КОНКУРС НА ЛУЧШУЮ НАУЧНУЮ РАБОТУ СТУДЕНТОВ ПО РАЗДЕЛУ
ТЕХНИЧЕСКАЯ КИБЕРНЕТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

СТУДЕНЧЕСКАЯ НАУЧНАЯ РАБОТА

“Исследование алгоритмов поиска максимальных независимых множеств на графах ”

Девиз : *Маниграф*

ДОНЕЦК 2001

РЕФЕРАТ

Работа содержит 40 страниц, 6 рисунков, 4 таблицы, 2 приложения, 3 источника.

В данной работе представлены результаты исследования алгоритмов для нахождения максимальных независимых множеств вершин. Был рассмотрен алгоритм Брона-Кэрбоша, и произведена его модификация.

Для проверки эффективности работы алгоритмов была разработана программа на языке Visual C++ 6.0, проведен ряд экспериментов и сделаны заключения относительно эффективности модификаций алгоритма по сравнению с оригиналом: показано, что полученные алгоритмы затрачивают на решение меньше времени.

ГРАФ, АЛГОРИТМ БРОНА-КЭРБОША, МАКСИМАЛЬНЫЕ НЕЗАВИСИМЫЕ МНОЖЕСТВА, УСОВЕРШЕНСТВОВАНИЕ АЛГОРИТМОВ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача о независимых множествах графа представляет большой интерес для исследования. Это связано не только с тем, что задача нахождения наибольшего независимого множества вершин является NP-полной: она является фундаментальной для теории графов. К задаче нахождения наибольшего независимого множества сводятся задачи о наименьшем покрытии, о наименьшем доминирующем множестве вершин, о наименьшем доминирующем множестве ребер, о наибольшем независимом множестве ребер([1], стр. 279). Каждая из этих задач и имеет множество прикладных применений, что позволяет говорить о необходимости разработки как можно более эффективных алгоритмов для их решения. В особенности для решения задачи о независимых множествах.

Задача о наибольшем независимом множестве вершин опирается на поиск всех максимальных независимых множеств. Это обусловлено тем, что независимые множества не являются матроидом, что делает невозможным применение жадных алгоритмов, основывающихся на свойствах элементов искомого множества.

Задача о нахождении всех независимых множеств является труднорешаемой, то есть все алгоритмы для ее решения имеют экспоненциальную временную сложность.

Простейший из способов решения этой задачи – полный перебор. При решении задачи таким образом будут построены все подмножества вершин, а потом будут отобраны удовлетворяющие следующим условиям:

- множество должно быть независимым;
- не должно содержаться в другом независимом множестве.

Неэффективность таких методов очевидна.

Существует второй подход – поиск с возвратами. Идея метода состоит в следующем: находясь в некоторой ситуации, производится попытка улучшить решение, если изменение не привело к успеху, то производится возврат к исходному состоянию и делается другая попытка. Наиболее известный, а также один из наиболее эффективных точных алгоритмов поиска максимальных независимых множеств вершин с возвратами – алгоритм Брона-Кэрбоша.

1 ПОСТАНОВКА ЗАДАЧИ

Задача отыскания всех максимальных независимых множеств может быть сформулирована следующим образом.

Пусть задан граф. Найти такое множество множеств вершин, что в каждом множестве любые две вершины несмежны и добавление любой вершины приведет к нарушению условия независимости.

Или:

пусть

$G = (V, \Gamma)$

найти:

$$W = \{S_l \mid S_l \subseteq V \,\&\, S_l = \{x_k \mid x_k \in V \,\&\, x_j \notin \Gamma(x_i) \,\&\, \neg\exists_{(x_m)}(S_l \cup x_m \in W)\}\}$$

2 ОБЩАЯ ИДЕЯ АЛГОРИТМА БРОНА-КЭРБОША

Пусть S_k – уже полученное множество из k вершин, Q_k – множество вершин, которое можно добавить в S_k, то есть $S_k \cap \Gamma(Q_k)=\varnothing$. Среди вершин Q_k будем различать те, которые уже использовались для расширения S_k (Q^-_k), и те, которые еже не использовались (Q^+_k). Тогда общая схема реализации поиска с возвратами будет стоять из следующих шагов.

Шаг вперед от k к k+1 состоит в выборе вершины $x \in Q^+_k$:

$$S_{k+1} = S_k \cup \{x\},$$
$$Q^-_{k+1} = Q^-_k \setminus \Gamma(x),$$
$$Q^+_{k+1} = Q^+_k \setminus (\Gamma(x) \cup \{x\})$$

Шаг назад от k+1 к k:

$$S_k = S_{k+1} \setminus \{x\},$$
$$Q^-_k = Q^-_k \cup \{x\},$$
$$Q^+_k = Q^+_k \setminus \{x\}$$

Если S_k максимальное, то $Q^+_k=\varnothing$. Если при этом $Q^-_k \neq \varnothing$, то S_k было получено раньше и не является максимальным. Таким образом, проверка максимальности задается следующим образом $Q^+_k = Q^-_k = \varnothing$.

Перебор можно улучшить, если заметить следующее. Пусть $\exists_{(x)}(x \notin Q^-_k \ \& \ \Gamma(x) \cap Q^+_k = \varnothing)$. Эту вершину никогда не удалить из Q^-_k, так как из Q^-_k, удаляются только вершины смежные с Q^+_k. Таким образом, существование такой вершины является достаточным условием для возвращения. Кроме того, $k \le N-1$, где $N = |V|$. ([1], стр. 275)

Остается проблема выбора вершины для расширения.

Нужно стремится минимизировать число шагов. Следовательно, нужно сосредоточить усилия на достижении условия $\exists_{(x)}(x \notin Q^-_k \ \& \ \Gamma(x) \cap Q^+_k = \varnothing)$. Можно выбирать любую вершину из Q^+_k. На шаге возврата эта вершина будет удалена из Q^+_k и добавлена в Q^-_k. Если выбрать вершину, такую что $x \in \Gamma(x_l \in Q^-_k) : \min(|\Gamma(x_l \in Q^-_k)|)$, то величина $|\Gamma(x \in Q^-_k) \cap Q^+_k|$ будет уменьшаться на 1 на каждом шаге возврата. ([2], стр. 48)

Очевидно, что существует необходимость хранить все множества $S_i, Q^+_i, Q^-_i \quad (i \le k)$, что можно организовать двумя способами: хранить массив множеств или «вталкивать» их в стек.

3 АЛГОРИТМ БРОНА-КЭРБОША

Все алгоритмы записаны на алгоритмическом языке, предложенном Ф.А.Новиковым. ([1], стр. 15)

Алгоритм Брона-Кэрбоша

Вход: граф G=(V,Г).

Выход: последовательность максимальных независимых множеств (W).

$Step1 : \{начальные\ установки\}$

$k := 0$

$S[k] := \varnothing$

$Q^-[k] := \varnothing$

$Q^+[k] := V$

$Step2 : \{расширение\}$

$select\ x \in \Gamma(x_l \in Q_k^-) : \min(\left|\Gamma(x_l \in Q_k^-)\right|)$ $\{выбор\ вершины\ для\ расширения\}$

$S[k+1] := S[k] \cup \{x\}$

$Q^-[k+1] := Q^-[k] \setminus \Gamma(x)$

$Q^+[k+1] := Q^+[k] \setminus (\Gamma(x) \cup \{x\})$

$k := k+1$

$Step3 : \{проверка\}$

$for\ y \in Q^-[k]\ do$

  $if\ \Gamma(y) \cap Q^+[k] = \varnothing\ then$

    $goto\ Step5$

  $end\ if$

$end\ for$

$Step4 : \{проверка\}$

$if\ Q^+[k] = \varnothing\ then$

  $if\ Q^-[k] = \varnothing\ then$

    $W := W \cup \{S[k]\}$ $\{максимальное\ независимое\ множество\}$

  $end\ if$

  $goto\ Step5$

$else$

  $goto\ Step2$

$end\ if$

$Step5 : \{шаг\ назад\}$

$x := Last(S[k])$ $\{последний\ добавленный\ элемент\}$

$k := k-1$

$S[k] := S[k+1] \setminus \{x\}$

$S[k] := S[k+1] \setminus \{x\}$

$Q^{-}[k] := Q^{-}[k] \cup \{x\}$

$Q^{+}[k] := Q^{+}[k] \setminus \{x\}$

if $k = 0$ & $Q^{+}[k] = \varnothing$ *then*

    stop {*выход* }

else

    goto *Step* 4

end *if*

4 МОДИФИКАЦИИ АЛГОРИТМА БРОНА-КЭРБОША

4.1 Модификация 1

Алгоритм Брона-Кэрбоша обладает весьма полезным свойством, – время его работы почти не зависит от расположения ребер в графе. Таким образом, временная сложность алгоритма является функцией только от количества вершин и ребер.

С другой стороны, это и недостаток алгоритма. Если рассматривать два графа, в одном из которых несколько искомых множеств, а в другом на порядок больше, то приблизительное равенство времени их обработки достижимо только путем усложнения обработки первого графа.

Для увеличения эффективности алгоритма необходимо найти «лишние» шаги, проделываемые им.

Рассмотрим работу алгоритма на пустом графе.

Первым найденным максимальным независимым множеством будет множество всех вершин графа, то есть других независимых множеств не будет. Следовательно, при эффективном решении, в течение обработки пустого графа должно быть N расширений.

Состояние переменных алгоритма после первого шага возврата:

$S[k] = V \setminus \{x_N\}$

$Q^-[k] = \{x_N\}$

$Q^+[k] = \varnothing$

Состояние переменных алгоритма после второго шага возврата:

$S[k] = V \setminus \{x_{N-1,} x_N\}$

$Q^-[k] = \{x_{N-1}\}$

$Q^+[k] = \{x_N\}$

На шаге 4 не выполнится условие $Q^+_k=\varnothing$, и выполнится ненужное расширение. При таком состоянии множеств выполнится условие $\exists_{(x)}(x \notin Q_k^- \;\&\; \Gamma(x) \cap Q_k^+ = \varnothing)$ на шаге 3, что приведет к выполнению шага возврата.

Если изменить шаг 5 таким образом, чтобы после его выполнения осуществлялся переход на шаг 3, то ненужное расширение выполняться не будет.

Это справедливо не только для пустых графов. Дело в том, что $(Q_k^+ = \varnothing \,\&\, Q_k^- \neq \varnothing) \Rightarrow \exists_{(x)}(x \notin Q_k^- \,\&\, \Gamma(x) \cap Q_k^+ = \varnothing)$, но не наоборот. Хотя в процессе работы алгоритма Брона-Кэрбоша возникают ситуации, когда переход на шаг возврата осуществляется по условию шага 4. Что снижает эффективность модификации, так как проверка условия шага 3 требует достаточно много времени.

Вопрос о целесообразности модификации сводится к вопросу о том, как часто при работе алгоритма Брона-Кэрбоша возникает ситуация, когда производится ненужное расширение, приводящее к шагу возврата.

На пустом графе все расширения, кроме N, будут лишними. На полном графе, таких расширений не будет. Таким образом, на пустом графе эффективнее работает модификация алгоритма, а на полном – оригинал. Можно предположить, что эффективность модификации алгоритма будет снижаться с увеличением числа ребер. Но при этом будет уменьшаться мощность множества Q^-, следовательно, уменьшается время, затрачиваемое на проверку условия шага 3. То есть эффективность модификации будет уменьшаться довольно медленно. Несмотря на это, графики временной сложности этих алгоритмов будут пересекаться, следовательно, должна существовать некоторая критическая заполненность графа, такая, что при дальнейшем его заполнении модификация алгоритма будет менее эффективной.

Алгоритм Брона-Кэрбоша с предложенными изменениями в литературе часто также называют алгоритмом Брона-Кэрбоша (например [1], [2]). Однако эти изменения приводят к неработоспособности алгоритма: к примеру, ошибка возникает при обработке пустых графов. Дело в том, что после шага возврата возможна ситуация, когда $k = 0 \,\&\, Q^+ \neq \varnothing \,\&\, \exists_{(x)}(x \notin Q_k^- \,\&\, \Gamma(x) \cap Q_k^+ = \varnothing)$. Это приведет к тому, что при k=0 будет переход на шаг возврата. Это в свою очередь делает шаг отрицательным, что является ошибкой, так как множества $S_{-1}, Q_{-1}^+, Q_{-1}^-$ не определены. Это произойдет только если все множества, которые можно построить, или уже были построены, или являются их подмножествами (по определению условия $\exists_{(x)}(x \notin Q_k^- \,\&\, \Gamma(x) \cap Q_k^+ = \varnothing)$). Во избежание ошибок, необходимо добавить в начало шага возврата проверки k на равенство нулю и, в случае успеха, прекращать работу алгоритма.

Модификация 1

Вход: граф G=(V,Г).

Выход: последовательность максимальных независимых множеств (W).

$Step1: \{начальные\ установки\}$

$k := 0$

$S[k] := \varnothing$

$Q^-[k] := \varnothing$

$Q^+[k] := V$

$Step2: \{расширение\}$

$select\ x \in \Gamma(x_l \in Q_k^-) : \min(\left|\Gamma(x_l \in Q_k^-)\right|)$ $\{выбор\ вершины\ для\ расширения\}$

$S[k+1] := S[k] \cup \{x\}$

$Q^-[k+1] := Q^-[k] \setminus \Gamma(x)$

$Q^+[k+1] := Q^+[k] \setminus (\Gamma(x) \cup \{x\})$

$k := k+1$

$Step3: \{проверка\}$

$for\ y \in Q^-[k]\ do$

$if\ \Gamma(y) \cap Q^+[k] = \varnothing\ then$

$goto\ Step5$

$end\ if$

$end\ for$

$Step4: \{проверка\}$

$if\ Q^+[k] = \varnothing\ then$

$if\ Q^-[k] = \varnothing\ then$

$W := W \cup \{S[k]\}$ $\{максимальное\ независимое\ множество\}$

$end\ if$

$goto\ Step5$

$else$

$goto\ Step2$

$end\ if$

$Step5: \{шаг\ назад\}$

$if\ k = 0\ then$

$stop$

$end\ if$

$x := Last(S[k])$ $\{последний\ добавленный\ элемент\}$

$k := k-1$

$S[k] := S[k+1] \setminus \{x\}$

$Q^-[k] := Q^-[k] \cup \{x\}$

$Q^+[k] := Q^+[k] \setminus \{x\}$

if $k = 0 \,\&\, Q^+[k] = \varnothing$ *then*

stop {*выход*}

else

*goto Step*3

end if

4.2 Модификация 2

Так как условие шага 3 включает в себя условие шага 4, то после проверки условия шага 3 проверять второе условие бессмысленно. Поэтому целесообразно переместить условие максимальности множества в шаг 3 и удалить шаг 4. Это поможет избежать заведомо лишних сравнений.

Модификация 2

Вход: граф G=(V,Г).

Выход: последовательность максимальных независимых множеств (W).

*Step*1 : {*начальные установки*}

$k := 0$

$S[k] := \varnothing$

$Q^-[k] := \varnothing$

$Q^+[k] := V$

*Step*2 : {*расширение*}

select $x \in \Gamma(x_l \in Q_k^-) : \min(\left|\Gamma(x_l \in Q_k^-)\right|)$ {*выбор вершины для расширения*}

$S[k+1] := S[k] \cup \{x\}$

$Q^-[k+1] := Q^-[k] \setminus \Gamma(x)$

$Q^+[k+1] := Q^+[k] \setminus (\Gamma(x) \cup \{x\})$

$k := k+1$

*Step*3 : {*проверка*}

f $Q^+[k] = \varnothing \,\&\, Q^-[k] = \varnothing$ *then*

$W := W \cup \{S[k]\}$ {*максимальное независимое множество*}

*goto Step*5

end if

$for\ y \in Q^{-}[k]\ do$
$if\ \Gamma(y) \cap Q^{+}[k] = \varnothing\ then$
$goto\ Step5$
$end\ if$
$end\ for$
$Step5 : \{шаг\ назад\}$
$if\ k = 0\ then$
$stop$
$end\ if$
$x := Last(S[k])\ \{последний\ добавленный\ элемент\}$
$k := k - 1$
$S[k] := S[k+1] \setminus \{x\}$
$Q^{-}[k] := Q^{-}[k] \cup \{x\}$
$Q^{+}[k] := Q^{+}[k] \setminus \{x\}$
$if\ k = 0\ \&\ Q^{+}[k] = \varnothing\ then$
$stop\ \{выход\}$
$else$
$goto\ Step3$
$end\ if$

4.3 Модификация 3

С другой стороны проверка условия шага 3 занимает слишком много времени. Нельзя исключать возможности того, что условие возврата, введенное в первоначальный алгоритм, необходимо для более быстрого функционирования алгоритма. Вернемся к модификации 1 и поменяем местами шаги 3 и 4. Переход с шага возврата будет происходить на шаг 4. Таким образом, перед проверкой основного условия, будет производиться «быстрая проверка», в большинстве случаев не влияющая на ход выполнения алгоритма. Но большого изменения времени работы алгоритма не ожидается, так как это условие выполняется достаточно редко.

Модификация 3

Вход: граф G=(V,Г).

Выход: последовательность максимальных независимых множеств (W).

$Step1 : \{начальные\ установки\}$
$k := 0$

$S[k] := \varnothing$

$Q^-[k] := \varnothing$

$Q^+[k] := V$

$Step2 : \{расширение\}$

$select\ x \in \Gamma(x_l \in Q_k^-) : \min(\left|\Gamma(x_l \in Q_k^-)\right|)$ {*выбор вершины для расширения*}

$S[k+1] := S[k] \cup \{x\}$

$Q^-[k+1] := Q^-[k] \setminus \Gamma(x)$

$Q^+[k+1] := Q^+[k] \setminus (\Gamma(x) \cup \{x\})$

$k := k+1$

$Step4 : \{проверка\}$

if $Q^+[k] = \varnothing$ *then*

  if $Q^-[k] = \varnothing$ *then*

    $W := W \cup \{S[k]\}$ {*максимальное независимое множество*}

  end if

  *goto Step*5

else

  *goto Step*2

end if

$Step3 : \{проверка\}$

for $y \in Q^-[k]$ *do*

if $\Gamma(y) \cap Q^+[k] = \varnothing$ *then*

    *goto Step*5

  end if

end for

$Step5 : \{шаг\ назад\}$

if $k = 0$ *then*

  stop

end if

$x := Last(S[k])$ {*последний добавленный элемент*}

$k := k-1$

$S[k] := S[k+1] \setminus \{x\}$

$Q^-[k] := Q^-[k] \cup \{x\}$

$Q^+[k] := Q^+[k] \setminus \{x\}$

if $k = 0\ \&\ Q^+[k] = \varnothing$ *then*

  stop {*выход*}

else

  *goto Step*4

end if

4.4 Модификация 4

У всех рассмотренных ранее алгоритмов есть один общий недостаток. При выборе вершины каждый из них просматривает множества $\Gamma(x) \cap Q_k^+$ для всех $x \in Q_k^-$. При проверке условия на расширяемость проверяются те же множества. Гораздо эффективнее будет совместить два действия и при проверке условия расширяемости находить следующую вершину для расширения. Таким образом, суммарная эффективность шагов 2 и 3 увеличится почти вдвое, так как эти действия составляют их основу.

Функция проверить_и_взять может быть реализована следующим образом:

Проверить_и_взять

Вход: Γ, Q^+, Q^-

Выход: x – вершина, которой следует расширить множество S.

$MinP = \infty$

$for\ u \in Q^-[k]\ do$

$\quad if\ \left|\Gamma(u) \cap Q^+\right| = 0\ then$

$\quad\quad x := nonvertex\ \{вершина, заведомо\ не\ входящая\ ни\ в\ один\ граф\}$

$\quad\quad stop$

$\quad end\ if$

$\quad if\ \left|\Gamma(u) \cap Q^+\right| < MinP\ then$

$\quad\quad MinP := \left|\Gamma(u) \cap Q^+\right|$

$\quad\quad x := first\left(\Gamma(u) \cap Q^+\right)\{первый\ элемент\ множества\}$

$\quad end\ if$

$end\ \ for$

Так как самым эффективным из рассмотренных алгоритмов предполагается модификация 2, модифицировать следует именно его.

Модификация 4

Вход: граф G=(V,Г).

Выход: последовательность максимальных независимых множеств (W).

$Step1: \{начальные\ установки\}$

$k := 0$

$S[k] := \varnothing$

$Q^-[k] := \varnothing$

$Q^+[k] := V$

*Step*2 : {*расширение*}

$S[k+1] := S[k] \cup \{x\}$

$Q^-[k+1] := Q^-[k] \setminus \Gamma(x)$

$Q^+[k+1] := Q^+[k] \setminus (\Gamma(x) \cup \{x\})$

$k := k+1$

*Step*3 : {*проверка*}

f $Q^+[k] = \varnothing$ & $Q^-[k] = \varnothing$ *then*

$W := W \cup \{S[k]\}$ {*максимальное независимое множество*}

*goto Step*5

end if

$x :=$ *проверить _ и _ взять*(Γ, Q^+, Q^-)

if $x = nonvertex$ *then*

*goto Step*5

else

*goto Step*2

end if

*Step*5 : {*шаг назад*}

if $k = 0$ *then*

stop

end if

$x := Last(S[k])$ {*последний добавленный элемент*}

$k := k-1$

$S[k] := S[k+1] \setminus \{x\}$

$Q^-[k] := Q^-[k] \cup \{x\}$

$Q^+[k] := Q^+[k] \setminus \{x\}$

if $k = 0$ & $Q^+[k] = \varnothing$ *then*

stop {*выход*}

else

*goto Step*3

end if

5 РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ИССЛЕДОВАНИЯ АЛГОРИТМОВ

Инструментом для кодирования алгоритмов был выбран язык высокого уровня MS Visual C++ 6.0.

Для реализации множеств были использованы связные однонаправленные списки.

Граф представляется списком вершин, а вершина номером (0 – N) и множеством Г(x).

Для хранения данных о предыдущих шагах используется стек со стандартным набором операций над ним, с той только разницей, что операции со стеком работают одновременно с переменными содержащими Q^+, Q^-, S

Остальные структуры, реализованные в программе, используются для вывода данных на экран и пользовательского интерфейса.

Необходимо отметить, что при каждом шаге любого из алгоритмов его действие заносится в протокол, который можно просмотреть после окончания работы алгоритма или сохранить на жесткий диск в виде текстового файла.

Предусмотрена случайная генерация графов, с задаваемой заполненностью.

Входными данными для всех алгоритмов является граф.

Выходными данными являются протокол, количество выполненных шагов, затраченное время, текстовый список результатов – максимальные независимые множества, найденные алгоритмом.

В программе отличается нумерация алгоритмов, алгоритмы с первого по пятый соответственно модификация 1, алгоритм Брона-Кэрбоша, модификация 2, модификация 3, модификация 4.

5.1 Блок-схемы алгоритмов

Блок-схемы алгоритмов представлены на Рисунках 5.1 – 5.5.

Считается, что в блоке выбора (условия) при выполнении условия движение производится по правому выходу.

Для модификации 4 принято, что константа nonvertex=-1.

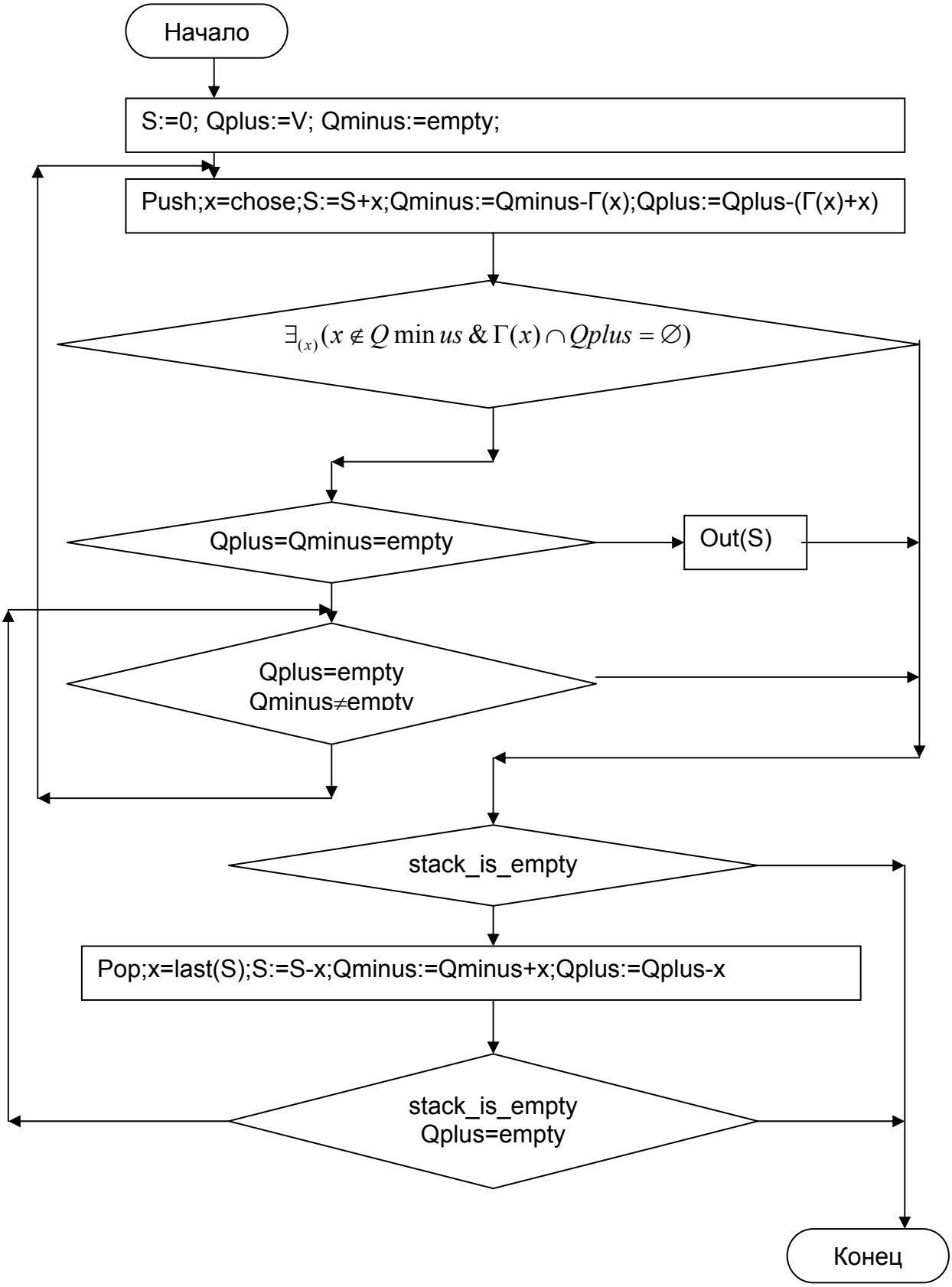

Рисунок 5.1 Алгоритм Брона-Кэрбоша

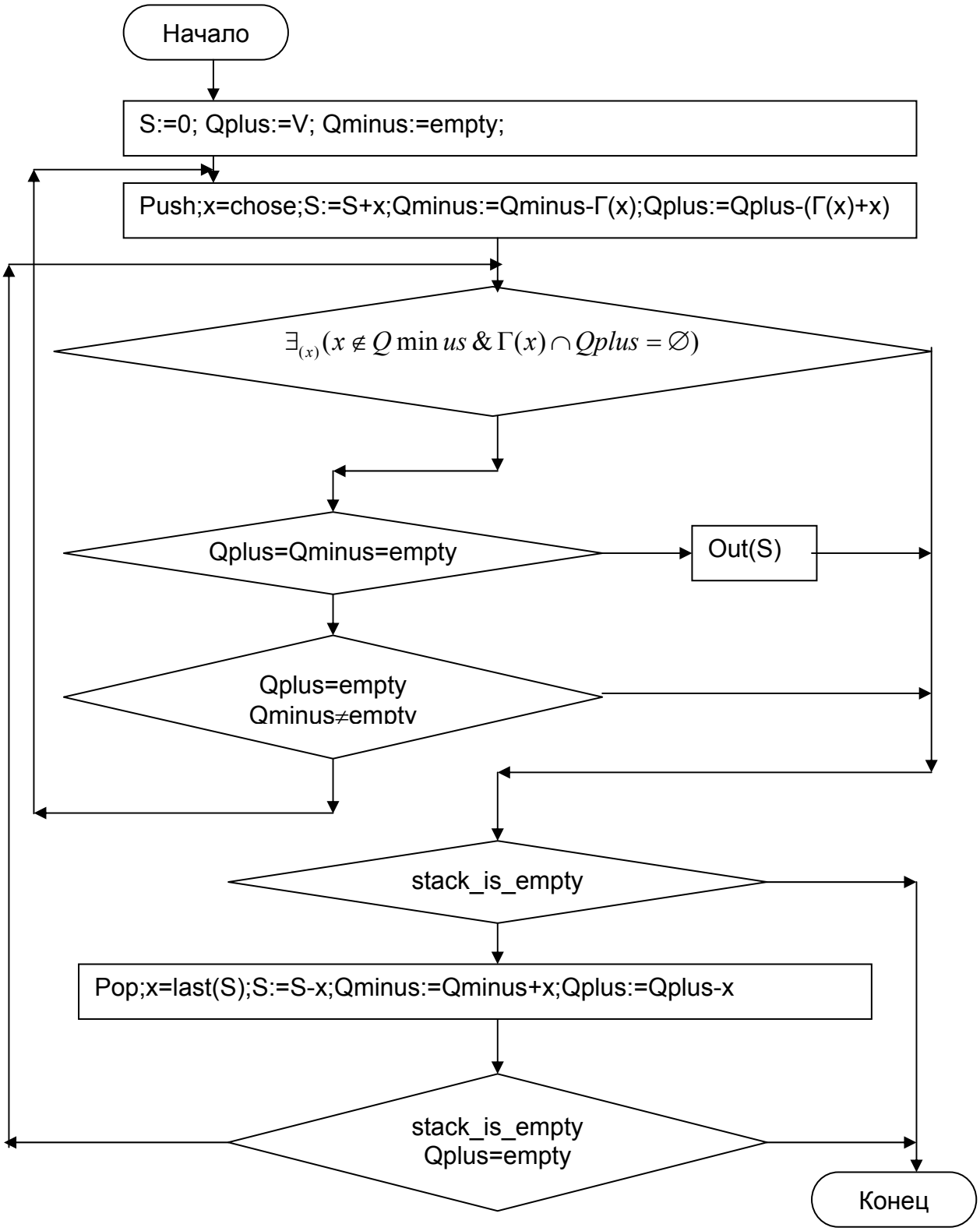

Рисунок 5.2 Модификация 1

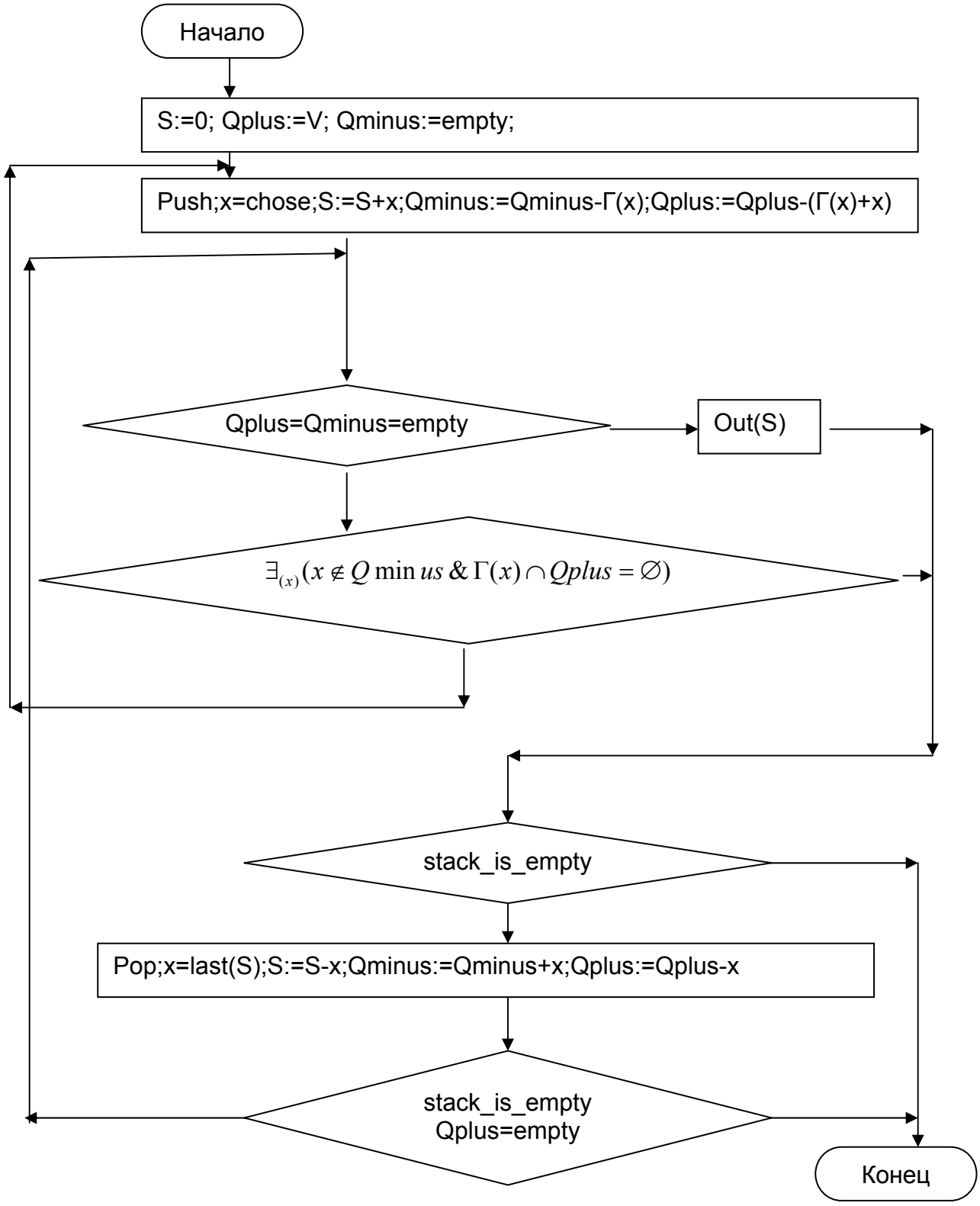

Рисунок 5.3 Модификация 2

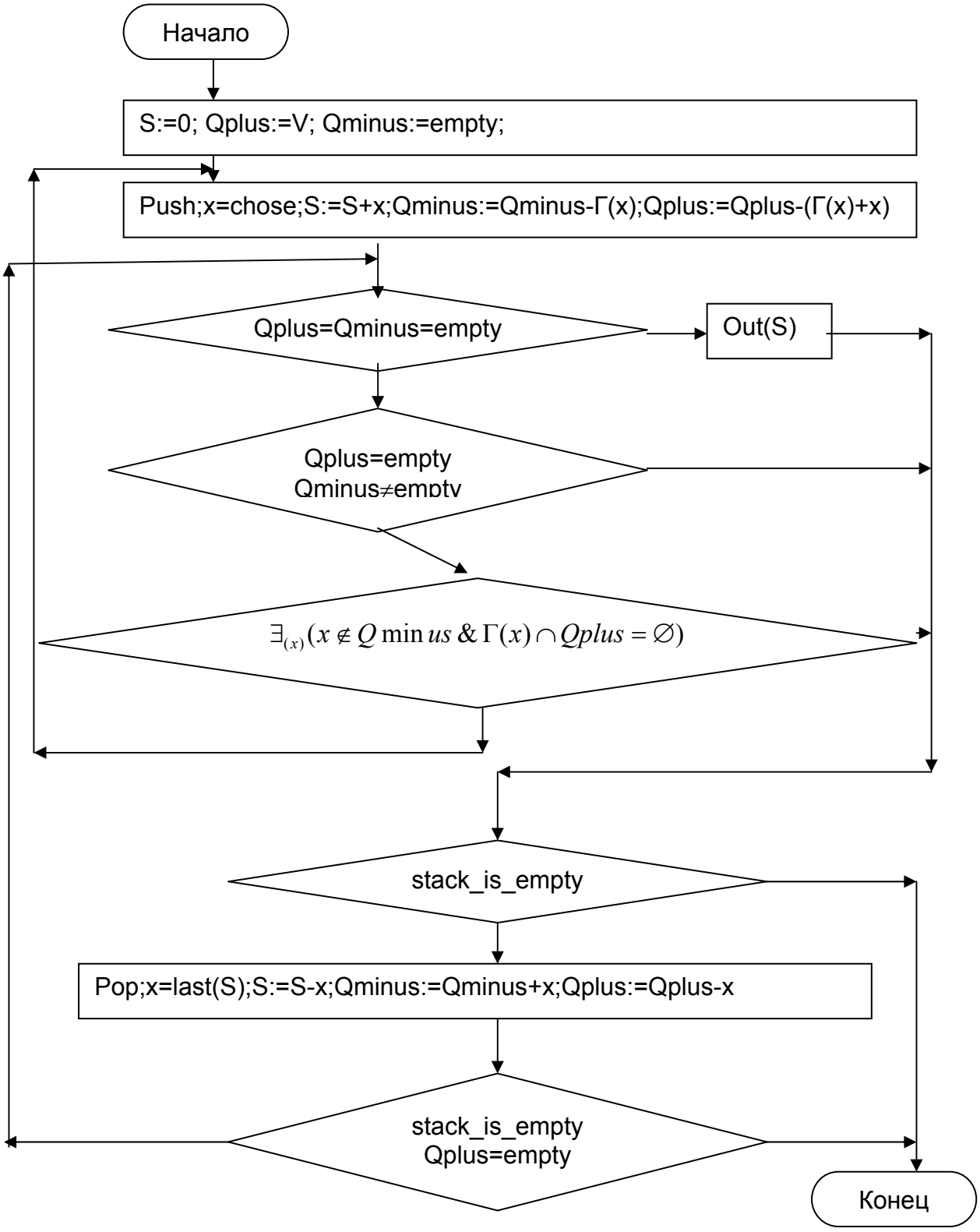

Рисунок 5.4 Модификация 3

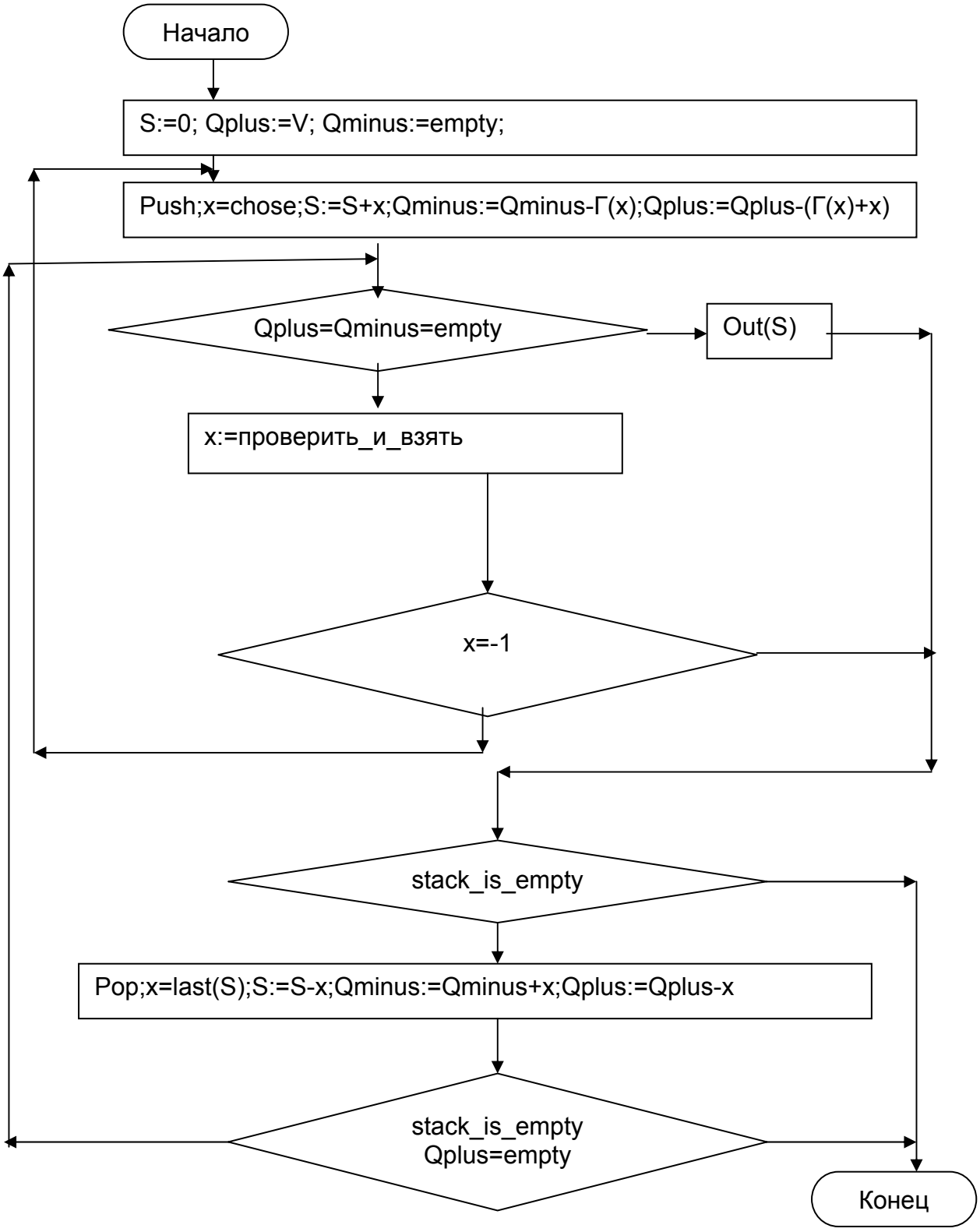

Рисунок 5.5 Модификация 4

6 ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ

6.1 Методика экспериментов

Все экспериментальные исследования проводились по следующему принципу:

1. Генерация графа – задается относительная заполненность графа в процентах (с точностью до 0,01). Далее производится переход от числа процентов к количеству ребер, которые необходимо добавить в граф, по правилу:

$$\frac{Per}{100} * \frac{N * (N-1)}{2},$$

где Per – относительная заполненность, N – количество вершин графа.

Если заполненность, которую необходимо получить, не превышает 50%, создается пустой граф на N вершинах. Далее, посредством стандартного генератора случайных чисел Visual C++, производится генерация пары чисел в пределах от 0 до N-1. Если такое ребро уже существует или сгенерированы одинаковые числа, – генерация повторяется, в противном случае добавляется ребро между вершинами, номера которых совпадают со сгенерированными, и генерируется следующая пара чисел. Если необходимая заполненность превышает 50%, то создается полный граф и из него удаляются ребра, генерируемые аналогично предыдущему случаю. Добавление (или удаление) ребер производится до достижения заданной заполненности.

Не всегда возможно создать граф с указанной заполненностью, например, для графа на 10 вершинах: 4 ребра – 8,89%, 5 ребер – 11,11%, следовательно, заполненность 10% недостижима. В таких ситуациях генерация производится таким образом, чтобы не превысить указанной заполненности.

2. Сохранение графа – граф сохраняется для возможности повторения эксперимента в будущем.
3. Эксперимент – граф последовательно обрабатывается исследуемыми алгоритмами. При этом если время выполнения менее 1 секунды, то проводится повторное исследование: алгоритм выполняется 100 раз и суммарное время работы делится на 100. Если при этом время остается

меньше 1 секунды, то проводятся дополнительные исследования с количеством запусков 1100, 2100, 3100, … Если время выполнения алгоритма более 1 секунды, но менее 30 секунд, алгоритм выполняется 10 раз. В зависимости от количества запусков алгоритма, время его работы определяется с разной точностью: 1 раз – с точностью до секунды, 10 раз – 0,1 секунды, 100 раз – 0,01 секунды, 1000 и более раз – 0,001.

6.1 Сравнение алгоритмов Брона-Кэрбоша и Модификации 1

В таблице 6.1 представлены экспериментальные данные по сравнению алгоритма Брона-Кэрбоша и Модификации 1. Эксперименты проводились на графах с 40 вершинами. На рисунке 6.1 представлен график зависимости времени работы рассматриваемых алгоритмов. Рассмотрим с помощью графика тенденцию изменения эффективности алгоритмов относительно друг друга.

При заполненности 0% алгоритм Брона-Кэрбоша затрачивает больше времени. На промежутке 0-5% время работы обоих алгоритмов растет, но время работы алгоритма Брона-Кэрбоша растет быстрее, из чего следует, что эффективность модификации возрастает. На промежутке 5-15% время, необходимое обоим алгоритмам продолжает расти, однако наклон кривой алгоритма Брона-Кэрбоша намного меньше, чем у Модификации 1, таким образом, эффективность модификации падает, однако Модификация 1 остается в 2-3 раза эффективнее. При дальнейшем увеличении заполненности, время работы алгоритмов уменьшается, причем уменьшение времени работы алгоритма Брона-Кэрбоша более стремительное. И уже при заполненности 75% алгоритм Брона-Кэрбоша оказывается более эффективным. На промежутке 75-100% уменьшение времени продолжается в таком же соотношении, следовательно, эффективность алгоритма Брона-Кэрбоша растет.

Так как время работы алгоритмов на графах с высокой заполненностью мало (<1с), для получения более достоверной информации о поведении алгоритмов на графах с высокой степенью заполненности, было принято решения провести дополнительные исследования на графах со 100 вершинами. К сожалению, время работы алгоритмов на таких графах при малой заполненности слишком

велико, поэтому исследовались только графы с заполненностью не менее 75%. Результаты исследования представлены в таблице 6.1 и на рисунке 6.2.

В общем, рассмотренная тенденция сохранилась, однако есть некоторые отличия. Во-первых, при приближении заполненности к 100% время работы Модификации 1 немного увеличивается (при более детальном рассмотрении рисунка 6.1 это оказывается справедливым и для него). Во-вторых, «критическая точка», после которой применение Модификации 1 нецелесообразно, сдвинулась в сторону большей заполненности. Логично предположить, что при увеличении числа вершин, «критическая точка» сдвинется еще ближе к полным графам.

Таблица 6.1 Экспериментальное сравнение алгоритма Брона-Кэрбоша и Модификации 1

Кол-во вершин	Заполненность	Время работы алгоритма Брона-Кэрбоша, (с)	Время работы алгоритма Модификация 1, (с)	Количество максимальных независимых множеств
40	0	0,48	0,008	1
40	5	12,5	2,3	3596
40	10	13,7	4,8	7092
40	15	14,6	6,9	9256
40	20	7,5	3,1	5166
40	25	4,0	1,6	2558
40	30	2,9	1,22	1669
40	35	2,0	0,9	1069
40	40	1,53	0,73	733
40	45	1,26	0,72	617
40	50	0,86	0,53	464
40	55	0,71	0,46	333
40	60	0,54	0,43	242
40	65	0,42	0,34	188
40	70	0,35	0,31	155
40	75	0,30	0,31	121
40	80	0,25	0,27	94
40	85	0,22	0,26	80

Таблица 6.1 продолжение				
40	90	0,20	0,26	64
40	95	0,18	0,27	38
40	100	0,18	0,29	40
100	75	9,3	8,3	1246
100	76	8,5	8,3	1233
100	77	7,8	7,8	1061
100	78	7,4	7,7	1039
100	79	6,9	7,3	1007
100	80	6,4	6,9	855
100	81	5,8	6,3	794
100	82	5,4	6,1	692
100	83	4,9	5,8	637
100	84	4,8	5,8	579
100	85	4,5	5,7	522
100	90	3,6	5,0	331
100	100	3,0	5,2	100

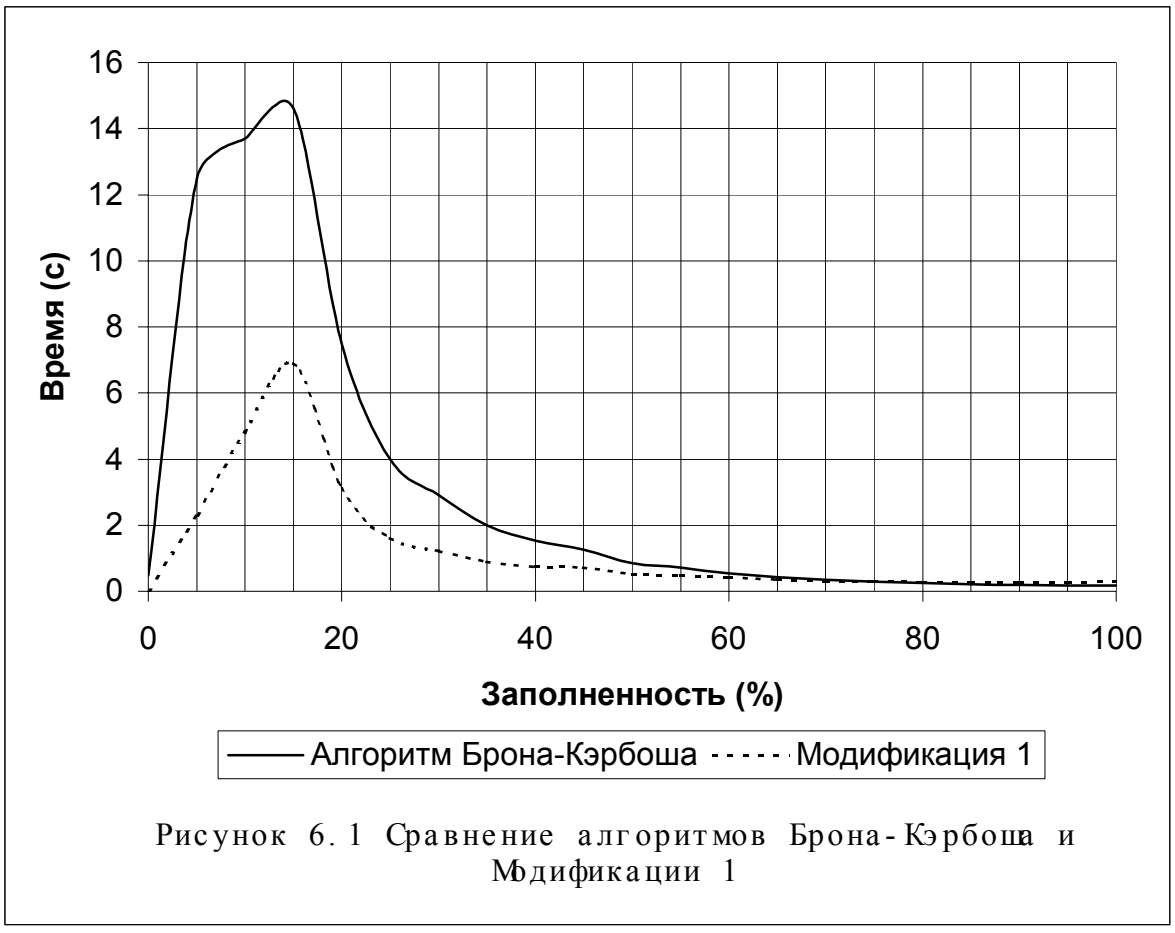

Рисунок 6.1 Сравнение алгоритмов Брона-Кэрбоша и Модификации 1

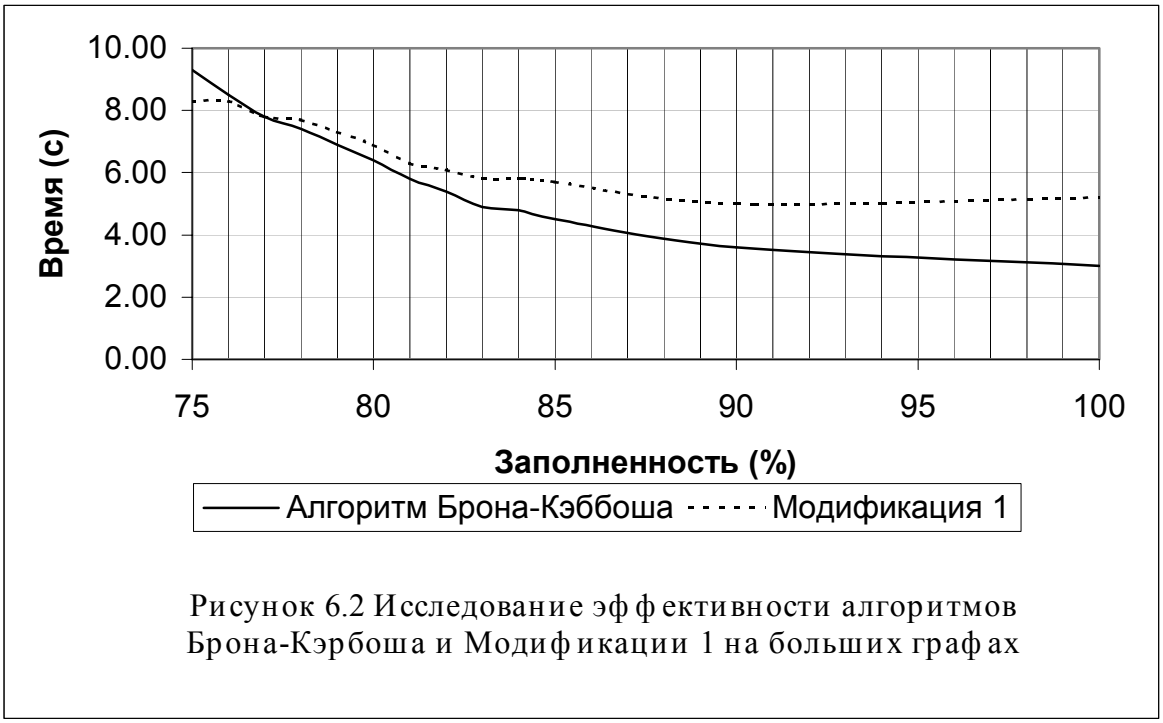

Рисунок 6.2 Исследование эффективности алгоритмов Брона-Кэрбоша и Модификации 1 на больших графах

6.2 Сравнение Модификаций 1, 2 и 3

Результаты тестирования алгоритмов приведены в таблице 6.2 (все эксперименты проводились на графах с 45 вершинами).

Модификация 2 на некоторых графах улучшила Модификацию 1, но на других ухудшила ее. При этом в среднем эффективность алгоритма не изменилась.

Что касается Модификации 3, то на большинстве графов она работает с такой же скоростью что и Модификация 1. Но на некоторых графах экономия времени становится заметной. Так при заполненности 34,9% экономия составляет 30мс, что на больших графах может увеличиться до нескольких часов. Но так как появление таких ситуаций крайне редко, рассматривать эти алгоритмы в будущем нецелесообразно.

Таблица 6.2 Сравнение алгоритма Брона-Кэрбоша и Модификаций 1, 2, 3 (45 вершин)

Кол-во ребер (%)	Алгоритм Брона-Кэрбоша	Модификация 1	Модификация 2	Модификация 3
0	0,74 с	0,011	0,011с	0,011
4,9	5мин 11с	3мин 16с	3мин 17с	3мин 17с

Таблица 6.2 продолжение

10	1мин 31с	46с	46с	46с
14,9	37с	16,4с	16,6с	16,5с
20	18,2с	8,5с	8,6с	8,5с
24,9	9,2с	3,6с	3,6с	3,5с
30	6,5с	2,7с	2,7с	2,7с
34,9	4,0с	1,88с	1,88с	1,85с
40	2,9с	1,47с	1,46с	1,45с
50	1,47с	0,88с	0,88с	0,87с
60	0,88с	0,6с	0,6с	0,5с
70	0,57с	0,48с	0,48с	0,47с
80	0,36с	0,41с	0,41с	0,40с
90	0,28с	0,39с	0,39с	0,39с
100	0,26с	0,43с	0,44	0,44

6.3 Сравнение Модификаций 1 и 4

В таблицах 6.3 и 6.4 представлены результаты экспериментальных исследований эффективности алгоритмов на графах на 20 и 50 вершинах соответственно.

Модификация 4 оказалась гораздо более эффективной, нежели все рассмотренные ранее алгоритмы. Для наглядности на основании таблиц были построены графики (см. рисунки 6.3 и 6.4) зависимости времени, потраченного на обработку от заполненности. При этом время работы алгоритма Брона-Кэрбоша принято за 1, а время работы остальных алгоритмов представлено как отношение ко времени работы алгоритма Брона-Кэрбоша.

Из графика к серии экспериментов на графах на 20 вершинах видна общая схема изменения эффективности алгоритмов относительно друг друга. Но при рассмотрении серии на 50 вершинах эта схема становится более наглядной и очевидной. Всплески графиков основанных на графах с небольшим количеством вершин можно объяснить погрешностью при измерении малых интервалов времени.

На рисунке 6.4 видно, что пустые графы обрабатываются рассматриваемыми модификациями приблизительно одинаково. При увеличении заполненности эффективность этих алгоритмов относительно алгоритма Брона-Кэрбоша

падает, причем параллельно. Минимум эффективности приходится на 10% (при 20 вершинах на 25%). После минимума эффективность обоих алгоритмов начинает расти, но у Модификации 4 этот рост более стремителен. Максимум эффективности приходится на 30-40%. Дальнейшее падение эффективности происходит также неодинаково: Модификация 1 теряет эффективность несколько быстрее и пересекает линию алгоритма Брона-Кэрбоша на 70-80%. После этого, ее эффективность продолжает уменьшаться, и на полных графах достигает наименьшего значения, намного меньшего, чем эффективность алгоритма Брона-Кэрбоша.

Модификация 1, в свою очередь, достигает соей наименьшей эффективности также на полных графах, однако она равна эффективности алгоритма Брона-Кэрбоша. Таким образом, для каждого графа, Модификация 4 оказывается более эффективной, чем наиболее эффективный для данного графа другой рассматриваемый алгоритм.

Для того, чтобы удостоверится в правильности выводов была дополнена серия опытов на больших графах (100 вершин): результаты работы алгоритма Брона-Кэрбоша и Модификации 1 взяты из таблицы 6.1, и проведены исследования алгоритма Модификация 4 на тех же графах. Результаты исследований представлены в таблице 6.5 и на рисунке 6.5. Они полностью подтверждают сделанные ранее выводы.

Таблица 6.3 Сравнение алгоритма Брона-Кэрбоша и Модификаций 1, 4 (20 вершин)

Кол-во ребер (%)	Алгоритм Брона-Кэрбоша (с)	Модификация 1 (с)	Модификация 2 (с)
0	0,03	0,001	0,001
4,7	0,09	0,014	0,013
10	0,11	0,02	0,02
14,7	0,07	0,019	0,018
20	0,08	0,02	0,02
24,7	0,08	0,03	0,03
30	0,05	0,02	0,01
34,7	0,05	0,02	0,017
40	0,06	0,03	0,02

Таблица 6.3 продолжение

44,7	0,05	0,03	0,02
50	0,04	0,02	0,019
54,7	0,04	0,02	0,02
60	0,03	0,02	0,018
64,7	0,02	0,02	0,017
70	0,02	0,02	0,016
74,7	0,02	0,02	0,016
80	0,02	0,02	0,017
84,7	0,02	0,02	0,015
90	0,018	0,02	0,015
94,7	0,017	0,02	0,015
100	0,017	0,02	0,017

Таблица 6.4 Сравнение алгоритма Брона-Кэрбоша и Модификаций 1, 4 (50 вершин)

Кол-во ребер (%)	Алгоритм Брона-Кэрбоша	Модификация 1	Модификация 2
0	0,96с	0,012с	0,011с
5	7мин 19с	3мин 57с	3мин 57с
10	12мин 40с	10мин 39с	10мин 37с
15	11мин 9с	1мин 8с	1мин 6с
20	56	36	34
25	22,7	12,2	11,3
30	10,9	5,2	4,6
34,9	6,5	3,0	2,5
40	4,9	2,3	1,8
50	2,2	1,28	0,95
60	1,22	0,86	0,60
70	0,71	0,16	0,45
80	0,47	0,52	0,34
90	0,34	0,47	0,29
100	0,27	0,45	0,27

Таблица 6.5 Сравнение алгоритма Брона-Кэрбоша и Модификаций 1, 4 (100 вершин)

Кол-во ребер (%)	Алгоритм Брона-Кэрбоша	Модификация 1	Модификация 2
75	9,3	8,3	5.8
76	8,5	8,3	5.4
77	7,8	7,8	5
78	7,4	7,7	4.9
79	6,9	7,3	4.6
80	6,4	6,9	4.2
81	5,8	6,3	4
82	5,4	6,1	3.8
83	4,9	5,8	3.6
84	4,8	5,8	3.6
85	4,5	5,7	3.5
90	3,6	5,0	3
100	3,0	5,2	3

Рисунок 6. 3 Сравнение времени работы алгоритмов на графах на 20 вершинах

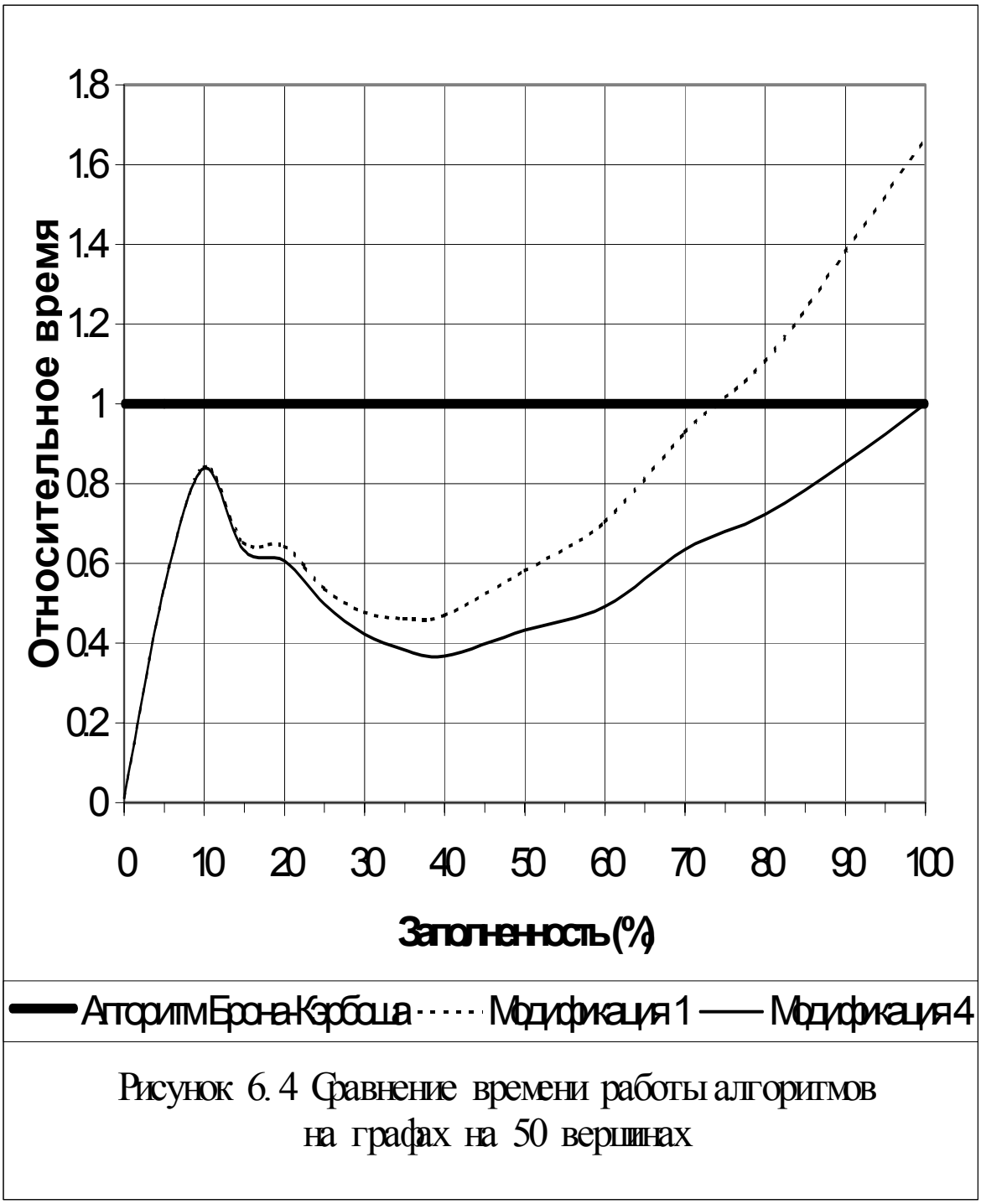

Рисунок 6.4 Сравнение времени работы алгоритмов на графах на 50 вершинах

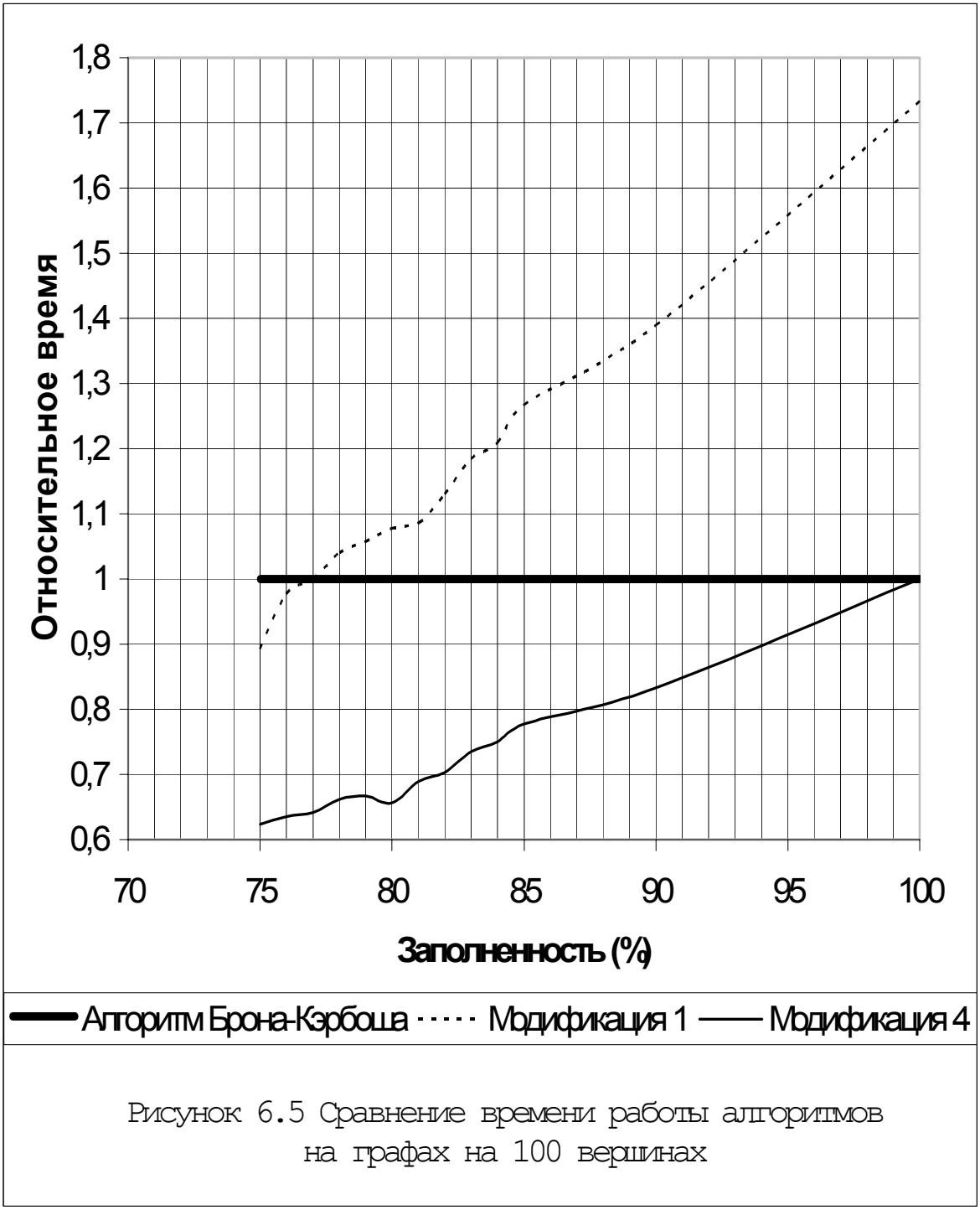

Рисунок 6.5 Сравнение времени работы алгоритмов на графах на 100 вершинах

ВЫВОДЫ

1. Обнаружены недостатки алгоритма Брона-Кэрбоша.
2. Проведена модификация алгоритма.
3. Показано, что полученная модификация эффективнее исходного алгоритма.

ПЕРЕЧЕНЬ ССЫЛОК

1. Ф.А. Новиков. Дискретная математика для программистов (учебник). СПб: Питер, 2001г. - 304стр.
2. Н. Кристофидес. Теория графов. Алгоритмический подход. М: Мир 1978г – 432стр.
3. М.Дж. Янг VISUAL C++ 6.0 Полное руководство. К: Ирина 2000г. – 1049стр.