УДК 004.051+519.6

Способы снижения вычислительной сложности алгоритмов, вытекающие из принципа формирования решений

© В.А. Овчинников

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

Анализируется класс способов снижения вычислительной сложности комбинаторно-оптимизационных алгоритмов на графах и множествах. Рассмотренные способы основаны на использовании пошагового принципа формирования решения. Приведена классификация способов указанного класса. Рассмотрены примеры выполнения оптимизирующих преобразований в итерационном алгоритме улучшения разрезания гиперграфа и последовательном алгоритме его начального разрезания. Для этих алгоритмов рассмотрены процессы и приведены формулы определения значений критериальных оценок и множества вершин-кандидатов. Получены оценки вычислительной сложности выполнения указанных действий для случаев без и с использованием способов снижения вычислительной сложности. Выполнена оценка эффективности применения этих способов. Определена возможность формализации рассматриваемых оптимизирующих преобразований.

Ключевые слова: *алгоритм, вычислительная сложность, граф, множество, оптимизирующие преобразования, пошаговый принцип, формирование решения, рекуррентные формулы.*

Введение. Большинство задач проектирования структур сложных систем имеет большую размерность входа — миллионы и более компонент. В связи с этим даже для полиномиальных алгоритмов актуальной является проблема сокращения времени работы за счет использования способов снижения их вычислительной сложности, т. е. выполнения оптимизирующих преобразований.

Рассматриваются способы, вытекающие из пошагового принципа формирования решения, в том числе и его итерационного улучшения. Этот принцип широко используется в комбинаторно-оптимизационных алгоритмах и определяет, например, метод вычисления значений критериальных оценок и формирования состава «рабочих» подмножеств. Снижение вычислительной сложности алгоритмов при применении способов данной группы достигается за счет сокращения количества операций или замены их на более эффективные.

Само преобразование алгоритма заключается в замещении его части, неэффективно выполняющей необходимые действия, — заменяемый фрагмент — на заменяющий фрагмент, обеспечивающий получение результата с меньшей вычислительной сложностью. Исход-

ный и преобразованный алгоритмы должны быть *эквивалентны*, т. е. на всех допустимых наборах входных данных задачи давать одинаковые результаты.

Классификация рассматриваемых способов снижения вычислительной сложности. Предлагаемые автором способы преобразования алгоритмов показаны на рис. 1. Возможность использования первых двух из указанных способов обусловлена также ограниченностью количества элементов, связанных с добавляемым/удаляемым компонентом при пошаговом формировании графа-результата. Третье и четвертое преобразование заключаются в замене операций над множествами на более эффективные, причем такая замена возможна при выполнении определенных условий.

В качестве примеров будет рассмотрена ограниченная выборка возможных применений указанных способов. Объектами преобразований в примерах будут графы, описанные в [1–4].

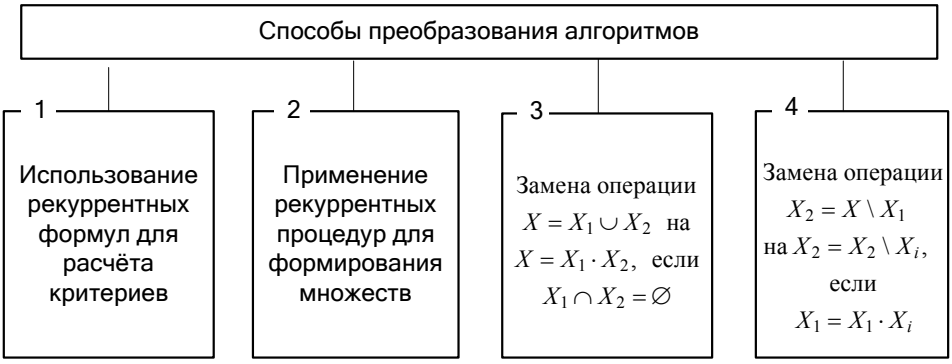

Рис. 1. Способы преобразования алгоритмов, вытекающие из принципа последовательного формирования решения

Способы, использующие рекуррентные формулы. *Первый способ* — применение рекуррентных формул для определения новых значений критериев и/или оценочных функций. В качестве примера рассмотрим алгоритм итерационного улучшения начального разрезания гиперграфа $H(X, U)$ парными перестановками вершин [2]. Критерий оптимальности — минимум количества ребер S, попадающих в разрез. Пусть имеется начальная компоновка схемы на две подсхемы. Соответственно ее модель — гиперграф $H(X, U)$ — разрезана на два куска $H_1\left(X_1^0, U_1^0\right)$ и $H_2\left(X_2^0, U_2^0\right)$, где верхний индекс «0» обозначает шаг обмена и $X_1^0 \cup X_2^0 = X$, $X_1^0 \cap X_2^0 = \varnothing$, $U_1^0 \cup U_2^0 = U$, $\left|U_1^0 \cap U_2^0\right| = S^0$, S^0 — количество ребер, соединяющих эти куски. Поскольку операцией преобразования разрезания гиперграфа является обмен вершинами между

множествами X_1 и X_2, на k-м шаге алгоритма для каждой перестановки $x_i \in X_1^{k-1}$ с $x_j \in X_2^{k-1}$ множества ребер U_1^k и U_2^k должны определяться по формулам $U_1^k = \Gamma\left(X_1^k\right)$, $U_2^k = \Gamma\left(X_2^k\right)$. Здесь $X_1^k = \left\{X_1^{k-1} \setminus x_i\right\} \cup x_j$, $X_2^k = \left\{X_2^{k-1} \setminus x_j\right\} \cup x_i$. Тогда показатель $S_{i,j}^k$ определяется по формуле

$$S_{i,j}^k = \left|\Gamma\left(X_1^k\right) \cap \Gamma\left(X_2^k\right)\right| S.$$

Вычислительная сложность определения количества ребер по данной формуле составит $O(n^2)$ при $n > m$ и $O(m^2)$ при $m > n$, где $n = |X|$, $m = |U|$ [2].

Выполнение операции попарного перемещения двух вершин x_i и x_j приводит к тому, что в разрез попадают или из разреза уходят только ребра, связанные с этими вершинами. Поэтому для каждого парного обмена достаточно определить количество ребер, приходящих в разрез и уходящих из него, и рассчитать их приращение. Тогда

$$S_{i,j}^k = S_{i,j}^{k-1} + \Delta S\left(x_i, x_j\right),$$

где $\Delta S\left(x_i, x_j\right) = \Delta S^+\left(x_i\right) + \Delta S^+\left(x_j\right) - \Delta S^-\left(x_i\right) - \Delta S^-\left(x_j\right)$ — приращение количества ребер в разрезе при перестановке вершин x_i и x_j; $\Delta S^+\left(x_i\right), \Delta S^+\left(x_j\right)$ — количество ребер, приходящих в разрез; $\Delta S^-\left(x_i\right)$, $\Delta S^-\left(x_j\right)$ — количество ребер, уходящих из разреза. Подсчет количества ребер, инцидентных, например вершине $x_i \in X_1^{k-1}$, и уходящих из разреза или приходящих в него, подразумевает реализацию следующего фрагмента алгоритма:

$$\forall u_f \in \Gamma x_i :$$

$$\left(X_f := \Gamma u_f,\ \left(\forall x_r \in X_f\right)\ \left(x_r \in X_2^{k-1} \cdot x_i\right) \Rightarrow \Delta S^-\left(x_i\right) := \Delta S^-\left(x_i\right) + 1,\right.$$

$$\left.\left(\forall x_r \in X_f\right)\ \left(x_r \notin X_2^{k-1}\right) \Rightarrow \Delta S^+\left(x_i\right) := \Delta S^+\left(x_i\right) + 1\right).$$

Учитывая, что $|\Gamma x_i| = \rho$, $|\Gamma u_j| = A$, где A — количество ребер, инцидентных вершине, ρ — количество вершин, инцидентных ребру, и считая $\left|X_1^k\right| = \left|X_2^k\right| = n/2$, получим асимптотическую оценку вычислительной сложности данного варианта определения количества ребер в разрезе $O(n)$.

Оценим возможность формализации данного способа. Пользователь может описать приведенные выше вычисления несколькими способами, например множество X_1^k определять как $X_1^k = X_1^{k-1} \setminus x_i$, $X_1^k = X_1^k \cdot x_j$,

или $X_1^k = X_1^{k-1} \setminus x_i \cdot x_j$ и т. п. Учтем тот факт, что этот способ может быть применен к алгоритмам, реализующим последовательное формирование графа результата посредством включения / исключения вершин или ребер. Это еще больше усложняет распознавание заменяемого фрагмента. Однако существует возможность выявления информации, которая укажет на принадлежность алгоритма к указанному классу. Такой анализ целесообразно выполнять по уграфу алгоритма [3, 4], поскольку он характеризуется высокой степенью формализации. Анализируется наличие в уграфе алгоритма цикла, содержащего операцию добавления или удаления вершины или ребра и операцию вычисления ее характеристик. Формальное правило такого анализа имеет вид [5]:

$$\exists G^3\left(X^3, U^3\right) \in G_y \left[t\left(x_i\right) \in Op_{\text{мод}}^G \,\&\, t\left(x_j\right) \in Op_{\text{хар}}^G / x_i, x_j \in X^3, i \neq j \right],$$

где $G^3\left(X^3, U^3\right)$ — конструкция типа «цикл» уграфа G_y алгоритма; $t\left(x_i\right)$ — операция, соответствующая вершине этого цикла; $Op_{\text{мод}}^G$ — множество операций модификации графовых моделей; $Op_{\text{хар}}^G$ — множество операций определения характеристик графовых моделей.

При обнаружении подобных конструкций в алгоритме целесообразно выдать разработчику запрос, предоставив ему возможность выполнить соответствующее преобразование вручную.

Второй способ — использование рекуррентных процедур для определения состава таких множеств, элементы которых в результате выполнения преобразований меняются частично. Данный способ реализован, например, в последовательном алгоритме разрезания гиперграфа [2]. На текущем шаге алгоритма множеством кандидатов *X*к на включение в формируемое подмножество X_l куска гиперграфа H_l являются вершины, которые смежны вершинам, уже вошедшим в кусок. При наличии в аналитическом представлении гиперграфа образа F_1X множества вершин относительно предиката смежности множество вершин-кандидатов определяется по формуле

$$X\text{к} = X_l^{\text{см}} \setminus X_l,$$

где $X_l^{\text{см}}$ — множество вершин, смежных вершинам:

$$X_l^{\text{см}} = F_1\left(X_l\right) = \cup F_1 X_i, \;\; x_i \in X_l.$$

Распишем процесс получения множества $X_l^{\text{см}}$:

$$F_1\left(X_l\right) = \left\{ F_1 x_i \cup F_1 x_j \cup F_1 x_k \cup \;\ldots\; \cup F_1 x_r \right\},$$

где $\left\{x_i, x_j, x_k, \;\ldots\; , x_r\right\} = X_l.$

Обозначим $F_1 x_i = X_i$, $F_1 x_j = X_j$, $F_1 x_k = X_k$, $F_1 x_r = X_r$ — множества вершин, смежных вершинам $x_i, x_j, x_k, \ldots, x_r$ соответственно. Будем считать, что $|X_i| = |X_j| = |X_k| = \ldots = |X_r| = \rho(A-1)$.

В связных гиперграфах количество вершин, одновременно смежных вершинам x_i и x_j, будет:

$$\left|\{X_i \cup X_j\}\right| = |X_i| + a_j |X_j| = \rho(A-1)(1+a_j),$$

где $a_j = |X'_j| / |X_j|$, $X_j \subseteq X_j$ — множество вершин, смежных вершине x_j и не смежных вершине x_i; $0 \le a_j \le 1$.

Количество операций сравнения при объединении множеств X_i и X_j равно $|X_i| \times |X_j|$, т. е. $\rho^2 (A-1)^2$. При объединении множеств $\{X_i \cup X_j\}$ и X_k это количество составит $\left(|X_i| + a_j |X_j|\right) \times |X_k|$, т. е. $\rho^2 (A-1)^2 \times (1+a_j)$ и т. д. Для получения оценки суммарного количества операций сравнения положим $a_j = a_k = \ldots = a$ $(0 < a < 1)$. Тогда количество операций сравнения, необходимых для определения $F_1(X_l)$, будет:

$$K_1 = \rho^2 (A-1)^2 \sum_{i=2}^{|X_l|} \left[1 + a(i-2)\right].$$

Асимптотическая оценка сложности формирования множества $X_l^{\text{см}}$ в худшем случае будет $O(n^2)$, если $|F_1 x|$ ограничена константой и $|X_l|$ ограничена n.

Очевидно, что мощность множества $X_l^{\text{см}}$ ограничена величиной n. Таким образом, асимптотическая оценка сложности формирования множества Xк будет равна $O(n^2)$.

Однако после включения в множество X_l некоторой вершины x_t множество Xк вершин-кандидатов может измениться только за счет вершин, инцидентных ребрам, приходящим в разрез, и не принадлежащих множеству Xк (рис. 2).

Таким образом, множество Xк достаточно корректировать, исключая из него вершину x_t и добавляя новые. Если определено множество $U_t^{\text{п}}$ ребер, приходящих в разрез, то множество $X_t^{\text{п}}$ инцидентных им вершин определяется по формуле

$$X_t^{\text{п}} = \bigcup_{u_j \in U_t^{\text{п}}} \{\Gamma u_j \setminus x_t\}.$$

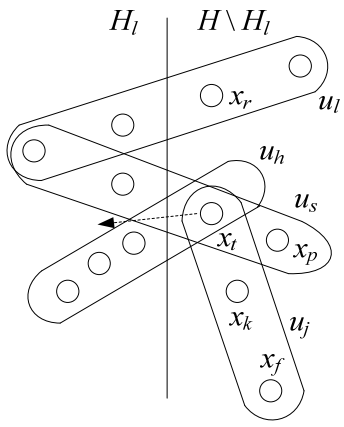

Рис. 2. Текущий шаг работы алгоритма, при выполнении которого в множество X_l включается вершина x_t

Тогда множество $X\text{к} = X\text{к} \setminus x_t$, $X\text{к} = X\text{к} \cup X_t^{\text{п}}$.

Так как $\left|X_t^{\text{п}}\right|_{\max} < \rho\left(A-1\right)$ и, согласно закону Рента,

$$\left|X\text{к}\right|_{\max} = \rho\left(A-1\right)n_l^p \text{, где } p = 0{,}5...0{,}75\,,$$

вычислительная сложность определения множество Xк будет от $O(n^{0{,}5})$ до $O(n^{0{,}75})$.

Возможность формализации данного способа такая же, как и у предыдущего.

Способы, выполняющие замену операций на более эффективные. *Третий способ* базируется на том, что при решении задач структурного анализа и синтеза обычно рассматриваются фрагменты графов с непересекающимися подмножествами вершин, а в ряде случаев и ребер. Очевидно, что при объединении непересекающихся подмножеств нет необходимости проверять, совпадают ли их элементы. Таким образом, при $X_1 \cap X_2 = \varnothing$ операцию $X = X_1 \cup X_2$ следует заменить на $X = X_1 \cdot X_2$, что позволяет снизить вычислительную сложность этой операции с $O(n^2)$ до $O(n)$ при представлении множеств векторами и до $O(1)$ при представлении списками с указателями их начала и конца. Следует, однако, помнить, что во втором случае множество X_2 не сохраняется.

Применение *четвертого способа* возможно, если при переносе элементов или подмножеств из одного множества в другое множества формируются следующим образом: $X_1 := X_1 \cdot X_i$, $X_2 = X \setminus X_1$. Если $|X_1|$ ограничена величиной n, вычислительная сложность определения множества X_2 равна $O(n^2)$. В этом случае множество X_2 следует получать как дополнение до него переносимых подмножеств

(элементов) $X_2 = X_2 \setminus X_i$, если $|X_i|$ ограничена константой. Тогда вычислительная сложность определения X_2 будет равна $O(n)$. Третье и четвертое преобразования можно формализовать.

Заключение. Представленные результаты исследований показали возможность и эффективность использования оптимизирующих преобразований, основанных на пошаговом принципе формирования решения. Асимптотическая оценка вычислительной сложности алгоритмов, основанных на методах последовательного формирования и итерационного улучшения, снижается с $O(n^2)$ до $O(n)$, что весьма существенно для алгоритмов решения задач структурного синтеза с большой размерностью входа.

ЛИТЕРАТУРА

[1] Овчинников В.А. Математические модели объектов задач структурного синтеза. *Наука и образование*, 2009, № 3. URL: http://technomag.bmstu.ru/doc/115712.html.

[2] Овчинников В.А. *Алгоритмизация комбинаторно-оптимизационных задач при проектировании ЭВМ и систем*. Москва, Изд-во МГТУ им. Н.Э. Баумана, 2001.

[3] Касьянов В.Н., Евстигнеев В.А. *Графы в программировании: обработка, визуализация и применение*. Санкт-Петербург, БХВ-Петербург, 2003.

[4] Овчинников В.А., Иванова Г.С. Информационно-логическая модель алгоритма. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2005, № 2 (59), с. 109–121.

[5] Иванова Г.С. *Методология и средства разработки алгоритмов решения задач анализа и синтеза структур программного обеспечения и устройств вычислительной техники*. Дис. ... д-ра техн. наук. Москва, 2007.

Статья поступила в редакцию 28.06.2013

Ссылку на эту статью просим оформлять следующим образом:

Овчинников В.А. Способы снижения вычислительной сложности алгоритмов, вытекающие из принципа формирования решений. *Инженерный журнал: наука и инновации,* 2013, вып. 11. URL: http://engjournal.ru/catalog/it/hidden/1046.html

Овчинников Владимир Анатольевич родился в 1939 г., окончил МВТУ им. Н.Э. Баумана в 1961 г. Д-р техн. наук, профессор кафедры «Компьютерные системы и сети» МГТУ им. Н.Э. Баумана, академик Международной академии информатизации. Автор свыше 120 научных работ в области вычислительной техники. Специализируется в области автоматизации проектирования компьютерных систем. e-mail: vaovchinnikov@gmail.com