АЛГОРИТМ ОПРЕДЕЛЕНИЯ КРАТЧАЙШИХ ПУТЕЙ МЕЖДУ ВСЕМИ ПАРАМИ ВЕРШИН В ГРАФЕ ПОСЛЕ СТЯГИВАНИЯ ДВУХ ВЕРШИН

Ладыженский Ю.В., Попов Ю.В.
Донецкий национальный технический университет, г. Донецк
кафедра прикладной математики и информатики
E-mail: ly@cs.dgtu.donetsk.ua

Abstract
Ladyzhensky Y.V., Popoff Y.V. An algorithm to define the shortest paths between all nodes in a graph after compressing of two nodes. *An algorithm to define a matrix of shortest paths between all nodes in a graph after compressing of two nodes is developed. A method to develop algorithms to define shortest paths in a dynamic graph is outlined. Experimental algorithm research is fulfilled.*

Введение

Цифровые системы логического управления на основе микросхем с программируемой логикой широко используются в промышленности. Моделирование систем логического управления является важным этапом проектирования, который обеспечивает проверку корректности функционирования системы. Высокая размерность и сложность цифровых устройств приводит к большим затратам времени и памяти при моделировании. Ускорение и ресурсы памяти можно получить при реализации моделей управления в компьютерных сетях. Некоторые критерии оптимальности отображения схем цифровых устройств на граф процессоров зависят от кратчайших путей между элементами схемы.

Анализ последних исследований показал, что существует множество алгоритмов определения кратчайших путей в графе [1, 2]. Наиболее эффективным алгоритмом для определения кратчайших путей между заданными парами вершин является алгоритм Дейкстры [1]. Последовательным применением этого алгоритма ко всем парам вершин можно найти кратчайшие пути между всеми парами вершин. Однако использование алгоритма Дейкстры для поиска путей между всеми парами вершин не является эффективным — для решения этой задачи есть более подходящий алгоритм Флойда [1]. Если граф обладает некоторыми особенностями или нам уже известны некоторые из существующих кратчайших путей, то эту информацию можно было бы учитывать для повышения скорости работы алгоритма. Алгоритм Флойда такую информацию не учитывает.

Таким образом, проблема быстрого поиска кратчайших путей между всеми парами вершин в графе после стягивания двух вершин не решается существующими алгоритмами.

Цели и задачи исследований, проведенных в статье, включают в себя разработку методики создания алгоритмов поиска кратчайших путей в динамических графах и разработку алгоритма определения кратчайших путей между всеми парами вершин в графе после стягивания двух вершин.

1. Операция стягивания двух вершин и определения стоимости пути

Задан граф $G=\{P,C\}$, где $P=\{p_i\}$ — множество вершин графа, а $C=\{c_{i,j}\}$ — множество дуг. $R=\{r_i\}$ — вектор весов вершин графа.

Стягиванием пары вершин i и j будем называть операцию, в которой: 1) все дуги, соединяющие стягиваемые вершины, удаляются; 2) все дуги, входящие во вторую вершину, перенаправляются в первую; 3) все дуги, исходящие из второй вершины, перенаправляются из первой; 4) вес первой вершины пересчитывается и становится равным суммарному весу стянутых вершин; 5) вес дуг пересчитывается и становится равным числу разных электрических цепей, соответствующих этим дугам; 6) вторая вершина удаляется (рис. 1).

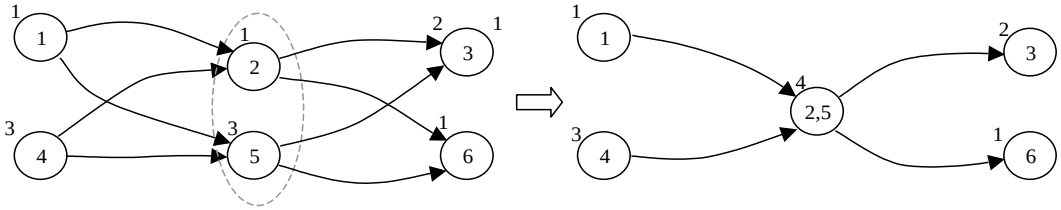

Рисунок 1 — Операция стягивания двух вершин

Обычно стоимость пути определяется как суммарный вес дуг, входящих в этот путь. Описанный в статье алгоритм используется в системе оптимального разрезания графа на части [3,4]. В этой системе стоимость пути определяется как суммарный вес всех вершин, входящих в путь, включая исходную и конечную вершины.

2. Случаи, в которых стоимость кратчайшего пути не изменяется

Проверим, изменится ли стоимость кратчайшего пути $t_{i,j}$ из вершины p_i в вершину p_j, если стягиваются вершины p_{i1} и p_{i2}. Результатом стягивания будет вершина p_{i1}, а вершина p_{i2} удаляется. Матрицу кратчайших путей на текущем шаге обозначим $T_k = \left\{t_{i,j}^k\right\}$, а на следующем шаге — $T_{k+1} = \left\{t_{i,j}^{k+1}\right\}$. $t_{i,j}$ не изменится в следующих случаях:

1) путь из вершины в нее же саму равен весу этой вершины и не может измениться: $t_{i,i} = r_i$;
2) если в графе после стягивания вершин нет пути из p_i в p_j через стянутую вершину: $\nexists m : t_{i,m}^{k+1} < \infty \,\&\, t_{m,j}^{k+1} < \infty$. Это возможно, если до стягивания вершин:
   - из p_i нельзя попасть ни в p_{i1}, ни в p_{i2}: $t_{i,i1}^k = \infty \,\&\, t_{i,i2}^k = \infty$ (рис. 2);
   - ни из p_{i1}, ни из p_{i2} нельзя попасть в p_j: $t_{i1,j}^k = \infty \,\&\, t_{i2,j}^k = \infty$ (рис. 3);
3) путь от вершины p_i до стягиваемых вершин длиннее, чем путь к вершине p_j: $t_{i,i1}^k > t_{i,j}^k \;\&\; t_{i,i2}^k > t_{i,j}^k$ (рис. 4);
4) если в графе после стягивания вершин существует путь через стянутую вершину и его стоимость при этом равна стоимости кратчайшего пути: $t_{i,j}^{k+1} = t_{i,i1}^{k+1} + t_{i2,j}^{k+1}$ (рис. 5 б, г):
   - если существует путь из p_i в p_j $(t_{i,j}^k < \infty)$, из p_i в p_{i1} $(t_{i,i1}^k < \infty)$ и из p_{i2} в p_j $(t_{i2,j}^k < \infty)$ и стоимость кратчайшего пути из p_i в p_j равна сумме стоимостей кратчайших путей из p_i в p_{i1} и из p_{i2} в p_j: $t_{i,j}^k = t_{i,i1}^k + t_{i2,j}^k$ (рис. 5 а,б);
   - если существует путь из p_i в p_j $(t_{i,j}^k < \infty)$, из p_i в p_{i2} $(t_{i,i2}^k < \infty)$ и из p_{i1} в p_j $(t_{i1,j}^k < \infty)$ и стоимость кратчайшего пути из p_i в p_j равна сумме стоимостей кратчайших путей из p_i в p_{i2} и из p_{i1} в p_j: $t_{i,j}^k = t_{i,i2}^k + t_{i1,j}^k$ (рис. 5 в,г).

   Не важно, проходит ли в исходном графе путь кратчайший путь через вершины p_{i1} и p_{i2} (рис. 5, а,б) или нет (рис. 5 в,г).

3. Случаи, в которых стоимость кратчайшего пути может измениться

Проверим, изменится ли стоимость кратчайшего пути $t_{i,j}$ из вершины p_i в вершину p_j, если стягиваются вершины p_{i1} и p_{i2}. Результатом стягивания будет вершина p_{i1}, а вершина p_{i2} удаляется. Матрицу кратчайших путей на текущем шаге обозначим $T_k = \left\{t_{i,j}^k\right\}$, а на следующем шаге — $T_{k+1} = \left\{t_{i,j}^{k+1}\right\}$. $t_{i,j}$ изменится в следующих случаях:

1) если после стягивания вершин путь через стянутую вершину станет короче (рис. 6). В графе до стягивания вершин это будет в следующих случаях:

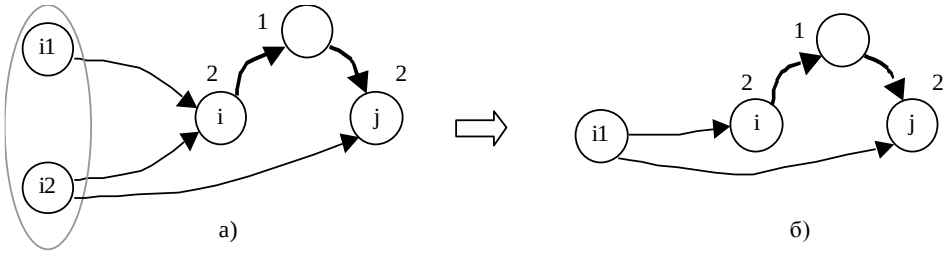

Рисунок 2 — Граф, в котором из вершины p_i нельзя попасть в стягиваемые вершины

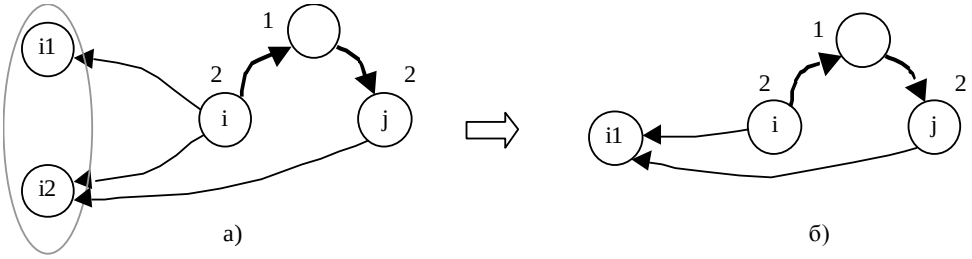

Рисунок 3 — Граф, в котором из стягиваемых вершин нельзя попасть в p_j

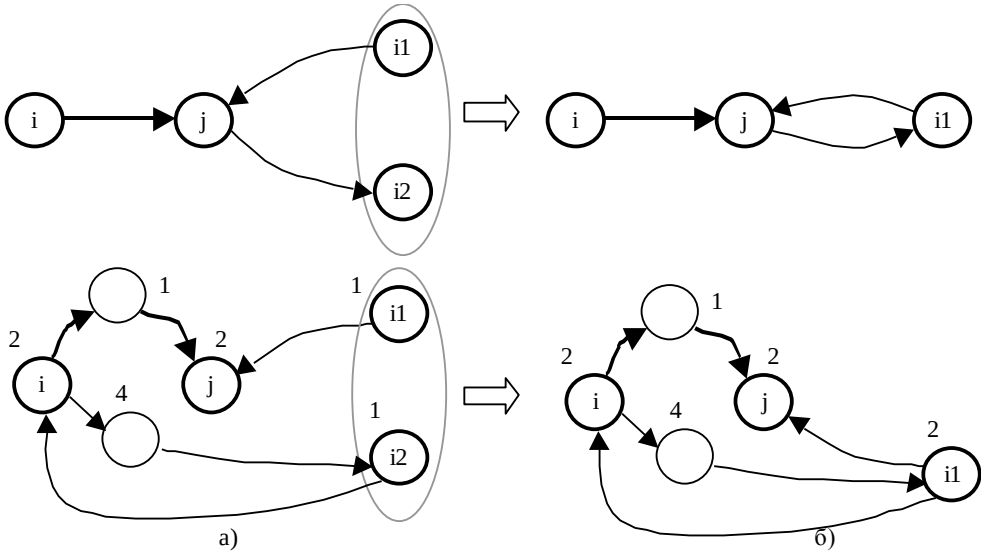

Рисунок 4 — Граф, в котором путь из p_i в стягиваемые длиннее, чем к p_j

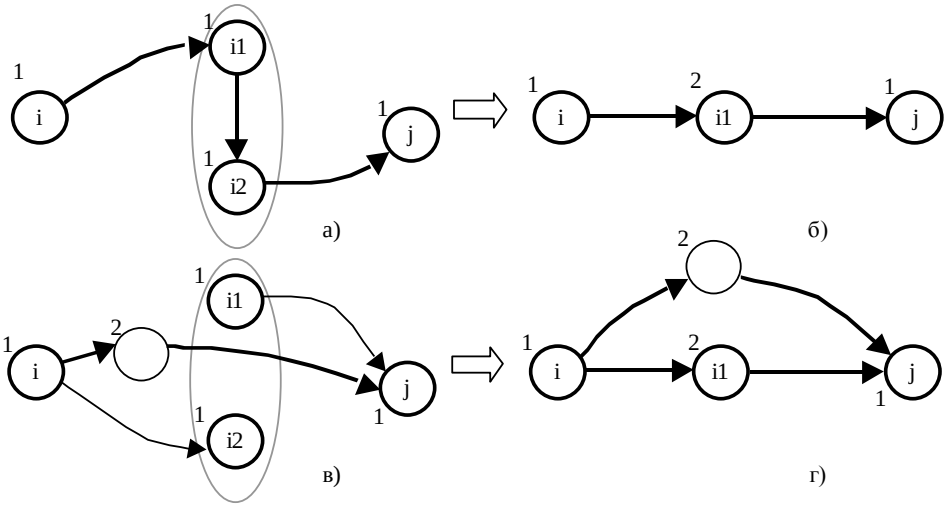

Рисунок 5 — Графы, в которых стоимость кратчайшего пути через стянутую вершину равна стоимости кратчайшего пути до стягивания вершин

– если есть путь из p_i в p_{i1} и из p_{i2} в p_j и нет пути из p_i в p_j, или стоимость этого пути больше суммы стоимостей кратчайших путей из p_i в p_{i1} и из p_{i2} в p_j (рис. 6 а,б):
$t^k_{i,i1} < \infty \,\&\, t^k_{i2,j} < \infty \,\&\, t^k_{i,j} > t^k_{i,i1} + t^k_{i2,j} \Rightarrow t^{k+1}_{i,j} = t^k_{i,i1} + t^k_{i2,j}$;

– если есть путь из p_i в p_{i2} и из p_{i1} в p_j и нет пути из p_i в p_j, или стоимость этого пути больше суммы стоимостей кратчайших путей из p_i в p_{i2} и из p_{i1} в p_j (рис. 6 в,г):
$t^k_{i,i2} < \infty \,\&\, t^k_{i1,j} < \infty \,\&\, t^k_{i,j} > t^k_{i,i2} + t^k_{i1,j} \Rightarrow t^{k+1}_{i,j} = t^k_{i,i2} + t^k_{i1,j}$;

2) если кратчайший путь из p_i в p_j проходит через одну из стягиваемых вершин: $t^k_{i,i1} + t^k_{i1,j} - r_{i1} = t^k_{i,j} \vee t^k_{i,i2} + t^k_{i2,j} - r_{i2} = t^k_{i,j}$. Путь может увеличиться до стоимости второй вершины (рис. 7):
$t^k_{i,i1} + t^k_{i1,j} - r_{i1} = t^k_{i,j} \Rightarrow t^{k+1}_{i,j} = t^k_{i,j} + r_{i2}$; $\quad t^k_{i,i2} + t^k_{i2,j} - r_{i2} = t^k_{i,j} \Rightarrow t^{k+1}_{i,j} = t^k_{i,j} + r_{i1}$
Путь из p_i в p_j может уменьшиться, если путь через p_{i2} короче (рис. 8):
$t^k_{i,i1} + t^k_{i1,j} - r_{i1} = t^k_{i,j} \,\&\, t^k_{i,i1} + t^k_{i2,j} < t^k_{i,j} \Rightarrow t^{k+1}_{i,j} = t^k_{i,i1} + t^k_{i2,j}$
$t^k_{i,i2} + t^k_{i2,j} - r_{i2} = t^k_{i,j} \,\&\, t^k_{i,i2} + t^k_{i1,j} < t^k_{i,j} \Rightarrow t^{k+1}_{i,j} = t^k_{i,i2} + t^k_{i1,j}$

3) если одна из рассматриваемых вершин является одновременно стягиваемой (рис. 9). Таких случаев может быть четыре:

– если $i = i1$ и существует путь из p_{i2} в p_j, то после стягивания стоимость кратчайшего пути из стянутой вершины в p_j не может превышать стоимости кратчайшего пути из p_{i2} в p_j плюс стоимость вершины p_{i1} до стягивания:
$i = i1 \,\&\, t^k_{i2,j} < \infty \Rightarrow t^{k+1}_{i,j} \le t^k_{i2,j} + r^k_{i1}$;

– если $i = i2$ и существует путь из p_{i1} в p_j, то после стягивания стоимость кратчайшего пути из стянутой вершины в p_j не может может превышать стоимости кратчайшего пути из p_{i1} в p_j плюс стоимость вершины p_{i2} (рис. 9 в,г):
$i = i2 \,\&\, t^k_{i1,j} < \infty \Rightarrow t^{k+1}_{i,j} \le t^k_{i1,j} + r_{i2}$;

– если $j = i1$ и существует путь из p_i в p_{i2}, то после стягивания стоимость кратчайшего пути из вершины p_i в стянутую вершину не может превышать стоимости кратчайшего пути из p_i в p_{i2} плюс стоимость вершины p_{i1} (рис. 9 а,б):
$j = i1 \,\&\, t^k_{i,i2} < \infty \Rightarrow t^{k+1}_{i,j} \le t^k_{i,i2} + r_{i1}$;

– если $j = i2$ и существует путь из p_i в p_{i1}, то после стягивания стоимость кратчайшего пути из вершины p_i в стянутую вершину не может превышать стоимости кратчайшего пути из p_i в p_{i1} плюс стоимость вершины p_{i2}: $j = i2 \,\&\, t^k_{i,i1} < \infty \Rightarrow t^{k+1}_{i,j} \le t^k_{i,i1} + r_{i2}$;

4) в случаях, когда стоимость кратчайшего пути может увеличиться, оценки 1), 2) и 3) служат ограничениями сверху: если путь увеличится, то он станет не больше чем эта оценка. Путь может увеличиться, но на меньшую величину, либо вообще не увеличиться (рис. 10). Такие пары вершин будем называть С-парами вершин.

Существование этого случая означает, что для поиска точных стоимостей кратчайших путей следует использовать модифицированный алгоритм Флойда. Модификация состоит в том, что поиск путей осуществляется не между всеми парами вершин, а только между С-парами вершин. При этом уже имеется информация о стоимости кратчайших путей между всеми остальными парами вершин. Если отбросить все пары вершин, стоимость кратчайших путей между которыми не изменилась и не стала меньше, то таких С-пар вершин останется не очень много.

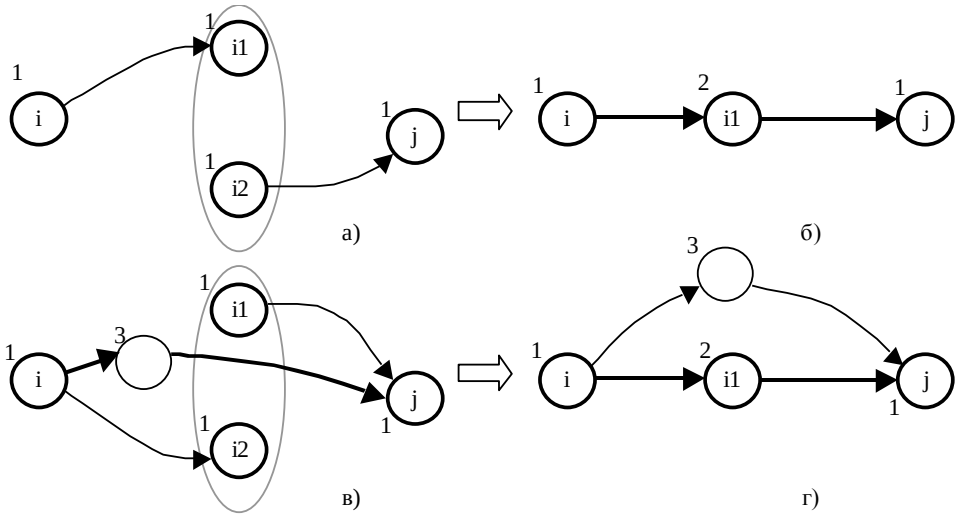

Рисунок 6 — Графы, в которых после стягивания появился более короткий путь

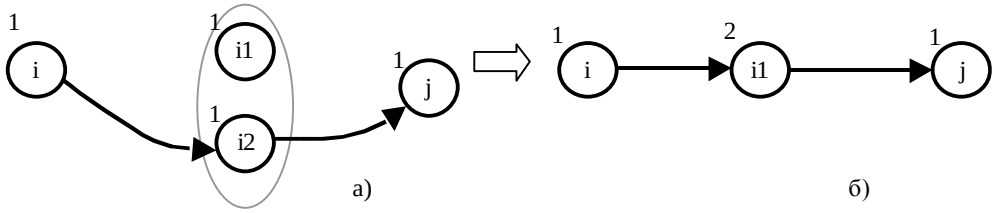

Рисунок 7 — Граф, в котором путь из p_i в p_j проходит через стягиваемую вершину

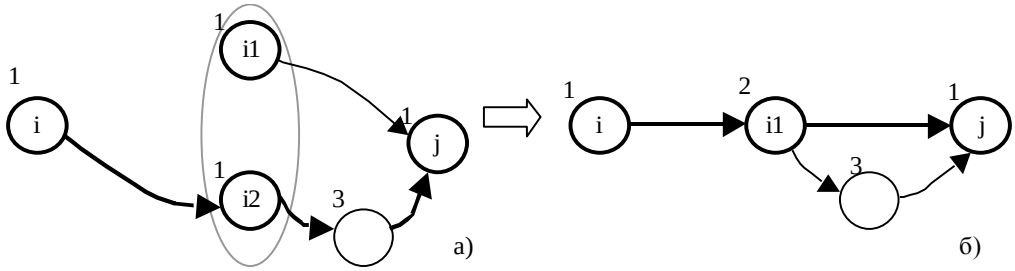

Рисунок 8 — Граф, в котором кратчайший путь из p_{i2} в p_j короче, чем из p_{i1} в p_j

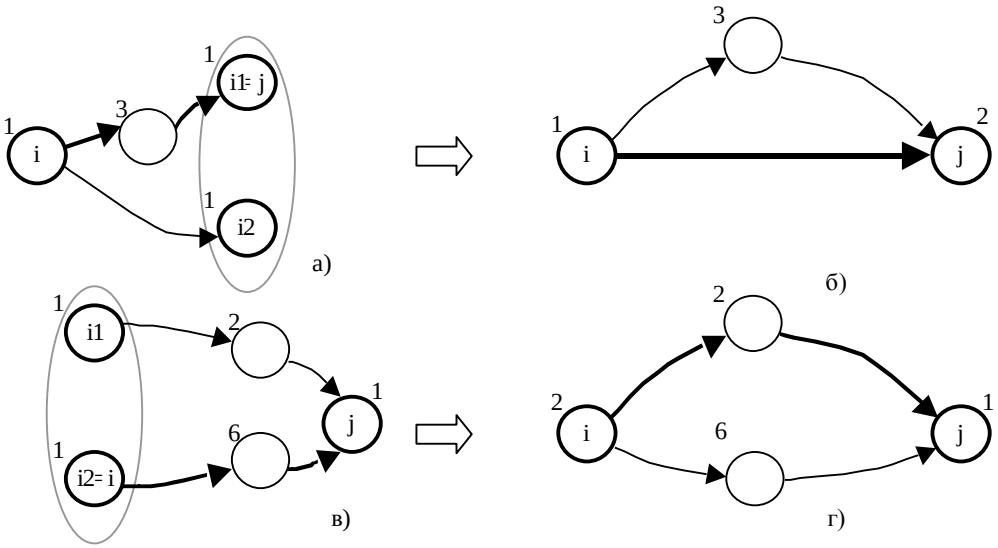

Рисунок 9 — Графы, в которых кратчайший путь между двумя вершинами уменьшается, потому что одна из вершин — стягиваемая

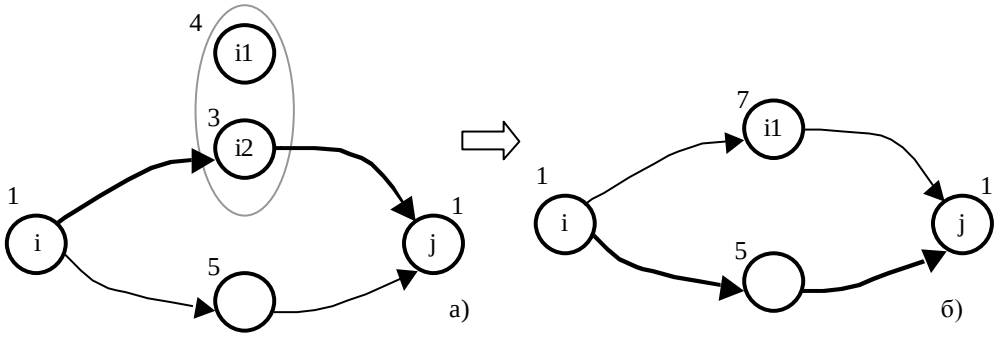

Рисунок 10 — Граф, в котором стоимость пути, меньше приведенных оценок сверху

4. Алгоритм определения кратчайших путей между всеми парами вершин в графе после стягивания двух вершин

Рассмотрим некоторый граф $G = \{P, C\}$. Допустим, нам нужно найти кратчайшие пути не между всеми парами вершин, а только между некоторыми заданными парами $L = \{p_i \to p_j\}$. Если значения кратчайших путей между парами вершин $\overline{L}$ (все остальные пары вершин, не входящие в множество L) не известны, то нельзя упростить алгоритм таким образом, чтобы не искать стоимости кратчайших путей между парами вершин $\overline{L}$. Это связано с тем, что стоимости кратчайших путей между парами вершин L зависят от стоимостей кратчайших путей $\overline{L}$. При применении алгоритма Флойда это означает, что для поиска кратчайших путей между парами вершин L требуется перебирать все возможные пары вершин, и те, которые входят в L, и те, которые входят в $\overline{L}$. Если значения кратчайших путей между парами вершин $\overline{L}$ уже известны, то для поиска кратчайших путей L можно перебирать только пары вершины, входящие в L и пропустить перебор пар вершин $\overline{L}$.

Учитывая все случаи, в которых стоимость кратчайших путей может измениться или не измениться, рассмотрим алгоритм быстрого пересчета матрицы кратчайших путей между всеми парами вершин для графа, в котором две вершины были стянуты в одну (рис. 11).

Алгоритм состоит из двух частей:

1) пересчет матрицы кратчайших путей. Для тех пар вершин, в которых стоимость кратчайших путей изменится, в матрицу записывается ограничение сверху;

2) для тех пар вершин, в которых $t_{i,j}^{k+1} > t_{i,j}^{k}$, производится проверка альтернативных путей по модифицированному алгоритму Флойда, в котором перебираются только заданные пары вершин. Вторую часть алгоритма можно не выполнять в случае, если точные стоимости кратчайших путей не интересуют, а важен только факт наличия пути и приблизительная оценка стоимости этого пути.

5. Экспериментальное исследование предложенного алгоритма

Корректность и скорость работы алгоритма проверялась на графах, соответствующих схемам iscas89 [5]. Результаты экспериментального тестирования алгоритма приведены в таблице 1.

Эксперименты проводились на компьютере Celeron 2.8 GHz 512 MB RAM. Оперативной памяти достаточно для того, чтобы во время экспериментов весь граф помещался в оперативной памяти и файл подкачки не использовался.

В таблице 1 указано имя схемы, на графе которой проводился эксперимент, количество вершин в графе, соответствующем в этой схеме. Одной вершине соответствует один элемент схемы. Во время эксперимента в графе, соответствующем схеме, стягивались две вершины, и после стягивания вычислялась матрица кратчайших путей двумя способами — при помощи алгоритма быстрого пересчета и при помощи алгоритма Флойда.

```
Вход:
   T: array [1..k,1..k] of real; // матрица кратчайших путей на предыдущем шаге
                                 // значение <0 означает пути нет
   R: array [1..k] of real;      // вектор весов всех вершин
   i1,i2:integer;                // номера стягиваемых вершин
                                 // результат- i1, вершина i2 удаляется
Выход:
   T1: array [1..k,1..k] of real; // кратчайшие пути после стягивания вершин

  T1:=T; //сохраняем старый вариант матрицы
  for i=0 to k do
  if i=i2 then continue; //пути из удаляемой вершины не интересуют
  if T[i][i1]<0 and T[i][i2]<0 then continue; //нельзя попасть в стягиваемую
  for j=0 to k do
    if i=j then continue; //Кратчайший путь из вершины в нее саму
                          // всегда равен стоимости этой вершины
    if j=i2 then continue;//Пути в удаляемую вершину не интересуют
    if T[i1][j]<0 and T[i2][j]<0 then continue; //В вершину нельзя попасть из
                  // стягиваемой, значит к этой вершине пути не изменятся
    if i=i1 then  //если рассматриваем путь из стянутой вершины
      if T[i2][j]>=0 and (T[i][j]<0 or T[i][j]+R[i2]>T[i2][j]+R[i1]) then
          //К какой вершине путь короче?
        T1[i][j]:=T[i2][j]+R[i1]
      else
        T1[i][j]:=T[i][j]+R[i2];
      continue;
    end if;
    if j=i1 then
      if T[i][i2]>=0 and (T[i][j]<0 or T[i][j]+R[i2]>T[i][i2]+R[i1]) then
         //К какой вершине путь короче?
          T1[i][j]:=T[i][i2]+R[i1]
        else
          T1[i][j]:=T[i][j]+R[i2];
      continue;
    end if;
    if T[i][i2]<0 or T[i][j]<T[i][i2] then continue;
    if T[i][i1]+T[i2][j]<=T[i][j] or T[i][j]<0 then
      //рассматриваем путь i - i1 - i2 - j
      T1[i][j]:=T[i][i1]+T[i2][j];
      continue;
    end if;
    if T[i][i2]+T[i1][j]<=T[i][j] or T[i][j]<0 then
      //рассматриваем путь i - i2 - i1 - j
      T1[i][j]:=T[i][i2]+T[i1][j];
      continue;
    end if;
    if T[i][i1]+T[i1][j]-R[i1]=T[i][j] then //путь может проходить через i1
      if T[i2][j]>=0 and T[i][j]+R[i2]>T[i][i1]+T[i2][j] then
          T1[i][j]:=T[i][i1]+T[i2][j]
        else
          T1[i][j]:=T[i][j]+R[i2];
      continue;
    end if;
    if T[i][i2]>=0 and T[i2][j]>=0 and T[i][j]>=0 and
       T[i][i2]+T[i2][j]-R[i2]=T[i][j] then //путь может проходить через i2
      if T[i1][j]>=0 and T[i][j]+R[i1]>T[i][i2]+T[i1][j] then
          T1[i][j]:=T[i][i2]+T[i1][j]
        else
          T1[i][j]:=T[i][j]+R[i1];
      continue;
    end if;
  end for //по j
end for //по i
```

Рисунок 11 — Алгоритм быстрого пересчета матрицы кратчайших путей

Общее количество экспериментов соответствует количеству способов выбрать две вершины из всего множества вершин: $N = C_n^2 = \frac{n!}{2!(n-2)!} = \frac{n*(n-1)}{2}$, где n — количество вершин в графе.

Время работы алгоритма приведено в среднем по каждому эксперименту. Для схем s27, s208 и s400 было проведено все множество экспериментов (стягивались все вершины со всеми остальными). Для схем s713 и s1196 были проведены не все возможные эксперименты, а только указанное количество. 11263 эксперимента для схемы s1196 проводились 23 часа.

Таблица 1 — Результаты экспериментального тестирования алгоритма

Схема	**Кол-во вершин**	**Кол-во экспериментов (всего / было проведено)**	**Время поиска приближенного решения, сек**	**Время поиска точного решения, сек (T_m)**	**Время работы алгоритма Флойда, сек (T_f)**	**Ускорение (T_f / T_m)**
s27	13	78 / 78	0,0001	0,0011	0,0034	3,1
s208	112	6216 / 6216	0,0012	0,0099	0,045	4,5
s400	185	17020 / 17020	0,0048	0,031	0,206	6,7
s713	412	84666 / 22318	0,057	0,45	2,37	5,2
s1196	547	149331 / 11263	0,082	0,72	5,64	7,8

Выводы

1. Разработан алгоритм определения кратчайших путей между всеми парами вершин в графе после стягивания двух вершин.
2. На примере этого алгоритма сформулирована методика построения алгоритмов поиска кратчайших путей между всеми парами вершин в динамических графах. Суть методики состоит в том, что следует определить случаи, в которых стоимость кратчайших путей не изменится, уменьшится или увеличится. На основании этих случаев строится алгоритм.
3. Проведено экспериментальное исследование полученного алгоритма. Показано, что для точного нахождения весов кратчайших путей между всеми парами вершин этот алгоритм работает в среднем в 7 раз быстрее алгоритма Флойда. Для поиска приблизительной оценки стоимостей кратчайших путей предложенный алгоритм работает в среднем в 70 раз быстрее алгоритма Флойда.

Перспективным направлением дальнейших исследований является повышение эффективности поиска точных значений кратчайших путей между всеми парами вершин.

Литература

1. Дискретная математика для программистов / Ф.А. Новиков — СПб: Питер, 2001. — 304 с.; ил.
2. Кристофидес Н. Теория графов. Алгоритмический подход. — М.: Мир, 1997.
3. Ладыженский Ю.В., Попов Ю.В. Программная система для исследования протоколов синхронизации при распределенном событийном логическом моделировании // Наукові праці Донецького национального технічного університету. Серія: Обчислювальна техніка та автоматизація. Випуск 74 / Редкол.: Башков Є.О. (голова) та ін. — Донецк: Вид-во ДонНТУ, 2004. — 427 с. — С. 201–209.
4. Ladyzhensky Y.V., Popoff Y.V. A program system for distributed event-driven logic simulation of VHDL designs // Proceedings of East-West Design & Test Workshop (EWDTW'04). — Yalta, Alushta, Crimea, Ukraine, September 23–26, 2004. — Kharkov: Kharkov National University of Radioelectronics, 2004. — 289 p. — P. 203 – 209.
5. Brglez F., Bryan D., Kozminski K. Combinational Profiles of Sequential Benchmark Circuits. ISCAS'89 Benchmark Circuits. – Proc. IEEE Int. Symposium on Circuits and Systems. — May 1989. — Pp. 1929–1934.