Московский Государственный Университет имени М.В.Ломоносова

Факультет биоинженерии и биоинформатики

Методы множественного выравнивания аминокислотных последовательностей на примере алгоритма MUSCLE.

Курсовая работа

студентки 2 курса

Поверенной Ирины Владимировны

Тьютор: Ройтберг Михаил Абрамович,

к.ф.-м.н., зав. лабораторией прикладной математики

Института математических проблем биологии РАН

Москва

2010

ОГЛАВЛЕНИЕ

1. **Постановка задачи.**

1.1. Что такое множественное выравнивание и где оно используется. MSA-программы.

Множественное выравнивание последовательностей $v_1,\ldots, v_n$ – это набор последовательностей $u_1,\ldots, u_n$, обладающих следующими свойствами:

a) Все последовательности u_i, где $i=1,\ldots,n$, имеют одинаковую длину l.

b) Для каждого i $(i=1,\ldots,n)$ последовательность u_i получена из v_i добавлением вхождений специального символа (этот символ называется «индел», в качестве индела ниже используется символ ‘-‘).

k-ой *колонкой* выравнивания $(k=1,\ldots,l)$ называется набор символов $<u_1[k], u_2[k], \ldots, u_n[k]>$. Неформально говоря, считается, что символы одной колонки соответствуют друг другу. В идеале, при выравнивании гомологичных последовательностей предполагается, что все эти символы произошли от одного символа общего предка сравниваемых последовательностей.

Множественные выравнивания белковых последовательностей имеют большую важность для многих приложений. В качестве примеров можно привести построение филогенетических деревьев и оценку их качества, предсказание вторичной структуры белков, построение доменных профилей, поиск активного центра, предсказание 3D-структуры. Существует большое количество алгоритмов множественного выравнивания последовательностей (multiple sequence alignment – MSA) [1]. Два свойства MSA-программ имеют первостепенное значение для пользователей: биологическая точность и вычислительная сложность (время и требования к памяти). Сложность имеет большую актуальность в связи с быстрым ростом баз данных последовательностей, которые сейчас содержат так много представителей крупных белковых семейств, что это превышает возможности большинства текущих программ. Получение биологически точных выравниваний также очень важно, ведь даже лучшие методы иногда не позволяют выровнять очевидные консервативные мотивы.

1.2. Программа MUSCLE.

Программа MUSCLE была разработана Робертом С. Эдгаром (R.C.Edgar) в 2004 году [2,3]. На сегодняшний день она считается наиболее быстрой и точной программой

множественного выравнивания белковых последовательностей. Существуют три варианта программы. Каждый из них предоставляет пользователю соответственно:

- наибольшую точность (MUSCLE с настройками по умолчанию)
- наиболее высокую скорость (MUSCLE-fast)
- тщательно подобранный компромисс между этими двумя вариантами (MUSCLE-p (синоним MUSCLE-prog)).

Сравнение скорости и точности программы MUSCLE с программами CLUSTALW, T-Coffee и MAFFT-скриптом FFTNS1 было произведено на основе результатов, полученных для тестовых баз данных: BAliBASE, PREFAB, SABmark и SMART. Авторами были протестированы все три варианта программы. По полученным данным выяснилось, что MUSCLE-fast является самым быстрым алгоритмом на всех тестовых БД, достигая ту же среднюю точность выравнивания, что и программа CLUSTALW, но тратя на это время на два-три порядка меньше. На обычном современном компьютере MUSCLE-fast может выровнять 1000 последовательностей средней длины 282 за 21 секунду (см. п.5).

1.3. Цель работы.

Цель работы – разобраться в алгоритме программы MUSCLE и сопутствующих алгоритмах и понятиях, по возможности, перепроверить результаты работ [2,3], касающиеся быстродействия программы и установить пределы ее применимости.

2. **Алгоритм Muscle: общее описание.**

2.1. Общая схема.

Работа алгоритма состоит из трех стадий. На первой стадии производится построение предварительного множественного выравнивания. Это выполняется путем построения предварительного дерева семейства последовательностей, а затем итеративного множественного выравнивания (progressive alignment) в соответствии с этим деревом [4,5].

На следующих стадиях это множественное выравнивание улучшается. На второй стадии итеративно выполняется улучшение дерева и для каждого нового дерева

стандартным способом строится итеративное множественное выравнивание. На третьей стадии выполняется улучшение выравнивания при фиксированном дереве за счет «перевыравнивания слоев» этого выравнивания.

Выигрыш во времени достигается программой MUSCLE за счет того, что при построении дерева в ней не производится попарное выравнивание всех последовательностей. Вместо этого на стадии 1 мера сходства между последовательностями грубо определяется на основе сравнения частот *k*-слов. Уточнение дерева на стадии 2 и последующее уточнение выравнивания на стадии 3 позволяет достичь хорошего (по сравнению с другими известными алгоритмами) качества, сохраняя при этом выигрыш по времени.

2.2. Стадия 1: предварительное дерево последовательностей и предварительное итеративное выравнивание.

На первой стадии строится итеративное множественное выравнивание на основе упрощенной меры сходства между последовательностями. Это множественное выравнивание уточняется на последующих стадиях.

Итеративное множественное выравнивание строится в соответствии со стандартной схемой (см. п. 2.1) Детали реализации этой схемы кратко описаны ниже, более подробно – см. раздел 3.

1) Мера сходства между последовательностями.

Сходство между последовательностями вычисляется на основе подсчета количества вхождений *k*-слов (подробнее см. п. 3.1). Это – одна из основных отличительных черт программы MUSCLE. Такое вычисление сходства позволяет избежать дорогостоящего попарного выравнивания всех последовательностей. Значение меры сходства – число из отрезка [0..1].

2) Оценка расстояния (матрица расстояний)

Для каждой пары последовательности вычисляется мера расстояния по формуле:

$$d_{\text{k-слова}} = 1 - F,$$

где F – это мера сходства (см. п.1.1).

3) Построение дерева.

По матрице расстояний с помощью метода UPGMA [6] или метода neighbor-joining [7] строится дерево; определяется его корень. Каждому листу построенного дерева соответствуют последовательность. По умолчанию используется метод UPGMA (см. п.3.2.).

4) Итеративное выравнивание.

Дерево обходится от листьев к корню. Для каждого узла *V* строится множественное выравнивание последовательностей, соответствующих листьям, достижимым из данного узла. Пусть *VL* и *VR* - сыновья узла *V*. Множественное выравнивание *A(V)* строится путем профиль-профильного выравнивания множественных выравниваний *A(VL)* и *A(VR)* (см. п.3.3).

2.3. Стадия 2: улучшение итеративного выравнивания за счет уточнения дерева.

Главная цель второй стадии заключается в улучшении дерева последовательностей и построении соответствующего ему нового итеративного выравнивания. Эта стадия может быть повторена несколько раз, однако, как правило, она выполняется один раз (см. п.6).

1) Мера сходства.

Сходство каждой пары последовательностей определяется как доля совпадений в парном выравнивании этих последовательностей, извлеченного из текущего множественного выравнивания. Как видим, и на этой стадии попарное выравнивание всех последовательностей не производится.

2) Оценка расстояния.

Расстояния определяются по доле совпадений на основе формулы Кимуры [8], см. п.3.4.

3) Построение дерева.

Дерево строится на основе вычисленной матрицы расстояний аналогично этапу 3 стадии 1.

4) Сравнение деревьев.

Определяется, есть ли отличия между предыдущим и новым деревом. Цель этапа – отметить в новом дереве все такие внутренние узлы, что соответствующее поддерево совпадает с некоторым поддеревом старого дерева и указать соответствующий узел старого дерева (т.к. листья сравниваемых деревьев помечены последовательностями из исходного набора, то узел нового дерева может соответствовать не более, чем одному узлу старого дерева). Это, очевидно, может быть выполнено за время $O(N)$, где N - количество последовательностей, параллельным просмотром деревьев от листьев к корню.

Если деревья совпадают (т.е. корень нового дерева помечен), то выполнение второй стадии прекращается. В противном случае – переходим к этапу 5 стадии 2.

5) Итеративное выравнивание.

Строится новое итеративное выравнивание в соответствии с новым деревом; в узлах, отмеченных на этапе 4 (там, где новое дерево совпадает со старым), используются старые выравнивания. В итоге в корне дерева получаем новое множественное выравнивание исходного набора последовательностей.

После этого выполнение второй стадии прекращается. В принципе, можно итерировать выполнение этой стадии, пока на этапе 4 не будет достигнуто совпадение деревьев или просто выполнять стадию 2 некоторое заранее фиксированное количество раз. Однако, как отмечает Р. Эдгар, это не приводит к улучшению качества итогового выравнивания.

2.4. Стадия 3: окончательное уточнение выравнивания.

На третьей стадии выполняется итерационное уточнение выравнивания при фиксированном дереве последовательностей. Для этого набор последовательностей разными способами разбивается на две части, и соответствующие подвыравнивания выравниваются между собой с помощью выравнивания профилей.

Разбиение множества последовательностей на две части производится согласованно с построенным деревом. А именно, из дерева удаляется некоторое ребро, что индуцирует разбиение листьев (а с ними – множества последовательностей) на два класса. В ходе выполнения стадии 3 перебираются все ребра дерева, т.е. описанные ниже этапы 1 – 3 выполняются *2N-3* раза.

1) Выбор разбиения на две части.

Ребра перебираются в порядке уменьшения расстояния от корня. Т.е. мы начинаем с ребер наиболее далеких от корня и последовательно доходим до ребер, выходящих из корня.

2) Получение профиля.

Профиль (множественное выравнивание) каждого подмножества извлекается из текущего множественного выравнивания. Столбцы, не содержащие остатков (т.е. содержащие только инделы), отбрасываются.

Замечание. В отличие от общепринятой терминологии, Р. Эдгар использует термин «профиль» в качестве синонима термина «множественное выравнивание». Мы ниже так же будем это делать. В частности «профиль-профильное выравнивание» (см. ниже) означает выравнивание множественных выравниваний.

3) Перевыравнивание

Два профиля, полученные на этапе 2 стадии 3, перевыравниваются друг с другом с помощью профиль-профильного выравнивания (см. ниже п. 3.3).

4) Принять / Отклонить

Вычисляется вес множественного выравнивания, которое получено на предыдущем этапе; вес рассчитывается по методу Sum of Pairs (SP-вес, см. п. 3.5). Если вес увеличивается, новое выравнивание сохраняется, в противном случае оно отбрасывается. Если после перебора всех ребер множественное выравнивание не изменилось, или если определенное пользователем максимальное число итераций была достигнуто, алгоритм прекращается, в противном случае он (в зависимости от настроек алгоритма) возвращается к шагу 1 этапа 3. В программе MUSCLE

пользователь может указывать максимальное количество итераций стадии 3; по умолчанию это количество равно 14.

3. **Алгоритм Muscle: детали реализации.**

3.1. Мера сходства, использующая *k*-слова.

k-слово - это слово (т.е. непрерывная символьная последовательность) длины *k*. Пусть *X* - символьная последовательность и τ - *k*-слово. Через L_X обозначается длина последовательности *X*, через $n_X(\tau)$ – количество вхождений слова τ в последовательности *X* (перекрытие вхождений допускается).

Пусть *X*, *Y* – две последовательности. Тогда в качестве меры их сходства используется величина:

$$F(X, Y) = \Sigma_{\tau} (\min (n_X(\tau), n_Y(\tau)) / (\min (L_X , L_Y) - k + 1)) \qquad (1)$$

(сумма берется по всем возможным *k*-словам *τ).* Знаменатель в формуле (1) является оценкой максимально возможного количества выровненных вхождений данного *k*-слова *τ* для последовательностей данных длин (оценка достигается только, если все буквы в *τ* – одинаковые). Числитель является оценкой максимально возможного количества выровненных вхождений данного *k*-слова *τ* с учетом количеств вхождений $n_X(\tau)$, $n_Y(\tau)$ в последовательности *X* и *Y.*

Отметим, что формулу (1) можно переписать следующим образом:

$$F(X, Y) = (\Sigma_{\tau} (\min (n_X(\tau), n_Y(\tau))) / (\min (L_X , L_Y) - k + 1), \qquad (1a)$$

откуда видно, что при любых *X*, *Y* выполнено

$$0 \le F(X, Y) \le 1.$$

Программа MUSCLE использует еще одну меру сходства F^{Binary}, которая определяется следующим образом:

$$F^{\text{Binary}}(X, Y) = (\Sigma_\tau \delta_{XY}(\tau)) / (\min (L_X, L_Y) - k + 1). \quad (2)$$

Здесь, если τ есть в обеих последовательности, то $\delta_{XY}(\tau)$ равен 1, иначе равен 0. Так как в одной последовательности относительно редко встречается много копий отдельного k -слова, это зачастую является хорошим приближением к F.

Бинарное приближение позволяет значительно улучшить скорость и сократить требования к памяти. Оно позволяет хранить сведения о вхождениях k-слов в виде битовых векторов и вычислять значение $\Sigma_\tau \delta_{XY}(\tau)$ с помощью битовых операций.

Вычисление величин F и F^{Binary} для пары последовательностей выполняется за время $O(L)$, где L – суммарная длина последовательностей. Общее время на вычисление всех мер сходства для N последовательностей – $O(N^2L)$. Для сравнения: в программе CLUSTALW [9] расчет мер сходства производится на основе выравнивания последовательностей и занимает время $O(N^2L^2)$, это наиболее времяемкий этап работы программы CLUSTALW. Потребности в памяти для вычисления значений F и F^{Binary} составляют $O(A^k)$, где A – длина используемого алфавита.

По умолчанию программа MUSCLE использует 6-слова и меру сходства F.

Подобно программе MAFFT [10], программа MUSCLE использует приведенные выше формулы не для исходного алфавита последовательностей, а для «классфикационного» алфавита (subset alphabet, см. [11]), т.е. для алфавита, в котором символы обозначают классы, содержащие два или более видов остатков. Очевидно, что значение меры сходства при использовании классификационного алфавита больше либо значению этой меры идентичности при подсчете в исходном алфавите.

Ранее было показано [12], что мера сходства k -слов хорошо коррелирует с долей совпадения. Если выбран алфавит, в котором есть высокие вероятности внутриклассовой замены и низкие вероятности междуклассовой, то можно ожидать, что его использование при вычислении мер (1) и (2) приведет к биологически более адекватным результатам, особенно, при сравнении эволюционно далеких гомологов. Однако доказательств этого (например, с помощью компьютерных экспериментов) нет.

По умолчанию MUSCLE использует 6-классовый алфавит Dayhoff (6-class Dayhoff alphabet [13]).

3.2. Кластеризация.

Программа MUSCLE реализует как метод UPGMA, так и метод neighbor-joining.

Считается, что neighbor-joining дает более точную оценку правильного эволюционного дерева [14]. В то же время известно, что точность выравнивания уменьшается со снижением идентичности последовательностей [15]. Поэтому пара из множества профилей, выровненная наиболее точно, есть пара с самой высокой идентичностью, т.е. пара с кратчайшим эволюционным расстоянием. Именно такая пара выбирается критерием ближайших соседей в UPGMA. В противоположность, neighbor-joining выбирает пару эволюционных соседей, т.е. пару, имеющую общего предка. Когда частота мутаций изменчива, эволюционный сосед может оказаться не ближайшим соседом. Поэтому дерево ближайших соседей может быть лучше, чем истинное эволюционное дерево, для построения итеративного выравнивания.

К тому же метод neighbor-joining тратит $O(N^4)$ времени, а метод UPGMA только $O(N^3)$ времени.

3.3. Профиль-профильное выравнивание.

Р. Эдгар определяет профиль как множественное выравнивание, в отличие от общепринятого употребления термина «профиль»: обычно профиль определяется как позиционно-зависимая весовая матрица (position-specific scoring matrix). В дальнейшем мы будем придерживаться терминологии, которая используется Р. Эдгаром. Выравнивание производится методом Смита-Ватермана с небольшими модификациями, связанными с определением весов «открытия» и «закрытия» удаленного фрагмента («gap»).

В этом разделе всюду X, Y – это исходные множественные выравнивания («профили»), к которым применяется процедура профиль-профильного выравнивания.

3.3.1. Профильная функция.

Для того чтобы применять методы парного выравнивания к профилям, для произвольной пары позиций сравниваемых профилей, т.е. пары столбцов множественного выравнивания, должен быть определен вес их сопоставления. Функция, которая задает вес сопоставления пары столбцов множественного

выравнивания, называется профильной функцией. Она является аналогом весовой матрицы замен (см., например, [16]). Программа MUSCLE использует две профильные функции PSP и LE.

3.3.1.1. Профильная функция PSP (от Profile Sum of Pairs), говоря неформально, определяется как взвешенная сумма весов попарных сопоставлений символов из данных столбцов с учетом:

- доли данного символа в данном столбце;
- априорной частоты символа в базе данных;
- априорных весов последовательностей.

Замечание. В приведенной ниже формуле Р.Эдгар никак не учитывает весов последовательностей. Создается впечатление, что веса последовательностей используются только при вычислении SP-весов в 3-й стадии. Более того, Р.Эдгар не касается подробно вопросов выбора весов последовательностей, отмечая лишь, что они выбираются так же, как в программе CLUSTALW. Поэтому ниже мы не будем касаться этого вопроса, считая, что для любой пары последовательностей вес равен 1.

Дадим определение более формально. Пусть i и j - типы аминокислот ($i, j = 1, \ldots, 20$), x, y – “колонки”, т.е. символьные вектора, содержащие символы из аминокислотного алфавита и специальный символ «индел». Мы используем следующие обозначения. p_i – базовая (априорная) вероятность символа i в базе данных; p_{ij} – вероятность того, что символы i и j выровнены друг с другом, S_{ij} – вес сопоставления i и j в соответствии с используемой матрицей замен, fx_i – наблюдаемая частота встречаемости символа i в колонке x; fx_G – наблюдаемая частота инделов («гэпов») в колонке x. Тогда значение PSP(x, y) вычисляется по формуле:

$$\mathrm{PSP}(x, y) = \Sigma_i \Sigma_j f x_i f y_j S_{ij} . \qquad (3)$$

Так как $S_{ij} = \log (p_{ij} / p_i \bullet p_j)$ [17], то формулу (3) можно переписать в виде:

$$\mathrm{PSP}(x, y) = \Sigma_i \Sigma_j f x_i f y_j \log (p_{ij} / p_i \bullet p_j) \qquad (3a)$$

Функция PSP используется в программах CLUSTALW и MAFFT. Она ориентирована на максимизацию SP-веса выравнивания объединенного множества последовательностей (см. п. 3.5), если считать штрафы за удаления нулевыми и если предположить, что в итоговом выравнивании мы не изменяем (перевыравниваем) столбцы исходных выравниваний. Программа MUSCLE позволяет использовать PSP-функцию на основе 200 PAM матрицы [18] и 240 PAM VTML матрицы [19].

Замечание 1. Последняя фраза взята из основной статьи про программу MUSCLE [3]. Однако, точный ее смысл не вполне ясен. Это связано с тем, что матрицы семейств PAM и VTML не используют напрямую формулу $S_{ij} = \log (p_{ij} / p_i \bullet p_j)$, а основаны на дополнительных математических преобразованиях (см. цитированные статьи). В этом вопросе мы будем разбираться. Кроме того, термин “240 PAM VTML matrix”, видимо, является опечаткой (должно быть “240 VTML matrix”, т.к. в Интернете встречается только в анализируемой работе Р.С.Эдгара и работах, которые ее пересказывают (не комментируя).

Замечание 2. Согласно данным работы [20] матрицы семейств VTML и BLOSUM дают лучшее качество парных выравниваний, чем матрицы PAM.

3.3.1.2. Другая профильная функция, используемая программой MUSCLE – это функция LE, основанная на методе *log-ожидания* (*log-expectation*, LE). Она представляет собой модифицированную версию профильной функции *LA*, основанной на методе log-средних (log-average, LA) которая была предложена на теоретических основаниях [21]:

$$LA(x, y) = \log \Sigma_i \Sigma_j \alpha(x, i) \bullet \alpha(y, j) \bullet p_{ij} / p_i p_j , \qquad (4)$$

где $\alpha(x, i)$ - оценка вероятности наблюдения i в столбце x. Оценочные вероятности α выводятся из наблюдаемых частот f, как правило, путем добавления эвристических псевдо-отсчетов или с помощью смесей Дирихле [22].

Функция LE определяется следующим образом:

$$LE(x, y) = (1 - f x_G) (1 - f y_G) \bullet \log \Sigma_i \Sigma_j f x_i \bullet f y_j \bullet p_{ij} / p_i p_j \qquad (5)$$

Как видно, оценки вероятности α в функции LA заменяются на наблюдаемые частоты f в функции LE. Кроме того, добавляется множитель $(1 - f x_G) (1 - f y_G)$, снижающий вес сопоставления для колонок, содержащих инделы. Р. Эдгар указывает, что это значительно улучшает точность выравниваний.

3.3.2. Штрафы за гэп.

Штрафы за гэп в профиль-профильном выравнивании определяются следующим образом. Пусть первый индел в гэпе называется *открытием гэпа*, последний – *закрытием гэпа*. Рассмотрим выравнивание двух профилей X и Y с гэпом длиной λ в X, открытие которого выровнено с позицией y_o в Y, а закрытие с y_c. Штраф за такой гэп вычисляется как

$$b\,(y_o) + t\,(y_c) + \lambda e, \qquad (6)$$

где b и t штрафы за открытие и закрытие гэпов, их значение изменяется в зависимости от положения в Y, а e - называется штрафом за удлинение гэпа, не зависящий от позиции.

Отметим, что при глобальном выравнивании без потери общности можно считать, что штраф за удлинение гэпа $e = 0$. Рассмотрим глобальное выравнивание последовательностей X и Y с длинами L_X и L_Y. Если добавить некоторую константу δ к каждому значению матрицы замены и $\delta/2$ к каждой позиции, содержащей гэп (что соответствует уменьшению штрафа за удаление e на $\delta/2$), то в результате к весу любого возможного выравнивания прибавляется постоянное значение $\delta(L_X+L_Y)/2$, и поэтому набор оптимальных выравниваний остается без изменений. С учетом системы определения веса с помощью матрицы замены S_{ij} и штрафа за удлинение e, можно выбрать $\delta/2 = e$ и, соответственно, использовать $S'_{ij} = S_{ij}+2e$ и $e' = 0$, чтобы получить то же оптимальные выравнивания. Постоянная $2e$ может быть добавлена к матрице замены во время компиляции. В программе MUSCLE это преобразование задействовано только в случае использования функции PSP, но не используется для функции LE.

Пусть $f_o(y)$ – доля открытий гэпа в позиции y в Y, а $f_c(y)$ – доля закрытий гэпа в этой позиции. MUSCLE вычисляет значения b и t (см. формулу (6)) следующим

образом (у Р. Эдгара очевидная «опечатка» – количество (“number”) вместо «доля», см. также замечание 2 ниже):

$$b\,(y) = g/2 \bullet (1 - f_o(y)) \bullet (1 + h_w(y) \bullet H\,), \quad (7)$$

$$t\,(y) = g/2 \bullet (1 - f_c(y)) \bullet (1 + h_w(y) \bullet H\,). \quad (8)$$

Здесь g - это параметр, который является штрафом за гэп в целом (per-gap), удобно считать, что половина этого штрафа берется при открытии гэпа, а половина – при закрытии. Остальные параметры рассмотрены ниже.

Замечание 1. Программа MUSCLE обрабатывает позиции открытия и закрытия симметрично, в отличие от программы CLUSTALW, где рассматривается только штраф в позиции открытия. С точки зрения Р.Эдгара (не очевидной и ничем не подкрепленной) это может вести к тому, что гэпы, которые вставляет CLUSTALW имеет тенденцию быть выровненными по левому краю. Представляет интерес разобраться, верна ли эта гипотеза Р. Эдгара.

Множитель $(1 + h_w(y) \bullet H)$ задает увеличение штрафов для гидрофобных участков. Величина $h_w(y)$ равна 1, если y лежит в блоке из w последовательных гидрофобных остатков, или нулю в противном случае; значения по умолчанию параметров w и H: $w = 5$, $H = 1{,}2$ (эти значения можно изменить в настройках [34]). Ужесточение штрафа в гидрофобных участках имеет целью затруднить появление гэпов во внутренних участках белковой глобулы, где вставки и удаления встречаются реже, чем на поверхности глобулы.

Введение множителей $(1 - f_o(y))$, $(1 - f_c(y))$ связано с желанием сблизить оптимизируемый вес профиль-профильного выравнивания и SP-вес итогового множественного выравнивания (см. ниже п. 3.5). Действительно, пусть выравниваемые профили соответствуют множествам X и Y исходного набора последовательностей U и гэп G вставляется в X таким образом, что открытие гэпа выровнено с позицией y в Y. Если последовательность $s \in Y$ имеет открытие гэпа в позиции y, то SP-вес не включает штраф за открытие G в любой паре последовательностей вида (s, t), где $t \in X$, т.к. в парном выравнивании для последовательностей (s, t) при этом образуется пара

противостоящих друг другу пробелов. Поэтому множитель (1 - $f_o(y)$) корректирует вклад открытий гэпа в SP-вес за счет учета уже существующих гэпов в *Y*.

Замечание 2. Как отмечает Р. Эдгар, «даже с этой поправкой, остаются другие вопросы, связанные с гэпами». Профильная функция PSP и приведенные выше штрафы за гэпы не позволяют построить выравнивание наибольшего возможного SP-веса среди выравниваний данных профилей (т.е. множественных выравниваний) *X*, *Y* при ограничении фиксированных колонок.

Нам кажется, что в формулах (7) и (8) правильнее использовать величину $f_G(y)$, а не величины $f_o(y)$ и $f_c(y)$. Но это требует дополнительного изучения.
Отметим, что с учетом сказанного выше, можно считать, что вес элонгации $e=0$.

3.3.3. Терминальные гэпы.

Терминальным гэпом называется гэп, который открывается в N-терминальной позиции последовательности, с которой он выравнивается, или закрывается в С-терминальной. В противном случае гэп называется *внутренним.*

Ранее высказывалось предположение [23,24], что методы глобального выравнивания испытывают трудности при восстановлении длинных делеций или вставок. По мнению Р. Эдгара, эти трудности связаны с выбором штрафов за терминальные гэпы. Программе CLUSTALW, которая не взимает штрафы за терминальные гэпы, как правило, не удается открыть необходимый внутренний гэп и, тем самым, она оказывается не в состоянии выровнять терминальные мотивы. Программа MAFFT, имеющая одинаковые штрафы за терминальные и внутренние гэпы, иногда с помощью вставки искусственного внутреннего гэпа выравнивает небольшое количество внутренних остатков с терминальными, что приводит к ошибкам.

По умолчанию в программе MUSCLE значение штрафа за терминальные гэпы равно половине штрафа за внутренний гэп. Это делается путем присвоения значения ноль величинам $b(1)$ (штраф за открытие в С-конце) и $t(L)$ (штрафу за закрытие в N-конце). Также предусмотрена возможность применения только полного штрафа, (как в программе MAFFT). Р. Эдгар утверждает, что решение с половиной штрафа за терминальные гэпы дает хорошие результаты для широкого диапазона входных данных.

Дальнейшее улучшение может иногда достигаться с помощью следующих оптимизаций. Если отношение длин двух профилей, которые выравниваются, превышает порог (по умолчанию 20%), то MUSCLE строит четыре различных выравнивания, в которых гэпы в обоих концах, в одном или нигде штрафуются полностью. *Консервативный вес* определяется путем вычитания всех штрафов за гэпы (как внутренних, так и терминальных) от веса выравнивания, в результате чего остается только сумма весов колонок, вычисленных по профильным функциям. Используется выравнивание с самым высоким консервативным весом.

3.4. Мера расстояния между последовательностями.

Аддитивная мера расстояния оценивается с учетом значения сходства.

Аддитивная мера расстояния d(A, B) между двумя последовательностями A и B удовлетворяет d(A, B) = d(A, C) + d(C, B) для любой третьей последовательности C, при условии, что A, B и C связаны между собой.

Идеальным, но в целом невозможным для использования, является *мутационное расстояние*, т.е. количество мутаций, которые произошли между последовательностями в ходе эволюции. В филогенетическом дереве исторический путь прокладывается от одной последовательности к другой через их последнего общего предка. Очевидно, что мутационное расстояние аддитивно.

Часто в качестве меры сходства используется доля совпадений *D*. Для близкородственных последовательностей 1 - *D* является хорошим приближением к мутационному расстоянию (такое решение предполагает, что замена в одной позиции является единственным разрешенным типом мутации, и что в одной позиции не может произойти больше одной мутации).

По мере рассмотрения все более удаленных гомологов растет вероятность множественных мутаций в одном месте. Для исправления этого, в MUSCLE используется следующая оценка расстояния:

$$d_{\text{Kimura}} = -\log_e (1 - D - D^2/5) \qquad (9)$$

Для $D \leq 0{,}25$ используется табулированные значения из исходных кодов CLUSTALW. Заметим, однако, что согласно источнику [8] в формуле расстояния Кимура переменная D – это доля несовпадений. Поэтому мы предполагаем, что в статье была допущена опечатка.

3.5. SP-вес.

3.5.1. Введение

На стадии 3 программа MUSCLE стремится максимизировать вес выравнивания. Этот вес, по определению, равен сумме весов выравниваний всех пар последовательностей (при этом вес удаленного участка считается пропорциональным длине фрагмента). Вес, определенный таким образом, называется *SP-вес.* Пусть *M(a, b)* – вес сопоставления символов *a, b; d* – штраф за индел. Тогда вес колонки $<a_1, a_2, \ldots, a_n>$ в выравнивании *n* последовательностей определяется по формуле

$$W(a_1, a_2, \ldots, a_n) = \sum_{1 \le i < j \le n} \sigma(a_i, a_j),$$

где

$$\sigma(a, a') = \begin{cases} M(a, a') & \text{– если и } a, \text{ и } a' \text{ – буквы исходного алфавита;} \\ -d, & \text{- если ровно один из символов } a, a' \text{ – индел;} \\ 0 & \text{- если оба символа } a, a' \text{ – инделы.} \end{cases}$$

Вычисление SP-веса доминирует во времени работы программы MUSCLE, поэтому при вычислении применяется ряд оптимизаций.

3.5.2. Вклад сопоставлений аминокислот

Рассмотрим сначала вклад в SP-вес от аминокислот, этот вклад обозначается SP^a. Пусть *s* и *t* - последовательности, *x* - колонка, *s*[*x*] – аминокислота из последовательности *s* в колонке *x*, и *S* (*i*, *j*) – вес замены аминокислот *i* и *j*. Введем (произвольным образом) порядки на множестве исходных последовательностей и на множестве типов аминокислот. Тогда,

$$SP^a = \Sigma_x \Sigma_s \Sigma_{t>s} S(s[x], t[x]) \qquad (10)$$

где *S(a, a') = M(a, a')*, если и *a*, и *a'* – буквы исходного алфавита и *S(a, a') =0* в противном случае.

Определим $\delta(s,i,x)$=1, если $s[x] = i$, 0 в противном случае, и $n_i[x] = \Sigma_s \delta(s, i, x)$. Очевидно, $n_i[x]$ – это *количество* аминокислот типа *i* в колонке *x*. Таким образом,

можно преобразовать сумму по парам последовательностей в сумму по парам аминокислотных типов:

$$SP^a = \Sigma_x \Sigma_i n_i \Sigma_{j>i} n_j S(i, j) + \Sigma_x \Sigma_i ((n_i^2 - n_i)/2) \bullet S(i, i). \quad (11)$$

Определим частоты следующим образом:

$$f x_i = n_i [x] / N \quad (12)$$

Подставляя частоты в (11), получим:

$$SP^a = N^2 \Sigma_x \Sigma_i f x_i \Sigma_{j>i} f x_j S(i, j) + (N^2/2) \Sigma_x \Sigma_i (f x_i^2 - f x_i / N) S(i, i). \quad (13)$$

Для простоты мы не учитываем взвешивание последовательностей, в случае использования весов следует величины n_i заменить на сумму весов тех последовательностей s, для которых $s[x]=i$. несложно показать, что (26) остается неизменным, если взвешивание используется. Обратите внимание, что для вычисления по формуле (10) требуется $O(N^2 L_p)$ шагов, а для вычисления по формулам (11) и (13) достаточно $O(NL_p)$ шагов. Для $N >> 20$, это является существенным улучшением. Здесь и далее L_p – это длина профиля.

3.5.2. Вклад сопоставлений аминокислот и инделов

Пусть SP^g - вклад штрафов за гэпы в SP, т.е.

$$SP^g = \Sigma_x \Sigma_s \Sigma_{t>s} G(s[x], t[x]) \quad (16)$$

где $G(a, a') = -d$, если ровно один из символов a, и a' – это индел и $G(a, a') = 0$ в противном случае. Таким образом, $SP = SP^a + SP^g$.

Замечание. Р. Эдгар утверждает, что аналоги формул (11), (13) не известны. Это кажется странным. Пусть $n^a[x]$ – количество символов аминокислот в столбце x; $n^g[x]$ – количество инделов в столбце x. Тогда аналогом формулы (11) является следующая формула:

$$SP^g = -d \bullet n^a[x] \bullet n^g[x] \quad (17)$$

Конец замечания.

Р. Эдгар отмечает, что при работе программы *MUSCLE* (см. п. 2.4) абсолютное значение SP веса не требуется; достаточно определить различие в SP-весе до и после повторного выравнивания пары профилей. Пусть SP (*s, t*) - вклад в SP-вес от пары последовательностей *s* и *t*, таким образом, что $SP = \Sigma_s \Sigma_{t>s} SP(s, t)$, и обозначим через X и Y два профиля, которые выравниваются при очередном выполнении этапа 3 стадии 3 алгоритма MUSCLE. Величину можно представить следующим образом:

$$SP = \Sigma_{s \in X} \Sigma_{t \in X: t>s} SP(s, t) + \Sigma_{s \in Y} \Sigma_{t \in Y: t>s} SP(s, t) + \Sigma_{s \in X} \Sigma_{t \in Y} SP(s, t) \quad (18)$$

Первые два слагаемых назовем внутрипрофильными, а третье – межпрофильным. Внутрипрофильные условия остаются неизменными при выполнении межпрофильного выравнивания, т.е. различие между старым и новым значением SP определяется различием в межпрофильном слагаемом

$$SP_{XY} = \Sigma_{s \in X} \Sigma_{t \in Y} SP(s, t) \quad (19)$$

Это в среднем уменьшает время вычисления примерно в два раза. Далее, время можно еще улучшить, заметив, что обычно изменений в выравнивании либо мало, либо вообще нет. Это наводит на мысль, что можно вычислить изменения SP, рассматривая только те колонки, где старое и новое выравнивание различаются.

Пусть $C_{ст}$ – множество колонок старого выравнивания, которые отсутствуют в новом множественном выравнивании и $C_{нов}$ – множество колонок нового выравнивания, которые отсутствуют в старом множественном выравнивании. Эти множества можно построить за время $O(L_p)$. Теперь искомая разность значений весов может быть вычислена как разность

$$SP_{нов} - SP_{ст}$$

где $SP_{нов}$ (соответственно, $SP_{ст}$) – это сумма величин *SP*, подсчитанных для колонок из множества $C_{нов}$ (соответственно, $C_{ст}$).

4. Алгоритм Muscle: анализ качества.

4.1. Общие сведения.

Результаты работы программы MUSCLE (версии MUSCLE и MUSCLE-p) сравнивались с результатами программ CLUSTALW [9], T-Coffee [23], а также двумя MAFFT-скриптами: FFTNS1 и NWNSI [10]. Тестируемые версии: MUSCLE 3.2, CLUSTALW 1.82, T-Coffee 1.37 и MAFFT 3.82.

Оценка качества работы программ проводилась двумя способами:

1) путем сравнения результатов работы программы с эталонными выравниваниями;
2) с помощью вычисления меры APDB [25], которая вычисляется исходя из пространственных структур сравниваемых белков.

В качестве источников эталонных выравниваний использовались базы данных BAliBASE [26,27], SABmark [28,29], SMART [30–32] и PREFAB [3]. При сравнении алгоритмических выравниваний с эталонными использовались две меры качества:

1) Q – Количество правильно выровненных пар аминокислотных остатков разделенное на общее число пар остатков в эталонном выравнивании.
2) TC - Количество правильно выровненных колонок разделенное на общее число колонок в эталонном выравнивании.

При вычислении меры APDB использовались структурные выравнивания из БД PREFAB.

При сравнении результатов измерений для различных программ использовался критерий Фридмана [33]. Выбор критерия Фридмана (а не критерия Вилкоксона) обосновывается тем, что критерий Фридмана накладывает меньше ограничений на распределение выборки.

4.2. Сравнение на основе BAliBASe.

При сравнении программ использовались разделы (reference groups) Ref 1–Ref5 БД BAliBASe 2.0. Остальные разделы содержат повторения, инверсии, а также трансмембранные белки, для которых еще нет испытанного алгоритма. Результаты сравнения приведены в таблицах 1-3.

Таблица 1. Полученные оценки Q и TC и затраченное время CPU в BAliBASE.

Метод	***Q***	**TC**	**CPU**

MUSCLE	0.896	0.747	97
MUSCLE-p	0.883	0.727	52
T-Coffee	0.882	0.731	1500
NWNSI	0.881	0.722	170
CLUSTALW	0.86	0.69	170
FFTNS1	0.844	0.646	16

В таблице 1 представлены средние значения Q и TC оценок для каждого метода, а также затраченное время (CPU) в секундах полученные по БД BAliBASE в целом. Таблицы 2 и 3 содержат средние значения оценок *Q* и *TC* отдельно по каждому разделу БД BAliBASE. Раздел Ref1 содержит 81 выравнивание последовательностей – это самый большой раздел, он составляет около 60% всей базы данных. Остальные разделы гораздо меньше. Например, разделы Ref4 и Ref5 содержат по 12 выравниваний. Как видно из таблиц 2 и 3, средние значения оценок *Q* и *TC* для разных разделов существенно различны.

Таблица 2. Значения оценки Q для различных разделов BAliBASE

Метод	**Ref1**	**Ref2**	**Ref3**	**Ref4**	**Ref5**
MUSCLE	0.887	0.935	0.823	0.876	0.968
MUSCLE-p	0.871	0.928	0.813	0.857	0.974
T-Coffee	0.866	0.934	0.787	0.917	0.957
NWNSI	0.867	0.923	0.787	0.904	0.963
CLUSTALW	0.861	0.932	0.751	0.823	0.859
FFTNS1	0.838	0.908	0.708	0.793	0.947

Таблица 3. TC оценка в различных группах BAliBASE

Метод	**Ref1**	**Ref2**	**Ref3**	**Ref4**	**Ref5**
MUSCLE	0.815	0.574	0.577	0.627	0.902

MUSCLE-p	0.795	0.558	0.55	0.598	0.891
T-Coffee	0.78	0.573	0.51	0.751	0.903
NWNSI	0.788	0.514	0.514	0.742	0.859
CLUSTALW	0.782	0.579	0.47	0.542	0.638
FFTNS1	0.732	0.496	0.35	0.451	0.831

4.3. Сравнение на основе SABmark.

Была использована версия 1.63 базы эталонных выравниваний SABmark. Эта база состоит из двух частей: Superfamily и Twilight. Структуры всех последовательностей известны. Twilight содержит 1994 домена из БД Astral с попарным сходством последовательностей e-values ≤ 1, разделенных на 236 класса согласно SCOP классификации. Superfamily содержит последовательности с парной идентичностью $\leq 50\%$, разделенных в 462 SCOP надсемейств. Каждая пара структур была выровнена с помощью структурных программ-выравнивателей: SOFI и CE.

Таблица 4. *Q* оценка и затраченное время CPU в SABmark

Метод	**All**	**Superfamily**	**Twilight**	**CPU**
MUSCLE	0.430	0.523	0.249	1886
T-Coffee	0.424	0.519	0.237	5615
MUSCLE-p	0.416	0.511	0.230	304
NWNSI	0.410	0.506	0.223	629
CLUSTALW	0.404	0.498	0.220	206
FFSNT1	0.373	0.467	0.190	75

В колонке All представлены средние значения Q для всех выравниваний из SABmark. В колонках Superfamily и Twilight - средние значения Q для соответствующих частей базы. Они вычисляются усреднением Q для каждой пары в отдельном множественном

выравнивании с последующим усреднением по всем множественным выравниваниям. Таким образом вводится поправка на зависимость пар последовательностей в данном множественном выравнивании.

4.4. Сравнение на основе SMART.

База SMART содержит множественные выравнивания, определенные экспертами (в основном, сигнальных доменов). При конструкции SMART были также использованы различные методы выравнивания последовательностей, поэтому SMART не совсем подходит как тестовая база данных. Однако эти выравнивания были проверены и исправлены экспертами, поэтому работа с этим банком представляет интерес.

Таблица 5. *Q* и TC оценки в SMART

Метод	***Q***	**TC**
MUSCLE	0.855	0.537
NWNSI	0.848	0.546
MUSCLE-p	0.836	0.505
T-Coffee	0.835	0.503
CLUSTALW	0.823	0.504
FFTNS1	0.817	

В таблице представлены средние значения Q и TC оценок точности 267 тестовых выравниваний из SMART. Каждое выравнивание содержит не более 100 последовательностей.

Замечание. TC оценка метода FFTNS1 по неизвестным нам причинам в статье не указана.

4.5. Сравнение на основе PREFAB.

Тестовая база данных PREFAB (protein reference alignment benchmark) была сконструирована Р.Эдгаром в 2004 г специально для тестирования качества работы программ множественного выравнивания.

Основной алгоритм построения PREFAB следующий. Сначала два белка выравнивались структурным методом, который не включал в себя сходство

последовательности. Затем для каждой последовательности проводился поиск по базе данных, в результате выбирались хиты с высоким весом. Запросы с полученными хитами объединялись и выравнивались между собой с помощью метода множественного выравнивания последовательностей. Точность оценивалась на первой паре сравнением с их структурным выравниванием. Более подробно см. [3].

Тестирование проводилось на БД PREFAB версии 3.0. Она содержит 1932 выравнивания в среднем из 49 последовательностей длины 240.

Таблица 6. Q оценка и затраченное время PREFAB

Метод	All	0–20%	20–40%	40–70%	70–100%	CPU
MUSCLE	0.645	0.473	0.813	0.937	0.980	$1.7 \cdot 10^4$
MUSCLE-p	0.634	0.460	0.802	0.942	0.985	$2.0 \cdot 10^3$
T-Coffee	0.615	0.464	0.795	0.935	0.976	$1.0 \cdot 10^6$
NWNSI	0.615	0.448	0.772	0.930	0.939	$1.4 \cdot 10^4$
FFTNS1	0.591	0.423	0.756	0.931	0.938	$1.0 \cdot 10^3$
CLUSTALW	0.563	0.382	0.732	0.916	0.930	$3.3 \cdot 10^4$

В колонке All представлены средние значения Q оценки для каждого выравнивания по всей БД PREFAB, в колонке CPU – затраченное время на построение выравнивания (в секундах). В остальных колонках представлены средние значения Q оценки для групп, в которых идентичность пары структур находится в данном диапазоне.

Как видно из таблицы, методу T-Coffee понадобилось 10 дней для завершения теста, в то время как методу MUSCLE меньше 5 часов, а методу MUSCLE-p примерно 30 минут.

Таблица 7. APDB оценки в PREFAB

Метод	APDB
NWNSI	62.0
MUSCLE	61.9

T-Coffee	61.9
MUSCLE-p	61.4
FFTNS1	60.8
CLUSTALW	59.1

В Таблице 7 представлены средние значения оценки APDB каждого метода на тестовой базе PREFAB. Как видно из таблицы, между четырьмя лучшими методами нет существенной статистической разницы. Однако они значительно лучше, чем метод FFTNS1 (MUSCLE-p > FFTNS1 с *P*-value= 0.009), а FFTNS1 значительно лучше метода CLUSTALW (*P* -value= $3 \cdot 10^{-5}$).

4.6. Статистический анализ.

Таблица 8. Статистическая значимость различий в результатах работы методов, вычисленная из Q оценок в тестовых базах данных BAliBASE и PREFAB

	MUSCLE	MUSCLE-p	T-Coffee	NWNSI	FFTNS1	CLUSTALW
MUSCLE		+0.001	(0.15)	+ 0.005	$+2 \cdot 10^{-6}$	+0.0002
MUSCLE-p	$-6 \cdot 10^{-6}$		(0.3)	(0.7)	+0.0002	+0.02
T-Coffee	–0.0002	(0.4)		(0.55)	$+7 \cdot 10^{-5}$	+0.01
NWNSI	$-<10^{-10}$	$-<10^{-10}$	-10^{-7}		+0.0001	(0.06)
FFTNS1	$-<10^{-10}$	$-<10^{-10}$	$-<10^{-10}$	$-<10^{-10}$		–0.04
CLUSTALW	$-<10^{-10}$	$-<10^{-10}$	$-<10^{-10}$	$-<10^{-10}$	–0.008	

Каждая ячейка таблицы содержит P-value различия между парой методов, определенного ранговым критерием Фридмана. Розовым цветом обозначены P-value, полученные из Q оценок в БД BAliBASE, голубым – в БД PREFAB. Если метод из левой колонки лучше метода из верхней строки, то перед P-value ставится плюс. Если наоборот - минус. Например, метод MUSCLE лучше метода T-Coffee с *P*-value = 0.0002. При *P*-value >0.05 различие в методах считается незначительным. В таблицы такие случаи обозначаются через круглые скобки.

Из Таблицы 8 видно, что метод MUSCLE лучше методов NWNSI, FFTNS1 и CLUSTALW, однако незначительно отличается от метода T-Coffee. Однако метод

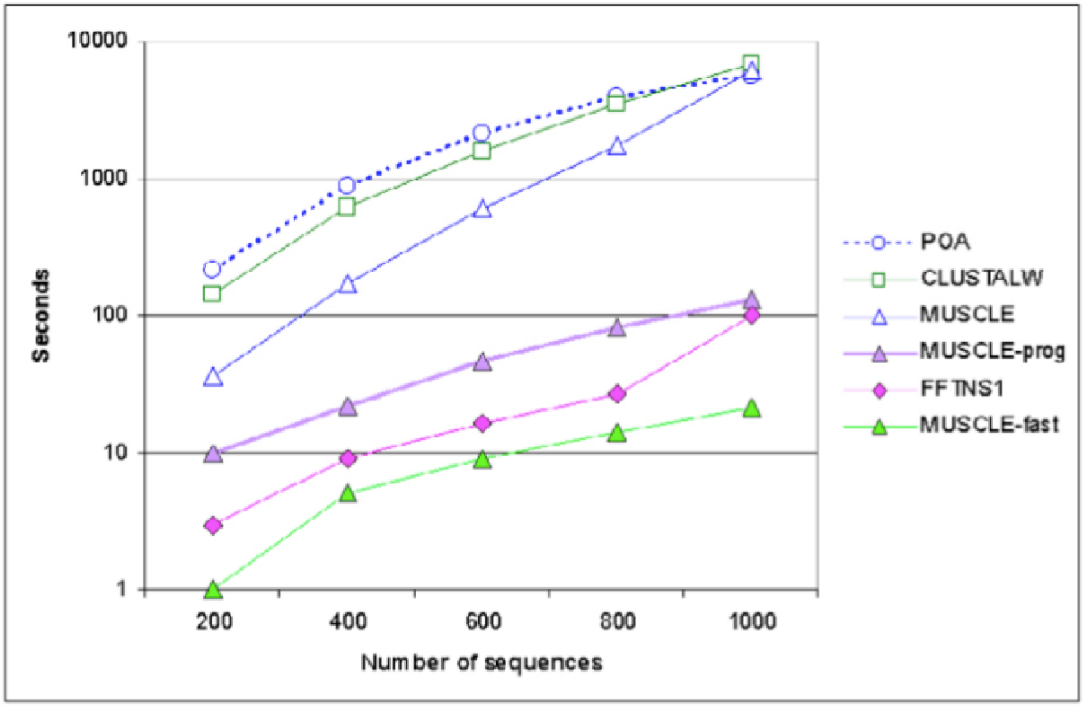
T-Coffee в отличие от метода MUSCLE гораздо более времяемкий, что можно наблюдать хотя бы из Таблицы 6.

5. Алгоритм Muscle: время работы.

5.1. Оценка, проведенная Р.Эдгаром.

Р. Эдгар оценивал скорость программ MUSCLE и MUSCLE-fast при большом количестве последовательностей следующим образом.

Рисунок 1. Время работы программ в зависимости от количества последовательностей.



Из 1000 гомологов белка ECH1_RAT (диенол-КоА изомераза, идентификатор в БД PDB 1DCI) была составлена тестовая выборка. Гомологи были получены с помощью программы PSI-BLAST в БД nr (NCBI non-redundant protein sequence database). Средняя длина последовательностей в выборке 282, максимальная длина 454, средняя парная идентичность 20%. Исследование проводилось на персональном компьютере с процессором Pentium 4 2.5 GHz. Программа MUSCLE-fast выровняла эту выборку за 21 секунду [3].

График времени работы MUSCLE-fast и других программ от количества последовательностей представлен на рис.1 (выше). По оси абсциссы – количество последовательностей, по оси ординаты – время в секундах.

Из рисунка видно, что при 200 последовательностях программа MUSCLE-fast приблизительно в 3 раза быстрее метода FFTNS1, который до этого считался наиболее быстрым. При 1000 последовательностях MUSCLE-fast быстрее FFTNS1 приблизительно в 5 раз. Р.Эдгар пишет, что при дальнейшем увеличении числа последовательностей такая тенденция остается, однако результаты опубликованы не были.

5.2. Оценка, проведенная нами.

Нами было выровнено 1000 последовательностей гомологов белка ARGB_ECOLI (ацетил-глутамат киназа, идентификатор в БД PDB 1GS5), выборка которых была получена с помощью программы BLASTP (параметры поиска стандартные) в БД nr. Средняя длина последовательностей 275. Исследование проводилось на компьютере с процессором Intel Core 2 Duo 1.83 GHz и оперативной памятью 2048 Mb. Время работы программы MUSCLE-fast составило 18 секунд. Также было выровнено 1000 негомологичных белков из организмов *Neisseria meningitidis* и *Escherichia coli K-12* (выборка была составлена случайным образом) средней длиной 331 за 48 секунд. Эти результаты подтверждают результаты работы [3].

Также было составлено несколько выборок из гомологов белка NADB_ECOLI (L-аспартат оксидаза, идентификатор в БД PDB 1KNP). Максимальная длина последовательностей – 742; средняя длина последовательностей – 535[1]. С помощью программы MUSCLE они были выровнены. Результаты о проведении работы программы MUSCLE представлены в таблице 9 (см. ниже).

При выравнивании 4700 последовательностей программа MUSCLE прекратила работу, так как лимит возможно используемой памяти (здесь 1660 Mb) был исчерпан. По умолчанию лимит равен 80 % от оперативной памяти либо 500 Mb (в случаях, когда MUSCLE не может определить размер оперативной памяти [34]).

К сожалению, Р.Эдгар не предоставил точные значения оценки скорости программы MUSCLE, заметив только, что при выравнивании 1000 последовательностей ее скорость сравнима со скоростями таких программ как CLUSTALW и Progressive POA (см. выше рис.1).

[1] Данные представлены для первой выборки из 500 последовательностей. В остальных выборках они варьировались на несколько аминокислотных остатков больше или меньше.

Таблица 9. Время, затраченное на построение выравнивания для различных выборок.

Количество последовательностей	Время, затраченное на обработку программой
500	9 мин 45 сек
1000	35 мин 07 сек
2000	2 часа 28 мин 32 сек
4700	-

6. **Программа Muscle и ее настройки.**

В программе MUSCLE настройки, обеспечивающие наибольшую точность, являются настройками по умолчанию. Поэтому, например, на 2-й стадии происходит только одна итерация, т.к. выяснилось, что последующие итерации не приносят видимого улучшения [3]. Однако при желании пользователь может увеличить количество итераций с помощью опции *maxtrees*, которая определяет максимальное количество новых деревьев во 2-й стадии.

Можно также изменить количество итераций во всем алгоритме. Это делается с помощью опции *maxiters*. Если ее значение равно 1, то проводится одна итерация, соответствующая 1-й стадии. При значении равном 2 выполняются 1-я и 2-я стадии. Значение большее или равное 3 указывает на максимальное количество итераций в 3-й стадии (т.е., если *maxiters* 3, то выполняется одна итерация, а если 4, то возможно две итерации). Значение *maxiters* по умолчанию 16.

В случае необходимости существует опция, ограничивающая выполнение алгоритма по времени, - *maxhours*. Если время выполнения превышает ее значение, то алгоритм завершается, а на выход выдается текущее выравнивание. По умолчанию эта опция не используется.

Также с помощью опции *treeX*, где X=1,2, можно сохранить филогенетическое дерево, полученное на соответствующей стадии.

К сожалению, в программе MUSCLE не предусматривается диалоговый режим, позволяющий просматривать результаты после определенной стадии и, если нужно,

продолжить работу. Однако программа выдает технические сообщения о затраченном времени, памяти, номере итерации и сколько данных уже обработано (в процентах).

MUSCLE поддерживает такие опции, как переуточнение уже существующего выравнивания (*refine*), построение выравнивания на основе известного дерева (*usetree*), построение профиль-профильного выравнивания множественных выравнивания *(profile*). С помощью *profile* можно также добавить последовательность к существующему выравниванию (если на место одного из входящего выравнивания поставить нужную последовательность).

Как уже упоминалось в п.1., кроме стандартной программы MUSCLE существуют также ее модифицированные варианты: MUSCLE-fast и MUSCLE-p. В случае MUSCLE-fast выполняется только первая стадия алгоритма с настройками, обеспечивающими самую высокую скорость (использование профильной функции PSP, F^{binary} как меры сходства и поиск диагоналей при построении выравнивания). В MUSCLE-p выполняются первые две стадии со стандартными настройками.

Полный список возможных настроек в программе MUSCLE можно увидеть [34].

7. Выводы.

Целью работы было разобраться в алгоритме программы MUSCLE, сопутствующих алгоритмах и понятиях, по возможности, перепроверить результаты работ [2,3], касающиеся быстродействия программы и установить пределы ее применимости.

Как было показано, программа MUSCLE получает более точные результаты за меньшее время по сравнению с другими программами множественных выравниваний. Это происходит благодаря большому количеству разнообразных приемов. Для быстродействия программы, по-видимому, наиболее важным является использование k-слов для подсчета расстояния между последовательностями (п. 3.1) и техника вычисления SP-весов (см. п. 3.5).

К сожалению, в статьях [2,3] существуют опечатки и непонятно написанные фрагменты. Вследствие этого в приведенном обзоре не затрагиваются некоторые технические детали (например, взвешивание последовательностей, реализация метода динамического программирования), однако основные особенности метода, как нам кажется, разобраны.

Мы планируем продолжить изучение методов множественного выравнивания.

Благодарим Сергея Александровича Спирина за предоставленные объяснения и помощь.

Список литературы.

1. Notredame C: **Recent progress in multiple sequence alignment: a survey.** *Pharmacogenomics* 2002, **3**(1)**:**131-144.
2. Edgar RC: **MUSCLE: multiple sequence alignment with high accuracy and high throughput.** *Nucleic Acids Res* 2004, **32**(5)**:**1792-1797.
3. Edgar RC: **MUSCLE: a multiple sequence alignment method with reduced time and space complexity**. *BMC Bioinform.* 2004 **5:** 113.
4. Hogeweg,P. and Hesper,B. (1984) **The alignment of sets of sequences and the construction of phyletic trees: an integrated method.** *J. Mol. Evol.*, 20(2), 175±186.
5. Feng,D.F. and Doolittle,R.F. (1987) **Progressive sequence alignment as a prerequisite to correct phylogenetic trees.** *J. Mol. Evol.*, 25(4), 351±360.
6. Legendre, Louis (1998). **Numerical ecology.** *Elsevier* . p. 319-321.
7. Saitou N, Nei M (1987). **The neighbor-joining method: a new method for reconstructing phylogenetic trees**. *Mol Biol Evol* **4** (4): 406–425 .
8. Kimura M: **The neutral theory of molecular evolution.** Cambridge University Press; 1983.
9. Thompson JD, Higgins DG, Gibson TJ: **CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.** *Nucleic Acids Res* 1994, **22**(22)**:**4673-4680.
10. Katoh K, Misawa K, Kuma K, Miyata T: **MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform.** *Nucleic Acids Res* 2002, **30**(14)**:**3059-3066.
11. G. Kucherov, L. Noe, M. Roytberg: **Subset seed automaton**. *Proceedings of the 12th International Conference on Implementation and Application of Automata (CIAA* 2007*), LNCS* 4783:180-191 , 2007.
12. Edgar RC: **Local homology recognition and distance measures in linear time using compressed amino acid alphabets.**

Nucleic Acids Res 2004, **32**(1)**:**380-385.

13. Dayhoff MO, Schwartz RM, Orcutt BC: **A model of evolutionary change in proteins in Atlas of protein sequence and structure,.**
(Edited by: Dayhoff MO, Ech RV). Maryland: National Biomedical Research Foundation 1978.
14. Felsenstein J: **Inferring Phylogenies.** Sunderland, Massachusetts: Sinauer Associates 2004.
15. Sauder JM, Arthur JW, Dunbrack RL Jr: **Large-scale comparison of protein sequence alignment algorithms with structure alignments.** *Proteins* 2000, **40**(1)**:**6-22.
16. Edgar RC, Sjolander K: **A comparison of scoring functions for protein sequence profile alignment.** *Bioinformatics* 2004, *in press*.
17. Altschul SF: **Amino acid substitution matrices from an information theoretic perspective.**
J Mol Biol 1991, **219**(3)**:**555-565
18. Jones DT, Taylor WR, Thornton JM: **The rapid generation of mutation data matrices from protein sequences.**
Comput Appl Biosci 1992, **8**(3)**:**275-282.
19. Muller T, Spang R, Vingron M: **Estimating amino acid substitution models: a comparison of Dayhoff's estimator, the resolvent approach and a maximum likelihood method.**
Mol Biol Evol 2002, **19**(1)**:**8-13.
20. Price GA, Crooks GE, Green RE, Brenner SE. **Statistical evaluation of pairwise protein sequence comparison with the Bayesian bootstrap.** *Bioinformatics*. 2005 Oct 15;21(20):3824-31
21. von Ohsen N, Zimmer R: **Improving profile-profile alignment via log average scoring.**
In: Algorithms in Bioinformatics, First International Workshop, WABI 2001 (Edited by: Gascuel O, Moret BME). Berlin: Springer-Verlag 2001, **2149:**11-26.
22. Sjolander K, Karplus K, Brown M, Hughey R, Krogh A, Mian IS, Haussler D: **Dirichlet mixtures: a method for improved detection of weak but significant protein sequence homology.**
Comput Appl Biosci 1996, **12**(4)**:**327-345.

23. Notredame C, Higgins DG, Heringa J: **T-Coffee: A novel method for fast and accurate multiple sequence alignment.**
J Mol Biol 2000, **302**(1)**:**205-217.

24. Thompson JD, Plewniak F, Poch O: **A comprehensive comparison of multiple sequence alignment programs.**
Nucleic Acids Res 1999, **27**(13)**:**2682-2690.

25. O'Sullivan O., Zehnder,M., Higgins,D., Bucher,P., Grosdidier,A. and Notredame,C. (2003) **APDB: a novel measure for benchmarking sequence alignment methods without reference alignments.** *Bioinformatics*, 19 Suppl. 1, I215–I221.

26. Bahr A, Thompson JD, Thierry JC, Poch O: **BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations.**
Nucleic Acids Res 2001, **29**(1)**:**323-326.

27. Thompson JD, Plewniak F, Poch O: **BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs.**
Bioinformatics 1999, **15**(1)**:**87-88.

28. Van Walle I., Lasters,I. and Wyns,L. (2004) **Align-m—a new algorithm for multiple alignment of highly divergent sequences.** *Bioinformatics*, DOI: 10.1093/bioinformatics/bth116.

29. Van Walle I, Lasters I, Wyns L. **SABmark - a benchmark for sequence alignment that covers the entire known fold space.** *Bioinformatics.* 2005 Apr 1;21(7):1267-8.

30. Schultz J., Milpetz,F., Bork,P. and Ponting,C.P. (1998) **SMART, a simple modular architecture research tool: identification of signaling domains.** Proc. Natl Acad. Sci. USA, 95, 5857–5864.

31. Schultz J., Copley,R.R., Doerks,T., Ponting,C.P. and Bork,P. (2000) **SMART: a web-based tool for the study of genetically mobile domains.** *Nucleic Acids Res*., 28, 231–234.

32. Ponting C.P., Schultz,J., Milpetz,F. and Bork,P. (1999) **SMART: identification and annotation of domains from signalling and extracellular protein sequences.** *Nucleic Acids Res.*, 27, 229–332.

33. Friedman M. (1937) **The use of ranks to avoid the assumption of normality implicit in the analysis of variance.** J. Am. Stat. Assoc., 32, 675–701.
34. http://www.drive5.com/muscle/muscle.pdf