УДК 004.4.42

Марченко Олександр Іванович
кандидат технічних наук,
доцент кафедри системного програмування і спеціалізованих комп'ютерних систем
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Марченко Александр Иванович
кандидат технических наук,
доцент кафедры системного программирования и специализированных компьютерных систем
Национальный технический университет Украины
«Киевский политехнический институт имени Игоря Сикорского»
Marchenko Oleksandr
PhD, Assistant Professor of Department of
System Programming and Specialized Computer Systems
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

Подзе Олександр Сергійович
магістрант
Національного технічного університету України
«Київський політехнічний інститут імені Ігоря Сікорського»
Подзе Александр Сергеевич
магистрант
Национального технического университета Украины
«Киевский политехнический институт имени Игоря Сикорского»
Podze Oleksandr
Student of the
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

МОДИФІКОВАНИЙ МЕТОД ВИДАЛЕННЯ СПІЛЬНИХ ВИРАЗІВ ЗА ДОПОМОГОЮ ГРАФУ ЗАЛЕЖНОСТІ СТАНІВ ТА ЗНАЧЕНЬ

МОДИФИЦИРОВАННЫЙ МЕТОД УДАЛЕНИЯ ОБЩИХ ВЫРАЖЕНИЙ ПРИ ПОМОЩИ ГРАФА ЗАВИСИМОСТИ СОСТОЯНИЙ И ЗНАЧЕНИЙ

MODIFIED COMMON SUBEXPRESSION ELIMINATION USING VALUE STATE DEPENDENCE GRAPH

Анотація. *Запропонований модифікований метод оптимізації трансляції видалення спільних виразів, який відрізняється від стандартного тим, що використовує граф залежності станів та значень у якості проміжного подання. Це дає можливість зменшити алгоритмічну складність методу, а також об'єм використаної пам'яті.*

Ключові слова: *оптимізації в трансляторах, граф залежності станів та значень, транслятори.*

Аннотация. *Предложен модифицированный метод оптимизации трансляции удаления общих выражений, который отличается от стандартного тем, что использует граф зависимости состояний и значений в качестве внутреннего представления. Это уменьшает алгоритмическую сложность метода, а также количество используемой памяти.*

Ключевые слова: *оптимизации в трансляторах, граф зависимости состояний и значений, трансляторы.*

Summary. *The paper presents modified common subexpression elimination method, which uses value-state dependence graph as its' intermediate representation. This allows reducing algorithmic complexity of method, as well as amount of consumed memory.*

Key words: *translator optimizations, value state dependence graph, translators.*

Розробку сучасного транслятору неможливо уявити без використання засобів оптимізації, які в автоматичному режимі надають можливість збільшити швидкодію, зменшити розмір скомпільованої програми, або зменшити об'єм використаної пам'яті. Одним із поширених методів оптимізації є оптимізація видалення спільних виразів. Ця оптимізація дозволяє в більшості випадків збільшити швидкодію вихідної програми без впливу на обсяг використаної пам'яті.

Вираз називається спільним, якщо його значення було обчислено раніше, і значення змінних, від яких цей вираз залежить, також не змінилося. Наприклад:

```
a:= b * c + g;
d:= b * c * e;
```

В наведеному фрагменті коду вираз b*c є спільним, і доцільно ввести додаткову змінну із проміжним результатом, використавши її при обчисленні двох змінних:

```
tmp:= b * c;
a:= tmp + g;
d:= tmp * e;
```

Стандартний метод вирішення проблеми видалення спільних виразів використовує граф потоку команд, і передбачає наступні етапи [1, с. 593]:

1. Побудова допоміжних множин in, out які можуть бути отримані в результаті проведення аналізу досяжних значень;

2. Пошук однакових виразів у програмі, які мають однакові досяжні значення у місці використання.

Проблема стандартного методу полягає в тому, що типова форма внутрішнього подання програми не ефективна для задач, які повинні враховувати потік даних в програмах — через це для проведення оптимізації необхідно спочатку сформувати множини досяжних значень для кожного виразу. Граф залежності станів та значень, запропонований в [2, с. 22] — функціональне подання програми, що відображає програму як потік даних, при цьому також містить інформацію щодо необхідного порядку виконання програми. Таким чином одна структура даних містить комбіновану інформацію, що традиційно зберігається в графі потоку даних, графі потоку команд, тощо. Загальна структура графу полягає в тому, що вершини відповідають обчисленням, а ребра відображають залежності одних обчислень від інших. Вхідна дуга відповідає за використання значення, яке є результатом обчислення для вершини, а вихідна — залежністю поточної операції від іншої. Приклад графу залежності станів та значень наведено на Рис. 1.

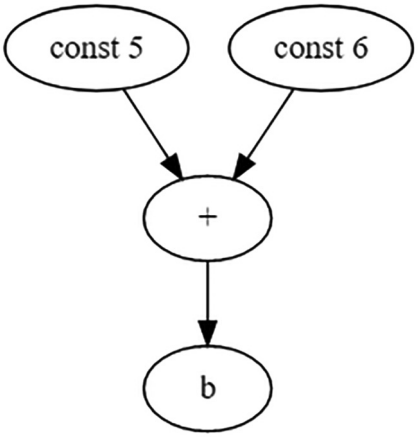

Рис. 1. Фрагмент графу залежності станів та значень

Граф, представлений на Рис. 1, відповідає виразу `B:= 5 + 6;`

Для графу залежності станів та значень використовується поняття попередника. Якщо існує шлях від вершини p до вершини q, то p являється попередником q. Множина вершин, які є попередниками p, позначається pred(p).

Завдяки тому, що вирази у графі відображені через вершини, а також через те, що залежності між змінними одразу використовують останнє записане значення, можна зробити висновок, що спільними виразами у графі залежності станів та значень можна вважати такі вершини p, q, які виконують однакову операцію обчислення, та їх множини попередників співпадають. Метод оптимізації видалення спільних виразів у псевдокоді буде мати наступний вигляд:

```
N_current = N
foreach(n in N_current)
   N_current:= N_current – n;
   if (n не має мітки)
      M_current = N_current
      foreach(m in M_current)
         M_current:= M_current – m
         if(n має мітку && m має мітку &&
            op(n) == op(m) && pred(n) ==
            pred(m))
            перемістити усі вхідні ребра
            від m до n;
            поставити мітку на m
foreach(n in N)
   if (n має мітку)
      видалити n
```

Такий підхід дозволяє видаляти спільні вирази з програми довільної глибини. Алгоритмічна склад-

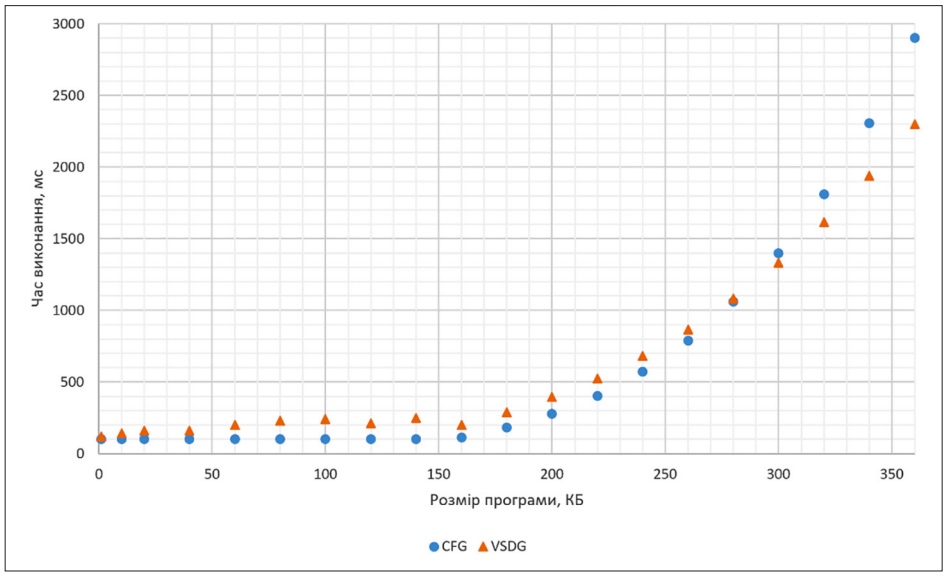

Рис. 2. Порівняння часу виконання запропонованого методу із стандартним

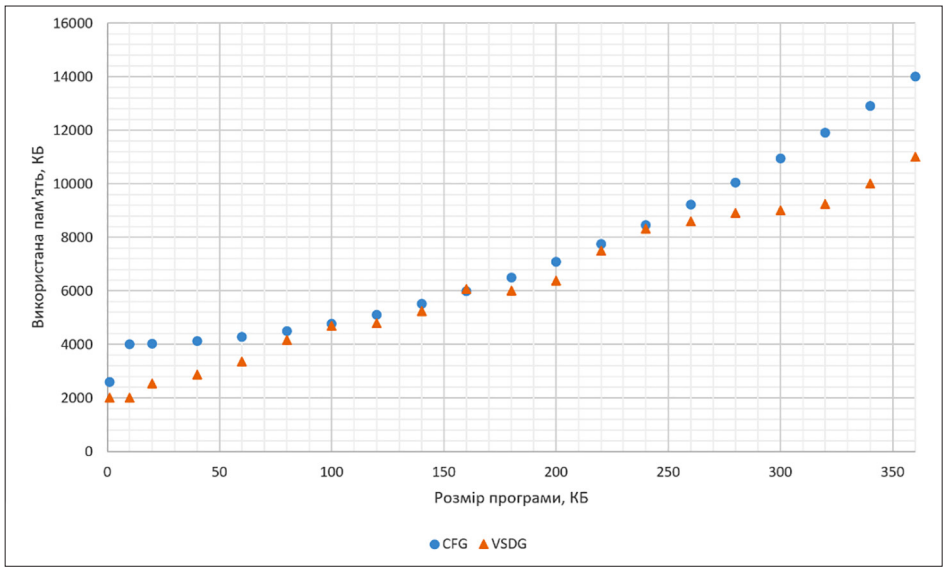

Рис. 3. Порівняння об'єму використаної пам'яті запропонованого методу із стандартним

ність запропонованого методу рівна $O(N^2)$ в найгіршому випадку, оскільки запропонований метод містить внутрішній цикл обходу по графу, тобто виконується перелік усіх пар вершин в графі. Просторова складність алгоритму складає O(N), оскільки використовується лише додаткова пам'ять на множину міток для вершин, та копію множини вершин графу.

Для стандартного методу етап проведення глобальної нумерації виразів, запропонований в [3, с. 5], має алгоритмічну складність $O(expr^3*join\_points)$, де expr — загальна кількість виразів у програмі, та join_points — кількість точок в програмі, де визначення змінних конфліктують між собою. Просторова складність стандартного методу складає $O(expr^2)$. Етап пошуку самих спільних виразів для стандартного методу має меншу складність, тому можна вважати вказані характеристики загальними показниками стандартного методу.

Зважаючи на вказане вище, запропонований метод дозволяє суттєво зменшити просторову та часову складність проведення оптимізації в трансляторах. Для перевірки теоретичних показників було створено два транслятори — один використовує стандартні структури даних у якості внутрішнього подання, а інший — граф залежності станів та значень. Кожен з них було перевірено на множині програм різної довжини, замірюючи загальний час роботи трансляторів. Результати практичних дослідів вказані на Рис. 2 та Рис. 3.

Висновки. Граф залежності станів та значень є досить новою формою проміжного подання програми в трансляторах. Існуючі типові методи оптимізації мають достатньо просту реалізацію в термінах такої структури. Запропонований модифікований метод показує кращі результати по параметрам швидкодії та використаної пам'яті на програмах великого обсягу.

Література

1. Ахо А., Сети Р., Ульман Д. Компиляторы: принципы, технологии и инструментарий: Пер. с англ. / Альфред Ахо, Рави Сети, Джеффри Ульман // издательский дом Вильямс, 2008.

2. Johnson N., Mycroft A. Combined Code Motion and Register Allocation using the Value State Dependence Graph. / Johnson N., Mycroft A. // 12th Intl. Conf. on Compiler Construction(CC'03) (April 2003), vol. 2622.

3. Saleena N., Paleri V. Global value numbering for redundancy detection: A simple and efficient algorithm / N. Saleena, V. Paleri // Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC '14, ACM, New York, NY, USA, 2014.