Доклад о линеаризации вариационного графа.

Я хочу доложить об достижениях в вопросе линеаризации вариационного графа. Задача формулируется следующим образом: имея вариационный граф нужно отсортировать его вершины таким образом, чтобы получить как можно меньше reversing edges и cut width, где reversing edges – это ребра, идущие из более поздней вершины сортировки в раннюю, а cut width – это «длина» всех ребер в том смысле, что мы считаем длиной ребра количество gap между соседними вершинами сортировки, через которые ребро проходит. Имеет смысл говорить об average cut width, равном cut width, деленном на количество gap, то есть на длину сортировки минус один.

Мы реализовали два алгоритма линеаризации: алгоритм, предложенный David Haussler и алгоритм, описанный Eades в статье 1993 года, с некоторыми изменениями, улучшающими его работу применительно к нашей ситуации.

Была проведена оценка сложности работы алгоритмов и тестирование на количество reversing edges, average cut width и время работы. Тестировались 3 алгоритма: Eades, Haussler и vg sort, базовая топологическая сортировка, реализованная в vg.

Оценка сложности приведена на слайдах 3 и 4. Как мы видим, алгоритм Haussler в худшем случае имеет кубическую сложность, в лучшем – линейную. Алгоритм Eades всегда работает за линейное время.

В процессе реализации алгоритма Haussler были обнаружены следующие проблемы.

Во-первых, алгоритм мог пропускать некоторые вершины потому, что после удаление минимального сечения они не попадали ни в in-growth, ни в out-growth (см пример на слайде 7). Эта проблема была решена следующим образом: после окончания основной работы алгоритма шла проверка на существование неотсортированных вершин и добавление их к вершине из отсортированной части, до которой можно добраться поиском в глубину.

Во-вторых, алгоритм мог «неправильно» работать на фрагментах, где на индел накладывается другой индел или SNP. Эта проблема обозначена на 8 слайде. Единственный способ решения – присвоение таких весов ребрам графа, чтобы поиск in- и out-growth всегда шел в нужном направлении.

Теперь о результатах тестирования.

Сначала поговорим про тестовый набор. Мы строили набор тестовых графов по следующему принципу: взяв backbone из 8000 вершин «накладывали» на него вариации по мере уменьшения их длины (от больших делеций и дупликаций к SNP). В результате получается вариационный граф с заданным набором вариаций. Далее мы увеличивали количество вариаций на этом графе и смотрели как меняются результаты работы алгоритмов. Сначала увеличивали количество всех вариаций одновременно в 2, 3, ..., 7 раз. Затем вернулись к исходному набору и увеличивали количество вариаций только одного типа в 2, 4, 8, 16, 32 раз.

Графики с результатами работы алгоритмов представлены на слайдах 12-17. Как мы видим, алгоритм Haussler дает большее число reversing edges, чем алгоритм Eades. Особенно заметно это становится при увеличении числа вершин в графе. Однако по average cut width ситуация обратная. Алгоритм Eades всегда проигрывает Haussler. Встроенная сортировка vg находится посредине по обоим параметрам. Однако такой результат базовой топологической сортировке связан с тем, что в тестовом наборе графы имели только один path, а именно backbone, с которого начиналось их построение. Есть основания полагать, что при увеличении числа path в графе, а следовательно, присвоении нетривиальных весов большему числу ребер, результаты реализованных сортировок будут намного опережать vg sort.

Наконец, время работы. Из-за кубической сложности алгоритма Haussler время его работы невероятно сильно растет с ростом размера графа. Так если граф из 16 000 вершин сортируется за 30 секунд, то уже для 256 000 вершин сортировка одного графа занимает более 9 часов.