УДК 681.3.06

АЛГОРИТМИ СИНТЕЗУ ОПТИМАЛЬНОЇ СИСТЕМИ ЗАХИСТУ ІНФОРМАЦІЇ

Ю.Ю. БОНЯ, О.М. НОВІКОВ

Розроблено математичну модель залежності середніх втрат при порушенні безпеки в інформаційно-телекомунікаційних системах з відкритою архітектурою для випадку використання не абсолютно надійних засобів захисту. Сформульовано задачу математичного програмування, яка виникає при синтезі економічно ефективної системи захисту. Розроблено алгоритм розв'язання задачі цілочисельного сепарабельного програмування з використанням локально-стохастичних алгоритмів. Проведено дослідження його працездатності та ефективності. Використання розробленого алгоритму в якості складової спеціалізованих систем автоматизованого проектування дозволяє прискорити синтез систем захисту інформації з мінімальною вартістю засобів захисту в інформаційно-телекомунікаційних системах з відкритою архітектурою.

ВСТУП

Проблема розробки формалізованих методик синтезу систем захисту інформації (СЗІ) в інформаційно-телекомунікаційних системах (ІТС) з відкритою архітектурою є актуальною з ряду причин. Розповсюджуються технології систем з відкритою архітектурою, бракує методик аналізу ризиків, пов'язаних з успішною реалізацією загроз інформації. Такі методики стають все важливішими. Слабка формалізація відомих методик аналізу ризиків та синтезу СЗІ, оптимальних за певними критеріями, не дозволяє широко застосовувати автоматизовані системи проектування СЗІ. Подолати ці перешкоди допоможе розробка адекватної математичної моделі оцінки ризиків і супутні алгоритми та методики.

Використання логіко-ймовірнісного математичного апарату, що застосовувався при дослідженні безпеки структурно складних систем, дозволяє описувати порушення безпеки і в ІТС з відкритою архітектурою [1]. Якщо йдеться про розробку автоматизованої системи проектування СЗІ, то поряд з моделлю ризику, що виникає при реалізації загроз, слід також розглядати конкретну постановку задачі оптимізації. Відомі декілька постановок таких задач, які відрізняються видом обмежень та критеріїв оптимізації. Найважливішими з них є мінімізація ймовірності порушення безпеки при накладанні обмежень на вартість засобів захисту та мінімізація вартості системи при

накладанні обмежень на величину ймовірності порушення чи величину ризику. Залежно від виду критеріїв та обмежень відповідна задача може бути використана для синтезу СЗІ в ІТС як державних структур, так і комерційних. Для останніх може бути характерним обмеження на бюджет витрат на СЗІ і мінімізація цих витрат. Для перших обмежуючим фактором є неможливість визначити втрати від реалізації загроз, оскільки їх матеріальний вираз не завжди можливий, а відомою є тільки ймовірність цих подій.

У роботі [2] було запропоновано задачу математичного програмування, яка виникає при синтезі СЗІ із мінімальною вартістю в ІТС із відкритою архітектурою за умови обмеження на величину залишкового ризику. Узагальнимо згадану задачу, ввівши до розгляду коефіцієнт ефективності $0 \le \alpha_{ij} \le 1$ i-го механізму захисту на j-му рівні стеку протоколів, та зробимо відповідні зміни у виразах для величини ризику.

Враховуючи ту особливість відкритих систем, що реалізація засобу захисту на нижчому рівні стеку гарантує захист на всіх вищих рівнях, ймовірність порушення безпеки на рівнях стеку $\{i_1, i_2, ..., i_s\} \subset W$ у результаті дії однієї загрози дорівнює

$$P_{i_1,\ldots,i_s} = \prod_{i_k \in \{i_1,\ldots,i_s\}} p_{i_k} \prod_{i_k \in W \setminus \{i_1,\ldots,i_s\}} (1 - p_{i_k}) \prod_{k=1}^{\min(i_1,\ldots,i_s)} (1 - q_k),$$

де $W = \{1, 2, \ldots, N\}$, N — кількість рівнів стеку протоколів; p_i — ймовірність спричинення збитків на i-му рівні через реалізацію загрози; q_i — ймовірність виключити негативну дію загрози на i-му рівні.

Для стеку з N рівнями середні втрати від однієї загрози можуть бути отримані у вигляді

$$Q = \sum_{s=1}^{N} \sum_{\substack{i_1, i_2, \ldots, i_s \subset W \\ i_1 < i_2 < \ldots < i_s}} P_{i_1, i_2, \ldots, i_s} \sum_{k=1}^{s} Q_{i_k},$$

де коефіцієнт Q_i — втрати від реалізації загрози на i-му рівні. Після спрощень та підстановок вираз Q можна записати

$$Q = \sum_{j=1}^{N} R_j \prod_{k=1}^{j-1} (1 - R_k) \prod_{k=1}^{j} (1 - \alpha_k M_k) \left(Q_j + \sum_{k=j+1}^{N} R_k Q_k \right), \qquad (1)$$

де зроблені позначення $R_i = p_i$ та $\alpha_i M_i = q_i$. Тут R_i — ймовірність реалізації загрози на i-му рівні; $\alpha_i = \alpha_{si}$ — коефіцієнт ефективності засобу захисту на i-му рівні стеку для деякої s-ї загрози, а бульова змінна M_i визначає, чи реалізується механізм захисту на i-му рівні стеку.

Через адитивність ризику перехід до випадку, коли розглядається L незалежних загроз, проводимо шляхом додавання у співвідношенні (1) до змінних індексу i номера загрози та підсумування за цим індексом.

Остаточний вигляд нової задачі математичного програмування буде таким:

$$\begin{cases} \sum_{i=1}^{L}\sum_{j=1}^{N} R_{ij}\prod_{k=1}^{j-1}(1-R_{ik})\prod_{k=1}^{j}\left(1-\alpha_{ik}M_{ik}\right)\left(Q_{ij}+\sum_{k=j+1}^{N}R_{ik}Q_{ik}\right)\le R_{\max}, \\ \sum_{i=1}^{L}\sum_{j=1}^{N} M_{ij}C_{ij}\to\min\limits_{M_{ij}\in\{0,1\}}, \end{cases} \tag{2}$$

де R_{ij} — ймовірність реалізації i-ї загрози на j-му рівні стеку; Q_{ij} — величина збитків при реалізації i-ї загрози на j-му рівні стеку; $R_{\max}$ — верхня границя втрат від порушення безпеки при реалізації L загроз; C_{ij} — вартість реалізації i-го механізму захисту на j-му рівні стеку.

ЕКВІВАЛЕНТНА ЗАДАЧА СЕПАРАБЕЛЬНОГО ПРОГРАМУВАННЯ

У загальному випадку задачу (2) можна звести до еквівалентної задачі сепарабельного програмування. Для початку будемо кодувати комбінації механізмів, які нейтралізують одну й ту саму i-ту загрозу у вигляді цілого числа

$$x_i=\sum_{j=1}^{N}2^{j-1}M_{ij}\in\{0,\ldots,2^N-1\}.$$

Тоді вирази для вартості системи і середніх втрат від порушення безпеки можна записати у вигляді $\sum_{i=1}^{L}F_i(x_i)$ та $\sum_{i=1}^{L}G_i(x_i)$ відповідно. Функції F_i, та G_i задаються таблично та залежать від цілочисельної змінної x_i неявним чином. Для їх обчислення необхідно декодувати значення змінної x_i та підставити отримані бульові значення у відповідні вирази задачі (2) для фіксованого i.

$$\begin{cases} F_i(x_i)=\sum_{j=1}^{N}M_{ij}C_{ij}, \quad x_i=0,\ldots,2^N-1, \\ G_i(x_i)=\sum_{j=1}^{N}R_{ij}\prod_{k=1}^{j-1}(1-R_{ik})\prod_{k=1}^{j}\left(1-\alpha_{ik}M_{ik}\right)\left(Q_{ij}+\sum_{k=j+1}^{N}R_{ik}Q_{ik}\right), \\ \qquad\qquad x_i=0,\ldots,2^N-1. \end{cases}$$

Задача у цьому випадку має вигляд

$$\min_{x_i\in\{0,\ldots,2^N-1\}}\left\{\sum_{i=1}^{L}F_i(x_i)\,\middle|\,\sum_{i=1}^{L}G_i(x_i)\le R_{\max}\right\} \tag{3}$$

та може розв'язуватися за допомогою методу послідовного аналізу варіантів (ПАВ) [3] або з використанням методу вектора спаду для задачі сепарабельного програмування [4].

Слід відзначити, що також легко розглянути випадки, коли деякі комбінації змінних M_{ij} для фіксованого i є забороненими (наприклад, коли

засіб не може бути реалізованим на певних рівнях стеку протоколів). У цьому випадку з усіх можливих 2^N значень x_i деякі потрібно відкинути, а решту занумерувати наново. Отримана задача запишеться як

$$\min_{x_i \in X_i} \left\{ \sum_{i=1}^{L} F_i'(x_i) \,\middle|\, \sum_{i=1}^{L} G_i'(x_i) \le R_{\max},\ X_i = \{0,\dots,l_i\},\ l_i < 2^N - 1 \right\},$$

де функції F_i' та G_i' задаються наново для всіх значень $x_i = 0,\dots,l_i$, що залишилися.

Окрім розглянутого вище загального випадку, коли ми не робимо ніяких припущень щодо значень коефіцієнтів міцності засобів захисту та можливих їх комбінацій, реалізованих на різних рівнях, існують ситуації, коли задача (2) стає лінійною.

ЛІНЕАРИЗАЦІЯ ЗАДАЧІ

Розглянемо ситуацію, коли для нейтралізації загрози обирається реалізація засобу захисту на єдиному рівні, або ми взагалі відмовляємося від його використання. Для випадку абсолютно надійних засобів захисту тільки серед таких конфігурацій СЗІ слід шукати оптимальну, а всі ті, в яких реалізація засобу відбувається на двох і більше рівнях, мають більшу вартість при тому самому значенні ризику. У цьому випадку задача (2) насправді буде лінійною [2]. Із введенням неодиничних коефіцієнтів ефективності засобів захисту їх дублювання на різних рівнях стеку протоколів вже буде виправданим, і для приведення задачі (2) до лінійного вигляду необхідно примусово накласти обмеження на структуру СЗІ, тоді для i-ї загрози справджується $\sum_{j=1}^{n} M_{ij} \le 1,\ 1 \le n \le N$, а разом з тим і $\prod_{j=1}^{n}(1-\alpha_{ij}M_{ij}) = 1 - \sum_{j=1}^{n}\alpha_{ij}M_{ij}$, бо всі перехресні добутки зникають. Отже, у цьому випадку обмеження на значення середніх втрат буде лінійним.

$$\sum_{i=1}^{L}\sum_{j=1}^{N} R_{ij} \prod_{k=1}^{j-1}(1-R_{ik}) \sum_{k=1}^{j} \alpha_{ik} M_{ik} \left(Q_{ij} + \sum_{k=j+1}^{N} R_{ik} Q_{ik} \right) \ge$$

$$\ge \sum_{i=1}^{L}\sum_{j=1}^{N} R_{ij} \prod_{k=1}^{j-1}(1-R_{ik}) \left(Q_{ij} + \sum_{k=j+1}^{N} R_{ik} Q_{ik} \right) - R_{\max}.$$

Після накладання додаткового обмеження на структуру розв'язку і тотожних перетворень вихідну задачу теж можна записати у лінійному вигляді.

$$\begin{cases} \sum_{i=1}^{L}\sum_{j=1}^{N} M_{ij} C_{ij} \to \min\limits_{M_{ij} \in \{0,1\}},\ \sum\limits_{j=1}^{N} M_{ij} \le 1,\ i = \overline{1, L}, \\ \sum_{i=1}^{L}\sum_{j=1}^{N} M_{ij} \left(\alpha_{ij} \sum\limits_{k=j}^{N} R_{ik} Q_{ik} \prod\limits_{k=1}^{j-1}(1-R_{ik}) \right) \ge \sum\limits_{i=1}^{L}\sum\limits_{j=1}^{N} R_{ij} Q_{ij} - R_{\max}. \end{cases} \quad (4)$$

При покладанні всіх $\alpha_{ij} = 1$ (тобто у випадку абсолютно надійних засобів захисту) задача (4) зводиться до задачі лінійного бульового програмування (ЛБП) з роботи [2].

Задачу (4) можна розв'язувати за допомогою точних методів бульового лінійного програмування (методу Балаша з вдосконаленням Джеофріона [5], методу ПАВ [3], *P*-методу [6, 7] та ін.) і методів локальної оптимізації (методу вектора спаду, локально-стохастичних алгоритмів, методу керованого випадкового пошуку [6, 8]).

Якщо задача (4) не розв'язується прямо, а відбувається перехід до еквівалентної задачі сепарабельного програмування, то функції F_i, та G_i з (3) виражаються наступним чином:

$$F_i(x_i) = \begin{cases} C_{i,x_i+1}, & x_i = 0,\ldots,N-1, \\ 0, & x_i = N, \end{cases}$$

$$G_i(x_i) = \begin{cases} \sum\limits_{j=1}^{N} R_{ij}Q_{ij} - \alpha_{i,x_i+1} \sum\limits_{k=x_i+1}^{N} R_{ik}Q_{ik} \prod\limits_{k=1}^{x_i} (1-R_{ik}), & x_i = 0,\ldots,N-1, \\ \sum\limits_{j=1}^{N} R_{ij}Q_{ij}, & x_i = N, \end{cases}$$

де значення змінної $x_i = 0,\ldots,N-1$ відповідають використанню i-го засобу захисту на $1,\ldots,N$-му рівнях стеку, а значення $x_i = N$ — відмові від використання.

РОЗВ'ЯЗАННЯ ЗАДАЧІ СЕПАРАБЕЛЬНОГО ПРОГРАМУВАННЯ ЗА ДОПОМОГОЮ ЛОКАЛЬНО-СТОХАСТИЧНИХ АЛГОРИТМІВ

У роботі [2] для розв'язання задачі (3) запропоновано використати метод вектора спаду, що належить до класу методів локальної оптимізації. Окрім оригінального методу вектора спаду також існує ряд інших локально-стохастичних методів [8], тому пропонується використати їх ідеї для подальшого вдосконалення алгоритму розв'язання задачі. Важливу роль при роботі локально-стохастичних алгоритмів грає процедура генерування випадкової перестановки, яка наводиться нижче.

Алгоритм генерування випадкової перестановки Π_α

1. Формуємо початкову послідовність індексів $J_0 = (j_1^0,\ldots,j_n^0)$ і задаємо деяке значення цілочисельного параметра α.

2. Покладаємо $k = n$.

3. За допомогою генератора псевдовипадкових чисел, рівномірно розподілених на інтервалі $(0, 1)$, отримуємо випадкове число ξ і обчислюємо $i = [(1-\xi^\alpha)k + 1]$.

4. Міняємо місцями i-й і k-й елементи послідовності J_{n-k}, отримуючи послідовність J_{n-k+1}.

5. Замінюємо k на $k-1$ і при $k>1$ переходимо до п.3, а при $k=1$ повертаємо випадкову перестановку $J = J_{n-1}$.

Якщо розглядати метод вектора спаду з радіусом околу $r=1$, то всім змінним можна надати деякий пріоритет, який буде відображати їх «перспективність» з точки зору руху в напрямку збільшення (для задачі вигляду (3)). Евристичне правило для обчислення таких пріоритетних коефіцієнтів для j-ї змінної можна записати так:

$$v_j = (F_j(x_j^0) - F_j(x_j^0+1)) \big/ (G_j(x_j^0+1) - G_j(x_j^0)).$$

Для випадку $x_j^0 = N$ подальший рух є неможливим, тому ця змінна не враховується при обчисленні пріоритетних коефіцієнтів. Випадок $G_j(x_j^0+1) = G_j(x_j^0)$ є нетиповим, бо зазвичай механізм, реалізований на більш високому рівні стеку, забезпечує менший рівень захисту.

Процедура Π_α має на меті зробити перебір напрямків зміни координат випадковим, не змінюючи суттєво порядок, що визначається за допомогою пріоритетних коефіцієнтів для відповідних змінних. Із збільшенням параметра α отримана перестановка J наближається до J_0, а при $\alpha = 1$ пріоритет не враховується взагалі і перестановка стає випадковою. Із врахуванням особливостей задачі алгоритм циклічного координатного підйому можна описати таким чином.

Алгоритм циклічного координатного підйому

1. У якості початкового наближення x^0 обираємо або нульовий вектор, або інший припустимий розв'язок задачі. Покладаємо $R=0$ і фіксуємо деяке значення параметра α.

2. Обчислюємо за формулою $v_j = (F_j(x_j^0) - F_j(x_j^0+1)) \big/ (G_j(x_j^0+1) - G_j(x_j^0))$ пріоритетні коефіцієнти v_j, $j = 1,\dots,L$ і будуємо адаптаційну послідовність $J_0 = (j_1^0,\dots,j_L^0)$ таку, що $v_{j_1^0} \ge \dots \ge v_{j_L^0}$. Виключаємо з J_0 всі індекси, які відповідають $x_i^0 = N$ (максимальному значенню змінної). Нехай кількість індексів після цієї процедури скоротилася з L до K.

3. За допомогою процедури Π_α генеруємо випадкову перестановку $J = (j_1,\dots,j_K)$, $K \le L$ елементів послідовності J_0.

4. Виходимо з точки x^0 та послідовно у порядку $j_1,\dots,j_K$ визначаємо координати припустимої точки з околу радіусу $r=1$, що поліпшує значення цільової функції.

5. Покладаємо $h=1$ та $x = x^0$.

6. Якщо $R - G_{j_h}(x_{j_h}) + G_{j_h}(x_{j_h}+1) \le R_{\max}$, то покладаємо $R = R - G_{j_h}(x_{j_h}) + G_{j_h}(x_{j_h}+1)$, $x_{j_h} = x_{j_h}+1$.

7. Якщо $h < K$, то замінюємо h на $h+1$ і переходимо до п.6.

8. Якщо $x \neq x^0$, то покладаємо $x^0 = x$ і переходимо до п.2. Інакше покладаємо $x^* = x$.

9. Повертаємо точку локального мінімуму x^* цільової функції відносно околу радіусу $r = 1$.

У роботі [8] також запропоновано алгоритм бікоординатного підйому, що є локально-стохастичним алгоритмом пошуку мінімуму в околі з радіусом $r = 2$. Для задачі, що розглядається, алгоритм бікоординатного підйому можна викласти у такому вигляді.

Алгоритм бікоординатного підйому

1. За допомогою алгоритму циклічного координатного підйому знаходимо точку x мінімуму цільової функції відносно околу одиничного радіусу. Обчислюємо значення $R = \sum_{i=0}^{L} G_i(x_i)$.

2. Розглянемо дві множини U та V, які визначаються таким чином: $U = \{i \mid x_i < N\}$, $V = \{i \mid x_i > 0\}$. Позначимо $N_U = |U|$ і $N_V = |V|$.

3. Формуємо множину Z, яка складається з трійок $\{z_i, f_i, q_i\}$, де $f_i = F_{z_i}(x_{z_i}) - F_{z_i}(x_{z_i+1})$, $q_i = '+'$ для $z_i \in U$ і $f_i = F_{z_i}(x_{z_i-1}) - F_{z_i}(x_{z_i})$, $q_i = '-'$ для $z_i \in V$. Впорядковуємо елементи множини Z за зростанням f_i.

4. Покладаємо $\bar{x} = x$.

5. Покладаємо $l = \min\{j \mid q_j = '-'\}$, $k = \max\{j \mid q_j = '+'\}$.

6. Якщо $k < l$, то x є точкою мінімуму цільової функції відносно околу радіуса два, і робота припиняється.

7. Маємо $k > l$. Якщо $z_k = z_l$, то переходимо до п.9.

8. Якщо виконується $R + G_{z_k}(x_{z_k}+1) - G_{z_k}(x_{z_k}) + G_{z_l}(x_{z_l}-1) - G_{z_l}(x_{z_l}) < R_{\max}$, то покладаємо $x_{z_k} = x_{z_k} + 1$, $x_{z_l} = x_{z_l} - 1$ і переходимо до п.2. Інакше — до наступного пункту.

9. Якщо $W_U = \{j \mid q_j = '+',\ j < k\} \neq \varnothing$, то покладаємо $k = \max\limits_{j \in W_U} j$ і переходимо до п.11. Інакше — до п.10.

10. Якщо $W_V = \{j \mid q_j = '-',\ j > l\} \neq \varnothing$, то покладаємо $l = \min\limits_{j \in W_V} j$, $k = \max\{j \mid q_j = '+'\}$ і переходимо до п.11. Інакше — до п.12.

11. Якщо $k < l$, то переходимо до п.12. Інакше — до п.7.

12. Якщо $x = \bar{x}$, то точка x є шуканою точкою локального мінімуму відносно околу радіуса два. Інакше за допомогою алгоритму циклічного координатного підйому, виходячи з точки x як початкової, знаходимо локально-оптимальний розв'язок x'. Якщо $x' = x$, то x є шуканою точкою локального мінімуму відносно околу радіуса два. Інакше переходимо до п.2, покладаючи $x = x'$.

ЗАСТОСУВАННЯ *P*-МЕТОДУ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ БУЛЬОВОГО ПРОГРАМУВАННЯ

Викладемо алгоритм розвязання задачі ЛБП (4), що виникає у процесі синтезу оптимальної СЗІ, з використанням Р-методу. Зробимо позначення:

$$b_{L+1} = R_{\max} - \sum_{i=1}^{L}\sum_{j=1}^{N} R_{ij}Q_{ij},\ \ a_{L+1,(i-1)N+j} = -\alpha_{ij}\sum_{k=j}^{N} R_{ik}Q_{ik}\prod_{k=1}^{j-1}(1-R_{ik}),$$

$$b_s = 1,\ \ a_{sk} = \begin{cases} 1, & (s-1)N < k < sN \\ 0, & \text{інакше} \end{cases},\ \ s = \overline{1, L},$$

$$c_{(i-1)N+j} = C_{ij},\ \ x_{(i-1)N+j} = M_{ij}.$$

Тоді вихідна задача (4) може бути записана так:

$$\min_{x_j \in \{0,\,1\}} \{J(x) = cx \mid Ax \le b,\ \ x = (x_1, \ldots, x_{LN}),\ \ j = 1, \ldots, LN\},$$

де A — матриця $((L+1)\times LN)$; b, c, x — вектори розмірності LN, і розв'язана із використанням *P*-методу [6, 7]. Цей метод відноситься до класу комбінаторних і полягає в тому, що всі значення вектора x неявно перебираються у порядку збільшення з точки зору лексикографічного впорядкування. При цьому безперспективні часткові розв'язки не добудовуються, що значно скорочує час пошуку.

Для доведення несумісності підсистеми $(l+1)$-го порядку для часткового l-розв'язку $(x_1, \ldots, x_l)$ використовуються як вихідні нерівності задачі (4), так і одна штучна, в якій ми відображаємо наше знання про верхню границю значення цільової функції, відому на даний момент. До того моменту, коли ми не знайшли перший припустимий розв'язок, можна покласти $J^* = \infty$. Якщо ж ми знаходимо деякий повний припустимий розв'язок $(x'_1, \ldots, x'_{LN})$, то необхідно одразу покласти $J^* = \sum_{j=1}^{LN} c_j x'_j - \delta$, де $\delta > 0$, і для всіх припустимих векторів $x \ne y$ виконується $|J(x) - J(y)| \ge \delta$. Для обчислень із скінченною точністю величина δ завжди відома. Так, для цілих c_j маємо $\delta = \text{НСД}(c_1, \ldots, c_{LN})$. Для раціональних коефіцієнтів неважко отримати аналогічний вираз. У разі, якщо ми покладемо $\delta = 0$, в результаті роботи алгоритму буде знайдено лексикографічно максимальний вектор з множини оптимальних розв'язків. Оскільки вектори перебираються у порядку їх лексикографічного зростання, то на це буде витрачатися додатковий час, що є небажаним.

Особливості задачі дозволяють зробити певні спрощення. Так, у загальному випадку в алгоритмі використовуються дві множини Z_j та W_j, які заповнюються перед початком його роботи. Якби знаки та значення коефіцієнтів матриці A не були відомі наперед, то ми б мусили обчислювати та зберігати всі $2LN \times (L+2)$ елементів множин Z_j та W_j. Насправді ж ми

знаємо, що в перших L нерівностях всі коефіцієнти є бульовими, а частина матриці A, яка відповідає за ці нерівності, має ту особливість, що в кожному її стовпці є лише одна одиниця, а в кожному рядку — рівно N одиниць. Для $(L+1)$-ї нерівності відомо: всі її коефіцієнти є від'ємними. Також відомо, що штучна $(L+2)$-га нерівність, яка використовується для відсіву за значенням цільової функції, має всі додатні коефіцієнти. Тому в нашому випадку для кожного $j=\{1,\ldots,LN\}$ множин Z_j та W_j можна записати

$$Z_j=\left\{z_j \mid z_j=\sum_{k=j}^{LN} a_{L+1,k}\right\},$$

$$W_j=\left\{w_{ij} \mid w_{ij}=\sum_{k=j}^{LN} a_{ik},\ i=1,\ldots,L,\ \text{та}\ \ w_{L+1,j}=\sum_{k=j}^{LN} c_k\right\}.$$

Як бачимо, для зберігання елементів множини Z_j достатньо LN комірок, а для зберігання елементів множини W_j — $2LN$ комірок (при цьому враховано, що w_{ij} для $i=1,\ldots,L$ повністю визначається N цілочисельними індексами, тому з усіх $2LN$ комірок половина завжди буде містити цілі числа).

Серед інших особливостей, за якими запропонований алгоритм відрізняється від оригінального *P*-методу, є умова, при виконанні якої ми робимо висновок про несумісність підсистеми $(l+1)$-го порядку для часткового l-розв'язку $(x_1,\ldots,x_l)$. Відмінність полягає в тому, що деякі коефіцієнти множин Z_j та W_j відомі наперед, тому не зберігаються, і відповідні умови мають бути записані в явному вигляді.

Нехай I — множина індексів, що складається з номерів тих обмежень, які можуть бути використані для доведення несумісності підсистеми $(l+1)$-го порядку для часткового l-розв'язку $(x_1,\ldots,x_l)$. Якщо для підсистеми l-го порядку ця множина невідома, то можна покласти $I=$ $=(1,\ldots,L+2)$. В іншому ж випадку, коли переходимо від підсистеми l-го порядку до підсистеми $(l+1)$-го порядку (очевидно, що ми були вимушені зробити такий перехід через те, що спроби довести несумісність попередньої системи були невдалими), то з множини I можна вилучити деякі індекси. Оскільки при виконанні однієї з нерівностей

$$\begin{aligned} &b_{L+1}-\textstyle\sum_{j=1}^{l} a_{L+1,j}x_j>0,\\ &J^*-\textstyle\sum_{j=1}^{l} c_j x_j>w_{L+1,l+1},\\ &b_p-\textstyle\sum_{j=1}^{l} a_{pj}x_j>w_{p,l+1},\quad p=1,\ldots,L, \end{aligned} \tag{5}$$

для певного значення $p=1,\ldots,L+2\in I$ відповідна p-та нерівність задачі (в якості $(L+2)$-ї нерівності використовуємо штучне обмеження на значення цільової функції) ніколи не буде порушуватися при збільшенні l та довіль-

ному доповненні часткового l-розв'язку, то ми вилучаємо індекс p з множини I. Таким чином, навіть якщо несумісність підсистеми $(l+1)$-го порядку не буде доведена, то при формуванні підсистеми $(l+2)$-го порядку до неї можна не включати нерівності з номерами, що були видалені з множини I. Разом з тим нерівності з (5), які порушуються, можуть бути використані для доведення несумісності підсистеми $(l+1)$-го порядку для часткового l-розв'язку $(x_1,\ldots,x_l)$.

Твердження 1. Якщо існує таке p, для якого порушується одна з нерівностей (5) і справджується відповідна нерівність

$$\sum_{j=1}^{l} a_{p,j}x_j > 1, \quad p=1,\ldots,L,$$

$$b_p - \sum_{j=1}^{l} a_{p,j}x_j <, \quad z_{l+1}, \quad p=L+1,$$

$$\sum_{j=1}^{l} c_j x_j > J^*, \quad p=L+2,$$

то підсистема $(l+1)$-го порядку для часткового l-розв'язку $(x_1,\ldots,x_l)$ є несумісною.

Для збільшення швидкості роботи оригінального *P*-методу його автори запропонували використовувати евристичне правило для впорядкування змінних. У нашому випадку воно має такий вигляд.

1. Для всіх x_j, $j=1,\ldots,LN$ знаходимо оцінку

$$v_j = \left(\frac{a_{L+1,j}}{b_{L+1}} - 1\right)c_j.$$

2. Змінні впорядковуємо у порядку зростання оцінок v_j, $j=1,\ldots,LN$.

Алгоритм *P*-методу

1. Покладаємо $t=1$, $(x_1,\ldots,x_{LN})=(0,\ldots,0)$, $J^*=\infty$.

2. Якщо виконується $R_{\max} \le \sum_{i=1}^{L}\sum_{j=1}^{N} R_{ij}Q_{ij}$, то переходимо до п.3. Інакше задача не має розв'язку, і робота алгоритму припиняється.

3. Змінюючи значення індексу l від t до $LN-1$ та перевіряючи умови теореми 1, знаходимо мінімальне l', за якого підсистема $(l'+1)$-го порядку для часткового l'-розв'язку $(x_1,\ldots,x_{l'})$ є несумісною, і переходимо до п.4. Якщо ж такого l' не існує, то покладаємо $x_l=0$, $l=t,\ldots,LN-1$ і переходимо до п.5.

4. Якщо всі компоненти часткового l'-розв'язку, що розглядається, дорівнюють 1, то припиняємо роботу алгоритму. При цьому останній припустимий розв'язок і буде оптимальним. Інакше будуємо новий

l''-розв'язок за правилами: покладаємо $l'' = \max\{j \mid j \in \{1,\dots,l'\}\ x_j = 0\}$ та $x_{l''} = 1$, а для випадку $l' > l''$ ще і $x_j = 0,\ j = l''+1,\dots,l'$ і переходимо до п.2.

5. Обчислюємо $D_i = b_i - \sum_{j=1}^{LN-1} a_{ij}\, x_j \quad \forall\ i = 1,\dots,L+1, \quad D_{L+2} = J^* - \sum_{j=1}^{LN-1} c_j x_j$. Якщо $D_i \geq 0$ для всіх $i = 1,\dots,L+2$, то, покладаючи $x_{LN} = 0$, отримуємо припустимий розв'язок $(x_1,\dots,x_{LN})$ початкової задачі і переходимо до п.7.

6. Покладаємо $x_{LN} = 1$. Якщо $D_i - a_{i,LN} \geq 0 \quad \forall i = 1,\dots,L+1$ та $D_{L+2} - c_{LN} \geq 0$, то отримуємо припустимий розв'язок задачі і переходимо до п.7. Інакше покладаємо $l' = LN - 1$ і переходимо до п.4.

7. Після отримання першого припустимого розв'язку $(x'_1,\dots,x'_{LN})$ починається відсів розв'язків за значенням цільової функції. З цією метою покладаємо $J^* = \sum_{j=1}^{LN} c_j x_j - \delta$, де δ визначено раніше, та $l' = LN - 1$ і переходимо до п.4.

ДОСЛІДЖЕННЯ РОЗРОБЛЕНИХ АЛГОРИТМІВ ТА ЇХ ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА

Викладені вище локально-стохастичні алгоритми розв'язання задачі дискретного сепарабельного програмування (3) і *P*-метод розв'язання задачі ЛБП (4) були запрограмовані мовою C# і їх ефективність досліджувалася на модельних прикладах з максимальною кількістю загроз $L = 20$ та кількістю рівнів стеку протоколів $N = 4$. При розв'язанні задач ЛБП також робилося порівняння з ефективністю роботи алгоритмів Балаша з модифікаціями Джеофріона та методом вектора спаду, що були реалізовані мовою FORTAN в бібліотеці чисельного аналізу НДОЦ МДУ [9] (відповідно до процедур MNR3R та MLC6R).

Для точних методів Балаша та *P*-методу ефективність виявилася недостатньою, і задачі з кількістю бульових змінних $LN = 80$ і $LN = 64$ не були розв'язані у відведений час. Натомість наближені методи показали високу ефективність і достатню для практичних застосувань точність.

Динаміку зміни вартості чергової припустимої конфігурації СЗІ у процесі роботи *P*-методу можна прослідкувати на рис. 1. Часові характеристики згаданих алгоритмів наведені у таблиці. Час роботи методу вектора спаду при розв'язанні задачі ЛБП і алгоритму бікоординатного підйому при розв'язанні задачі сепарабельного програмування є поліноміальним (рис. 2).

При розв'язанні одних і тих самих задач (або задач, записаних у еквівалентному вигляді) за допомогою різних алгоритмів були отримані такі результати. Оскільки задачі великої розмірності за допомогою точних методів розв'язувалися дуже повільно, то для порівняння використовувалася задача

розмірністю $LN = 60$ (15 загроз). Метод Балаша, Р-метод знайшли точний розв'язок за порівняно малий час порядку 50 с. За допомогою наближених методів отримувався або той же глобальний мінімум (як для більшості задач ЛБП, що розв'язувалися за допомогою методу вектора спаду), або деякі локальні мінімуми.

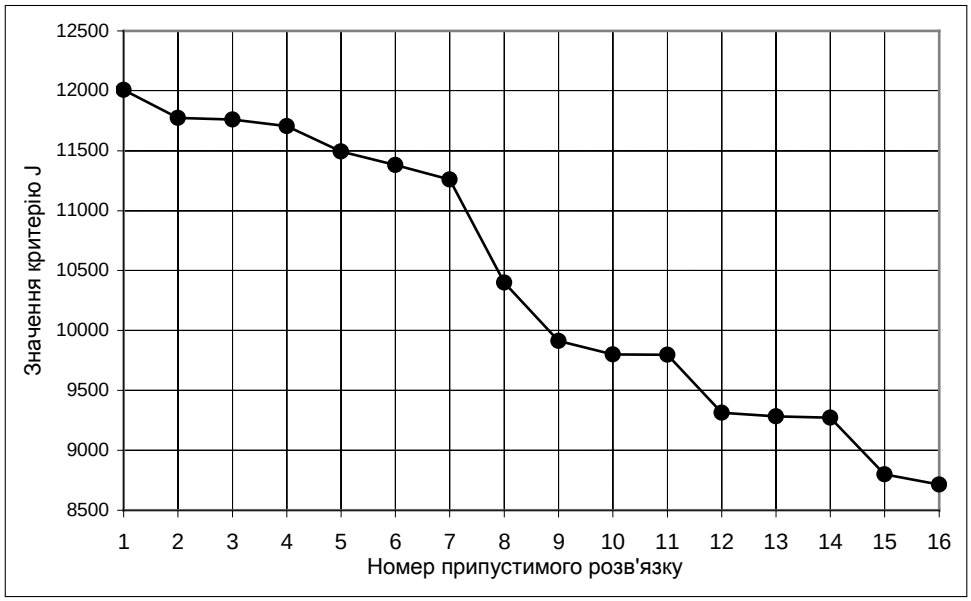

Рис. 1. Зміна значення критерію якості у процесі роботи Р-методу

Використана реалізація методу вектора спаду завжди починає пошук з нульового вектора, тому порівняння результатів його роботи для різних початкових значень не проводилося. Для тих задач ЛБП, в яких метод вектора спаду не знайшов глобального мінімуму, значення критерію в точці локального мінімуму відрізнялося від оптимального на величину не більше 2,5%.

Часові характеристики алгоритмів

Кількість загроз *L*	Час роботи алгоритмів			
	MNR3R, с	MLC6R, мс	*P*-метод, с	Бікоординатний підйом, мс
10	0,0313	0,195	0,08	0,4995
11	0,1458	0,255	0,31	0,5715
12	0,3327	0,336	0,78	0,7309
13	0,5195	0,405	3,20	0,7985
14	1,526	0,508	11,53	0,9440
15	32,33	0,621	49,25	1,1275
16	–	–	131	–
17	–	–	465	–
18	–	–	1879	–
19	–	–	6215	–
20	–	1,305	–	2,0333

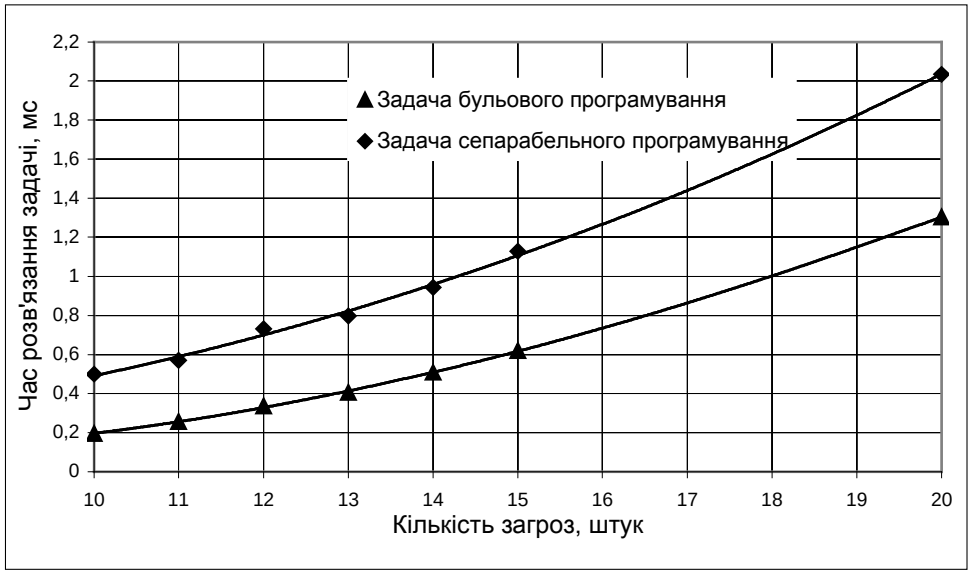

Рис. 2. Залежність часу роботи наближених методів від розмірності задачі

Алгоритм бікоординатного підйому починав пошук з випадкового вектора, тому в більшості випадків він отримував різні локальні мінімуми. Через високу швидкість алгоритму його можна запускати повторно. Конкретна кількість запусків з різних початкових точок може залежати від часу, який виділяється на розв’язання задачі. Значення відносних відхилень критерію від оптимального значення наведено на рис. 3.

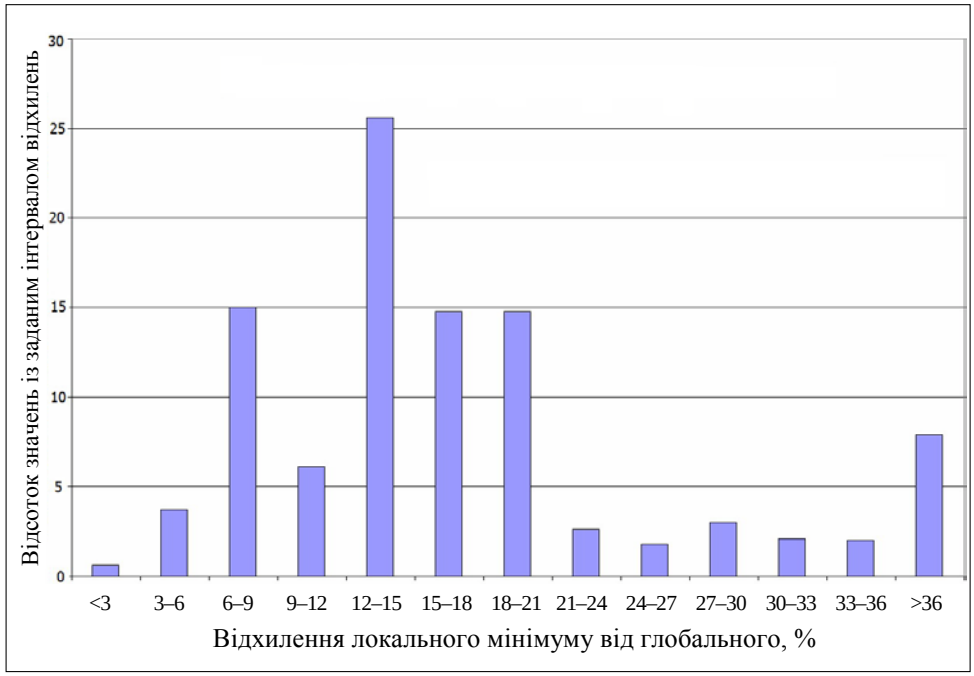

Рис. 3. Значення критерію в точках локальних мінімумів

ВИСНОВКИ

Розроблено алгоритми синтезу СЗІ з мінімальною вартістю засобів захисту в ІТС з відкритою архітектурою. Розглянуто задачі математичного програмування, які виникають в процесі синтезу оптимальних СЗІ при врахуванні додаткових особливостей. Запропоновано алгоритм розв'язання задачі цілочисельного сепарабельного програмування з використанням локально-стохастичних алгоритмів. При порівнянні його з точним Р-методом та методом вектора спаду для розв'язання задачі ЛБП він показав кращі або рівні показники і може бути використаний як складова системи автоматизованого проектування СЗІ, оптимальної за економічним критерієм. Недостатня точність розв'язання задачі сепарабельного програмування в порівнянні з задачею ЛБП нівелюється тим фактом, що в ній значно легше описати заборону певних комбінацій засобів захисту на різних рівнях стеку, як це часто буває на практиці. Всі розроблені алгоритми підтвердили свою працездатність і ефективність. Вони можуть бути використані при проектуванні реальних систем СЗІ, що задовольняють вимогам використання технології систем з відкритою архітектурою та для яких реально виконати чисельну оцінку ймовірнісних характеристик загроз безпеки і втрат від реалізації цих загроз.

ЛІТЕРАТУРА

1. *Новиков А., Тимошенко А.* Построение логико-вероятностной модели защищенной компьютерной системы // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. — 2001. — Вип. 3. — С. 101–105.
2. *Боня Ю.Ю., Новиков А.Н.* Синтез систем защиты информации с минимальной стоимостью механизмов защиты информации // Проблемы управления и информатики. — 2006. — №3. — С. 147–156.
3. *Зайченко Ю.П.* Исследование операций: Учебник. 6-е изд. — Киев: Слово, 2003. — 688 с.
4. *Боня Ю.Ю., Новиков О.М.* Синтез оптимальної системи захисту інформації на основі методу вектора спаду // Наук. вісті НТУУ «КПІ». — 2006. — № 4. — С. 107–112.
5. *Geoffrion Arthur M.* Integer Programming by Implicit Enumeration and Balas' Method // SIAM Review. — 1967. — **9**, № 2. — P. 178–190.
6. *Сергиенко И.В.* Математические модели и методы решения задач дискретной оптимизации. — Киев: Наук. думка, 1985. — 384 с.
7. *Артеменко В.І., Сергієнко І.В.* Про метод розв'язування задач линійного програмування з бульовими змінними // Доп. АН УРСР. — 1980. — № 4. — Серія А. — С. 68–71.
8. *Сергиенко И.В., Лебедева Т.Т., Рощин В.А.* Приближенные методы решения дискретных задач оптимизации. — Киев: Наук. думка, 1980. — 276 с.
9. *Систематический* каталог библиотеки численного анализа НИВЦ МГУ. — http://srcc.msu.su/num anal/lib na/cat/cat0.htm.

Надійшла 08.02.2007