Висновки

Згідно з проведеними оцінками використання ПФЕ 2^K при кількості факторів *K>11* стає недоцільним (часові витрати на синтез спрощених моделей перевищують часові витрати на оптимізацію на точних моделях). Це при оцінці одного компромісного розв'язку задачі оптимізації. При бажанні оцінки декількох розв'язків (**w**=var) межа ефективності ПФЕ 2^K збільшується, що графічно ілюструє крива оцінки чотирьох компромісних розв'язків: експеримент не ефективний при K>14.

У дійсності необхідності у проведенні ПФЕ 2^K при K>6 немає через надлишковість – поліноміальні моделі, отримані на їх основі мають статистично незначущі коефіцієнти при взаємодіях вищих порядків (q>5). Проведення ДФЕ 2^{K-P} збільшує межу ефективності застосування методики проведення машинних БФЕ. На графіку (рис.2) визначена область ефективності застосування методики БФЕ, яка обмежена кривою насиченого ДФЕ та кривою застосування точних моделей в процедурах оптимізації. Це область, в якій необхідно застосовувати розумний компроміс між точністю (чим більша репліка, тим нижча точність) та часовими витратами (чим більша репліка, тим менші часові витрати).

1. Казимира І.Я. Методи та засоби забезпечення ефективності процедур оптимального проектування ІС на схемотехнічному етапі. Автореф. дис. ... канд. техн. наук. Львів, 1998. 2. Koval V.A., Blyzniuk M.B., Kazymyra I.Y. Simplified Models of IC's for the Acceleration of Circuit Design // Mixed Design of Integrated Circuits and Systems, editors: A.Napieralski et al. – Kluwer Academic Publishers, Boston/Dordrecht/London, 1998. P.149–155. 3. Koval V.A., Blyzniuk M.B., Kazymyra I.Y. ICs Circuit Design Using Two-Level Simulation Technique // Proc. of the 4th International Workshop "Mixed Design of Integrated Circuits and Systems" (MIXDES'97). Poznan, 1997. P.163–166. 4. Kazymyra I.Y., Blyzniuk M.B., Lobur M.V. Preliminary Estimation of the Efficiency of Optimization Problems Solution in Circuit Design. Proceedings of the 5th International Conference "Mixed Design of Integrated Circuits and Systems" (MIXDES'98). Lodz, 1998. P.143–146

УДК 681.03.13

Нестор Н.І.
Львівський технічний коледж

ПАКЕТ ПРОЦЕДУР ДЛЯ РОЗВ'ЯЗАННЯ ОПТИМІЗАЦІЙНИХ ЗАДАЧ МЕТОДОМ ГІЛОК ТА ГРАНИЦЬ

Розглядаються практичні аспекти реалізації методу гілок та границь. Наводиться структура пакета процедур для виконання основних операцій при розв'язанні оптимізаційних задач. Пакет спроектований як програмне ядро, яке може використовуватися для різноманітних задач вичерпного пошуку із поверненнями.

Формулювання задачі

Метод гілок та границь є відомим часто вживаним способом чисельного розв'язування комбінаторних задач, у тому числі і NP-повних. Цей метод є варіантом пошуку із повер-

ненням. У загальному випадку ми припускаємо, що розв'язок задачі є вектором (s_1, s_2, **. . . .**) скінченної, але невизначеної довжини, який задовольняє певні обмеження. Кожне s_i є елементом скінченної лінійно впорядкованої множини S_i. Отже при вичерпному пошуку повинні розглядатися елементи множини $S_1 \times S_2 \times \ldots \times S_i$ для i=0, 1, 2, **. . .** як можливі рішення. Метод гілок та границь базується на припущенні, що кожне розв'язання має деяку вартість, і що потрібно знайти оптимальне – розв'язання з мінімальною вартістю). Для застосування методу гілок та границь вартість повинна бути чітко визначена для часткових розв'язків. Крім того, для всіх часткових розв'язків (s_1, s_2, **. . . ,** $\mathbf{s}_{к-1}$) і для всіх розширень (s_1, s_2, **. . . ,** $\mathbf{s}_{к-1}$, $s_к$) ми повинні мати

$$cost(s_1, s_2, ..., s_{k-1}) \le cost(s_1, s_2, ..., s_{k-1}, s_k) \quad (1)$$

Якщо вартість має таку властивість, то ми можемо відкинути частковий розв'язок (s_1, s_2, **. . . ,** $\mathbf{s}_{к-1}$, $s_к$), якщо його вартість більша або дорівнює вартості раніше обчислених розв'язків.

Дуже часто накладаються ще додаткові обмеження, які зв'язані функціональною залежністю із шуканим розв'язком. Такі обмеження, як правило, прискорюють розв'язання задачі, оскільки сприяють відсіканню підмножин варіантів, які їх не задовольняють. Задачу, особливості реалізації якої ми будемо розглядати далі, сформулюємо так: знайти мінімум вартості

$$\min cost(s_1, s_2, ..., s_{k-1}, s_k) \le cost(s_1, s_2, ..., s_{k-1}) \quad (2)$$

при обмеженнях

$$y = f(s_1, s_2, ..., s_{k-1}, s_k) \quad (3)$$

Одна із таких задач описана в роботі [1], при програмуванні якої і було розроблено достатньо універсальне ядро для застосування в різноманітних алгоритмах розв'язання оптимізаційних задач за методом гілок та границь.

Структури даних

При розв'язанні задач за методом гілок та границь найбільше ресурсів оперативної пам'яті та процесорного часу витрачається на формування дерева розв'язків і навігації по ньому. Тому при програмній реалізації особливу увагу треба звернути на оптимізацію власне цієї частини. Дерево розв'язків може бути наведене різними способами, серед яких найчастіше вживаються такі [2]: у формі багатозв'язного списку, послідовне подання, у вигляді бінарного дерева.

Перша форма подання є природною – і в цьому її основна перевага. Недоліком є те, що вершини дерева розв'язків мають різні степені, які до того ж змінюються в ході розв'язання задачі Це вимагає резервувати під вершину елемент із кількістю полів за максимально можливим степенем, що призводить до надлишкових витрат оперативної пам'яті і знижує універсальність процедур оперування з деревом або вимушує оперувати із елементами зі змінною кількістю полів, що суттєво ускладнює процедури динамічного використання оперативної пам'яті при змінах дерева розв'язків.

Послідовне подання дерева в пам'яті може використовуватися лише при постійності або рідкій мінливості структури дерева і для розв'язання задач за методом гілок та границь практично непридатний.

Спрощення внутрішнього подання бінарного дерева випливає із того, що кожний елемент зв'язного списку, який відповідає бінарному дереву, може містити лише два вка-

зівники для адресування інших елементів списку. Це означає, що для подання бінарного дерева достатньо двозв'язного списку. Будь-яке m – арне дерево може бути перетворене у бінарне. Перетворення дерева розв'язків у бінарне показане на рис.1. Це перетворення відрізняється від наведеного в [2] і краще відображає специфіку дерева розв'язків оскільки дуги, напрямлені праворуч, зв'язують вершини, які відповідають розгалуженням на підзадачі одного рівня, а дуги, напрямлені донизу, зв'язують суміжні рівні розгалуження.

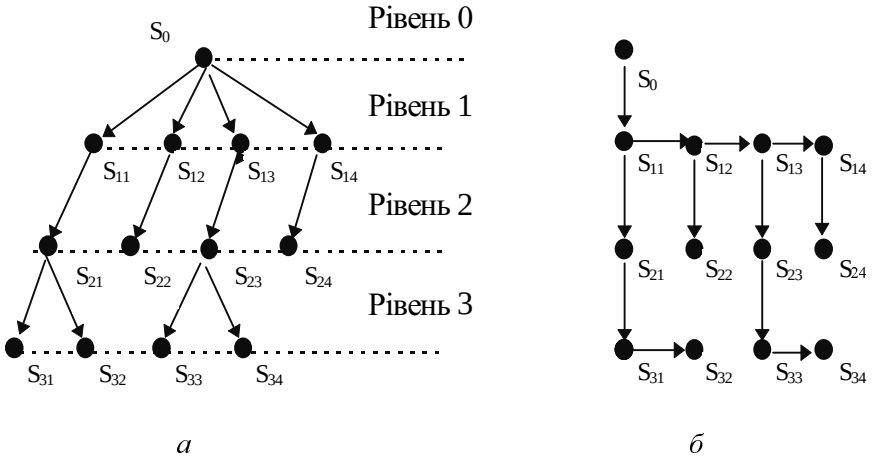


Рис.1. Перетворення дерева розв'язків у еквівалентне бінарне дерево

Перехід від дерева розв'язків до його бінарного еквівалента не тільки полегшує аналіз логічної структури, але також спрощує внутрішнє подання. Для відображення бінарного дерева в пам'яті комп'ютера у вигляді зв'язного списку кожен елемент списку може містити лише три поля, в одному з яких зберігаються дані, які відповідають елементу, а в двох інших полях – вказівники на наступний елемент в одному рівні і на елемент в наступному (нижчому) рівні.

Конкретна реалізація елементів спискової структури дерева розв'язків наведена на рис.2. Четверте поле в елементі відведене для ідентифікатора вершини. Як ідентифікатор використовується код, який містить два числа, запаковані в одне слово – номер рівня і номер "сина" в рівні. Очевидно, що такий ідентифікатор не обов'язково унікальний, але оскільки до кожної вершини існує тільки один шлях, то така неоднозначність колізій не викликає.

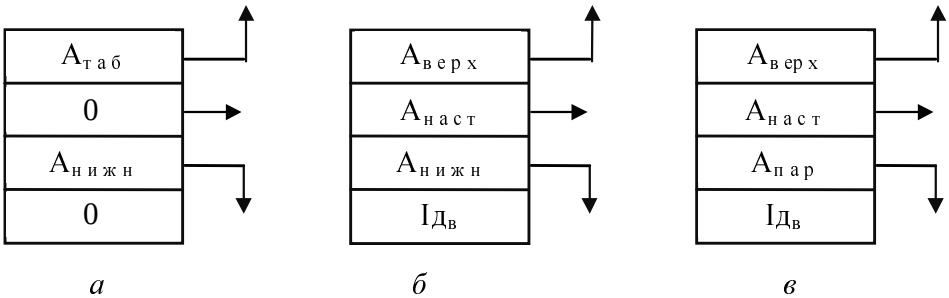


Рис.2. Елементи списку дерева рішень:
а – 0-вершина; б – проміжна вершина; в – висяча вершина;
$А_{таб}$ – посилання на таблицю опису дерева; $А_{верх}$– посилання
на елемент вищого рівня ("батька"); $А_{нижн}$– посилання на елемент нижчого рівня ("сина");
$А_{пар}$– посилання на таблицю параметрів вершини; $Ід_{в}$– ідентифікатор вершини

Кожний останній “син” позначається від’ємною адресою $A_{наст}$, а сама адреса є посиланням на першого “сина”. Висяча вершина має від’ємну адресу-посилання $A_{верх}$ на “батька”. Отже, спискова структура дозволяє рухатися по дереву донизу-догори-ліворуч-праворуч, від будь-якої вершини дерева. Це дає можливість вибирати оптимальну стратегію обходу дерева розв’язків у кожному конкретному алгоритмі.

Кожна висяча вершина має посилання на блок параметрів, який для даного часткового рішення містить вартість, значення параметра, на який накладається обмеження, а також нижні оцінки параметрів для розширень цього рішення. Проміжні вершини таких посилань не мають.

У процесі розв’язання задачі часткові розв’язки звичайно розширюються внаслідок розгалуження висячих вершин із мінімальною оцінкою вартості. Для прискорення пошуку вершини – претендента на подальше продовження часткового розв’язку висячі вершини прошиваються окремим списком. Кожен елемент цього списку аналогічний елементу дерева розв’язків і містить чотири поля: посилання на висячу вершину, посилання на наступний елемент списку, посилання на попередній елемент і ідентифікатор висячої вершини. На рис.3 наведено приклад подання бінарного дерева розв’язків з рис.1 і списку вільних вершин. Позначення елементів списку такі ж, як на рис.1. Елементи списку висячих вершин ідентифікуються іменами елементів дерева, на які вони посилаються. Вільні стрілки є посиланнями на блоки параметрів висячих вершин.

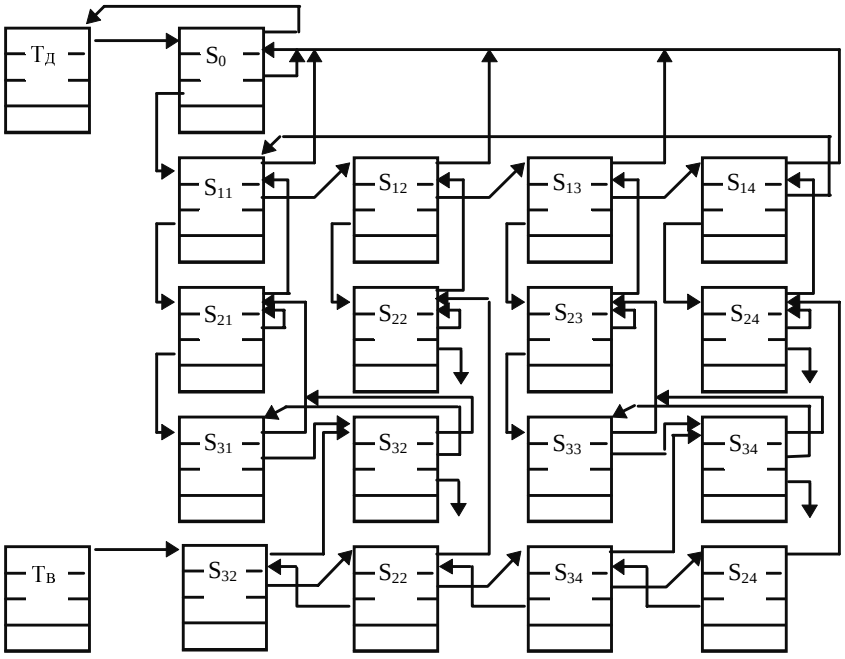


Рис.3. Приклад подання дерева розв’язків і списку висячих вершин
$T_д$, $T_в$ – таблиці дерева і списку висячих вершин відповідно

Усі списки розміщуються в одному масиві, який заповнюється динамічно. У процесі формування дерева розв’язків масив фрагментується і для оптимального використання пам’яті необхідні спеціальні процедури її виділення та вивільнення. Управління пам’яттю здійснюється за допомогою трьох списків вільної пам’яті:

- розміром в одне слово;
- розміром у два слова;
- розміром більше двох слів.

Перший список – простий лінійний циклічний список. Другий – двонапрямлений лінійний циклічний список. І третій – двонапрямлений лінійний циклічний список із вказанням довжини блока вільної пам'яті. Ці списки дозволяють реалізувати прості алгоритми динамічного управління пам'яттю.

Кожен список має свою таблицю опису з п'яти полів, в яких послідовно розміщені такі дані: вказівник на перший елемент списку, кількість елементів, робочі дані, адреса останнього долученого елемента, вказівник на елемент, з яким перед тим проводилися якісь дії.

У таблиці опису дерева рішень замість робочих даних вказується максимальна кількість рівнів.

Основні процедури

Коротко опишемо основні процедури маніпулювання зі списками.

Процедура формування таблиць опису списків резервує місце під таблиці опису списків і обнулює їх вміст. Кількість таблиць є вхідним параметром. Надалі кожен список ідентифікується порядковим номером таблиці.

Початкове встановлення таблиці вільної пам'яті. Процедура формує таблицю опису списку вільної пам'яті розміром більше двох слів. Встановлюються три однакові вказівники на початок блоку вільної пам'яті і кількість блоків, що дорівнює одиниці. Формуються вказівники на початку блоку вільної пам'яті і довжина цього блоку.

Долучення елемента до списку вільної пам'яті. Залежно від розміру вивільненого фрагмента пам'яті процедура долучає до одного із списків вільної пам'яті цей фрагмент.

Процедура пошуку блоку вільної пам'яті дозволяє знайти у списках вільної пам'яті блок потрібного розміру і видає адресу його початку. Якщо знайдений фрагмент має потрібну довжину, то він вилучається з відповідного списку, а якщо більшу, то залишок блоку долучається до відповідного списку вільної пам'яті.

Процедура створення і долучення вершини дерева створює і долучає до дерева розв'язків вершину одного з трьох типів: 0-вешину, вершину в цьому ж рівні, вершину наступного (нижчого) рівня. Тип вершини задається у вхідних даних. Перевіряється і встановлюється в спеціальній змінній помилки значення, яке дає можливість визначити коректність цієї дії, наприклад, фіксується неможливість долучення вершини наступного рівня до проміжної. Процедура використовує як внутрішню процедуру пошуку блоку вільної пам'яті.

Процедури навігації по дереву розв'язків дають можливість виконувати такі дії: встановити поточне значення вказівника на 0-вершину, встановити елемент дерева за значенням поточного вказівника, здійснити крок в одному з можливих напрямів від даного елемента. Усі процедури перевіряють коректність цих дій. Ще дві процедури дозволяють оперувати з параметрами вершин – долучати параметри до висячої вершини і читати параметри вершини.

Процедури для роботи із списком висячих вершин дерева розв'язків побудовані за аналогією із процедурами роботи з деревом розв'язків і виконують такі дії: долучення вершини до списку висячих, вилучення вершини, просування на крок вперед або назад. Використовуються процедури роботи зі списками вільної пам'яті.

До пакету також входить низка процедур тестування, налагодження і виведення проміжних результатів, які спрощують процес програмування і налагодження програми оптимізації загалом.

Процедури динамічного розподілу пам'яті вкористовуються також для резервування місця під вхідні і проміжні дані всієї програми.

Розроблений пакет процедур запрограмований на алгоритмічних мовах ФОРТРАН-77, ПАСКАЛЬ і в середовищі системи програмування DELPHI-4.0.

1. Нестор Н.І. Оптимізація послідовності технологічних операцій за критеріями виходу придатних виробів і вартості виготовлення // Вісн. ДУ "Львівська політехніка". 1999. № 373. С.42–46. 2. Костин А.Е., Шаньгин В.Ф. Организация и обработка структур данных в вычислительных системах. М., 1987.

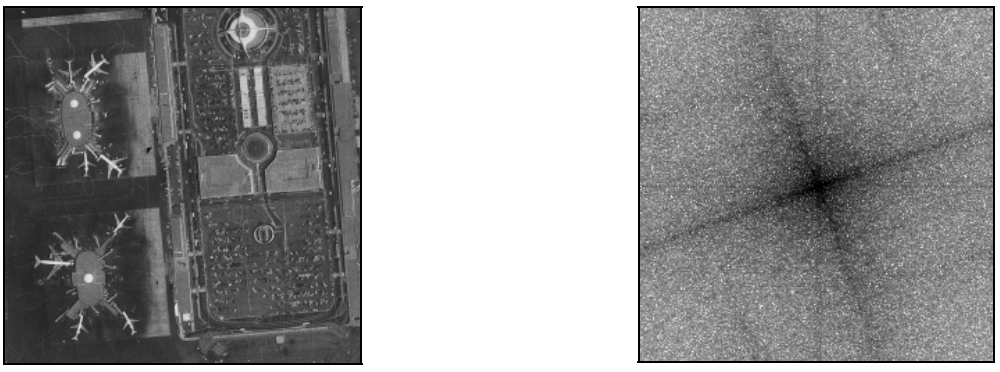
УДК 621.391.53.08

Клепфер Є., Прудиус І.*, Голотяк Т.*
Львівський науково-дослідний радіотехнічний інститут
*НУ "Львівська політехніка", кафедра РТП

АДАПТИВНИЙ МЕТОД ФОРМУВАННЯ РАДІОМЕТРИЧНИХ ЗОБРАЖЕНЬ

Розглянуто адаптивний метод формування радіометричних зображень. Проведено порівняльний аналіз методів формування зображень на основі l_2-похибки.

Дослідження різних типів радіозображень показали [1-4], що їх просторові спектри мають певні особливості у розташуванні просторових гармонік. Так, для зображення (рис.1,а) його просторовий спектр, визначений як перетворення Фур'є від інтенсивності, має такий вигляд (рис.1,б). Як видно з рисунка, гармоніки просторового спектра зображення мають специфічне розташування, тобто вони розташовані вздовж осей, зсунутих на певний кут відносно координатних осей у площині просторових частот. В ідеальному випадку забезпечувати оцінку всіх гармонік у просторовому спектрі зображення має антена. Звідси випливає, що узгодження характеристик антенної системи і зображення буде полягати у знаходженні такої геометрії антенної системи, яка б мала таку ж структуру просторового спектра, як і зображення. Однак на практиці це вимагає створення антен з великим розміром апертури, що не завжди вдається реалізувати через наявні конструкційні обмеження. Тому, враховуючи дані обмеження, пропонується система, яка забезпечує прийом найбільш



а *б*

Рис.1. Зображення (а) та його просторовий спектр (б)