Приложения теории графов. Алгоритмы поиска максимального потока в графе. Алгоритм Форда-Фалкерсона. Алгоритм Диница.

План.

Алгоритмы поиска максимального потока в графе.

Алгоритм Форда-Фалкерсона

Алгоритм Диница

Цели:

1. Формирование знаний об основных понятиях и определениях потока в сети.

2. Формирование знаний об основных способах и алгоритмах нахождения максимального потока и минимального разреза в сети.

Задачи:

1. Сформировать теоретические знания необходимые при нахождении максимального потока и минимального разреза в сети.

2. Содействовать расширению профессиональной компетенции в области основных понятий и способов нахождении максимального потока и минимального разреза в сети как инструмента для выполнения проектирования кабельной структуры компьютерной сети.

Определение 1: Сетью называется связный орграф без петель.

Определение 2: Потоком в сети называется некоторая функция, которая ставит в соответствие дуге некоторое число-вес дуги.

Для определения потока в сети используют алгоритм Форда-Фалкерсона:

а) ищем любую цепь из истока графа в сток;

б) каждой дуге приписываем возможный больший поток из истока в сток (записываем его через дробь с весом дуги; при этом поток не может превысить вес дуги, но может быть ему равен);

в) если поток становится равен весу дуги, то эта дуга является насыщенной, то есть через нее нельзя пройти при рассмотрении цепей в графе;

г) так перебираем все возможные цепи, пока станет невозможно попасть из истока в сток;

д) поток в сети будет равен сумме потоков всех дуг, инцидентных стоку графа (следует заметить, что сумма потоков всех дуг, инцидентных стоку графа равна сумме потоков всех дуг, инцидентных истоку графа).

Задача: Дана сеть. Определить поток в сети.

Решение:

Следуя вышеописанному алгоритму, рассмотрим следующие цепи:

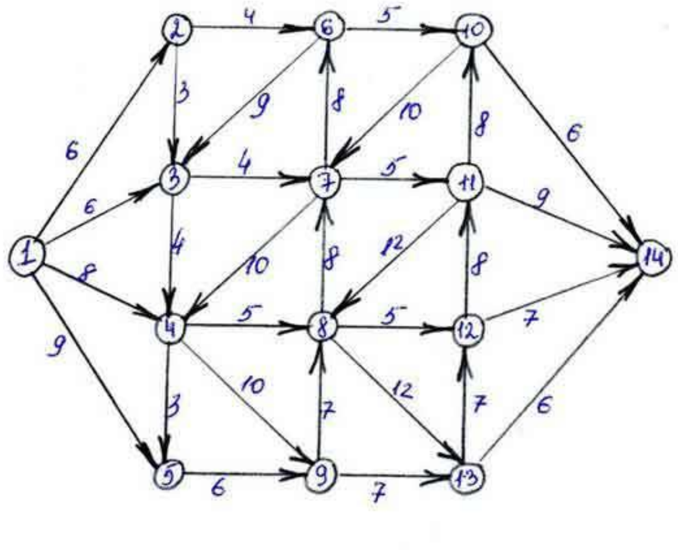
1-5-9-13-14;

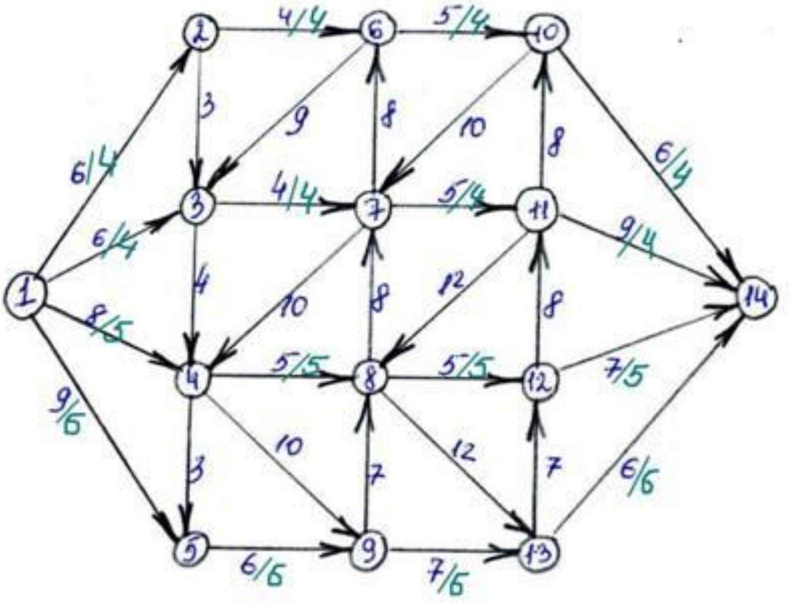
1-4-8-12-14;

1-3-7-11-14;

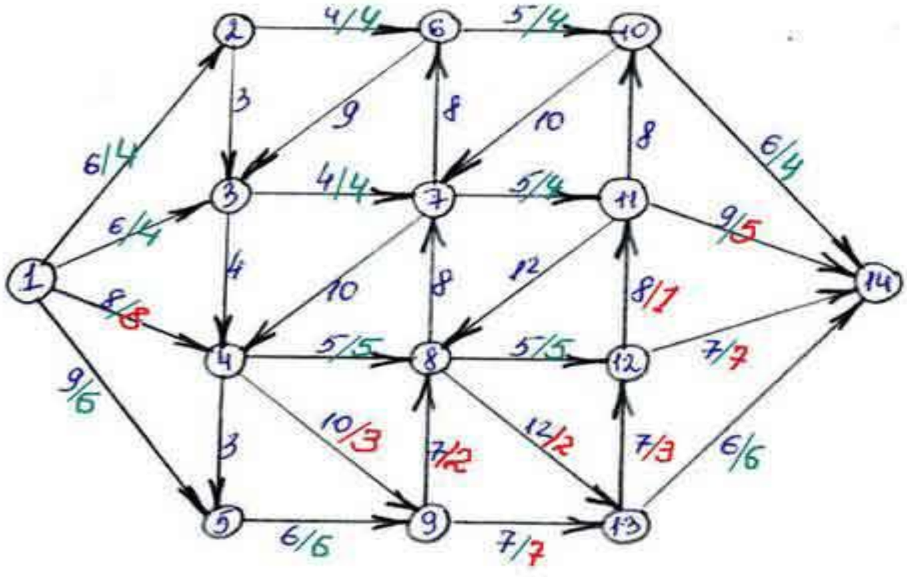
1-2-6-10-14

Расставив потоки у соответствующих дуг, получим:

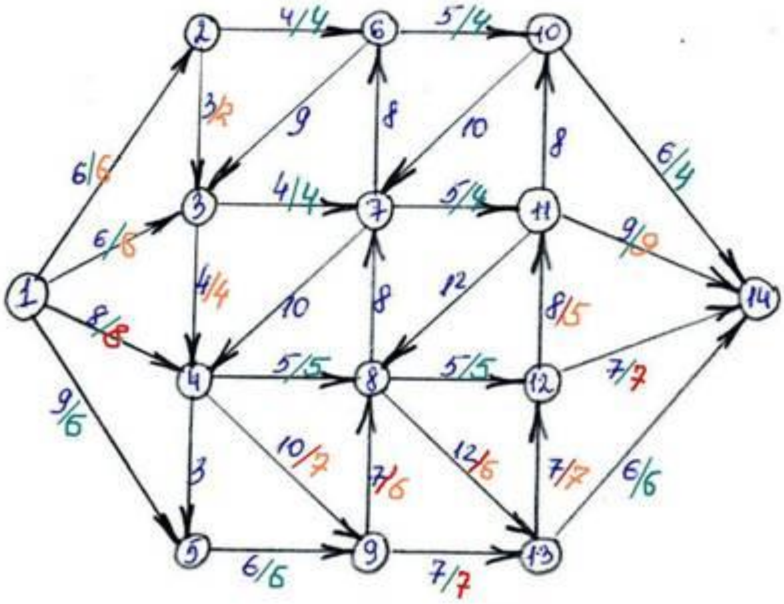




Далее будем выбирать цепи так, чтобы насыщались дуги, инцидентные истоку.
1-4-9-13-12-14 (здесь поток каждой дуги увеличивается на единицу);
1-4-9-8-13-12-14 (здесь также на единицу);
1-4-9-8-13-12-11-14 (и здесь тоже на единицу).
После этой последовательности цепей получим насыщенную дугу 1-4.



Следующая последовательность цепей дает нам окончательный результат:
1-3-4-9-8-13-12-11-14 (здесь поток каждой дуги увеличивается на двойку);
1-2-3-4-9-8-13-12-11-14 (здесь поток каждой дуги опять увеличивается на двойку).
<u>Поток</u> в сети равен 4+9+7+6=6+6+8+6=26.
<u>Насыщенные дуги</u>: 1-4,1-3,1-2, 5-9, 9-13, 13-14, 13-12, 4-8, 8-12, 12-14, 3-7, 11-14, 2-6, 3-4.



Алгоритм Диница

Этот алгоритм, опубликованный в 1970 г., имел огромное значение.

На протяжении 10 лет все достижения в решении задачи о максимальном потоке базировались на методе Диница.

Основная идея метода – алгоритм состоит из фаз, на которых поток увеличивается сразу вдоль всех кратчайших цепей определенной **длины.**

Для этого на i-й фазе строится вспомогательная бесконтурная сеть (layered network). Эта сеть содержит все увеличивающие цепи, длина которых не превышает k_i, где k_i – длина кратчайшего пути из s в t. Величину k_i называют длиной вспомогательной сети.

Рассмотрим работу алгоритма на i-й фазе:

Шаг 1. Построение вспомогательной сети.

С помощью поиска в ширину мы движемся из источника сети в сток по допустимым дугам, добавляя их в S_k и увеличивая k. Дуга u добавляется с $c_k(u) = c(u) – f(u)$. Как только достигнут сток сети, он помечается величиной k и k становится «фиксированной». Мы продолжаем поиск в ширину, но он уже не ведется из вершин v, для которых $q(s, v) \geq k$. Таким образом, вспомогательная сеть будет содержать подсеть исходной. Если поиск в ширину не достиг стока, то работа алгоритма прекращается. Если k больше (k может только увеличиваться) k_i, то мы находимся на i + 1 фазе с $k_{i+1} = k$.

Шаг 2. Поиск псевдомаксимального потока.

В построенной нами бесконтурной сети длины k ищется псевдомаксимальный поток. Под псевдомаксимальным потоком понимается такой поток f, для которого не существует увеличивающих цепей длины k. Найденный поток переносится в исходную сеть. Затем мы вновь переходим к первому шагу.

Для построения псевдомаксимального потока используется поиск в ширину (с ограничением на длину пути). Пусть на j-ой итерации был найден путь из s в t. Пустим по этому пути поток f_j. Это значит, что как минимум одна дуга вспомогательной сети является насыщенной. Удалим все насыщенные дуги. В результате могут образоваться «тупики»: вершины, из которых не выходит ни одна дуга (кроме стока), вершины, в которые не входит ни одна дуга (кроме источника) и изолированные вершины. Их также следует удалить со всеми инцидентными им дугами. Это в свою очередь может привести к образованию новых тупиков. Корректировка производится, пока во вспомогательной сети не останется ни одного тупика. Изменим пропускные способности оставшихся дуг по формуле $c_k(u) = c_k(u) – f_j(u)$.

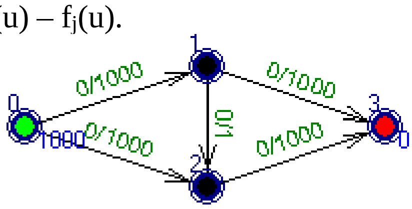


Рис. 1 Сеть с двумя тупиками

На рис. 4 вершины 2 и 5 являются тупиками. После их удаления будут последовательно удалены все вершины сети.

Поиск f_j следует продолжать до тех пор, пока вспомогательная сеть $\neq \varnothing$. Очевидно, что псевдомаксимальный поток $f = \sum f_j$. Псевдомаксимальный поток можно хранить в какой-либо структуре, но на практике найденные потоки f_j обычно сразу переносятся в исходную сеть S. Заметим, что после нахождения f_i и корректировки сети, поиск в ширину можно продолжать с ближайшей к s не подвергшейся изменениям дуги найденного пути.

После завершения работы алгоритма исходная сеть будет содержать максимальный поток.

Рассмотрим сеть, приведенную на рис. 2.

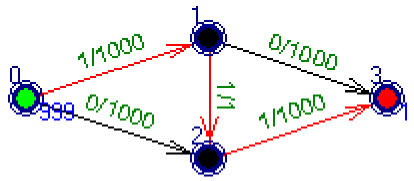


Рис. 2. Сеть с максимальным потоком, найденным по алгоритму Диница

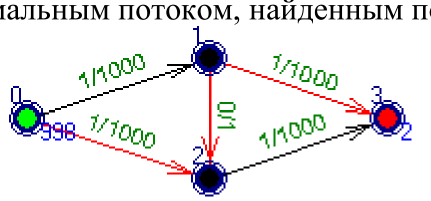


Рис. 3. Вспомогательная сеть на 1 фазе, $k_1 = 2$

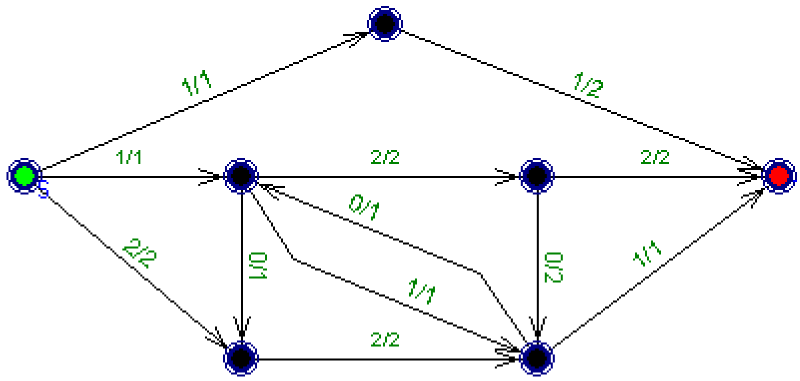


Рис. 4. Вспомогательная сеть на 2 фазе, $k_2 = 3$

На рис. 7 хорошо видно, что псевдомаксимальный поток сети обычно не является максимальным.

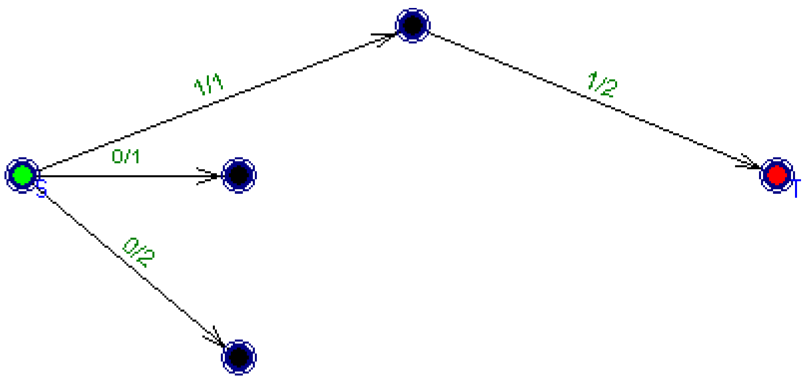


Рис. 5. Вспомогательная сеть на 3 фазе, $k_3 = 5$

Метод поразрядного сокращения невязок

В зарубежной литературе этот метод получил название capacity scaling. Он был рассмотрен Эдмондсом и Карпом в 1972 г. и независимо Диницом в 1973 г. Если с его помощью модифицировать алгоритм кратчайших увеличивающих цепей или алгоритм Диница, то можно получить оценки быстродействия $O(m^2 \log U)$ и $O(mn \log U)$ соответственно, где $U = \max(C_{ij})$.

Возьмем достаточно большое целое K (т. н. масштаб). Пусть все пропускные способности дуг округлены с недостатком до ближайшего кратного числу K и формируют сеть CS_k. Тогда при увеличении потока в этой сети, он будет возрастать на целое, кратное K. Таким образом, для построения потока мощности M требуется не более M / K итераций. Как только в сети CS_k найден максимальный поток, уточним пропускные способности дуг, выбрав K' и округлив их с недостатком до ближайшего кратного числу K', где K' < K.

После корректировки сети CS_k и потока f(u), получим сеть CS_k' с потоком f'(u). Если K кратно K', то имеющийся поток останется целочисленным.

Полученный на предыдущей итерации поток следует увеличить, чтобы он вновь стал максимальным. Подбирая, таким образом, масштаб, мы все более полно будем использовать пропускные способности дуг. В случае целочисленных пропускных способностей, максимальный поток будет получен после его увеличения в масштабе K = 1.

На практике выбирают $K = \log_2 U$. На i-ой итерации берется масштаб $2^{(K - i + 1)}$. В целочисленном случае для нахождения максимального потока достаточно K + 1 итерации.