Алгоритм Диница с масштабированием. Аналогично Форду-Фалкерсону с масштабированием, будем по очереди запускать алгоритм Диница, ставя ограничения на величину остаточных пропускных способностей, необходимых для того, чтобы ребро попадало в остаточную сеть. Общий вид алгоритма будет выглядеть как-то так:

```
int max_flow() {
    for (SCALE = 1 << 30; SCALE > 0; SCALE >>= 1) {
        while (bfs(s, t)) {
            while(path = dfs(s, t)) {
                relax(path);
            }
        }
    }
}
```

Какую мы получим выгоду от масштабирования? При переходе между итерациями мы знаем, что величина текущего минимального разреза не больше, чем $2 \cdot SCALE \cdot E$. То есть мы знаем, что теперь у нас будет суммарно не более $O(E)$ насыщений в рамках одной итерации.

Раньше мы оценивали Диница так: у нас был dfs, в котором мы считали число успешных и неуспешных итераций a_i и b_i. Мы получали, что $\sum a_i = O(VE)$, $\sum b_i = O(E)$. Сделаем новую оценку, используя тот факт, что насыщений всего $O(E)$. Здесь t_d — число путей длины d.

$$\sum_{d=0}^{V}\sum_{i=0}^{t_d}(a_{d,i}+b_{d,i}) \le \sum_{d=0}^{V}(d\cdot t_d + E) \le VE + V\cdot\sum_{d=0}^{V} t_d = O(VE)$$

.

В итоге на каждой итерации масштабирования мы сделаем $O(VE)$ шагов, получаем оценку на время работы $O(VE\log C)$.

Оценки Карзанова. Пусть в нашей сети единичные пропускные способности (например, в задаче о поиске максимального паросочетания). Тогда утверждается, что алгоритм Диница отработает за $O(E\sqrt{V})$.

Как это показать? Сначала покажем, что в рамках одной слоистой сети алгоритм делает $O(E)$ шагов. Для этого надо оценить сумму a_i, которая не превышает E, потому что после каждого успешного действия с ребром оно насыщается (читай — удаляется).

Отдельно покажем, что если определить $P = \sum p(v,u)$ (где $p(v,u)$ — потенциал вершины, сколько через нее максимально может пройти), то алгоритм Диница будет работать за $O(\sqrt{P}VE)$.

Определим вершинный разрез как такое множество вершин, через которое будет проходить любой путь из s в t.

Сделаем первые $\sqrt{P}$ итераций Диница. Теперь в нашей сети будет не меньше, чем $\sqrt{P}$ слоев. Поскольку эти слои не пересекались, то оставшийся поток в сети не больше минимума потока через вершинные разрезы, то есть он не больше, чем $\sqrt{P}$. А значит, алгоритм Диница найдет не более чем $\sqrt{P}$ путей. Таким образом, мы получим сложность $O(\sqrt{P}VE)$. В применении к задаче о паросочетании $P = V$, а в рамках одной слоистой сети происходит не $O(VE)$, а $O(E)$ шагов, откуда получается сложность $O(E\sqrt{V})$.

Стоимостные потоки. Введем w — дополнительную стоимость потока вдоль ребра. Мы будем считать, что $w_{v,u} = -w_{u,v}$. Общей стоимостью назовем $\sum \frac{f_{v,u} \cdot w_{v,u}}{2}$ (у нас дважды учитывается поток вдоль одного ребра из-за обратных ребер). Обычно стоимость хотят минимизировать.

Выделяют несколько задач: min-cost-flow, min-cost-k-flow, min-cost-max-flow. В первой мы минимизируем вес, во второй мы ищем минимальный вес потока величины k, а в третьей найти максимальный поток, а среди таких потоков поток минимальной стоимости. Есть еще задача о циркуляции минимальной стоимости, к которой все эти задачи сводятся, если замкнуть $t \to s$ правильно заданным ребром.

Критерий оптимальности потока. Поток считается минимальным по стоимости потоком размера k, если и только если в его остаточной сети нет циклов отрицательного веса. Почему? Ну пусть мы нашли какой-то другой поток размера k, который имел меньшую стоимость. Если рассмотреть $f_1 - f_2$, то это будет циркуляцией. Если рассмотреть ее декомпозицию, в ней будут только циклы, при этом стоимость отрицательная. Значит, какой-то из циклов имеет отрицательную стоимость. Противоречие.

В доказательстве было узкое место: у нас могли нарушиться условия на пропускные способности, и поэтому не факт, что мы можем добавить этот новый цикл. Но на самом деле это правда, потому что f_1 и f_2 были корректными потоками. А именно, при добавлении у нас поток по ребру будет не больше чем максимум из потоков по ребру, что остается корректным потоком.

Алгоритм поиска min-cost-k-flow. Если у нас был поток f минимальной стоимости веса k, то поток минимальной стоимости веса $k+1$ можно получить вдоль кратчайшего в плане стоимости пути в остаточной сети G_f. А это значит, что если мы сначала найдем минимальный поток веса 0 (брать отрицательные циклы, пока можем), а затем по очереди брать кратчайшие пути p из s в t.

Почему утверждение верно? Пусть был какой-то другой поток лучше. Тогда это значит? Что $w(f_1) + w(p) > w(f_2)$. Рассмотрим разность между этими потоками. В ней есть отрицательный цикл c. Рассмотрим $p + c$. Если его декомпозировать, то получим пути, каждый из которых не дешевле, чем p (потому что p был кратчайшим), и циклы, которые имели неотрицательную стоимость (иначе ими можно было бы уменьшить f_1). Тогда это значит, что поток $p + c$ имеет стоимость не меньше, чем p, противоречие.