цілі і задачі досліджень. Завдання, на вирішення якого спрямовано представлене дослідження, полягає у розробці та дослідженні деяких аспектів кластерних технологій моделювання задач Монте_Карло. Повільна збіжність методу є його істотним|суттєвим| недоліком|нестачею|, проте|однак| в даній статті|розділі| необхідно розробити його розподілені обчислювальні модифікації, які забезпечують високий порядок|лад| збіжності. При цьому для прискорення процесу обчислень алгоритм паралельних обчислень необхідно представити у вигляді наступної схеми. Генератор випадкових чисел генерує кожному «обчислювачу» випадкове число. Таким чином формується пошукова послідовність. Проміжні розрахунки здійснювати незалежно на різних, окремо взятих лезах кластера – «обчислювачах», а результати вже повинні оброблятися на окремо узятому MASTER – лезі – «аналізаторі». Такий підхід дозволяє позбавитися від неодмінної присутності між генератором випадкових чисел і «обчислювачем» маршрутизатора_комунікатора. Виграш продуктивності при цьому оцінити експериментальним шляхом. Таким чином, ставиться задача розробки відповідних ефективних алгоритмів розподіленого моделювання.

Практичну ілюстрацію механізму роботи розробленого підходу і деякі принципові особливості його реалізації необхідно розглянути|розглядувати| на прикладі розв'язку |на фоні| крайових|крайових| задач|задач| і задач|задач| з|із| початковими умовами для лінійних диференціальних рівнянь. Це пояснюється|тлумачить| тим, що зазначений клас

задач|задач| є|з'являється| однією із цікавих і перспективних областей додатків|застосування| методу Монте-Карло.

Вивести і проаналізувати аналітичні співвідношення, які дають статистичну оцінку досліджуваних величин, які застосовуються для наближеного вирішення досліджуваного класу задач.|задачі|

Показати, що розроблені алгоритми стійкі для будь-яких вхідних даних, мають максимальну паралельну форму, і, отже, мінімальний можливий час реалізації на паралельних обчислювальних системах|устроях|.

Виконати порівняльний аналіз розв'язків поставленої задачі числово-аналітичним методом і методом статистичних випробувань. Це дозволить якнайкраще оцінити похибку обчислень методом Монте-Карло. Для реалізації такого підходу ставиться задача розробки числово-аналітичного алгоритму для розв'язку досліджуваного класу задач.

Викладення основного матеріалу досліджень

Персональний обчислювальний кластер як ефективний засіб технології проведення складних розрахунків. Застосування|вживання| паралельних обчислювальних систем (*ПОС*|) є|з'являється,являється| стратегічним напрямом|направленням| розвитку обчислювальної техніки. Ця обставина викликана|спричинена| не тільки|не лише| принциповим обмеженням максимально можливої швидкодії звичайних|звичних| послідовних ЕОМ, але й практично постійним існуванням обчислювальних задач|задач|, для вирішення яких можливостей|спроможностей| існуючих засобів|коштів| обчислювальної техніки завжди виявляється|опиняється| недостатньо. До таких задач |задач| відносяться, наприклад, чисельне моделювання процесів гідродинаміки і металургійної теплофізики| [11,12], задачі |задачі| розпізнавання образів|зображень|, оптимізаційні задачі |задачі| з|із| великим числом параметрів, моделювання клімату, генна інженерія, проектування інтегральних схем, аналіз забруднення навколишнього середовища [13], рішення|розв'язання,вирішення,розв'язування| широкого кола|кола| багатовимірних|багатомірних| нестаціонарних задач |задач| [14] і т. д.

У наші часи істотний|суттєвий| інтерес до побудови|шикування| багатопроцесорних паралельних обчислювальних систем (кластерів) визначається застосуванням|застосуванням| стандартних загальнодоступних технологій і компонентів. Це зумовлено|зумовлює| рядом|низкою| чинників|факторів|. Відзначимо основні з|із| них. По-перше, зростання|зріст|, відповідно до потреб ринку, продуктивності таких стандартних мережевих|мережних| технологій як *Ethernet*| (характерна послідовним

підвищенням швидкості передачі інформації до 10, 100, 1000 Мбіт/с, застосуванням|застосування| комутаторів замість модулів з|із| розділюваним середовищем) дозволяє розглядати|розглядувати| їх як комунікаційне середовище|середовище| для багатопроцесорних обчислювальних систем. По_друге, дуже важливим чинником|фактором| стало збільшення популярності вільно поширюваної операційної системи *Linux*|. Дана операційна система спочатку позиціонувалась|позиціонувала| як варіант *UNIX*| для платформ на базі архітектури *Intel*|, але|та| досить|достатньо| швидко з'явилися|появлялися| версії для інших популярних мікропроцесорів, у тому числі й для лідерів за продуктивністю протягом останніх років – мікропроцесорів *Alpha*|.

З урахуванням|з урахуванням | економічних реалій нашої країни використання систем, побудованих|спорудити| на базі стандартних технологій, стає більш ніж актуальним. Причому залежно від особливостей задач|задач| і величини бюджету проект системи може мати всілякі|різноманітні| варіанти конфігурації. Найбільш доступним варіантом можна вважати стандартні материнські плати для процесорів *Intel*| *Pentium*| *III* і мережеві|мережні| адаптери *Fast*| *Ethernet*|. Вузли кластера об'єднуються між собою за допомогою комутатора *Fast*| *Ethernet*, розрахованого| на відповідне число портів. Кількість вузлів у системі та їх конфігурація залежить від вимог, що висуваються до обчислювальних ресурсів конкретними завданнями,|задачами| і залежать від фінансових можливостей користувачів|спроможностей|.

Крім того, гостра конкуренція між виробниками комп'ютерної техніки широкого вжитку приводить|наводить| до того, що ситуація з|із| цінами на ринку комплектуючих змінюється досить|достатньо| динамічно, особливо у зв'язку з випуском виробів нових моделей. Враховуючи також широкий асортимент сучасної електронної продукції, можна з|із| упевненістю констатувати, що при використанні стандартних комплектуючих можлива побудова|шикування| потужних|могутніх| обчислювальних систем загального|загального| призначення у вельми|дуже| стислі терміни з|із| максимально повним|цілковитим| урахуванням|урахуванням| потреб і можливостей|спроможностей| різних користувачів.

Останнім часом розвиток суперпродуктивних обчислювальних систем визначається застосуванням персональних обчислювальних кластерів (ПОК). Персональний обчислювальний кластер – це обчислювальна система, всі вузли якої розміщені в одному корпусі. На думку багатьох|низки| авторів [15 – 17], реального перелому в опануванні технологій паралельних обчислень|підрахунків| можна досягти розвитком багатопроцесорних обчислювальних систем *MPP*-архітектури| саме шляхом запроваджен-

ня|застосування| ПОК. Таким чином, створюється фундамент піраміди апаратних засобів для їхнього застосування в технології паралельних обчислень|підрахунків| у вигляді ПОК, аналогічний вже наявному фундаменту піраміди апаратури для традиційних технологій послідовних обчислень|підрахунків| у вигляді ПЕОМ. Відзначимо,|помітимо| що технології паралельного програмування і паралельних обчислень|підрахунків| можна вважати результатом широкого застосування|застосування| ПОК. Початок масового використання ПЕОМ припав на другу половину 1980 -х років, у той же час, за нашими прогнозами, середину першого десятиліття XXI -го ст. можна буде вважати|лічити| початком поширення|розповсюдження| ПОК у формі|у формі| мультипроцесорних обчислювальних систем з|із| розподіленою пам'яттю. Сфера застосування таких систем – опанування технологій паралельного програмування, створення|створіння| і налагодження паралельних програм, модельна прогонка розробленого ПЗ, набуття|надбання| навичок|навичок| керування багатопроцесорними системами та ін.

У сучасних умовах кластерні системи конструюються|сконструйовують| з|із| обчислювальних вузлів на базі стандартних процесорів, з'єднаних високошвидкісною системною мережею (інтерконектом|), а також, як правило, допоміжною і сервісною мережами. Проте|однак|, останнім часом лідери-виробники апаратних засобів комп'ютерної техніки пропонують свій формфактор, зокрема , компанії *IBM|*, *Verari|*, *LinuxNetworx|* та ін. мають у своєму розпорядженні кластерні рішення|вирішення,розв'язання,розв'язування|, побудовані|спорудити| на базі лезових| технологій (так званих блейд-технологій|). Дійсно, формуючи кластер на базі | серверів-лез, користувач фактично отримує|одержує| готове шасі, оснащене необхідним інтерконектом| і засобами керування. Досить розглянути|розглядувати| розробки|вирішення,розв'язання,розв'язування| від компаній *HP|* *BladeSystem|* *c7000|*, *IBM|* *BladeCenter|*, *SUN|* *Blade|* *6000*, *Dell|* *PoweEdge1955|*, щоб|аби| виявити цілий ряд|низку| можливостей для побудови|шикування| кластерів на їх основі. Блейд -системи компактні й зручні в обслуговуванні, їхнє конструктивне рішення|вирішення,розв'язання,розв'язування| дозволяє зручно формувати необхідну конфігурацію, проте|однак| воно трохи дорожче в реалізації порівняно з традиційним підходом.

Розглянемо|розглядуватимемо| деякі особливості такої реалізації кластерних систем. Кожен вузол працює під керівництвом своєї копії стандартної операційної системи, в більшості випадків це *Linux|*. Склад і потужність вузлів можуть бути різними в рамках|у рамках| одного класте-

ра, проте|однак| частіше будуються однорідні кластери. Вибір конкретного комунікаційного середовища|середовища| (інтерконекту|) зумовлюється багатьма чинниками|факторами|: особливостями розв'язуваних задач|задач|, доступним фінансуванням та ін.

На рис. 1 представлена блок -схема персонального обчислювального кластера. Він містить один майстер -вузол ($PM000$) і п'ять обчислювальних *slave*-вузлів ($PN001$, $PN002$, $PN003$, $PN004$, $PN005$) , три керовані комутатори ($SW1$, $SW2$, $SW3$), проміжні буфери пам'яті комутаторів, реконфігуровану мережу для обміну даних між обчислювальними вузлами, віртуальні локальні мережі ($VS123$, $VS23$, $VS012$, $VS022$, $VS032$, $VS042$, $VS052$, $VS013$, $VS023$, $VS033$, $VS043$, $VS053$), механізм резервування ключових компонентів, а також передбачає мережене завантаження вузлів. У майстер_вузлі та *slave*-вузлах застосовуються одні й ті самі комплектуючі (материнські плати, процесори, інтегровані мережеві плати *Fast Ethernet*, зовнішні мережеві плати *Gigabit Ethernet*). Зокрема, майстер -вузли обладнано додатково жорсткими дисками (*HDD*), *CD/DVD,* дисководами (*FDD*).

Особливості функціонування персонального обчислювального кластера полягають у наступному. Після подачі енергоживлення на блок $ATX1$ і надходження зовнішнього сигналу *PUSK* з панелі П00 починається запуск та ініціалізація роботи майстер -вузла модуля багатопроцесорної системи. Безпосередньо завантаження ОС може здійснюватися або з жорсткого диска, або з *CD/DVD*-пристрою. Після завантаження операційної системи запускається конфігураційний скрипт, який налаштовує роботу *DHCP*-сервера, крім того, тут визначається кількість обчислювальних вузлів системи, в разі потреби налаштовується доступ у середовище Інтернет або зовнішню мережу, при цьому зазначаються основні налаштування й параметри. Оскільки активізація режиму *Power on After Power Fail / Former-Sts* і подача живлення на відповідний блок ($ATX2$) запускає всі обчислювальні *slave*-вузли, то й завантаження операційної системи обчислювальних вузлів відбувається групами, без перевантаження першої мережі. Після завантаження всіх обчислювальних вузлів кластера завершується робота скрипту і кластер готовий до виконання паралельних обчислень.

Майстер_вузол ($PM00$) через комутатор $SW1$ забезпечує спрямування потоку даних керування, діагностики та прийому / передачі умов вирішуваних завдань до *slave*- вузлів. *Slave*-вузли відповідно до алгоритму вирішуваного завдання і виконуваних процесів реалізують режим необхідних обчислень. Обмін даними між обчислювальними вузлами винесено до окремої мережі, організованої за допомогою керованих комутаторів $SW2$ – $SW3$. Для досягнення максимальної ефективності

кластерної системи здійснюється процес реконфігурації структури другої мережі залежно від класу вирішуваних завдань. Прийом / передача даних у *slave*-вузлах відбувається за допомогою проміжних буферів пам'яті. Результати обчислень через першу комунікаційну мережу, за допомогою керованого комутатора $SW1$, передаються майстер -вузлу ($PM000$), де здійснюється необхідний режим видачі й обробки даних.

В якості конструктива обрано єдиний корпус, що являє собою осередок обчислювальної шафи. Це пов'язане з тим, що, з одного боку, при необхідності можна декілька ПОК розміщати в єдиному корпусі, а з іншого –при такому підході забезпечується компактність, успішне охолодження й легкий доступ до гнізд і елементів плат, які налагоджуються. ПОК включає вертикальне, паралельне одне стосовно другого, розташування системних плат, що відповідає ідеї «Blade» - серверів.

Персональний обчислювальний кластер має такі розміри: ширина 19', висота 10,9', глибина 9'. Вага пристрою приблизно 7 кг.

Після завантаження операційної системи доступ до ПОК можна отримати за стандартними мережними протоколми (*telnet, ssh, rsh*), як до звичайного ПК. Дякуючи чому для організації суперкомп'ютера на основі робочого ПК і ПОК необхідний лише мережний зв'язок між ними, який може бути організований як через локальну LAN так і через глобальну мережу internet.

Відзначимо деякі особливості зазначеного персонального обчислювального кластера (рис. 1), які дозволяють виділити його серед аналогічних багатопроцесорних систем:

по-перше, висока швидкодія обчислень під час вирішення сильнозв'язаних завдань, забезпечення високошвидкісного доступу до пам'яті вузлів кластера й обмін даними між ними, знизивши завантаження каналу, що проходить між вузлами кластера, за рахунок формування окремої обчислювальної мережі й реалізації механізмів *channel bonding* і *VLAN*;

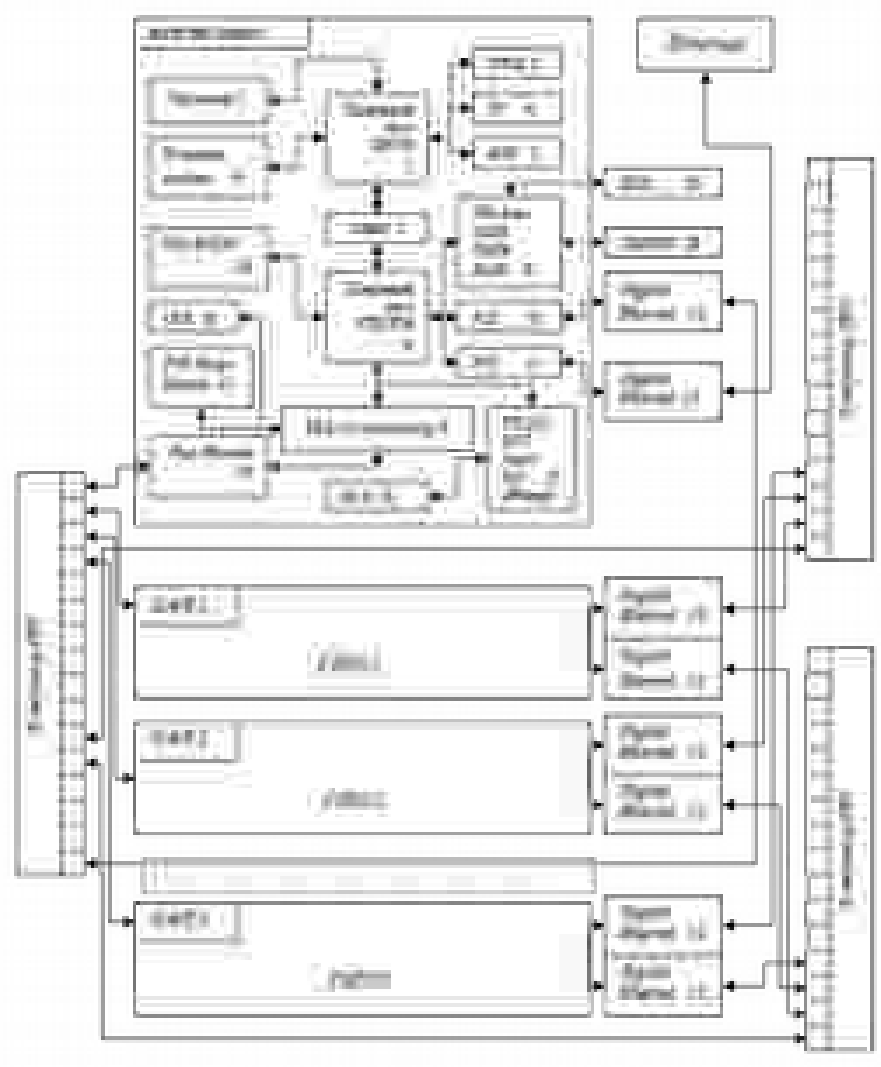


Рис. 1. Блок_схема персонального обчислювального кластера

по_друге, за рахунок проміжного буфера пам'яті з'явилася можливість «розвантажити» центральний процесор CPU в моменти передачі та прий-

ому пакетів між вузлами кластера, що зумовлює підвищення продуктивності обчислювальної системи в цілому;

по_третє, висока ефективність кластерної системи, яка досягається за рахунок адаптації структуру її мережі до вирішення кожного конкретного типу завдань;

по_четверте, за рахунок модульного принципу побудови спростити проектування, нарощування або заміну кластерних вузлів, що вийшли з ладу, а також роботу й експлуатацію системи в цілому;

по_п'яте, за рахунок мережевого завантаження модуля багатопроцесорної системи підвищеної готовності, а також введення механізму резервування ключових компонентів модуля істотно зменшити кількість її компонентів і тим самим значно підвищити надійність функціонування;

по_шосте, забезпечення здатності до перенесення програмного забезпечення на інші кластерні системи з метою виконання подібних розрахунків.

Особливості реалізації паралельних обчислень методом Монте_Карло. |підрахунків|Значний обсяг програмного|програмового| забезпечення, розробленого раніше для послідовної (однопроцесорної) обчислювальної техніки, не має необхідного запасу паралелізму, і спроби його розпаралелювання на рівні окремих циклів, як правило, не мають добрих результатів. Продуктивність коду залишається низькою. Серед різноманіття числових методів у математичних розрахунках останнім часом окреме місце посідає|позичає| якраз метод Монте_Карло. Проте як було раніше|однак| зазначено|помітимо| застосування|застосування| цього методу вимагає великих обчислювальних потужностей, що до певного часу стримувало процес його використання|застосування|. Зауважимо|помітимо|, що за допомогою методу Монте_Карло можна безпосередньо знайти наближений розв'язок задачі в єдиній фіксованій точці|точці| , не знаючи розв'язку задачі для основних точок сіткової області. Цією обставиною метод Монте_Карло, наприклад, у застосуванні до задачі|задачі| Діріхле, різко відрізняється від звичайних|звичних| стандартних способів розв'язку.

Спрощену схему обчислень методом Монте_Карло зображено на рис. 2.

Рис. 2. Схема обчислень|підрахунків| методом Монте_Карло

Застосування|застосування| цього методу дає можливість|спроможність| по-новому розглянути ідею розпаралелювання обчислень|підрахунків| і застосування|застосування| кластерних технологій обчислення|підрахунку|. Справді, проміжні розрахунки у методі Монте-Карло можуть здійснюватися незалежно на різних, окремо взятих, лезах кластера – «обчислювачах», а результати оброблятися на якомусь окремо взятому *master*-лезі – «аналізаторі» [18]. У зв'язку з цим для даної кластерної системи алгоритм паралельних обчислень можна уявити у вигляді схеми, поданої на рис. 3.

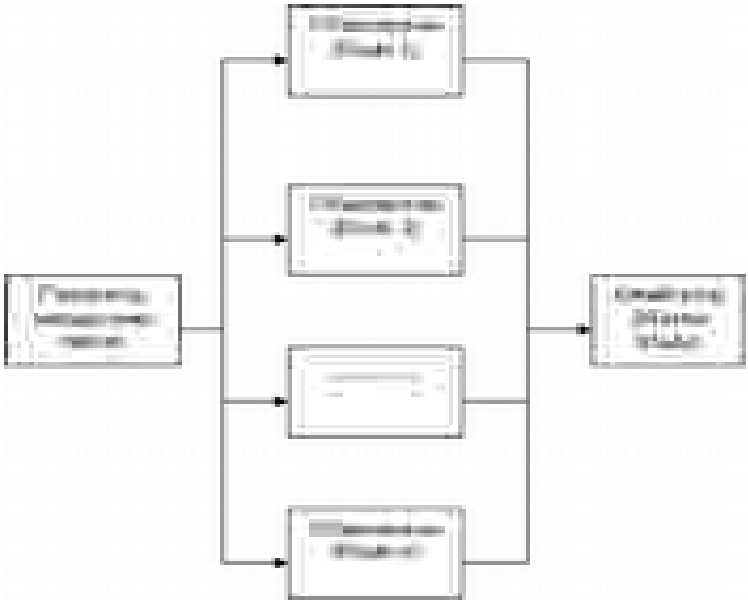


Рис. 3. Алгоритм паралельних обчислень|підрахунків| методом Монте-Карло

Згідно із такою схемою, один генератор випадкових чисел видає кожному «обчислювачу» випадкове число, яке вже було згенеровано. Така схема має істотний недолік, що знижує продуктивність паралельної кластерної системи. Інформація постійно передається комутаційними каналами, які мають відповідну латентність. Зазначені обставини і будуть знижувати продуктивність та швидкодію системи в цілому. Звичайно, ефективність такої системи залежатиме і від якості генератора випадкових чисел, оскільки існує досить висока ймовірність того, що серед послідовності випадкових чисел виникатимуть повторення або досить

близькі за величиною значення. Хоча можна зауважити, що зазначені обставини істотно не вплинуть на допустиму похибку обчислень. Досвід експлуатації персонального обчислювального кластера дозволив удосконалити схему обчислень, зображену на рис. 3.

На рис. 4 подано|уявляти| модифікований алгоритм обчислень|підрахунків| методом Монте ‗Карло. Тут кожен обчислювач має власний генератор випадкових чисел. Це дозволяє уникнути неодмінної присутності між генератором випадкових чисел і «обчислювачем» маршрутизатора‗комунікатора|. Очевидн|вочевидь|о, що таке рішення|вирішення| дозволяє прискорити процес обчислень|підрахунків| . Виграш у продуктивності можна оцінити експериментальним шляхом|колією|.

Таким чином, алгоритми Монте ‗Карло є стійкими по відношенню до будь‗яких вхідних даних, мають максимальну паралельну форму і, отже, мінімальний можливий час реалізації на паралельних обчислювальних пристроях|устроях|. Якщо можна призначити один процесор на один вузол розрахунку, то стає можливим проведення розрахунків у всіх вузлах сіткової області паралельно й одночасно.

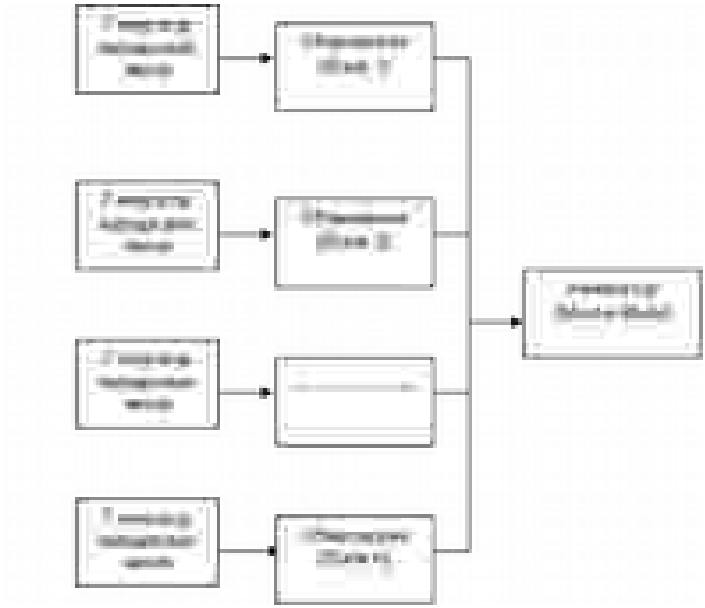


Рис. 4. Модифікований алгоритм паралельних обчислень|підрахунків| методом Монте_Карло

Аналіз проблеми блукань і розв'язування крайових|крайових| задач. Крайові|крайові| задачі|задачі| й задачі|задачі| з|із| початковими умовами для лінійних диференціальних рівнянь є|з'являються| однією з найцікавіших сфер застосування|застосування| методу Монте_Карло. Зв'язок між цими задачами|задач| був відомий давно [5 – 9]. Проте|однак| можливість|спроможність| застосування|застосування| цього зв'язку для фактичного відшукання|відшукування| результатів|вирішень| з'явилася|появлялася| тільки завдяки розвитку обчислювальних машин. Застосування|застосування| цього методу дає можливість|спроможність| по_новому сприйняти ідею розпаралелювання обчислень|підрахунків| і застосування|застосування| кластерних технологій обчислення|підрахунку|.

Щоб пояснити основну ідею методу, розглянемо задачу Діріхле для рівняння Лапласа. Нехай маємо певну однозв'язну область G, на межі якої задано функцію $f(Q)$. Потрібно знайти таку функцію $U(P)$, яка всередині даної області G задовольняє рівняння Лапласа, тобто

$$\Delta U = 0, \tag{1}$$

а на межі області P набуває таких заданих значень:

$$U_{|\Gamma} = f(Q). \tag{2}$$

Зазвичай цю задачу зводять до певної кінцево _різницевої схеми. Область G вкривається вузлами квадратної сітки. У внутрішніх вузлах шукають значення функції $U(P)$, виходячи з такої системи рівнянь

$$U(P) = \frac{1}{4}[U(P_1) + U(P_2) + U(P_3) + U(P_4)]. \tag{3}$$

Тут символами $\{P_1, P_2, P_3, P_4\}$ позначають чотири вузли, сусідні з внутрішнім вузлом P, вони лежать в області G або на її межі.

Розглянемо пов'язану з цією системою теоретико _ймовірнісну схему. Уявімо собі частинку, яка змушена пересуватися по вузлах сітки з такими цілочисловими координатами (i, j) на площині:

$$\left.\begin{array}{l} x_i = x_0 + i\eta,\ y_j = y_0 + j\eta \\ (i, j = 0, \pm 1, \pm 2, \cdots,) \end{array}\right\},$$

при цьому

$$\Delta x_i = x_{i+1} - x_i,\ \Delta y_j = y_{j+1} - y_j.$$

Припустимо, що сітка S_η складається з внутрішніх і граничних вузлів, у яких задано граничні умови першого роду. Граничні вузли являють собою сукупність лінійних точок $M_{pq}(x_p,y_q)$, яка апроксимує криволінійну межу Γ області G з точністю до η. Частинка M здійснює рівномірне випадкове блукання серед вузлів сітки (3). Зокрема, перебуваючи у внутрішньому вузлі $M_{i0,j0}$ сітки S_η, ця частинка за один перехід, з однією і тією самою ймовірністю, яка дорівнює ¼, може переміститися в один з сусідніх вузлів, зокрема, або в $M_{i-1,j}(x_{i-\eta},y_j)$, це буде крок уліво, або в $M_{i+1,j}(x_{i+\eta}, y_j)$ – крок управо, або в $M_{i,j-1}(x_i, y_{j-\eta})$ – крок униз, або $M_{i,j+1}(x_i, y_{j+\eta})$ – крок угору. Причому, кожен такий одиничний перехід є абсолютно випадковим і не залежить від положення частинки та її попередніх переміщень (історії блукань). Вважатимемо, що блукання частинки M закінчується, як тільки вона потрапляє на межу Γ_η; в цьому випадку межа Γ_η є «поглинальним екраном». Можна довести [5], що блукання точки M через скінченне число кроків закінчується саме на цій межі.

Якщо частинка M почала своє блукання з фіксованої внутрішньої точки $M_{i0,j0}$ на сітці S_η, то скінченну сукупність її послідовних положень можна записати таким чином:

$$M_{i0,j0}, M_{i1,j1}, \dots, M_{iS,jS},$$

причому

$$M_{i_k j_k} \in \Gamma\eta \ (k = 0,1,\cdots,S-1).$$

Тут вираз $M_{i_k j_k} \in \Gamma\eta$ відображає траєкторію частинки (з кількістю кроків S), яку прийнято називати історією блукання.

Рівномірне випадкове блукання частинки|частинки| можна організувати за допомогою рівномірно розподіленої послідовності однорозрядних випадкових чисел [5; 7 – 10], що набувають таких значень:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Для цього, наприклад, досить проводити|виробляти| розіграш, тобто випадкову вибірку з|із| чисел {0 – 9}, дотримуючись інструкції, поданої в табл. 1.

Таблиця 1

Визначення кроку частинки|частинки| залежно від випадкового числа

Випадкове число	Характер переміщення
0 або 4	$\Delta x_i = \eta$ (крок управо)
1 або 5	$\Delta Y_Y = \eta$ (крок угору)
2 або 6	$\Delta\Delta x_i = -\eta$ (крок уліво)

3 або 7	$\Delta\Delta Y_Y = -\eta$ (крок униз)

Випадкові числа беруться з готових таблиць або визначаються генератором псевдовипадкових чисел [7]. Останній спосіб набув широкого застосування під час роботи на ПОК, оскільки він не дозволяє занадто завантажувати пам'ять системи. Частинка, що почала своє випадкове блукання з внутрішнього вузла $M_{i0,j0}$, після першого кроку з імовірністю, яка дорівнює ¼, потрапляє в один з чотирьох сусідніх вузлів, а саме:

I. $M_{i,j}$, $M_{i-1,j}$, … , ;
II. $M_{i,j}$, $M_{i+1,j}$, … , ;
III. $M_{i,j}$, $M_{i,j-1}$, … , ;
IV. $M_{i,j}$, $M_{i,j+1}$, … .

За формулою повної|цілковитої| ймовірності|ймовірності| встановлюємо, що:

$$P(i,j,p,q)=\frac{1}{4}P(i-1,j,p,q)+\frac{1}{4}P(i+1,j,p,q)+ \\ +\frac{1}{4}P(i,j-1,p,q)+\frac{1}{4}P(i,j+1,p,q). \qquad (4)$$

Звідси, перемножуючи обидві частини рівності (4) на граничні значення γ_{pq} і підсумовуючи за всіма можливими значеннями p і q, одержимо, що

$$\vartheta_{ij}=\frac{1}{4}(\vartheta_{i-1,j}+\vartheta_{i+1,j}+\vartheta_{i,j-1}+\vartheta_{i,j+1}). \qquad (5)$$

Величини ϑ_{ij} допускають експериментальне визначення, для цього треба замінити математичне очікування ϑ_{ij} емпіричним математичним очікуванням, тоді відповідний вираз набуде такого вигляду:

$$U_{ij}=\frac{1}{N}\sum_{k=1}^{w}\varphi(x_p^{(k)}y_q^{(k)}). \qquad (6)$$

Формула (6) дає статистичну оцінку величини $U_{i,j}$ і може бути застосована до наближеного розв'язку задачі Діріхле. Проілюструємо розроблений алгоритм, розв'язуючи конкретні приклади.

Обчислювальні експерименти. Обчислювальні експерименти виконувалися на основі застосування персонального обчислювального кластера, блок_схема якого наведена на рис. 1, при цьому методика відповідних

розрахунків проводилася з урахуванням розробленого підходу. Розглянемо характерні приклади обчислень.

Приклад 1. Методом Монте_Карло знайти величину $U(2,2)$, якщо

$$\Delta U(x,y)=0, \text{ в області } G\{0\le x\le 4;\ 0\le y\le 4\}, \tag{7}$$

а

$$\left.\begin{aligned} U(x,0)&=0, \quad 0\le x\le 4;\\ U(4,y)&=y, \quad 0\le y\le 4;\\ U(x,4)&=x, \quad 0\le x\le 4;\\ U(0,y)&=0, \quad 0\le y\le 4. \end{aligned}\right\} \tag{8}$$

Розв'язок. Для квадрата G з межею Γ побудуємо квадратну сітку S, в якій крок $\eta = 1$. Далі розглядаємо серію рівномірних блукань частинки серед вузлів сітки S_η, виходячи з початкового положення (2,2). Блукання закінчуються на межі Γ, у граничних вузлах сітки $G_к$, заданих умовами (8). Переміщення частинки відбувалося відповідно до описаної вище інструкції (табл. 2), причому появу випадкових чисел 8 і 9 вважаємо стоянням частинки на місці.

У табл. 2 наведено траєкторії для 20 історій двовимірного випадкового блукання, коли число $N = 20$.

На підставі формули (6) можемо встановити, що:

$$U(2,2)=\frac{1}{20}\sum_k \varphi(x_p^{(k)} y_q^{(k)}=\frac{1}{20}\cdot 20=1.$$

Зауважимо|помітимо|, що в даному випадку відомий точний розв'язок задачі|задачі| Діріхле (7), (8), а саме:|вирішення|

$$U(x,y)=\frac{xy}{4}.$$

Тому

$$U(2,2)=\frac{2\cdot 2}{4}=1.$$

Таким чином, за допомогою методу статистичних випробувань одержали точне значення величини $U(2,2)$.

Приклад 2. Розглянемо задачу про температурне поле вугільної адіабатичної пластини [19]. Температурне поле $T(x,y)$ має задовольняти таке рівняння:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0, \tag{9}$$

а також систему довільно вибраних граничних значень температури:

$$T = \begin{cases} F_1(x) \text{ уздовж границі } y = \ell, \\ F_2(x) \text{ уздовж границі } y = 0, \\ G_1(y) \text{ уздовж границі } x = L, \\ G_2(y) \text{ уздовж границі } x = 0, \end{cases} \tag{10}$$

Потрібно визначити температурне поле пластини, якщо $F_1(x) = F_2(x) = 1$ і $G_1(y) = G_2(y) = 0$, а геометричні розміри пластини $l = 1$ і $L = 2l$.

Таблиця 2

Траєкторії блукання робочої точки

|точки|

№ блукан_ня, k	Траєкторія блукання	Значення функції $u(x,y)$ у точці виходу на межу області G
1	(2,2) > (2,3) > (2,2) > (2,1) > > (3,1) > (3,2) > (3,1) > (3,2) > > (2,2) > (2,3) > (2,3) > (2,2) > > (2,1) > (2,0);	0
2	(2,2) > (2,3) > (3,3) > (3,2) > (4,2)	2
3	(2,2) > (2,3) > (2,2) > (2,3) > (2,4)	2
4	(2,2) > (1,2) > (1,2) > (0,2);	0
5	(2,2) > (2,3) > (2,4);	2
6	(2,2) > (2,1) > (2,0);	0
7	(2,2) > (1,2) > (2,2) > (3,2) > (3,1) > (3,2) > (2,2) > (1,2) > (0,2);	0
8	(2,2) > (1,2) > (0,2);	0
9	(2,2) > (2,1) > (2,2) > (3,2) > (3,3) > (3,3) > (2,3) > (1,3) > (0,3);	0
10	(2,2) > (1,2) > (0,2);	0
11	(2,2) > (2,2) > (2,2) > (2,1) > > (2,2) > (3,2) > (3,1) > (3,1) > (4,1)	1

12	(2,2) > (2,2) > (2,1) > (2,1) > (4,1);	0
13	(2,2) > (2,1) > (3,1) > (3,0) ;	0
14	(2,2) > (3,2) > (4,2);	2
15	(2,2) > (2,3) > (2,4);	2
16	(2,2) > (2,3) > (2,3) > (1,3) > (0,3);	0
17	(2,2) > (3,2) > (4,2);	2
18	(2,2) > (3,2) > (3,1) > (2,1) > > (2,2) > (3,2) > (4,2);	3
19	(2,2) > (3,2) > (4,2);	2
20	(2,2) > (2,3) > (2,3) > (2,3) > (2,4);	2
	Σ	20

Виконаємо порівняльний аналіз розв'язків поставленої задачі числово - аналітичним методом і методом статистичних випробувань.

Отже, прямокутна область заданого розміру вкривається сітковими вузлами таким чином:

$$x_i = x_0 + i\eta,\ y_j = y_0 + j\eta\ , \qquad (11)$$

при цьому

$$\left.\begin{aligned} Dx1 = x_{i+1} - x_i = \frac{2}{2m_x},\ m_x = 10 \\ Dy1 = y_{j+1} - y_j = \frac{2}{2m_y},\ m_y = 5 \end{aligned}\right\},\ \eta = 0{,}1.$$

Доцільність такого нормування вузлів приводить до їх рахунковості через цілочислові значення параметрів m_x і m_y. Маємо таку потужність рахунковості:

$$\overline{i = 1,2m_x - 1},\quad \overline{j = 1,2m_y - 1}.$$

Граничні поверхні виражаються|виказують| параметрично:

$$x_0 = 0,\ \ y_0 = 0,\ \ x_{2m_x} = 2,\ \ y_{2m_y} = 1,$$

при цьому, $x_{m_x}\ y_{m_y}$ і (x_{m_x}, y_{m_y}) відповідають серединним поверхням і центральному вузлу. Замість незалежних змінних x та y введемо нормовані одиницею, тобто

$$\left.\begin{aligned}\varepsilon_x &= \frac{x - x_i}{x_{i+1} - x_i} \in [-1, +1],\\ \varepsilon_y &= \frac{y - y_j}{y_{j+1} - y_j} \in [-1, +1].\end{aligned}\right\} \tag{12}$$

Тоді рівняння (9) для рівномірно розподілених вузлів ($x_{i+1} - x_i = x_i - x_{i-1}$, $y_{j+1} - y_j = y_j - y_{j-1}$) виявиться інваріантним відносно сіткових вузлів, а саме:

$$\frac{\partial^2 T(\varepsilon_x, \varepsilon_y)}{\partial \varepsilon_x^2} + \frac{\partial^2 T(\varepsilon_x, \varepsilon_y)}{\partial \varepsilon_y^2} = 0. \tag{13}$$

Подібний підхід допускає можливість|спроможність| будувати|шикуванні| алгоритми з урахуванням|з урахуванням| апріорної інформації про шуканий розв'язок|вирішення|. Припустимо, що шуканий розв'язок рівняння (9) належить до класу аналітичних функцій. Отже, стає можливим існування шуканої функції у вигляді рядів Тейлора:|вирішення|

$$T_{j+\varepsilon_y,1}(x, y) = \sum_{n=0}^{\infty} \varepsilon_y^n T_{j,n+1}(\varepsilon_x) \tag{14}$$

або

$$T_{i+\varepsilon_x,1}(x, y) = \sum_{n=0}^{\infty} \varepsilon_x^n T_{i,n+1}(\varepsilon_y). \tag{15}$$

Після|потім| підстановки (14), (15) у рівняння (13) одержимо|одержуватимемо| сукупність диференціальних рівнянь такого|слідуючого| вигляду|виду|:

$$(n+1)(n+2) T_{j,n+3}(\varepsilon_x) = -T''_{j,n+1}(\varepsilon_x) \tag{16}$$

або

$$(n+1)(n+2) T_{i,n+3}(\varepsilon_x) = -T''_{i,n+1}(\varepsilon_y). \tag{17}$$

Легко помітити, що тейлорівські | компоненти (16), (17) можуть бути виражен|виказувати|і через дані Коші $\{T_{j,1}(\varepsilon_x), T_{j,2}(\varepsilon_x)\}$, $\{T_{i,1}(\varepsilon_y), T_{i,2}(\varepsilon_y)\}$ та їх похідні за незалежними змінними ξ_x, ξ_y. Тоді для різних значень n (наприклад: 0, 1, 2, 3) відповідні рівняння можна записати таким чином:

$$\left.\begin{aligned} T_{J,3}(\varepsilon_x) &= -\frac{1}{2!}T''_{j,1}(\varepsilon_x), \\ T_{J,4}(\varepsilon_x) &= -\frac{1}{3!}T''_{j,2}(\varepsilon_x), \\ T_{J,5}(\varepsilon_x) &= +\frac{1}{4!}T^{(4)}_{j,1}(\varepsilon_x), \\ T_{J,6}(\varepsilon_x) &= +\frac{1}{5!}T^{(4)}_{j,2}(\varepsilon_x), \end{aligned}\right\} \qquad (18)$$

і т. д.

Отже, замість ряду|низки| Тейлора (14) ми маємо локальний розв'язок задачі Коші відповідно до вузлів $(j=\overline{1,2m_y-1})$, а саме:

$$T_{y+\varepsilon_{y,1}}(x,y)=\sum_{n=0}^{\infty}(-1)^n\left[\frac{\varepsilon_y^{2n}}{(2n)!}T^{(2n)}_{y,1}(\varepsilon_x)+\frac{\varepsilon_y^{2n+1}}{(2n+1)!}T^{(2n)}_{y,2}(\varepsilon_x)\right], \qquad (19)$$

при цьому дані Коші $\{T_{J,1}(\varepsilon_x), T_{J,2}(\varepsilon_x)\}$ являють собою невідомі функції змінної ε_x.

Довизначаючи розв'язок (19) граничними умовами Діріхле, і вважаючи, що $\varepsilon_x=\pm 1$, отримаємо|одержуватимемо| розділений розв'язок задачі Коші в такому|слідуючого| вигляді|виду|:

$$\sum_{n=0}^{\infty}(-1)^n\frac{1}{(2n)!}T^{(2n)}_{J,1}(\varepsilon_x)=\frac{1}{2}\left[T_{j+1,1}(\varepsilon_x)+T_{j-1,1}(\varepsilon_x)\right], \qquad (20)$$

або

$$\sum_{n=0}^{\infty}(-1)^n\frac{1}{(2n+1)!}T^{(2n)}_{j,2}(\varepsilon_x)=\frac{1}{2}\left[T_{j+1,1}(\varepsilon_x)-T_{j-1,1}(\varepsilon_x)\right], \qquad (21)$$

при цьому $j=\overline{1,2m_y-1}$, а $T_{0,1}(\varepsilon_x), T_{2m_y,1}(\varepsilon_x)$ – відомі граничні функції першого роду.

Диференціюючи (19) за ε_y і розділяючи дані Коші, одержимо, що|одержуватимемо|:

$$\sum_{n=0}^{\infty}(-1)^n\frac{1}{(2n)!}T^{(2n)}_{j,1}(\varepsilon_x)=\frac{1}{2}\left[T_{j+1,2}(\varepsilon_x)+T_{j-1,2}(\varepsilon_x)\right] \qquad (22)$$

або

$$\sum_{n=0}^{\infty}(-1)^n \frac{1}{(2n-1)!} T_{j,1}^{(2n)}(\varepsilon_x) = \frac{1}{2}\left[T_{j+1,2}(\varepsilon_x) - T_{j-1,2}(\varepsilon_x)\right], \quad (23)$$

де $\{T_{0,2}(\varepsilon_x), T_{2m_y,1}(\varepsilon_x)\}$ – відомі граничні функції другого роду.

Таким чином, редукуючи нескінченні ряди (20) – (23) з урахуванням, що n прямує до ($n = M \in Z$), ми одержуємо частинні математичні моделі з довільним порядком точності. При цьому, якщо $M = 1$, то маємо такі звичайні кінцево-різницеві схеми:

– для задачі Діріхле:

$$T_1(i,j) = \frac{1}{4}[T_1(i-1,j) + T_1(i+1,j) + T_1(i,j-1) + T_1(i,j+1)]; \quad (24)$$

– для задачі|задачі| Неймана:

$$T_2(i,j) = \frac{1}{4}[T_2(i-1,j) + T_2(i+1,j) + T_2(i,j-1) + T_2(i,j+1)]. \quad (25)$$

Отримані співвідношення (24) та (25) можна застосовувати і для методу статистичних випробувань. Так, просте випадкове блукання частинки прямокутною сіткою легко поширюється і на задачі Неймана, причому $T_2(i,j)$ у (25) являє собою градієнту функцію. Можливі також інші модифікації процесу вибірки. Припустимо, наприклад, що вирішено змінити ймовірності переходу частинки в певній точці. Ця зміна може бути уточнена, якщо прийняти в ролі вагових коефіцієнтів схему p_α / r_α, за якою p_α являє собою гіпотетичну ймовірність одержання заданого результату на будь-якій стадії, а використаний процес вибірки реалізує цей результат з імовірністю r_α. Подібну процедуру «блукання» частинки можна застосовувати на кожному етапі випадкового процесу за умови, що обчислюються добутки відповідних вагових коефіцієнтів до кінця кожної історії.

Вельми|дуже| неприємною рисою немодифікованого випадкового блукання, на якому базувався попередній розгляд, є повільна збіжність. Із|із| самої природи випадкового процесу випливає, що потрібно зробити велику кількість кроків, щоб|аби| кудись дійти. За таких обставин зазвичай|звично| доцільно розглянути|розглядувати| використання спеціальних методів вибірки стосовно окремих груп [7].

Для розв'язання задачі|задачі| (9), (10) граничні умови (10) конкретизуємо такими|слідуючими| вихідними|початковими| даними:

$$T[j,0] = T[j,2m_x] = 0,$$
$$T[0,i] = T[2m_x,i] = 1, \qquad (26)$$

при цьому $j = \overline{1,2m_y - 1}, i = \overline{1,2m_x - 1}$. Переміщення частинки визначалося відповідно до поданої вище ілюстрації (табл. 2). Для організації випадкових блукань використовуємо рівномірно розподілену послідовність випадкових чисел, підібрану за допомогою ПОК. Результати розрахунків у колонках $[j,1]$, $[j,m_x]$ при різних значеннях величини N оброблялися у вигляді відносної похибки, що має такий вигляд:

$$\beta[j,i] = \frac{|T_t[j,i] - T_p[j,i]|}{T_t[j,i]} \cdot 100\%, \qquad (27)$$

при цьому $T_t[j,i]$ відповідає точному розв'язку, $T_p[j,i]$ визначають за схемою методу Монте_Карло. Результати розрахунків зведено в табл. 3.

Із порівняльного аналізу результатів моделювання випливає, що зі збільшенням числа N відносна похибка зменшується. У прикутових вузлах (1, 1), (m_j,1) похибка не зменшується через близькість граничних вузлів (0, 0) і (m_j, 0), де функція має розрив першого роду відповідно до умови (26).

Таблиця 3

Відносна похибка результатів розв'язку задач методом Монте_Карло при різних значеннях блукань робочої частинки N

Номер блукання	*N* = 1 000		*N* = 10 000		Аналітичний розв'язок
	β, %	$\overline{m}_x$	β, %	$\overline{m}_x$	*T(x,y)*
(1,1)	6,332	0,5273	2,359	0,5076	0,4959
(2,1)	1,314	0,5938	0,614	0,5897	0,5861
(3,1)	1,121	0,6858	0,236	0,6798	0,6782
(4,1)	0,967	0,7725	0,209	0,7667	0,7651
(5,1)	0,913	0,8948	0,101	0,8876	0,8867
(6,1)	0,967	0,7725	0,209	0,7667	0,7651
(7,1)	1,121	0,6858	0,236	0,6798	0,6782

(8.1)	1.314	0.5938	0.614	0.5897	0.5861
(9.1)	6.332	0.5273	2.359	0.5076	0.4959

Висновки. У наведеній статті набув подальшого розвитку процес математичного моделювання прикладних задач|задач| на основі використання персонального обчислювального кластера. |задач| Досвід|дослід| експлуатації перших паралельних систем показав, що для їх ефективної роботи|застосування| потрібно радикально змінювати|зраджувати| структуру числових методів. У зв'язку з цим розроблено відповідні розподілені алгоритми, виявлено і показано особливості моделювання прикладних задач|задач| на основі застосування|застосування| багатопроцесорних систем.

Останнім часом можна говорити про відродження методу Монте-Карло. Це пояснюється|тлумачить| тим, що цей метод ідеально підходить|пасує| до кластерної системи. Причому, чим більше процесорів буде в кластері, тим ефективніше буде розв’язуватися|розв'язуватиметься| задача|задача|. Метод Монте-Карло справив|виявляв| і продовжує справляти істотний|суттєвий| вплив на розвиток методів обчислювальної математики. Його застосування|застосування| виправдане, в першу чергу, до розв’язку таких задач|задачах|, які допускають теоретико-ймовірнісний| опис. Це пояснюєтьс|тлумачить|я як природністю одержання|отримання| відповіді |із|з певною заданою ймовірніст|ймовірністю|ю у задача|задачах|х |із|з імовірнісним змісто|змістом|м, так і суттєви|суттєвим|м спрощенням процедури розв’язку|вирішення|. За таких обставин у даній статті|розділі| висвітлено кластерні проблеми застосування методу Монте-Карло до розв’язування багатовимірни|багатомірних|х зада|задач|ч.

Повільна збіжність методу є його недоліком|нестачею|. Проте|однак| в проведених дослідженнях показано, що формування вибіркових випадкових чисел стосовно окремих груп дозволяє |суттєвий| істотно підвищити точність методу.

Крім того, показано, що метод Монте-Карло досить|достатньо| вдало|успішний| пристосований до розв’язування багатовимірних|багатомірних| задач|задач|. Наприклад, застосовуючи звичайний|звичному| метод розв’язку систем лінійних алгебраїчних рівнянь для обчислення|підрахунку| одного невідомого, треба визначити також і решту. У даному способі цього робити|чинити| не потрібно, кожного разу визначається тільки одна необхідна координата розв’язуваної|рішати| системи.

Крайові|крайові| задачі|задачі| та задачі|задачі| з|із| початковими умовами для лінійних диференціальних рівнянь є|з'являються| однією з найціка-

віших сфер застосування|застосування| методу Монте -Карло, а саме це стало можливим|появлялася| тільки завдяки розвитку кластерних обчислювальних систем. З цих причин у статті|розділі| наведено приклади розв'язку задач |задач| Неймана та Діріхле за допомогою методу Монте - Карло.

Застосування цього методу дає можливість по -новому подивитись на ідею розпаралелювання обчислень та використання кластерних технологій обчислення. У статті запропоновано модифікований алгоритм паралельних обчислень за методом Монте -Карло. Тут кожен обчислювач має власний генератор випадкових чисел. При цьому проміжні розрахунки здійснюються незалежно на різних, окремо взятих лезах кластера – «обчислювачах», а результати вже обробляються на якомусь окремо взятому *master*-лезі – «аналізаторі». Ця обставина дозволяє позбутися неодмінної присутності маршрутизатора -комунікатора між генератором випадкових чисел та «обчислювачем». Очевидно, що таке рішення дозволяє прискорити процес обчислень.

Показано, що розроблені паралельні алгоритми методу Монте -Карло стійкі для будь -яких вхідних даних, мають максимальну паралельну форму і, отже, мінімальний можливий час реалізації на паралельних обчислювальних пристроях|устроях| . Якщо можна призначити один процесор на один вузол розрахунку, то стає можливим проведення обчислень у всіх вузлах сіткової області паралельно й одночасно.

Бібліографічні посилання

1. **Михайлов Г.А.** Численное статистическое моделирование. Методы Монте - Карло / Г.А. Михайлов, А.В. Войтишек. – М., 2006. – 368 с.

2. **Михайлов Г. А.** Оптимизация весовых методов Монте -Карло по вспомогательным переменным / Г. А. Михайлов, И. Н. Медведев // Сиб. матем. журн. – 2004. – № 45. – С. 399 – 409.

3. Інтернет-ресурс http://ru.wikipedia.org/wiki/метод_Монте-Карло.

4. Інтернет-ресурс http://www.riskglossary.com/link/monte_carlo_method.htm.

5. **Ермаков С.М**. Метод Монте -Карло и смежные вопросы / С.М. Ермаков. – М., 1971. – 471 с.

6. **Соболь И.М**. Метод Монте-Карло / И.М. Соболь. – М., 1968. – 64 с.

7. **Браун Дж**. Методы Монте -Карло / Дж. Браун // Современная математика для инженеров; под ред. Э.Ф. Беккенбаха. – М., 1958. – 500 с.

8. **Китов Н.А**. Электронные цифровые машины и программирование / Н.А. Китов, Н.А. Криницкий. – М., 1959. – 572 с.

9. **Демедович Б.П**. Основы вычислительной математики / Б.П. Демедович, И.А. Марон. – М., 1963. – 660 с.

10. **Березин И.С.** Методы вычислений / И.С. Березин, Н.П. Шибков. – Т. 1. – М., – 1966. – 632 с.

11. **Коздоба Л. А.** Вычислительная теплофизика / Л.А. Коздоба. – К., 1992. – 224 с.

12. **Иващенко В.П**. Параллельные вычисления и прикладные задачи металлургической теплофизики / В.П. Иващенко, Г.Г. Швачич, А.А. Шмукин // Системні технології. Регіональний збірник наукових праць. – Вип. 3(56). – Т. 1. – 2008. – С. 123 – 138.

13. **Швачич Г.Г**. К вопросу конструирования параллельных вычислений при моделировании задач идентификации параметров окружающей среды / Г.Г. Швачич // Математичне моделювання. – 2006. – № 2 (14). – С. 23 – 34.

14. **Швачич Г.Г.** О параллельных компьютерных технологиях кластерного типа решения многомерных нестационарных задач / Г.Г. Швачич // Materiály IV mezinárodní vědecko- praktická konference [«Vědecký potenciál světa - 2007»]. – D. 7. – Technick é vědy. Matematika. Fyzika. Moderní informační technologie. Výstavba a architektura. – Praha: Publishing House «Education and Science» s.r.o. – P. 35 – 43.

15. **Воеводин Вл. В**. Вычислительное дело и кластерные системы / Вл.В. Воеводин, С.А. Жуматий. – М., 2007. – 150 с.

16. **Баканов В.М**. Персональный вычислительный кластер как недостающее звено в технологии проведения сложных технологических расчетов / В.М. Баканов // Метизы. – 2006. – 2 (12). – С. 33 – 36.

17. **Shvachych G.G**. Prospects of construction highly –productive computers systems on the base of standard technologies / G.G. Shvachych // IV Intrenational Conference «Strategy of Quality in Indastry and Education »; May 30 – June 6, 2008, Varna; Bulgaria. – Proceedings. – V. 2. – P. 815 – 819.

18. **Швачич Г. Г.** Некоторые особенности применения метода Монте - Карло для распределенного моделирования многомерных нестационарных задач / Г.Г. Швачич // Сборник научных трудов по материалам международной конференции «Научные основы внедрения новых технологий в эпоху Нового Возрождения». – 2009. – С. 340 – 341.

19. **Шнейдер П**. Инженерные проблемы теплопроводности / П. Шнейдер; пер. с англ. – М., 1960. – 478 с.

Надійшла до редколегії 25.01.11

УДК 519.8

[13]Т.О. Шевченко
Дніпропетровський національний університет імені Олеся Гончара

МЕТОД РОЗВ'ЯЗАННЯ ЗАДАЧІ ОПТИМАЛЬНОГО РОЗБИТТЯ МНОЖИНИ З РУХОМИ ГРАНИЦЯМИ І ЦЕНТРАМИ ПІДМНОЖИН

Розглядається динамічна задача оптимального розбиття деякої місцевості «гірського» типу на підобласті. Одночасно, центри підобластей, рухаючись із початкових положень по поверхні заданої місцевості, повинні дістатись пунктів призначення найкоротшим шляхом. Запропоновано та обґрунтовано метод розв'язання представленної задачі.

Рассматривается динамическая задача оптимального разбиения некоторой местности «горного» типа на подобласти. Одновременно, центры подобластей, двигаясь из начальных положений по поверхности заданной местности, должны добраться в пункты назначения кратчайшим образом. Предложено и обосновано метод решения представленной задачи.

The dynamic problem of a certain rock type area partitioning to the subsets is considered. Simultaneously the centers of subsets moving from initial points by surface of that area must get finisf points by shortest way. The method of cited problem is offered and justified.

Ключові слова: оптимальне розбиття множин (ОРМ), динамічна задача ОРМ, фазова траєкторія, оптимальне керування.

Вступ. Неперервні задачі оптимального розбиття множин, які є неклассичними задачами нескінченновимірного математичного програмування, детально досліджені у монографії [1]. В [1] зазначено, що одним з напрямів розвитку теорії неперервних задач ОРМ, є дослідження динамічних задач. У [2] дається огляд основних напрямків розвитку таких задач. Згідно з [2] є три основні типи урахування динаміки в задачах ОРМ:

1) коли цільовий функціонал задачі залежить від функцій попиту і вартості, які є фазовими траєкторіями деякої заданої керованої системи [3];

2) коли виконується п.1 і швидкість змінення функції попиту в кожній точці множини, що розглядається, залежить від приналежності цієї точки до поточної підмножини, яка залежить від часу [4];
3) коли виконується п.2 і центри підмножин змінюються з часом або за деяким заданим законом, або підлягають керуванню.

У даній роботі наведена постановка задачі, що відтворює варіант у 3), тобто динамічної задачі оптимального розбиття множини на підмножини, границі яких можуть рухатися з часом під впливом деякого керування; центри підмножин також є рухомими і потребують оптимального у певному сенсі переводу із початкових точок у задані. Передбачається, що центри – це проекції точок, що рухаються по заданій поверхні. Запропоновано та обгрунтовано метод розв'язання такої задачі.

Постановка задачі. Нехай задано набір функцій $u_i(x,t) \in L_2(\Omega \times [0,T]), i = \overline{1,N},$ де $\Omega \subset R^2$ – обмежена, вимірна за Лебегом множина. Через $P_N(\Omega)$ позначимо клас всіляких розбиттів множини Ω на N підмножин, що не перетинаються

$$P_N(\Omega) = \left\{ \overline{\omega} = (\Omega_1, ..., \Omega_N) : \bigcup_{i=1}^{N} \Omega_i = \Omega,\ mes(\Omega_i \cap \Omega_j) = 0,\ i \neq j,\ i, j = \overline{1,N} \right\}.$$

Дозволимо границям між підмножинами мінятися з часом і позначимо через $\overline{\omega}(t) = (\Omega_1(t), ..., \Omega_N(t))$ розбиття множини Ω, що відповідає моменту часу t і визначає зміну стану $\rho(x,t)$ деякої системи так, що $\forall x \in \Omega_i(t)$ виконується

$$\begin{aligned} &\dot{\rho}(x,t) = \rho(x,t) + u_i(x,t) + f(x,t), \quad \forall t \in [0,T],\ i = \overline{1,N}, \\ &\rho(x,0) = \rho_0(x). \end{aligned} \tag{1'}$$

Нехай на множині Ω задана деяка поверхня функцією $\mu(x)$.

З кожною підмножиною $\Omega_i(\cdot)$ будемо пов'язувати точку $\tau_i \in \Omega,\ i = \overline{1,N},$ – її «центр», який в свою чергу може змінювати своє місце розташування за законом

$$\begin{aligned} &\ddot{\tau}_{1i}(t) = a_i(t)\cos\varphi_i(t), \\ &\ddot{\tau}_{2i}(t) = a_i(t)\sin\varphi_i(t), \qquad t \in [0,T], \end{aligned} \tag{2'}$$

$$\begin{aligned} &(\tau_i(0), \dot{\tau}_i(0)) = (\tau_i^0, v_i^0), \\ &(\tau_i(T), \dot{\tau}_i(T)) = (\tau_i^T, 0), \end{aligned} \quad i = \overline{1,N}. \tag{3'}$$

і є проекцією деякої точки $(\tau_{1i},\tau_{2i},\tau_{3i})$, де $\tau_{3i}=\mu(\tau_{1i},\tau_{2i})$, на множину Ω. Одже, деякі точки $(\tau_{1i},\tau_{2i},\tau_{3i})$ рухаються поверхнею $\mu(x)$ під впливом керувань $a_i(t)$ та $\varphi_i(t)$ їхніми проекціями. Такі точки будемо називати «рухомими», а їх проекції на множину Ω – «рухомими центрами» підмножин $\Omega_i(\cdot)$.

Потрібно визначити такі функції $\overline{\omega}^*(t)=(\Omega^*_1(t),...,\Omega^*_N(t)$ розбиття множини Ω на N неперетинних підмножин $\overline{\omega}^*(t)=(\Omega^*_1(t),...,\Omega^*_N(t))$, $t\in[0,T]$, такі функції керування $a_i(t)$ та $\varphi_i(t)$, $i=\overline{1,N}$, і відповідний закон руху «рухомих центрів» підмножин множиною Ω, при яких функціонал

$$J(\overline{\omega}(\cdot),\tau(\cdot))=\gamma_0\int_0^T\sum_{i=1}^N\int_{\Omega_i(t)}\left[\left(c(x,\tau_i(t),t)+A_i\right)\rho(x,t;\overline{\omega}(t))+u_i^2(x,t)\right]dxdt+ \tag{4'}$$

$$+\gamma_1\left\{\sum_{i=1}^N\int_{\tau_i(0)}^{\tau_i(T)}dL_i+\int_0^T\sum_{i=1}^N a_i^2(t)dt\right\}$$

досягав би свого найменшого значення, за умов (2')-(3').

Тут $\gamma_0,\gamma_1\geq 0$ – константи, які визначають приорітет відповідного доданку; $A_i, i=\overline{1,N},$ – фіксовані величини; $c(x,\tau_i(t),t)$ – задана функція, L_i – довжина i-ї кривої (траєкторії руху i-ї «рухомої точки» по поверхні $\mu(x)$), що з'єднує точку, з якої починається рух, τ_i^0, та кінцеву точку τ_i^T; $\tau_i(t)\in\Omega, i=\overline{1,N}$, – «рухомі центри» підмножин, якими треба керувати оптимальним чином за допомогою функцій $a_i(t)$ та $\varphi_i(t)$, $i=\overline{1,N}$; $T>0$ – фіксована тривалість керованого процесу; функція $\rho(x,t)$ є розв'язком задачі Коші (1'); $f(x,t),\rho_0(x)$ – відомі функції своїх аргументів.

Опис методу розв'язання поставленої задачі ОРМ. Перший доданок функціоналу (4') – це цільовий функціонал, наведений в [4], задачі ОРМ з рухомими границями між підмножинами. Другий доданок функціоналу (4') містить суму криволінійних інтегралів першого роду.

Введемо наступні позначення

$$\dot{\tau}_{1i}(t) = v_{1i}(t),$$
$$\dot{\tau}_{2i}(t) = v_{2i}(t), \qquad t \in [0,T],\ i = \overline{1,N},$$

тоді умови (2') та (3') перепишуться, відповідно, у вигляді

$$\begin{aligned} &\dot{\tau}_{1i}(t) = v_{1i}(t),\\ &\dot{v}_{1i}(t) = a_i(t)\cos\varphi_i(t), \qquad t \in [0,T],\ i = \overline{1,N},\\ &\dot{\tau}_{2i}(t) = v_{2i}(t),\\ &\dot{v}_{2i}(t) = a_j(t)\sin\varphi_i(t), \end{aligned} \tag{2}$$

$$\begin{aligned} &\tau_i(0) = (\tau_{1i}^0, \tau_{2i}^0), \quad v_i(0) = (v_{1i}^0, v_{2i}^0),\\ &\tau_i(T) = (\tau_{1i}^T, \tau_{2i}^T), \quad v_i(T) = (0,0), \end{aligned} \qquad i = \overline{1,N}. \tag{3}$$

Зведемо криволінійний інтеграл першого роду в (4') до звичайного інтегралу:

$$\sum_{i=1}^{N} \int_{\tau_i(0)}^{\tau_i(T)} dL_i = \sum_{i=1}^{N} \int_0^T \sqrt{\dot{\tau}_{1i}^2(t) + \dot{\tau}_{2i}^2(t) + \dot{\tau}_{3i}^2(t)}\,dt = \sum_{i=1}^{N} \int_0^T \sqrt{v_{1i}^2(t) + v_{2i}^2(t) + \mu^2(\tau_{1i}(t), \tau_{2i}(t))}\,dt$$

.

Фізичний зміст криволінійного інтегралу першого роду – це маса кривої, яку ми і будемо мінімізувати.

Користуючись тими ж міркуваннями, що і у [4] перепишемо розглянуту задачу у вигляді задачі оптимального керування і одразу введемо характеристичні функції підмножин. Отримаємо наступну задачу

$$\begin{aligned} J(\lambda(\cdot,\cdot), \tau(\cdot)) = \gamma_0 \int_0^T \sum_{i=1}^{N} \int_\Omega \Big[\big(c(x, \tau_i(t), t) + A_i\big)\rho(x,t) + u_i^2(x,t) \Big] \lambda_i(x,t)\,dx\,dt + \\ + \gamma_1 \Bigg\{ \sum_{i=1}^{N} \int_0^T \sqrt{v_{1i}^2(t) + v_{2i}^2(t) + \mu^2(\tau_{1i}(t), \tau_{2i}(t))}\,dt + \\ + \int_0^T \sum_{i=1}^{N} a_i^2(t)\,dt \Bigg\} \to \min_{\lambda \in \Lambda_1}, \end{aligned} \tag{4}$$

$$\Lambda_1(t) = \{\lambda(x,t) = (\lambda_1(x,t), \cdots, \lambda_N(x,t)) : \ 0 \le \lambda_i(x,t) \le 1,\ i = \overline{1,N},$$
$$\sum_{i=1}^{N} \lambda_i(x,t) = 1, \text{ майже для всіх } x \in \Omega\}, \quad t \in [0,T],$$

$$\begin{aligned} &\dot{\rho}(x,t) = \rho(x,t) + u(x,t;\bar{\omega}(t)) + f(x,t), \quad u \in U(t), \quad t \in [0,T],\\ &\rho(x,0) = \rho_0(x). \end{aligned} \tag{1}$$

за умов (2), (3).

Тут

$$U(t)=\{u(x,t;\overline{\omega}(t)):\ u(x,t;\overline{\omega}(t))=u_i(x,t)\ \textit{майже для всіх}$$
$$x\in\Omega_i(t),\ i=\overline{1,N};\quad \overline{\omega}(t)=(\Omega_1(t),\cdots,\Omega_N(t))\in P_N(\Omega)\},\quad \forall t\in[0,T].$$

Задача (1_4) є задачею оптимального керування, в якій присутньо два вида керування: 1) керування границями підмножин – вектор_функція $\lambda(\cdot,\cdot)$; 2) керування «рухомими центрами» підмножин – вектор_функції $a(t)$ та $\varphi(t)$, тобто на скільки треба зміщуватися і під яким кутом, відповідно.

Умови оптимальності мають наступний вигляд:

а) стаціонарність за $\tau_1(t),\tau_2(t),v_1(t),v_2(t),\rho(x,t)$:

$$\begin{aligned}
\frac{\partial\Psi_1}{\partial t}&=-\frac{\partial H}{\partial\tau_1},\\
\frac{\partial\Psi_2}{\partial t}&=-\frac{\partial H}{\partial v_1},\\
\frac{\partial\Psi_3}{\partial t}&=-\frac{\partial H}{\partial\tau_2},\\
\frac{\partial\Psi_4}{\partial t}&=-\frac{\partial H}{\partial v_2},\\
\frac{\partial\Psi_5}{\partial t}&=-\frac{\partial H}{\partial\rho},\qquad \forall x\in\Omega,
\end{aligned}\tag{5}$$

де функція Гамільтона_Понтрягіна має вид

$$H(\lambda(x,t);\tau_1(t),\tau_2(t),v_1(t),v_2(t),\rho(x,t);\Psi_1(t),\Psi_2(t),\Psi_3(t),\Psi_4(t),\Psi_5(x,t))=$$
$$=\sum_{i=1}^{N}\left(\Psi_{1i}(t)v_{1i}(t)+\Psi_{2i}(t)a_i(t)\cos\varphi_i(t)+\Psi_{3i}(t)v_{2i}(t)+\Psi_{4i}(t)a_i(t)\sin\varphi_i(t)\right)+$$
$$+\int_\Omega\Big(\rho(x,t)+\sum_{i=1}^{N}u_i(x,t)\lambda_i(x,t)+f(x,t)\Big)\Psi_5(x,t)dx-$$
$$-\gamma_0\int_\Omega\sum_{i=1}^{N}\left((c(x,\tau_i,t)+a_i)\rho(x,t)+u_i^2(x,t)\right)\lambda_i(x,t)dx-$$
$$-\gamma_1\sum_{i=1}^{N}\sqrt{v_{1i}^2(t)+v_{2i}^2(t)+\mu^2(\tau_{1i}(t),\tau_{2i}(t))};$$

б) трансверсальність за $\rho(\cdot,\cdot)$: $\forall x\in\Omega$

$$\Psi_5(x,T)=0;\tag{6}$$

в) оптимальність за $a(\cdot)$:

$$a^*(t):\quad \max_a H(\lambda(\cdot,\cdot),\tau(\cdot),v(\cdot),\rho(\cdot,\cdot),\Psi_1(\cdot),\Psi_2(\cdot),\Psi_3(\cdot),\Psi_4(\cdot),\Psi_5(\cdot,\cdot)),$$

тобто

$$a_i(t)=\frac{\Psi_{2i}(t)\cos\varphi_i(t)+\Psi_{4i}(t)\sin\varphi_i(t)}{2}; \qquad (7)$$

оптимальність за $\varphi(\cdot)$:

$$\varphi^*(t):\quad \max_\varphi H(\lambda(\cdot,\cdot),\tau(\cdot),v(\cdot),\rho(\cdot,\cdot),\Psi_1(\cdot),\Psi_2(\cdot),\Psi_3(\cdot),\Psi_4(\cdot),\Psi_5(\cdot,\cdot)),$$

тобто

$$\varphi_i(t)=arctg\frac{\Psi_{4i}(t)}{\Psi_{2i}(t)}; \qquad (8)$$

оптимальність за $\lambda(\cdot,\cdot)$:

$$\lambda^*(x,t):\quad \max_{\lambda\in\Lambda_1} H(\lambda(\cdot,\cdot),\tau(\cdot),v(\cdot),\rho(\cdot,\cdot),\Psi_1(\cdot),\Psi_2(\cdot),\Psi_3(\cdot),\Psi_4(\cdot),\Psi_5(\cdot,\cdot)).$$

Враховуючи структуру множини Λ_1 можна записати: $\forall x\in\Omega,\ t\in[0,T]$

$$\lambda_i^*(x,t)=\begin{cases}1, & q_i(x,\tau_i,t)=\max\limits_{k=1,N} q_k(x,\tau_k,t),\\ 0, & \textit{в усіх інших випадках};\end{cases} \qquad (9)$$

де

$$q_i(x,\tau_i,t)=u_i(x,t)\Psi_5(x,t)-\gamma_0\left[(c(x,\tau_i,t)+a_i)\rho(x,t)-u_i^2(x,t)\right]$$

Для чисельного розв'язання отриманої крайової задачі побудовано алгоритм на основі методу Ньютона.

Результати експерементів. Розглянемо наступну поверхню $\mu(x)=80e^{-12(x_1-0.9)^2-7(x_2-1.3)^2}+280e^{-8(x_1-1.8)^2-11(x_2-2)^2}$, графічне зображення (рис. 1, 2):

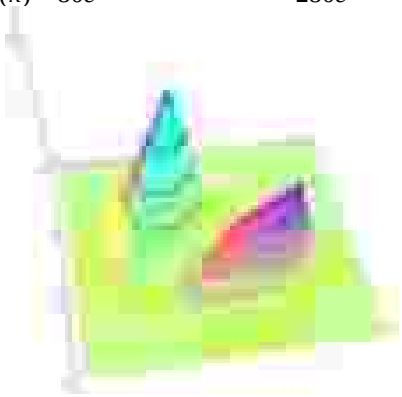


Рис.1. Поверхня $\mu(x)$

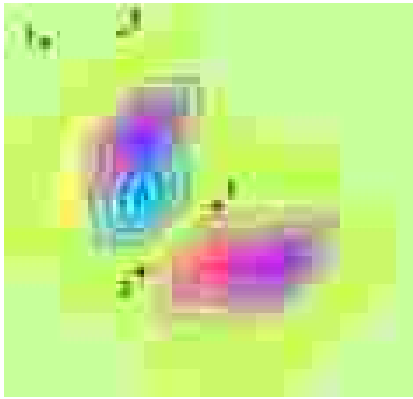
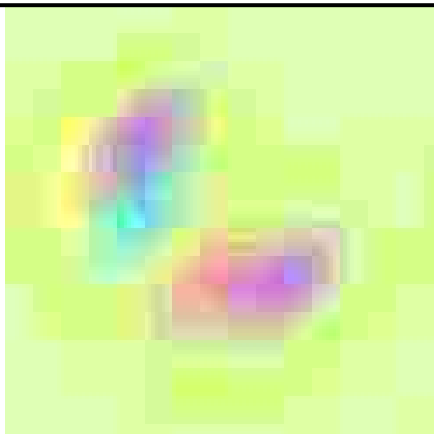
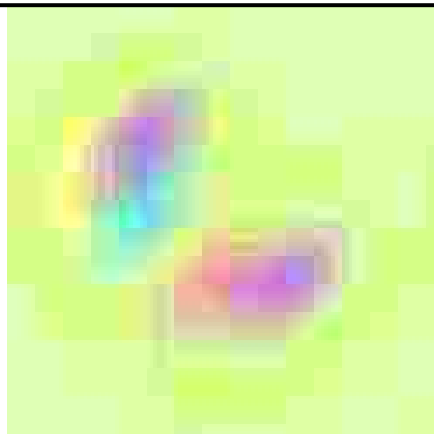
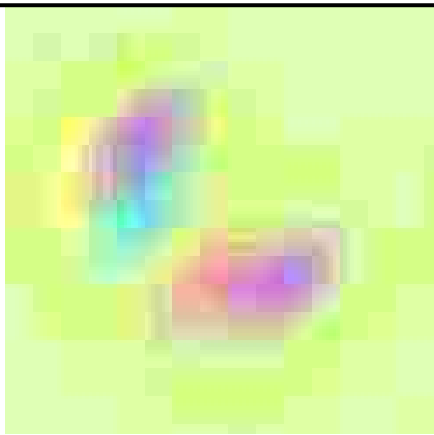
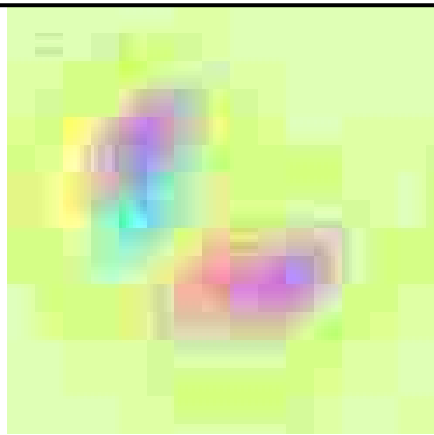


Рис.2. Лінії рівня

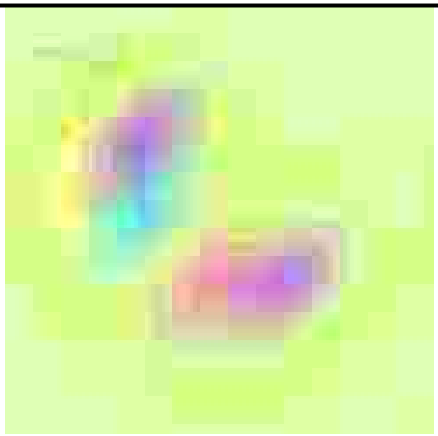
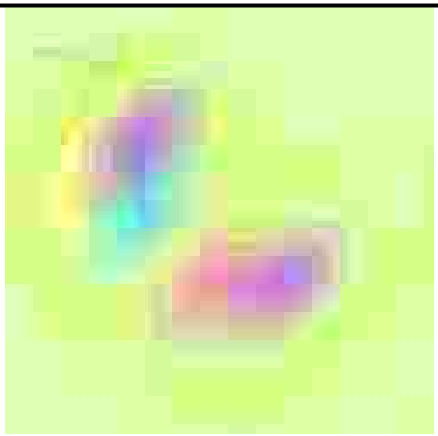
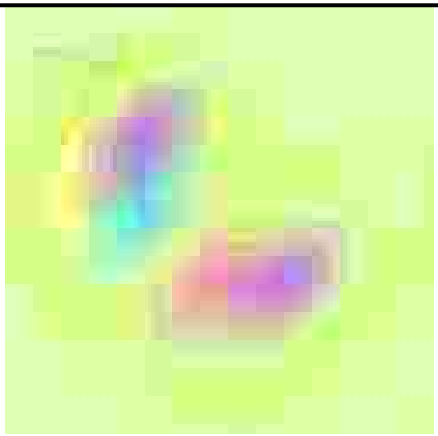
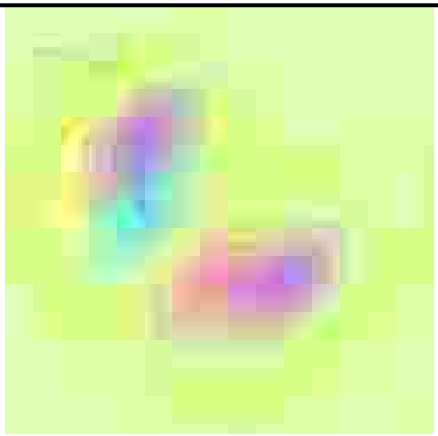
В таблиці 1 наведені результати роботи алгоритму для обраних фіксованих параметрів задачі. Алгоритм було реалізовано на мові Pascal у середовищі Delphi 7.

Табл. 1.

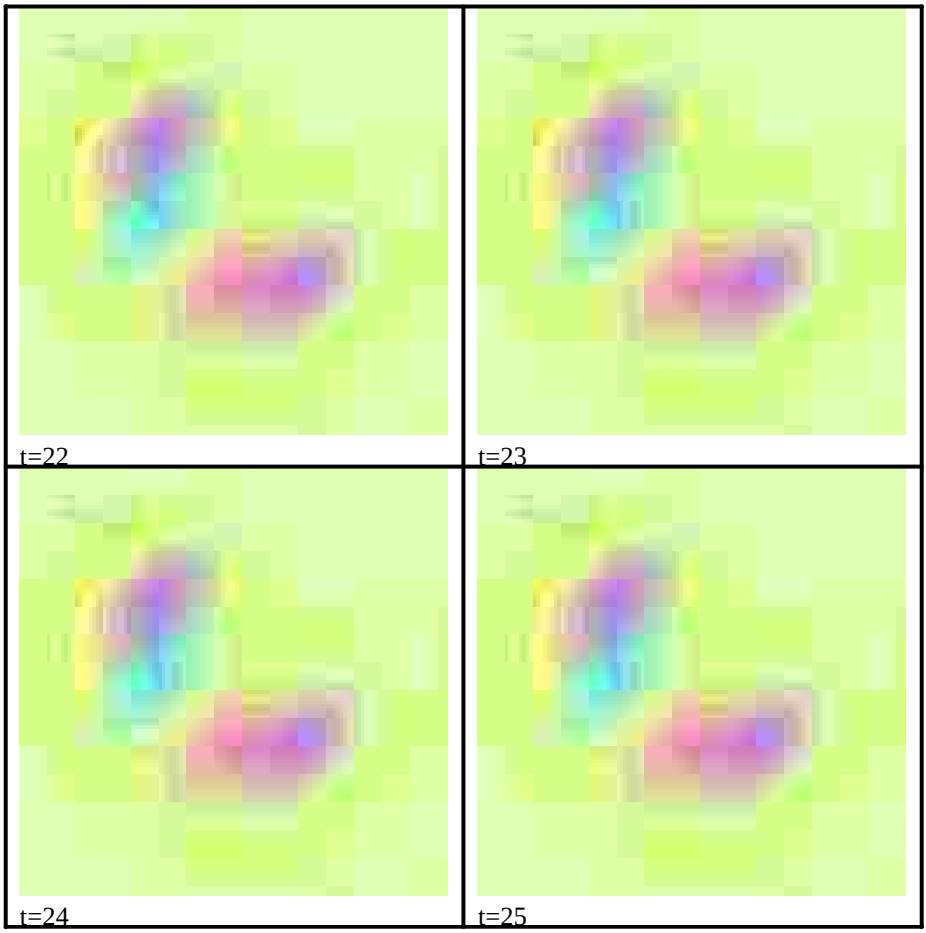
Обрані стани динамічної системи

 t=0	 t=1
 t=2	 t=3

Продовження таблиці 1

 t=11	 t=12
 t=13	 t=14

Продовження таблиці 1



В даному прикладі було взято такі параметри:

$\rho_0(x) = 1$ $\Omega = [0,3] \times [0,3]$ $\tau_1(0) = (0.3,0.3)$, (0) $(0.8,0.2)$

$u_i() = a_i + b_i$, $i = 1,2.$ $\tau_1(T) = (1.5,1.5)$, $(\ )$ $(1,2)$

$a_{11} = 0.1$, 0 $v_1(0) = (0.1,0.1)$, (0) $(0.3,0.2)$

$a_{22} = 0.3$, 0 $T = 2.5$

$$\mu(x,y) = 80^{-12(x_1-0.9)^2-7(y-1.3)^2} + 280e^{-8(x_1-1.8)^2-11(y-2)^2}$$

Висновки. У роботі запропоновано і обґрунтовано метод розв'язання динамічної задачі ОРМ з рухомими границями і центрами підмножин. На основі запропонованого методу розв'язання задачі побудовано відповідний алгоритм її розв'язання, роботу якого реалізовано за допомогою створеного програмного забезпечення і перевірено на деяких тестових моделях. Треба зауважити також, що з прикладної точки зору такі задачі розбиття виникають при моделюванні ситуацій які виникають при пересуванні певних об'єктів гірською місцевістю.

Бібліографічні посилання

1. **Киселева Е. М.** Непрерывные задачи оптимального разбиения множеств: теория, алгоритмы, приложения: Монография / Е. М. Киселева, Н. З. Шор – К., 2005. – 564 с.
2. **Кісельова О.М.** Напрямки розвитку динамічних задач оптимального розбиття множин. / О.М. Кісельова, Л.С. Коряшкіна, Т.О. Шевченко // Системний аналіз та інформаційні технології: матеріали 12-ї Міжнародної науково-технічної конференції SAIT 2010, Київ, 25-29 травня 2010р. – К.: ННК «ІПСА» НТУУ «КПІ» 2010. – С. 98.
3. **Коряшкіна Л.С.** Нові підходи до розв'язання динамічної задачі оптимального розбиття множин / Л.С. Коряшкіна, Т.О. Шевченко // Питання прикладної математики і математичного моделювання. – Д.: ДНУ 2009, – С. 220-231.
4. **Коряшкіна Л.С.** Особливості розвязку задачі оптимального розбиття множин з рухомими границями між підмножинами / Л.С. Коряшкіна, Т.О. Шевченко // Питання прикладної математики і математичного моделювання. – Д.: ДНУ 2008, – С. 167-178.

Надійшла до редколегії 07.06.11

УДК 656.612

[14]В.Г. Шерстюк
Херсонский национальный технический университет

МЕТОД ОЦЕНКИ ПОДОБИЯ СИТУАЦИЙ В СЦЕНАРНО-ПРЕЦЕДЕНТНОЙ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЕ УПРАВЛЕНИЯ ПОДВИЖНЫМ ОБЪЕКТОМ

Проведено аналіз особливостей прийняття рішень оператором рухомого об'єкту, визначено необхідність розробки адекватної функції подібності для пошуку аналогічних ситуацій та формалізовано гранулярну структуру прецеденту. Запропоновано оцінки подібності значень атрибутів прецеденту для всіх їх типів, отримано загальний метод оцінки подібності прецедентів, що може бути застосований у сценарно-прецедентних інтелектуальних системах реального часу.

Проведен анализ особенностей принятия решений оператором подвижного объекта, определена необходимость разработки адекватной функции подобия для поиска аналогичных ситуаций и формализирована гранулярная структура прецедента. Предложены оценки подобия значений аттрибутов прецедента для всех ее типов, получен общий метод оценки подобия прецедентов, который может быть использован в сценарно-прецедентных интеллектуальных системах реального времени.

The decision making features are analized by the operator of moving object, the necessity of adequate similarity function development to findsimilar situations is determined and precedent granular structure is formalized. For all its types evaluations of precedent attributes values similarity are proposed, a general method of precedents' similarity evaluation is obtained. It can be used in scenario-precedent intelligent real-time systems.

Ключевые слова: прецедентная интеллектуальная система, подвижный объект, функция подобия.

Введение. Обеспечение гарантированной безопасности движения управляемых оператором подвижных объектов (ПО) представляет собой важную научно-техническую проблему. Особенно важно поддерживать безопасность ПО в стесненных навигационных условиях, когда из-за неполноты и неопределенности исходной информации, значительных объе-

мов вычислений при существенных ограничениях во времени складываются информационно сложные ситуации для оператора [1], создающие риск экономических, экологических и, зачастую, человеческих потерь. Цена любой ошибки оператора ПО в таких ситуациях может быть непомерно высокой.

Поскольку большинство аварийных ситуаций (до 75 % [2]) происходит вследствие влияния «человеческого фактора», одним из направлений повышения надежности и безопасности управления ПО может являться разработка и внедрение новых интеллектуальных технологий, основным принципом реализации которых является использование методов современного искусственного интеллекта, в том числе методов интеллектуальной поддержки принятия решений оператором.

Необходимо отметить, что вопросы разработки и практического применения интеллектуальных систем (ИС) для решения задач поддержки принятия решений операторами в процессе управления ПО в реальном масштабе времени в настоящее время проработаны недостаточно, а в связи с участившимися случаями инцидентов и аварий *актуальны* и представляют значительный интерес для исследований.

Постановка задачи. Проведенный анализ предметной области показывает, что процессу принятия решений оператором ПО присущ ряд особенностей:

большая размерность решаемых задач и значительные объемы обрабатываемых данных;

существенная неопределенность стохастических воздействий внешней среды на ПО;

неполнота и неточность информации, получаемой от бортовых навигационных технических средств;

нестационарность и нелинейность моделей движения ПО, приводящая к их неадекватности задаче управления;

неоднозначность и противоречивость существующих нормативных регуляторов движения ПО;

нескоординированность и противоречивость целей операторов различных ПО, взаимодействующих на ограниченном пространстве движения;

непредсказуемость действий операторов окружающих ПО в слож-

ных, и особенно в предаварийных, ситуациях;

сложность формализации процедур принятия решений оператором ПО в информационно сложных ситуациях.

Указанные особенности в условиях дефицита времени на принятие решений исключают возможность использования классических и современных подходов теории управления, а также значительно усложняют применение известных методов искусственного интеллекта.

Вместе с тем, наблюдается значительная повторяемость принимаемых оператором ПО решений в сходных навигационных ситуациях. Кроме того, статистически подтвержден [3] факт первоочередного использования оператором в качестве основания для принятия решения предыдущего опыта и знания общих закономерностей процесса движения. В [1] показано, что опыт, навыки и эвристические возможности оператора позволяют компенсировать влияние недостоверности знаний на процесс движения ПО, а в информационно сложных ситуациях оператор может выполнять функции регулятора в контуре управления ПО.

Вышеизложенные соображения приводят к идее использования для управления ПО сценарно-прецедентного подхода, являющегося дальнейшим развитием методов принятия решений по прецедентам (Case-based reasoning), основанных на предположении, что в подобных ситуациях, как правило, принимаются схожие решения.

В [4] предложена концепция ИС управления ПО, основанная на интеллектуальной системе поддержки принятия решений (ИСППР) оператором. Структура и возможности ИСППР, в основе которой лежит сценарно-прецедентный подход к принятию решений, позволяющий решить основные проблемы предметной области, представлены в [5;6].

Известно, что краеугольным камнем сценарно-прецедентной ИС является используемая в ней функция подобия, позволяющая для текущей ситуации найти аналогичную, имевшую место ранее. Несмотря на то, что функционирование сценарно-прецедентной ИС во многом зависит от размера и построения хранилища прецедентов, от структуры прецедентов и способов представления информации, от алгоритмов адаптации решений и т. д., эффективность ИС во многом определяется выбором адекватной функции подобия [7].

Как правило, в прецедентных ИС используют функции подобия, основанные на использовании различных модификаций метода ближайшего

соседа. Однако, данному методу свойственны сложность выбора метрик для определения степени подобия и неэффективность при работе с неполными и неточными («зашумленными») данными [8]. Поскольку предметная область является открытой и динамичной, способы определения подобных ситуаций на основе деревьев решений, семантической индексации и т. д. [9] в ней непригодны.

Таким образом, при разработке сценарно_прецедентной ИС управления ПО одним из важнейших и не до конца исследованных вопросов является выбор функции подобия ситуаций. В данной работе ставится задача выбора адекватной функции подобия для сценарно_прецедентной ИС, предназначенной для открытой динамичной предметной области.

Метод решения. Сценарно_прецедентные ИСППР – класс интеллектуальных систем автоматизированного вывода решений, основанный на принципах:

1) повторяемости ситуаций;

2) возможности использования ранее принятых решений в случае возникновения сходных проблемных ситуаций;

3) представления решений в форме сценариев [10].

Прецедент включает в себя описание проблемной ситуации, принятое решение и полученный результат. Поиск подобных ситуаций в ИС управления ПО может производиться на основе исходной информации о текущей навигационной ситуации, для выполнения поиска должна быть задана функция оценки подобия и выбрано корректное множество сопоставляемых параметров [11].

Предположим, что хранилище прецедентов $\Im$ содержит M прецедентов

$$\Im = \{e_1, e_2, ...e_M\}.$$

Прецедент e_i содержит описание навигационной ситуации s_i, принятого решения r_i и полученного результата u_i

$$e_i = \langle s_i, r_i, u_i \rangle .$$

В общем случае результат u_i может представлять собой некоторую

следующую ситуацию s_j, полученную из s_i путем использования заданного в r_i сценария управляющих воздействий, и в дальнейшем изложении может быть опущен.

Определим $A = \{a_1, a_2, ...a_N\}$ как множество атрибутов, описывающих ситуацию s_i, $B = \{\varsigma_1, \varsigma_2, ...\varsigma_L\}$ как множество доступных сценариев управляющих воздействий.

Тогда прецедент $e_i \in \Im$ может рассматриваться как

$$e_i = \left\langle x_i^{a_1}, x_i^{a_2}, ...x_i^{a_N}, \varsigma_i \right\rangle,$$

где $x_i^{a_k}$ – приписанное атрибуту a_k прецедента e_i значение.

Каждое значение атрибута $x_i^{a_k}$, в свою очередь, имеет следующую гранулярную структуру

$$x_i^{a_k} = \langle Tag, DataLow, DataHigh, Func \rangle,$$

где $DataLow$ – нижний предел числового интервала; $DataHigh$ – верхний предел числового интервала; $Func$ – функция принадлежности; Tag – тип числового значения: 0 – значение отсутствует, NULL; 1 – «точное» числовое значение (находится в $DataLow$); 2 – «приближенное» (*Rough*) значение, используется интервал $[DataLow..DataHigh]$; 3 – «нечеткое» ($Fuzzy$) значение, используется интервал $[DataLow..DataHigh]$ и функция принадлежности $Func$.

Известно, что нечеткие множества слабо пригодны для формализации неточности источников информации в выбранной предметной области, поскольку функции принадлежности нечетких множеств в большинстве случаев заранее неизвестны, а производить их построение экспертным методом в режиме реального времени не представляется возможным. В то же время, существует теория приближенных множеств (Rough Set) [12], в основу концепции которых положена идея неточных знаний – знание считается неточным, если оно содержит неточные концепты. Неточные концепты могут быть определены приблизительно в доступном пространстве знаний с использованием двух точных концептов, называемых нижним и верхним приближением, без каких либо предположений о распределении

значений внутри полученных границ интервала.

Предложенный подход позволяет организовать доступную информацию таким образом, чтобы неточность измерений представлять с помощью аппарата приближенных множеств, а неопределенности других типов представлять с помощью аппарата нечетких множеств. Кроме того, неполнота доступной информации может быть представлена с помощью специального символа NULL.

Поскольку часть исходной информации может быть «точной», другая часть – неполной и неточной, атрибуты прецедентов, хранимых в $\Im$ и описывающих ситуации могут иметь неоднотипные значения для одних и тех же атрибутов, что порождает проблему их сравнения.

Для двух рассматриваемых прецедентов e_i и e_k нахождение оценки сходства по значению атрибута a_l ($x_i^{a_l}$ и $x_k^{a_l}$) предполагает следующие варианты сравнений:

1. Двух «точных» значений (Tag=1). Для этого случая ИСППР должна иметь предустановленный и регулируемый с помощью обратных связей порог различимости значений λ_l, тогда оценка сходства по атрибуту a_l

$$\xi_{ik}^{l} = 1 - \frac{\left| x_i^{a_l} - x_k^{a_l} \right|}{\lambda_l}.$$

2. «Точного» (Tag=1) и нечеткого (Tag=3) значения. В этом случае оценка сходства может быть получена с использованием функции принадлежности «точного» значения $x_i^{a_l}$ нечеткому множеству A_k^l, $x_k^{a_l} \in A_k^l$:

$$\xi_{ik}^{l} \quad \mu_{A_k^l}\left(x_i^{a_l}\right).$$

3. «Точного» (Tag=1) и приближенного (Tag=2) значения. В этом случае для оценки сходства проверяют «попадание» значения $x_i^{a_l}$ в границы интервала, образованного нижним $\underline{IND}\left(x_k^{a_l}\right)$ и верхним $\overline{IND}\left(x_k^{a_l}\right)$ приближением $x_k^{a_l}$:

$$\xi_{ik}^{l} = \begin{cases} 1, \text{ если } x_i^{a_l} \in \left[\underline{IND}\left(x_k^{a_l}\right)\overline{IND}\left(x_k^{a_l}\right)\right] \\ 0, \text{ если } x_i^{a_l} \notin \left[\underline{IND}\left(x_k^{a_l}\right)\overline{IND}\left(x_k^{a_l}\right)\right] \end{cases}$$

4. Любого значения (Tag=1..3) с неизвестным (Tag=0). Поскольку неизвестное значение может быть любым, предполагается

$$\xi_{ik}^{l} = 1.$$

5. Приближенного значения (Tag=2) с другим приближенным. В этом случае оценку сходства проводят следующим образом:

$$\xi_{ik}^{l} = \begin{cases} 0, \text{ если } \left[\underline{IND}\left(x_i^{a_l}\right)\overline{IND}\left(x_i^{a_l}\right)\right] \cap \left[\underline{IND}\left(x_k^{a_l}\right)\overline{IND}\left(x_k^{a_l}\right)\right] = \varnothing \\ \dfrac{\overline{IND}\left(x_i^{a_l}\right) - \underline{IND}\left(x_k^{a_l}\right)}{\overline{IND}\left(x_k^{a_l}\right) - \underline{IND}\left(x_i^{a_l}\right)} \text{ если } \underline{IND}\left(x_i^{a_l}\right) \subseteq \underline{IND}\left(x_k^{a_l}\right) \subseteq \overline{IND}\left(x_i^{a_l}\right) \\ \dfrac{\underline{IND}\left(x_i^{a_l}\right) - \overline{IND}\left(x_k^{a_l}\right)}{\underline{IND}\left(x_k^{a_l}\right) - \overline{IND}\left(x_i^{a_l}\right)} \text{ если } \underline{IND}\left(x_k^{a_l}\right) \subseteq \underline{IND}\left(x_i^{a_l}\right) \subseteq \overline{IND}\left(x_k^{a_l}\right) \\ 1, \text{ если } \left[\underline{IND}\left(x_i^{a_l}\right)\overline{IND}\left(x_i^{a_l}\right)\right] = \left[\underline{IND}\left(x_k^{a_l}\right)\overline{IND}\left(x_k^{a_l}\right)\right] \end{cases}.$$

6. Нечеткого значения (Tag=3) с другим нечетким. Для оценивания сходства используем нечеткое отношение покрытия.

Пусть U – нечеткое множество с универсумом A (полагаем, что A и P – конечные множества. Покрытием U будем называть семейство $\Sigma = \{A_p\}_{p \in P}$ нечетких множеств с общим универсумом A, если $U = \bigcup_{i \in P} A_p$. Конечным отношением, ассоциируемым с семейством Σ, является нечеткое бинарное отношение, определенное условием $R_\Sigma(x, y) = \vee_{p \in P} \{A_p(x) \wedge A_p(y)\}$.

Любое покрытие $\Sigma = \{A_p\}_{p \in P}$ можно интерпретировать как нечеткое отображение, и нечеткое отображение есть соответствие между множеством атрибутов A и множеством значений P.

Нечеткое отношение будем называть отношением сходства на нечетком множестве U в случае, если выполняются свойства:

1) $R\Sigma(x,y) = R\Sigma(y,x) \quad \forall x, \forall y \in A$;

2) $R\Sigma(x,y) \le R\Sigma(x,x) \vee R\Sigma(y,y) \quad \forall x, \forall y \in A$;

3) $R\Sigma(x,x) = U(x) \quad \forall x \in A$.

Свойства (2) и (3) есть свойства соответственно симметричности и слабой рефлексивности.

Любое нечеткое бинарное отношение, ассоциированное с покрытием, является отношением сходства. Для каждого покрытия Σ нечеткого множества U существует отношение сходства $R\Sigma$ на U, ассоциированное с Σ, и наоборот, для каждого отношения сходства R на U существует покрытие Σ универсума U такое, что $R = R\Sigma$.

Отношение подобия составляет частный случай отношения сходства, в случае если обладает свойствами симметричности и транзитивности.

Множество α-уровня нечеткого множества U определим как четкое множество $U^{\alpha} = \{x \in A | U(x) \ge \alpha\}, \alpha \in [0,1]$.

Если $\Sigma = \{A_p\}_{p \in P}$ – покрытие нечеткого множества U, то, очевидно, $\Sigma^{\alpha} = \{A_p\}_{p \in P}$ – четкое покрытие U^{α} для любых $\alpha \in [0,1]$.

Выражение для оценки подобия между двумя нечеткими значениями $x_i^{a_l}$ и $x_k^{a_l}$ должно зависеть от значений их функций принадлежности и от выбранного отношения подобия. На практике выберем, основываясь на мерах возможности и необходимости по Суджено [13]

$$\xi_{ik}^{l} = \alpha \Big| \sup_{x_k^{a_l} \in A_k^l} \mu_{A_k^l}^{\alpha}\left(x_k^{a_l}\right) = \inf_{x_i^{a_l} \in A_i^l} \mu_{A_i^l}^{\alpha}\left(x_i^{a_l}\right).$$

7. Приближенного (Tag=2) и нечеткого (Tag=3) значения. В данном случае наиболее целесообразно привести приближенную оценку к нечеткой, воспользовавшись понятием области неразличимости IND приближенного множества [14]:

$$IND\left(x_i^{a_l}\right) = \overline{IND}(x_i^{a_l}) - \underline{IND}(x_i^{a_l}).$$

Тогда можно получить функцию принадлежности

$$\mu_{A_i^l}^{IND}(x_i^{a_l}) = \frac{\left|P^l \cap IND(x_i^{a_l})\right|}{\left|IND(x_i^{a_l})\right|}, \ \mu_{A_i^l}^{IND}(x_i^{a_l}) \in [0,1].$$

Зная ее значение, можно провести оценку сходства значений таким образом, как предложено выше для случая двух нечетких значений атрибута.

Получив оценки сходства по значениям всех известных атрибутов прецедентов e_i и e_k, можно оценить степень подобия навигационных ситуаций, описываемых данными прецедентами:

$$\xi_{ik} = \frac{\sum_{j=1}^{m} \omega_j \cdot \xi_{ik}^j}{\sum_{j=1}^{m} \omega_j},$$

где ω_j – весовой коэффициент j-го атрибута, а m – число атрибутов в рассматриваемых прецедентах. Весовой коэффициент может быть предустановлен, а может изменяться в процессе функционирования ИС с помощью механизмов обратной связи, например самообучением системы. С помощью этого коэффициента может быть реализована семантическая индексация прецедентов, могут присоединяться механизмы оценки уместности (релевантности) прецедентов для текущей ситуации [8].

На основе полученной оценки подобия прецедентов навигационные ситуации могут быть признаны идентичными ($\xi_{ik} = 1$), подобными ($\lambda^* \le \xi_{ik} \le 1$) или различающимися ($\xi_{ik} \le \lambda^*$), при этом порог различимости ситуаций λ^* задается и регулируется оператором.

Выводы. Анализ особенностей принятия решений оператором при управлении ПО показал возможность использования для реализации ИС управления ПО сценарно-прецедентного подхода. Эффективность реализации сценарно-прецедентной ИС во многом определяется выбором адекватной функции подобия, позволяющей корректно сравнивать текущую

навигационную ситуацию с прецедентами, накопленными в хранилище.

Предложенная в работе модель оценки подобия прецедентов на основе покоординатного сравнения значений атрибутов может быть использована в условиях неполноты и неточности исходной информации. Гранулярность структуры прецедента позволяет использовать для формализации различных факторов неопределенности математический аппарат нечетких и приближенных множеств. В работе представлены способы оценки подобия атрибутов прецедентов для различных комбинаций типов значений атрибутов. Использование регулируемых порогов различимости для атрибутов и ситуаций в целом, а также весовых коэффициентов позволяет в дальнейшем развивать методы поиска адекватных прецедентов для текущей навигационной ситуации.

Предложенный подход может быть реализован на практике в сценарно-прецедентной ИСППР оператора ПО. Применение подобной ИС в информационно сложных ситуациях позволяет избавиться от информационных и временных перегрузок оператора, минимизировать влияние «человеческого фактора» и снизить риск аварийных ситуаций.

Библиографические ссылки

1. **Сиек Ю.Л.** Принципы синтеза интеллектуальных систем управления морскими динамическими объектами / Ю.Л. Сиек, Соэ Мин Лвин // Искусственный интеллект. – 2009. – №4. – С.448-456.

2. **Топалов В.П.** К проблеме человеческого фактора в судоходстве / В.П. Топалов, В.Г. Торский, Ю.В. Торский // Судовождение. – 2004. – Вып. 8. – С. 94-102.

3. **Вильский Г.Б.** Навигационная безопасность при лоцманской проводке судов / Г.Б. Вильский, А.С. Мальцев, В.В. Бездольный, Е.И. Гончаров. – Одесса-Николаев, 2007. – 456с.

4. **Шерстюк В.Г.** Принципы интеллектуальной поддержки принятия решений по управлению движением судна // В.Г. Шерстюк / Вестник Херсонского национального технического университета. – 2009. – №3(36). – С.133-141.

5. **Шерстюк В.Г.** Интеллектуальные системы поддержки принятия решений по управлению судном в условиях неполной и противоречивой информации / В.Г. Шерстюк, А.П. Бень // Судовождение. – 2007. – Вып. 14. – С.141-144.

6. **Шерстюк В.Г.** Гибридная интеллектуальная СППР для управления судном / В.Г. Шерстюк, А.П. Бень // Искусственный интеллект. – 2008. – №3. – С.490-500.

7. **Leake D.** Case-based reasoning – experiences, lessons, and future directions /

D. Leake // AAAI Press-MIT Press, 1996. – 318 p.

8. **Варшавский П.Р.** Моделирование рассуждений на основе прецедентов в интеллектуальных системах поддержки принятия решений // П.Р. Варшавский, А.П. Еремеев / Искусственный интеллект и принятие решений. – 2009. – №1. – С.45-57.

9. **Aamodt A.** Case-based reasoning: foundational issues, methodological variations, and system approaches / A. Aamodt, E. Plaza // AI Communications. – 1994. – Vol. 7. – №1. – P.39-59.

10. **Шерстюк В.Г.** Формальная модель гибридной сценарно_прецедентной СППР / В.Г. Шерстюк // Автоматика. Автоматизация. Электротехнические комплексы и системы. – 2004. – Вып. 1. – С.114_122.

11. **Шерстюк В.Г.** Сценарно_прецедентный подход к формированию управляющих воздействий в системе управления морского подвижного объекта / В.Г. Шерстюк // Проблемы информационных технологий. – 2009. – №2(6). – С.69_77.

12. **Pawlak Z.** Rough Sets / Z. Pawlak, W. Jerzy, R. Slowinski, W. Ziarko // Communications of ACM. – 1995. – V.38(11). – pp. 88-95.

13. Нечеткие множества в моделях управления и искусственного интеллекта / Под ред. Д.А.Поспелова. – М., 1986. – 312 с.

14. **Pawlak Z.** Rough Sets, Rough Relations and Rough Functions / Z. Pawlak // Fundamenta Informaticae. – 1996. – V. 27. – №2(3). – p. 103-108.

Надійшла до редколегії 15.12.10