Вычисление статистик π, λ, c на графе

Миронович Светлана

Рассмотрим процесс нахождения значений $\pi(l)$ и $\lambda(l)$ на ориентированном графе. Заметим, что принцип нахождения π отличается от процесса нахождения λ только тем, на каком графе выполняется поиск: на графе линейных или разностных переходов. Поэтому будем рассматривать некий обобщенный граф, в котором есть только два типа ребер (ребра веса 0 и ребра веса 1), и для этого графа будем искать маршруты длины l с минимальным количеством единиц на ребрах (если на вход подают граф разностных переходов, то это то же самое, что и $\pi(l)$; если же на вход подали граф линейных переходов, то это то же самое, что и $\lambda(l)$). Для простоты найденный такой маршрут мы будем называть $\pi(l)$ вне зависимости от того, на каком графе это значение вычислялось.

Пусть у нас есть граф с n вершинами и m ребрами. Формализуем задачу. Нам требуется найти маршрут длины l с минимальным количеством единиц на нем. Поскольку ребра могут иметь вес только 0 или 1, то эта задача эквивалентна более общей задаче: нахождение самого легкого (дешевого) маршрута длины l в графе. Будем называть вершины по нормерам: вершина 0, 1, 2 и т.д. Заметим также, что относительно графов линейных переходов и разностных переходов нам известно, что в вершину 0 нет никаких ребер (за исключением петли в саму себя). А в таком случае добавим ребра из 0 во все остальные вершины, и присвоим всем ребрам вес 0. Понятно, что эти ребра не изменят финальный ответ, поскольку с увеличением числа l искомые маршруты будут представлять собой хождение по некоторому циклу с минимальным "удельным"весом единиц в нем (т.е. по такому циклу, что отношение его веса к его длине будет минимальным). Но эти ребра не могут задать такой цикл, поскольку ни одно ребро не ведет в 0.

Итак, мы получили задачу, при которой у нас есть граф, в котором:

- все ребра имеют вес только 0 или 1
- из вершины 0 есть ребра в каждую вершину, и вес ребра 0
- нас интересует самый дешевый маршрут длины l в графе

Заметим, что это то же самое, что найти самый дешевый маршрут длины $l+1$, который начинается в вершине 0, поскольку вес маршрута от ребра из вершины 0 не меняется, и из вершины 0 мы можем добраться в любую вершину за одно ребро. Итого, мы получаем новую задачу: на полученном графе самый дешевый путь длины l из вершины 0. Можем вспомнить алгоритм Беллмана-Форда, который вычисляет кратчайшие пути из одной вершины графа во все остальные. Этот алгоритм делает n шагов, и на каждом шаге k называет самые дешевые пути длины k до каждой из вершин. Т.е. инвариант алгоритма Беллмана-Форда: на каждом шаге l мы знаем самый дешевый путь длины l до каждой из вершин. Но поскольку нам известны легчайшие пути длины l до каждой из вершин, то это значит, что один из этих путей и есть искомый для $\pi(l)$ путь. Т.е. для того, чтобы найти легчайший путь длины l, нам достаточно пройтись по легчайшим путям длины l до каждой из вершин, и найти самый дешевый из них. Следовательно, инвариант алгоритма Беллмана-Форда нам подходит.

Заметим, что истинная наша цель состоит не в том, чтобы для любого l найти значение $\pi(l)$, но в том, чтобы найти значение предела: $\lim_{l\to\infty} \frac{\pi(l)}{l} = c$. Ранее было доказано:

$$c = min_{cycle \in Graph} \frac{weight(cycle)}{length(cycle)}.$$

Назовем такой цикл, в котором достигается минимум, оптимальным циклом.

Заметим, что на $n+1$-ом шаге алгоритма Беллмана-Форда каждая вершина либо недостижима, либо ее маршрут уже вошел в цикл. Т.е. если мы рассмторим оптимальный маршрут длины $n+1$ для любой вершины, там либо уже будет видна повторяющаяся последовательность вершин (т.е. цикл, например "3 2 5 1 3 2 5 1 3 2 5..."), либо за такое количество шагов-ребер нельзя добраться до данной вершины. Это следует из того, что в графе только n вершин, и это значит, что за $n+1$ шаг мы точно посетим какую-то вершину дважды. Но когда мы приходим в какую-то вершину, мы из нее уходим всегда по одному и тому же пути, и следовательно, когда мы приходим в вершину, в которой мы уже были, мы уже не можем не идти по циклу, который построили.

Т.к. алгоритм Беллмана-Форда находит оптимальные по весу маршруты, и в пределе оптимальный маршрут - это маршрут, зацикленный в оптимальном цикле, то это означает, что все вершины, которые образуют оптимальный цикл, начиная с некоторого шага алгоритма Беллмана-Форда будут исключительно находиться внутри этого цикла (а точнее, для всех вершин оптимального цикла на $n+1$ шаге алгоритма уже можно построить цикл, в котором они находится, и это будет именно оптимальный цикл). В таком случае, для того, чтобы найти оптимальный цикл, поступим следующим образом:

- Совершим $n+1$ шаг алгоритма Беллмана-Форда. Ассимптотическая сложность: $O(nm)$.

- Для каждой вершины найдем цикл, в котором она находится. Для этого просто будем идти в обратном направлении по тому пути, который нам предложит алгоритм Беллмана-Форда, до тех пор, пока не встретим ту вершину, с которой мы стартовали обход в обратном направлении. Более того, когда мы построили какой-то цикл из вершин $u1u2..uk$, начиная из $u1$, мы уже можем не строить циклы для $u2, u3, ..., uk$, поскольку они, очевидно, будут находиться том же самом цикле. Это означает, что на этом этапе нам нужно не более $O(n)$ шагов, поскольку в каждой вершине мы будем не более 2 раз.

- Среди полученных циклов найдем оптимальный. Этот шаг имеет сложность $O($сумма длин всех найденных циклов$)$, и очевидно, это не более чем $O(n)$.

  Итого общая сложность нахождения величины $c : O(nm + n + n) = O(nm)$.

  Проблему в данном случае представляет то, что в худшем случае $O(m) = O(n^2)$, а также тот факт, что на самом деле $n = 2^k$ и изменять мы будем не n, а k. Итого общая сложность нахождения c для некоторой криптосистемы из k блоков в худшем случае: $O(2^{3k})$.