УДК 519.854.33

Рощин В. О., Боярчук Д. О., Ляшко В. І., Шило П. В.

АЛГОРИТМ ГЛОБАЛЬНОГО РІВНОВАЖНОГО ПОШУКУ РОЗВ'ЯЗАННЯ ЗАДАЧІ ПРО ПОКРИТТЯ

У статті проаналізовано кращі відомі алгоритми розв'язання задачі про покриття. Запропоновано і досліджено новий алгоритм, заснований на використанні методів глобального рівноважного пошуку, повторного локального пошуку та адаптивного настроювання повторності. Наведено результати обширного обчислювального експерименту, які показали переваги розробленого алгоритму над кращими відомими алгоритмами.

Ключові слова: задача про покриття, метод глобального рівноважного пошуку, адаптивне настроювання повторності, обчислювальний експеримент, ефективність алгоритму.

Вступ

Відомо, що задачі про покриття множини мають багато практичних застосувань. Серед них слід відзначити задачі, що виникають при розв'язанні проблем оптимального розміщення об'єктів, мінімізації логічних схем і алгоритмів, складанні розкладів, при аналізі й синтезі живучості транспортних мереж. До них відносяться також задачі планування, тестування, пошуку комп'ютерних вірусів, інформації, балансування складальної лінії. Крім того, слід виділити дуже важливі з точки зору практичних застосувань задачі постачання, маршрутизації, планування роботи льотних екіпажів. І, нарешті, досить часто алгоритми розв'язання задачі про покриття входять до складу алгоритмічного забезпечення сучасних комп'ютерних технологій. У [6; 11] наведено огляди із застосування таких задач.

Постановка задачі

Типову задачу про покриття множини (SCP) можна сформулювати так. Припустимо, що задано скінченну множину $E=\{e_1,...,e_m\}$ та сім'ю $S=\{S_1,...,S_n\}$ її підмножин. Зв'яжемо з кожною підмножиною S_j сім'ї S деяку вагу (ціну) c_j. Задача полягає в тому, щоб вибрати деяку кількість підмножин із сім'ї S (підсім'ю F сім'ї S) таким чином, щоб вони утворювали покриття множини E і щоб сумарна вага (вартість) цього покриття була мінімальною (мінімум береться по множині всіх можливих покриттів). Слід зазначити, що F називається покриттям множини E, якщо кожний елемент e_i міститься не менш ніж в одній із підмножин S_j, що входять в F.

Позначимо $A=[a_{ij}]_{m\times n}$ матрицю інциденцій елементів E і підмножин $S_j: a_{ij}=1$, якщо $e_i\in S_j$ і $a_{ij}=0$ при $e_i\notin S_j$. Кожну підсім'ю F сім'ї S задаємо за допомогою вектора x, у якого компонента $x_j=1$, якщо підмножина S_j входить у покриття F, і $x_j=0$ у протилежному випадку.

Тоді задача про покриття множини може бути сформульована як задача цілочислового лінійного програмування вигляду: знайти

$$\min\left\{f(x)=\sum_{j=1}^{n}c_jx_j\right\} \qquad (1)$$

при обмеженнях

$$\sum_{j=1}^{n}a_{ij}x_j\geq 1,\ i\in M=\{1,...,m\}, \qquad (2)$$

$$x_j=0\vee 1,\ j\in N\in\{1,...,n\}, \qquad (3)$$

де $a_{ij}\in\{0,1\}, i\in M, j\in N$.

Для розв'язання цієї задачі можна застосовувати відомі методи цілочислового програмування, зокрема метод гілок і меж із використанням як оцінки оптимального значення цільової функції відповідної неперервної задачі. Однак, метод гілок і меж при розв'язанні задачі вигляду (1)–(3) великої розмірності не приводить до успіху. Специфічна структура задачі (1)–(3) дає можливість розробляти ефективніші, порівняно із загальними, спеціальні методи її розв'язання.

Задача SCP є NP-складною, тому розробка наближених алгоритмів її розв'язання представляє особливий інтерес. Оскільки точні методи потребують значних обчислювальних затрат для розв'язання задач SCP великої розмірності, наближені алгоритми використовуються для швидкого знаходження якісних розв'язків.

Розроблено багато наближених алгоритмів розв'язання задач SCP. У першу чергу слід згадати жадібні алгоритми [12–14; 18]. Незважаючи на швидкість, з їхньою допомогою рідко знаходяться розв'язки хорошої якості.

Для підвищення якості отримуваних розв'язків сучасні наближені методи розробляються на основі методів відпалу (SA) [15], нейронних мереж (NN) [17], генетичних алгоритмів (GA) [5; 8], Meta-RaPS [16]. Крім того, в деяких наближених алгоритмах застосовується лагранжева релаксація початкової задачі [7; 9; 10; 19], що дає їм можливість використовувати інформацію про конкретну задачу.

У роботах [3; 4] запропоновано наближені алгоритми розв'язання задачі про покриття множини мінімальної потужності. Вони показали відмінні результати як за швидкістю отримання, так і за якістю розв'язків. Таким чином для 20 із 70 відомих у світовій літературі тестових задач великої розмірності знайдено нові рекорди, а для інших задач рекорди повторено. Це є особливим успіхом для такого класу задач.

У цій роботі для розв'язання задач вигляду (1)–(3) розроблено наближений алгоритм, що ґрунтується на використанні методу глобального рівноважного пошуку (ГРП). Цей метод добре зарекомендував себе при розв'язанні задач про максимальний розріз графу, багатовимірний рюкзак, квадратичної булевої задачі і багатьох інших [1]. Він вважається одним із кращих сучасних наближених методів дискретної оптимізації.

У схемі запропонованого алгоритму ГРП (GES) як пошуковий використовується алгоритм із роботи [3]. Крім того, застосування лагранжевої релаксації дає можливість отримувати і використовувати інформацію про конкретну розв'язувану задачу так, як це зроблено в [2].

Алгоритм

Представимо запропонований алгоритм розв'язання задачі про покриття у вигляді такої процедури:

procedure GES

1. *initialization_algorithm's_parameters* ($ngen, maxnfail, \mu, K$),

де μ – вектор значень температури, K – кількість температурних стадій,

maxnfail – параметр рестарту, *ngen* – кількість розв'язків, генерованих на одній стадії, $x_{best} \leftarrow 1$ (одиничний вектор), $\tilde{S} \leftarrow \varnothing$ { $\tilde{S}$ – множина знайдених розв'язків};

2. **for** $na = 1$ to *maxna* **do**
3. **if** ($\tilde{S} = \varnothing$) **then**
4. $x \leftarrow construct\_random\_solution$
5. $x_{min} \leftarrow x$
6. $\tilde{S} \leftarrow x_{min}$
7. **end if**
8. $nfail \leftarrow 0$
9. **while** ($nfail < maxnfail$) **do**
10. $x_{oldmin} \leftarrow x_{min}$
11. **for** $k = 0$ to K **do**
12. $calculate\_generation\_probabilities\ (p(\mu_k), \tilde{S})$
13. **for** $g = 0$ to *ngen* **do**
14. $x \leftarrow generate\_solution\ (x_{min}, p(\mu_k))$
15. $R \leftarrow search\_method(x)$
16. $\tilde{S} \leftarrow \tilde{S} \cup R$
17. $x_{good} \leftarrow \arg\min_{x \in \tilde{S}} f(x)$
18. **if** $f(x_{good}) \le f(x_{min})$ **then**
19. $x_{min} \leftarrow x_{good}$
20. **if** $f(x_{min}) \le f(x_{best})$ **then** $x_{best} \leftarrow x_{min}$
21. **end if**
22. **end for**
23. **end for**
24. **if** ($f(old\_x_{min}) = f(x_{min})$) **then** $nfail \leftarrow nfail + 1$

25. **else** $nfail \leftarrow 1$
26. $\tilde{S} = \{x_{min}\}$
27. **end while**
28. $\tilde{S} = \varnothing$
29. **end for**

end GES

K=21, *ngen*=28, *maxnfail*=1.

Зовнішній цикл (рядки 2–29) цієї процедури, в якому проводиться задана кількість повторів, дає можливість повторювати пошук найкращого розв'язку.

«Температурний» цикл (рядки 11–23) є основним в структурі алгоритму ГРП. Для нього необхідно задати значення величин K (кількість виконань циклу) і температури μ_k, $k = 0,...,K$. У циклі проводиться серія стартів пошуку найкращого розв'язку для зростаючих значень температури. Температурний цикл і його повторення дають можливість гнучко чергувати режими звуження і розширення зони пошуку розв'язку, що призводить до високої ефективності методу ГРП.

Припустимо, що $\tilde{S}$ – підмножина множини S допустимих розв'язків задачі (1)–(3), знайдених алгоритмом ГРП, і

$$\tilde{S}_j^1 = \left\{x \,\middle|\, x \in \tilde{S},\ x_j = 1\right\},\ \tilde{S}_j^0 = \left\{x \,\middle|\, x \in \tilde{S},\ x_j = 0\right\},\ j = 1,...,n.$$

Якщо $\tilde{S} = \varnothing$, то з допомогою операторів рядків 4–6 знаходиться перший розв'язок та ініціалізуються необхідні дані.

Компоненти вектора імовірності $p(\mu_k) = (p_1(\mu_k),..., p_n(\mu_k))$ розраховуються згідно з формулою

$$p_j(\mu_k) = \frac{1}{\dfrac{1 - p_j(\mu_0)}{p_j(\mu_0)} \exp\left\{(\mu_k - \mu_0)\right\}\left(f_j^0 - f_j^1\right)}, \quad (4)$$

де $f_j^u = \min\limits_{x \in S_j^u} f(x),\ u \in \{0,1\}$.

Ясно, що імовірність $p(\mu_k)$ генерування початкових розв'язків (рядок 12) для процедури *generate_solution* $(x_{min}, p(\mu_k))$ (рядок 14) обчислюється, виходячи з понять, запозичених у методу відпалу, і залежить від поточної температури μ_k і знайденої раніше множини $\tilde{S}$ допустимих розв'язків.

Значення μ_k, $k = 0,...,K$, що визначають криву відпалу, звичайно обчислюються за формулами $\mu_0 = 0, \mu_{k+1} = \alpha\mu_k$, $k = 1...,K-1$. Величини μ_1 і $\alpha > 1$ підбираються так, щоб $\left\|x_{min} - p^K\right\| \cong 0$, де x_{min} – рекордний розв'язок. Крива відпалу універсальна і не налаштовується на окрему задачу, а використовується при розв'язанні всіх задач. Масштабуються тільки коефіцієнти цільової функції так, щоб значення рекорду дорівнювало заданій величині. Вектор $p_j(\mu_0)$, j=1,...,n, є розв'язком релаксованої початкової задачі (1)–(3). Таким чином, у рамках схеми ГРП досягається налаштування на конкретну розв'язувану задачу.

Так званий покращуючий цикл (рядки 9–27) виконується до тих пір, доки *maxnfail* разів не відбудеться покращення розв'язку x_{min}. Процедура *calculate_generation_probabilities* $(p(\mu_k), \tilde{S})$ (рядок 12) дає можливість обчислювати імовірність випадкового збурення розв'язку x_{min} за формулою (4). Цикл знаходження нових розв'язків (рядки 13–22) повторюється задане число *ngen* разів. За допомогою процедури *generate_solution* $(x_{min}, p(\mu_k))$ випадковим чином генерується розв'язок x, який є початковим для процедури *search_method*(x) (рядок 15). При великих температурах значення компонент, однакових для кращих знайдених розв'язків, змінюються мало. Температурний цикл дає можливість здійснювати диверсифікацію пошуку (при малих температурах) і його інтенсифікацію (при великих температурах). Таким чином, при підвищенні температури генеровані розв'язки набувають ознак, властивих рекордним розв'язкам, і в результаті збігаються до розв'язку x_{min}.

Наведемо процедуру *генерація_розв'язку*:

procedure *generate_solution* $(x_{min}, p(\mu_k))$

1. $x \leftarrow x_{min}$; $j \leftarrow 1$
2. **while** ($j \le n$)
3. **if** $x_j = 1$ **then**
4. **if** ($p_j(\mu_k) <$ *random*[0,1]) **then**
5. $x_j \leftarrow 0$
6. **end if**
7. **end if**
8. **else**
9. **if** ($p_j(\mu_k) \ge$ *random*[0,1]) **then**
10. $x_j \leftarrow 1$
11. **end if**
12. **end else**
13. $j \leftarrow j + 1$
14. **end while.**

У рядку 1 даної процедури присвоюються початкові значення вектору x і змінній j, в циклі рядків 2–14 – збурення вектору x_{min}. Випадкова, рівномірно розподілена на відрізку [0,1] величи-

на *random*[0,1] (рядки 4 і 9) використовується для моделювання випадкових подій, що ведуть до зміни розв'язку x_{min}.

При виборі алгоритму для процедури $search\_method(x)$ потрібно зважати на специфіку розв'язуваної задачі, тому бажано використати алгоритм, що дозволяє інтенсифікувати пошук і ефективно досліджувати області початкового розв'язку для його поліпшення. У процедурі $search\_method(x)$ нами використовувався алгоритм випадкового локального пошуку, запропонований в [3], схема якого наводиться далі.

RandomLocalSearchMG (*F*)

1. Calc(*n_control*, *n_cover*, *F*)
2. **while**(*F* не є покриттям)
3. Формування множини Max_cover
4. Формування множини BestMove
5. $S_j \leftarrow$ RandomSelectElement(BestMove)
6. $F = F \cup S_j$
7. ReCalc(*n_control*, *n_cover*, *F*)
8. Формування множини Redundant
9. **while**(Redundant не порожня)
10. $S_j \leftarrow$ RandomSelectElement(Redundant)
11. $F = F \setminus S_j$
12. ReCalc(*n_control*, *n_cover*, *F*)
13. Формування множини Redundant
14. **end while**
15. Формування множини Cand_Redundant
16. **if** (Cand_Redundant не порожня)
17. $S_j \leftarrow$RandomSelectElement(Cand_Redundant)
18. goto line 6
19. **end if**
20. **end while**
21. **end.**

Розглянемо детальніше цю процедуру. У рядку 1 здійснюється виклик процедури Calc. В ній для підмножин $S_j \in F$ розраховуються величини $n\_control_j$, а для підмножин $S_j \notin F$ – величини n_cover_j. Тут $n\_control_j$ – кількість покритих елементів з *M*, що покриваються тільки підмножиною S_j, а n_cover_j – число непокритих елементів з *M*, що покриваються підмножиною S_j. У рядку 3 формується множина Max_cover, що складається з підмножин S_j з максимальним значенням n_cover_j.

У рядку 4 формується множина BestMove, яка складається з підмножин S_j, що входять у множину Max_cover з максимальним значенням gmod. У рядку 5 випадково вибирається підмножина S_j з множини Max_cover з однаковою імовірністю. Процедура ReCalc викликається в рядках 7, 12. У ній для підмножин $S_j \in F$ перераховуються величини $n\_control_j$, а для підмножин $S_j \notin F$ – величини n_cover_j. У рядках 8 і 13 формується множина Redundant, що складається з підмножин $S_j \in F$, для яких $n\_control_j = 0$. Рядок 15 слугує для формування множини Cand_Redundant, яка складається з підмножин $S_j \notin F$, таких, що $n\_cover_j > 0$, і додавання S_j в *F* приведе до появи непорожньої множини Redundant. Якщо множина Cand_Redundant не порожня, то з неї з однаковою імовірністю випадково вибирається підмножина S_j (рядок 17) і здійснюється перехід на рядок 5. Блок, що складається з рядків 15–19, дає можливість робити ходи, відмінні від ходів жадібного алгоритму.

Алгоритм RandomLocalSearchMG(F) базується на використанні повторного жадібного алгоритму, який на кожному ході (рядки 3–6) намагається максимізувати число покритих елементів. У роботі [3] запропоновано вибирати ходи, що максимізують таку функцію:

$$gmod(F) = L_0 \times numbcover + \sum_{k=0}^{L_{\max}} L_k \times numbcontrol_k,$$

де $numbcover$ – число покритих елементів, а $numbcontrol_k$ – число підмножин $S_j \in F$, які контролюють *k* елементів, $L_{\max} = |F|$, |*F*|–потужність множини *F*. Значення L_k, $k = L_{\max}, .., 1$, вибираються таким чином:

$$L_k = \begin{cases} 1 + L_{k+1} + |F| \times (L_{k+1} - L_{k+2}), \text{ якщо } k \le |F|, \\ 0, \text{ якщо } k > |F|. \end{cases}$$

Оскільки числа L_k можуть бути дуже великими, що незручно при обчисленнях, була використана «відсічена» модифікована функція:

$$L_k = \begin{cases} |F|^{L_{\max} - k} \text{ при } k \le L_{\max}, \ L_{\max} \le |F|. \\ 0 \text{ при } k > L_{\max}. \end{cases}$$

Результати обчислювальних експериментів

Для дослідження ефективності різних наближених алгоритмів проведено обчислювальні експерименти з використанням PC з Intel® Core QUAD CPU Q9550 2.83GHz і 8.0GB оперативної пам'яті. Алгоритм ГРП реалізовано мовою C++. У всіх розрахунках параметри цього алгоритму однакові: $K = 21$, $ngen = 27$, $maxnfail = 1$. На початку кожного температурного циклу як вектор $p_j(\mu_0)$, *j*=1,…,*n*, вибирався розв'язок релаксованої початкової задачі

(1)–(3). Для температурного розкладу використовувалися такі значення: $\mu_0 = 0$, $\mu_1 = 10^{-9}$, $\mu_k = \mu_{k-1}\frac{\log 10 - \log \mu_1}{19}$, $k = 2,...,20$.

Було розв'язано 65 випадково згенерованих тестових задач 4–6, A–D і NRE–NRH великої розмірності, які доступні в OR-library (*http://mscmga.ms.ic.ac.uk/jeb/orlib/scpinfo.html*). Характеристики цих задач наведено в табл. 1, де ρ-щільність матриці $[a_{ij}]_{m\times n}$ (під щільністю розуміємо відношення числа одиничних елементів до загального числа елементів матриці), $c_j \in [1,100]$.

Табл. 2 містить кількість рекордів, одержаних різними алгоритмами, із 65 відомих рекордів.

У табл. 3 наведено результати розв'язання задач вигляду (1)–(3) трьома алгоритмами. В ній прийнято такі позначення: BKS – найкращі відомі рекорди з [16]. Третя–шоста колонки відповідно містять рекорди та час їхнього отримання алгоритмами Greedy та Meta–RaPS[16]. У сьомій колонці наведено рекорди, знайдені запропонованим алгоритмом. Восьма та одинадцята колонки містять відповідно середні значення favr цільової функції та часу timeavr (в сек.) при 100 спробах використання запропонованого алгоритму. У дев'ятій колонці наведено кількість nbest (із 100) знайдених алгоритмом ГРП рекордів. Десята колонка містить найменший час mintime (в сек.) пошуку розв'язку однієї з сотні задач розробленим алгоритмом. У дванадцятій та тринадцятій колонках знаходяться відповідно оптимальне значення цільової функції LPE задачі (1),(2) та час LPEtime (в сек.) його отримання.

Табл. 4 і 5 містять відповідно середні значення відхилень знайдених рекордів від найкращих відомих та середні значення часу розв'язання задач різними алгоритмами.

Таблиця 1. **Характеристики тестових задач**

Задача	Кількість розв'язаних задач	*m*	*n*	r	Інформація про оптимальний розв'язок
4	10	200	1000	2 %	Відомий
5	10	200	2000	2 %	Відомий
6	5	200	1000	5 %	Відомий
A	5	300	3000	2 %	Відомий
B	5	300	3000	5 %	Відомий
C	5	400	4000	2 %	Відомий
D	5	400	4000	5 %	Відомий
NRE	5	500	5000	10 %	Невідомий
NRF	5	500	5000	20 %	Невідомий
NRG	5	1000	10000	2 %	Невідомий
NRH	5	1000	10000	5 %	Невідомий

Таблиця 2. **Кількість знайдених відомих рекордів**

CFT	Meta–RaPS	BeCh	IGA	Be	PROGRES	Greedy	GES
65	65	61	61	22	20	0	65

Таблиця 3. **Результати розв'язання задач алгоритмами Greedy, Meta-RaPS[16] та GES**

Задача	BKS	Greedy	Time	Meta-RaPS	Time	GES	favr	nbest	mintime	timeavr	LPE	LPEtime
1	2	3	4	5	6	7	8	9	10	11	12	13
41	429	439	0	429	1.36	429	429.00	100	0.00	0.00	429	0
42	512	547	0	512	0.24	512	512.00	100	0.00	0.00	512	0.03
43	516	546	0	516	0.29	516	516.00	100	0.00	0.00	516	0
44	494	510	0	494	0.39	494	494.07	93	0.00	0.59	494	0
45	512	519	0	512	0.9	512	512.00	100	0.00	0.00	512	0
46	560	594	0	560	0.1	560	560.00	100	0.00	0.00	557.25	0
47	430	447	0	430	0.04	430	430.00	100	0.00	0.00	430	0
48	492	502	0	492	1.46	492	492.30	70	0.00	0.63	488.67	0.03
49	641	672	0	641	3.47	641	641.00	100	0.00	0.00	638.54	0.03
410	514	521	0	514	0.08	514	514.00	100	0.00	0.00	513.5	0
51	253	271	0	253	1.55	253	253.00	100	0.00	0.08	251.23	0

Закінчення таблиці 3. **Результати розв'язання задач алгоритмами Greedy, Meta-RaPS[16] та GES**

Задача	BKS	Greedy	Time	Meta-RaPS	Time	GES	favr	nbest	mintime	timeavr	LPE	LPEtime
1	2	3	4	5	6	7	8	9	10	11	12	13
52	302	329	0	302	0.59	302	302.00	100	0.00	0.09	299.76	0
53	226	232	0.01	226	1.14	226	226.00	100	0.00	0.00	226	0
54	242	253	0	242	0.32	242	242.00	100	0.00	0.02	240.5	0.03
55	211	220	0	211	0.33	211	211.00	100	0.00	0.00	211	0
56	213	234	0	213	0.14	213	213.00	100	0.00	0.00	212.5	0.03
57	293	302	0	293	1.03	293	293.00	100	0.00	0.05	291.78	0
58	288	308	0.01	288	0.08	288	288.00	100	0.00	0.00	287	0.05
59	279	290	0	279	0.04	279	279.00	100	0.00	0.00	279	0.03
510	265	275	0	265	0.03	265	265.00	100	0.00	0.00	265	0
61	138	147	0	138	0.25	138	138.50	75	0.05	0.91	133.14	0.05
62	146	160	0	146	0.02	146	146.00	100	0.00	0.09	140.46	0
63	145	152	0	145	0.02	145	145.00	100	0.00	0.00	140.13	0
64	131	137	0	131	0.34	131	131.00	100	0.00	0.00	129	0
65	161	178	0	161	1.02	161	161.00	100	0.00	0.01	153.35	0
a1	253	271	0	253	6.22	253	253.47	53	0.33	3.63	246.84	0.16
a2	252	267	0	252	0.28	252	252.00	100	0.03	1.63	247.5	0.14
a3	232	244	0	232	16.94	232	232.00	100	0.00	1.40	228	0.13
a4	234	246	0	234	0.04	234	234.00	100	0.00	0.03	231.4	0.16
a5	236	247	0.01	236	9.37	236	236.00	100	0.00	0.98	234.89	0.2
b1	69	73	0	69	0.14	69	69.00	100	0.00	0.02	64.54	0.19
b2	76	78	0	76	0.53	76	76.00	100	0.00	0.02	69.31	0.22
b3	80	85	0	80	0.62	80	80.00	100	0.02	1.08	74.16	0.17
b4	79	85	0	79	2.25	79	79.00	100	0.00	0.20	71.22	0.22
b5	72	76	0	72	0	72	72.00	100	0.00	0.01	67.67	0.2
c1	227	246	0	227	0.43	227	227.00	100	0.05	1.32	223.8	0.34
c2	219	231	0.02	219	12.89	219	219.00	100	0.03	1.87	212.85	0.36
c3	243	256	0	243	26.24	243	243.00	100	0.06	1.28	234.58	0.38
c4	219	239	0	219	24.29	219	219.14	90	0.05	6.06	213.85	0.36
c5	215	228	0.01	215	1.79	215	215.00	100	0.00	0.17	211.64	0.34
d1	60	68	0.01	60	3.13	60	60.00	100	0.01	0.13	55.31	0.45
d2	66	70	0	66	13.59	66	66.00	100	0.00	0.12	59.35	0.61
d3	72	78	0.02	72	1.31	72	72.00	100	0.00	0.17	65.07	0.42
d4	62	65	0	62	0.2	62	62.00	100	0.00	0.32	55.84	0.53
d5	61	71	0.01	61	0.29	61	61.00	100	0.00	0.01	58.62	0.45
nre1	29	31	0.02	29	0.73	29	29.00	100	0.00	0.10	21.38	1.53
nre2	30	34	0	30	46.17	30	30.00	100	0.02	5.32	22.36	1.47
nre3	27	32	0	27	5.95	27	27.00	100	0.00	0.34	20.49	1.44
nre4	28	32	0.01	28	39.64	28	28.00	100	0.00	0.66	21.35	1.42
nre5	28	31	0	28	0.81	28	28.00	100	0.00	0.06	21.32	1.49
nrf1	14	17	0.02	14	4.29	14	14.00	100	0.02	0.79	8.81	2.81
nrf2	15	16	0.02	15	3.8	15	15.00	100	0.00	0.31	9.99	3.77
nrf3	14	16	0.01	14	1.84	14	14.00	100	0.05	0.99	9.49	3.41
nrf4	14	15	0.01	14	5.44	14	14.00	100	0.02	1.19	8.47	2.92
nrf5	13	15	0.01	13	33.27	13	13.80	20	5.03	7.43	7.84	3.03
nrg1	176	194	0.03	176	298.97	176	176.00	100	0.72	9.65	159.89	5.92
nrg2	154	165	0.03	154	222.34	154	155.09	9	1.98	30.39	142.07	5.72
nrg3	166	179	0.01	166	21.56	166	167.26	7	3.39	33.57	148.27	6.36
nrg4	168	184	0.03	168	194.21	168	169.26	21	4.42	30.71	148.95	6.14
nrg5	168	181	0.02	168	47.57	168	168.01	99	1.50	17.71	148.23	5.92
nrh1	63	71	0.04	63	3917.08	63	63.93	7	17.43	7.96	48.13	8.3
nrh2	63	69	0.03	63	238.45	63	63.28	72	2.63	31.85	48.64	8.86
nrh3	59	65	0.05	59	783.2	59	59.56	46	2.67	37.68	45.2	8.08
nrh4	58	66	0.03	58	1358.28	58	58.00	100	2.31	16.10	44.04	7.77
nrh5	55	62	0.04	55	5.62	55	55.00	100	0.08	2.17	42.37	8.56

Таблиця 4. **Результати розв'язання задач алгоритмами Greedy, Meta–RaPS[16] та GES**

Алгоритм / Задача	CFT	Meta–RaPS	GES	BeCh	IGA	PROGRES	Be	Greedy
4	0.00	0.00	0.00	0.00	0.00	0.57	0.06	3.78
5	0.00	0.00	0.00	0.09	0.00	0.88	0.18	5.51
6	0.00	0.00	0.00	0.00	0.00	0.69	0.56	7.22
A	0.00	0.00	0.00	0.00	0.00	0.75	0.82	5.61
B	0.00	0.00	0.00	0.00	0.00	0.00	0.81	5.57
C	0.00	0.00	0.00	0.00	0.00	0.87	1.93	6.88
D	0.00	0.00	0.00	0.00	0.32	0.00	2.75	9.79
NRE	0.00	0.00	0.00	0.00	0.00	0.00	3.5	12.75
NRF	0.00	0.00	0.00	0.00	0.00	1.43	7.16	12.98
NRG	0.00	0.00	0.00	0.13	0.13	1.18	4.83	8.49
NRH	0.00	0.00	0.00	0.63	1.30	1.68	8.12	11.78
Разом	0.00	0.00	0.00	0.08	0.16	0.72	2.36	8.21

Таблиця 5. **Порівняння середнього часу (в сек.) розв'язання задач**

Алгоритм / Задача	CFT (DECstation 5000/240)	Meta–RaPS (Intel PIV 1.7 GHz)	BeCh (Graphics Indigo (R4000, 100))	GES (Intel QUAD Q9550 2.83GHz)
4	6.5	0.83	57.59	0.12
5	3.2	0.58	190.87	0.02
6	9.4	0.33	20.21	0.20
A	106.6	6.57	52.79	1.53
B	7.4	0.72	54.9	0.27
C	66	13.13	70.38	2.14
D	17.2	3.70	81.41	0.15
NRE	118.2	18.66	3082.55	1.30
NRF	109	9.73	976.90	2.14
NRG	504.8	156.93	4540.83	24.41
NRH	858.2	1260.53	2240.70	19.15
Разом	164	133.79	1033.5	4.3

Висновки

Слід зазначити, що розглянуті тестові задачі мають рідко заповнені матриці обмежень. Вони набагато складніші для розв'язання, ніж задачі з великою щільністю матриць. Аналіз результатів експериментальних розрахунків показав, що метод ГРП з успіхом вдалося поширити на новий клас задач про покриття множини. Розроблений алгоритм ГРП продемонстрував свої переваги щодо більшості порівнюваних алгоритмів. Це стосується, зокрема, можливості та незначного часу отримання відомих рекордів для розглянутої множини тестових задач.

Список літератури

1. Сергиенко И. В. Проблемы дискретной оптимизации: сложные задачи, основные подходы к их решению / И. В. Сергиенко, В. П. Шило // Кибернетика и системный анализ. – 2006.– № 4. – С. 3–25.
2. Шило В. П. Решение многомерных задач о ранце методом глобального равновесного поиска / В. П. Шило // Теория оптимальных решений. – К. : Ин-т кибернетики им. В. М. Глушкова НАН Украины, 2000. – С. 10–13.
3. Шило В. П. Решение задачи о покрытии минимальной мощности / В. П. Шило // Компьютерная математика. – 2013. – Вып. 2. – С. 152–160.
4. Шило П. В. Применение «бесполезных» ходов при решении задачи о покрытии / П. В. Шило // Компьютерная математика. – 2014. – Вып. 1. – С. 150–158.
5. Aickelin U. An indirect genetic algorithm for set covering problems / U. Aickelin // Journal of the Operational Research Society. – 2002. – Vol. 53. – P. 1118–1126.
6. Balas E. A class of location, distribution and scheduling problems: Modeling and solution methods / E. Balas // Proceedings of the Chinese–US Symposium on System Analysis ; P. Gray, L. Yuanzhang (ed.). – 1983. – J. Wiley and Sons. – New York. – P. 36–65.
7. Beasley J. E. A Lagrangian heuristic for set covering problems / J. E. Beasley. – 1990. – Naval Research Logistics- 37. – P. 151–164.
8. Beasley J. E. A genetic algorithm for the set covering problem / J. E. Beasley, P. C. Chu // European Journal of Operational Research. – 1996. – Vol. 94. – P. 392–404.
9. Caprara A. A heuristic method for the set covering problem / A. Caprara, M. Fischetti, P. Toth // Operations Research. – 1999. – Vol. 47 (5). – P. 730–743.
10. Ceria S. A Lagrangian-based heuristic for large-scale set covering problems / S. Ceria, P. Nobili, A. Sassano // Mathematical Programming. – 1998. – Vol. 81. – P. 215–228.
11. Ceria S. Set Covering Problem / S. Ceria, P. Nobili, A. Sassano // Annotated Bibliographies in Combinatorial Optimization ; M. DellÁmico, F. Maffioli, and S. Martello (Eds). – New York : J. Wiley and Sons, 1998. – P. 415–428.

12. Chvatal V. A greedy heuristic for the set covering problem / V. Chvatal // Mathematics of Operations Research. – 1979. – Vol. 4. – P. 233–235.
13. Feo T. A probabilistic heuristic for a computationally diffcult set covering problem / T. Feo, M. G. C. Resende // Operations Research Letters. – 1989. – Vol. 8. – P. 67–71.
14. Haouari M. A probabilistic greedy search algorithm for combinatorial optimization with application to the set covering problem / M. Haouari, J. S. Chaouachi // Journal of the Operational Research Society. – 2002. – Vol. 53. – P. 792–799.
15. Jacobs L Note: A local-search heuristic for large set–covering problems / L. Jacobs, M. Brusco // Naval Research Logistics. – 1995. – Vol. 42. – P. 1129–1140.
16. Lan G. An effective and simple heuristic for the set covering problem / G. Lan, G. W. DePuy, G. E. Whitehouse // European Journal of Operational Research. – 2007. – Vol. 176. – P. 1387–1403.
17. Ohlsson M. An effcient mean field approach to the set covering problem / M. Ohlsson, C. Peterson, B. Soderberg // European Journal of Operational Research. – 2001. – Vol. 133. – P. 583–595.
18. Vasko F. J. An effcient heuristic for large set covering problems / F. J. Vasko, G. R. Wilson // Naval Research Logistics Quarterly. – 1984. – Vol. 31. – P. 163–171.
19. Yelbay B. The Set Covering Problem Revisited: An Empirical Study of the Value of Dual Information [Electronic resource] / B. Yelbay, S. I. Birbil, K. Bulbul. – November 14, 2010. – Mode of access: http://www.optimization-online.org/DB_FILE/2010/11/2814.pdf. – Title from the screen.

V. Roschyn, D. Boyarchuk, V. Lyashko, P. Shylo

SOLVING SET COVERING PROBLEM BY GLOBAL EQUILIBRIUM SEARCH

Best known algorithms for solving the set covering problem were analyzed. A new algorithm based on the global equilibrium search method and iterative local search with adaptive iterative tuning is proposed and studied. The results of extensive computational experiments demonstrate the advantages of the proposed algorithm over best known algorithms.

Keywords: Set covering problem, global equilibrium search method, adaptive iterative tuning, computational experiment, efficiency of algorithm.

Матеріал надійшов 21.05.2014

УДК 004.896

Мейтус В. Ю.

СТВОРЕННЯ НАПІВІНТЕЛЕКТУАЛЬНИХ КОМП'ЮТЕРНИХ СИСТЕМ: ОСНОВНІ ПРОБЛЕМИ

У роботі розглянуто проблеми, пов'язані зі створенням напівінтелектуальних комп'ютерних систем. Основне завдання таких систем полягає в адаптації їхньої поведінки до навколишнього середовища, з яким вони взаємодіють. Для реалізації адаптації така система використовує закладені в ній алгоритми аналізу та синтезу (на відміну від інтелектуальних). Поведінка системи визначається в процесі рішення послідовності задач, пов'язаних з динамічно змінюваним зовнішнім середовищем. А інтелект проявляється у правильному розумінні і вирішенні цих задач з урахуванням можливих змін навколишнього оточення. Проблема створення напівінтелектуальних систем полягає в розробці методів відстеження змін в оточенні системи та обліку цих змін при формуванні її поведінки.

Ключові слова: комп'ютерні системи, інтелект, модель предметної області, напівінтелектуальні системи, технології.

Один з найважливіших напрямів використання комп'ютерів у сучасних умовах – це комп'ютеризація складних систем, у яких комп'ютерам відводяться суттєві завдання подання та обробки інформації, пов'язаної з роботою таких систем. Незалежно від того, яка сис-