Гибкое выравнивание белковых структур. Сервисы POSA, FATCAT.

Реферат студента 4 курса

Факультета Биоинженерии и Биоинформатики

Алешина Василия Алексеевича

22 декабря 2014

Содержание

Введение …………………………………………………………………………………………3

Обсуждение………………………………………………………………………………………4

Заключение……………………………………………………………………………………...11

Список литературы……………………………………………………………………………..12

Введение

Решение задачи пространственного выравнивания является интересной и важной с научной точки зрения. Как известно, гомологичные белки имеют сходство топологии и/или последовательности. Особый интерес представляют белки, которые имеют высокое сходство топологии при отсутствии гомологии по последовательности, и наоборот, белки с высоким сходством последовательностей, но различием в топологии, так как сопоставление таких белков позволяет выявить мотивы и аминокислотные остатки, важные для поддержания укладки белка, либо для возникновения подвижности и изменения укладки молекулы. Накопленные таким образом знания в перспективе будут полезны для моделирования и создания белков с заданными свойствами, что найдет широкое применение в промышленности и медицине. Существует много эвристических подходов к решению задачи пространственного выравнивания, многие из которых для совмещения молекул сперва совмещают их центры масс, а затем оптимизируют углы поворота. Однако целый ряд случаев нельзя разрешить с помощью такого «жесткого» выравнивания молекул, если например, один из доменов или участков последовательности повернут относительно остального белка. Не может помочь в этом случае иногда даже учет вторичной структуры белка, так как изгиб может и часто изменяет число спиралей и бета-тяжей, например в структурах спектрина 2spc и 1aj3 одна из спиралей сгибается пополам (рис. 1b из статьи [1]). Для решения этой проблемы применяют метод «гибкого» выравнивания, когда допускается внесение поворотов в структуру белка для наилучшего совмещения. Рассмотрим этот метод подробнее.

***Комментарий****. Сопоставляя рисунки (a) и (b) возникает вопрос, неужели DALI сделал выравнивание спирали II лишь до черной стрелки? Ответ: да.  
Вопрос второй: зачем же вводить поворот №1? Ответ: он не обязателен, но система подсчета весов алгоритма FATCAT этот поворот не уничтожила (см. далее ступени алгоритма).*

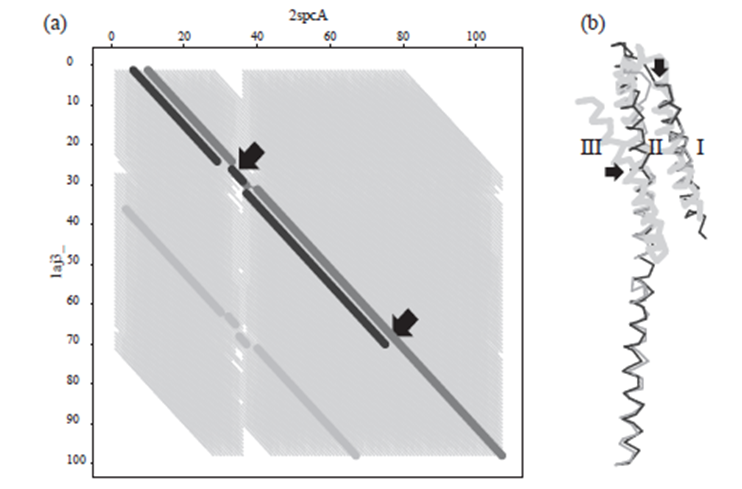


Рисунок 1. Из статьи [1]. Сравнение структур спектринов 1aj3 и 2spc. (а) матрица выровненных фрагментов по итогам работы алгоритмов FATCAT[1], DALI[2] и CE[3] показаны серым, черным и светло-серым соответственно. (b) пространственное совмещение структур 2spc (черным) по итогам работы алгоритмов DALI (1aj3 толстым серым) и FATCAT (1aj3 тонким серым). Черные стрелки на рисунках (а) и (b) обозначают места изгибов, внесенных алгоритмом FATCAT. Римские цифры (b) обозначают альфа-спирали.

Обсуждение

Очевидно, что алгоритмы гибкого выравнивания остро необходимы по крайней мере для выравнивания одинаковых или похожих по структуре белков, находящихся в разных конформациях, а также для поиска точек в последовательности белка, внесение поворота в которое значительно улучшает выравнивание (поиск таких точек для множества выравниваний позволит статистически обосновать предположение о подвижности молекулы в этой области). Первый алгоритм гибкого выравнивания FlexProt[4] появился в 2002 году, уже в 2003 году появился алгоритм FATCAT[1], который лучше справлялся с минимизацией числа вносимых в выравнивание поворотов (по данным авторов FATCAT, на тех же примерах, что использовали авторы FlexProt [1,4]).

Тут необходимо сделать важное пояснение. Когда алгоритм делает структурное выравнивание по Сα-атомам, он, в случае жесткого выравнивания, имеет право лишь выровнять пару атомов или пропустить тот или иной атом, как бы внеся гэп. Наиболее часто используемой мерой сходства является RMSD выравнивания, который алгоритмы стараются минимизировать, создавая при этом достаточно длинное выравнивание. Очевидно, что в общем, чем меньше выравнивание, тем лучше RMSD, и наоборот. В случае алгоритмов гибкого выравнивания разрешается также вносить повороты, и тоже очевидно, что чем больше поворотов, тем меньше RMSD при той же длине выравнивания. Соответственно ясно, что биологически осмысленным результатом работы алгоритма гибкого выравнивания должно стать выравнивание, при создании которого алгоритм минимизировал RMSD, число поворотов и максимизировал длину выравнивания. Функция оценки веса выравнивания, входящие в нее веса для поворотов, гэпов и продления выравнивания, а также некоторые особенные для каждого алгоритма константы и этапы будут тем, что различает все алгоритмы гибкого выравнивания, которые основываются на подсчете RMSD (алгоритмы гибкого сейчас выравнивания используют только этот подход). Так как алгоритм FATCAT (по [1], а также неопубликованным данным Алексеевского А.В.) вносит наименьшее число поворотов при сопоставимом RMSD и длине выравнивания, а также дает хорошие результаты без использования поворотов, то рассмотрим этот алгоритм для понимания проблем и способа их решения при создании гибкого выравнивания.

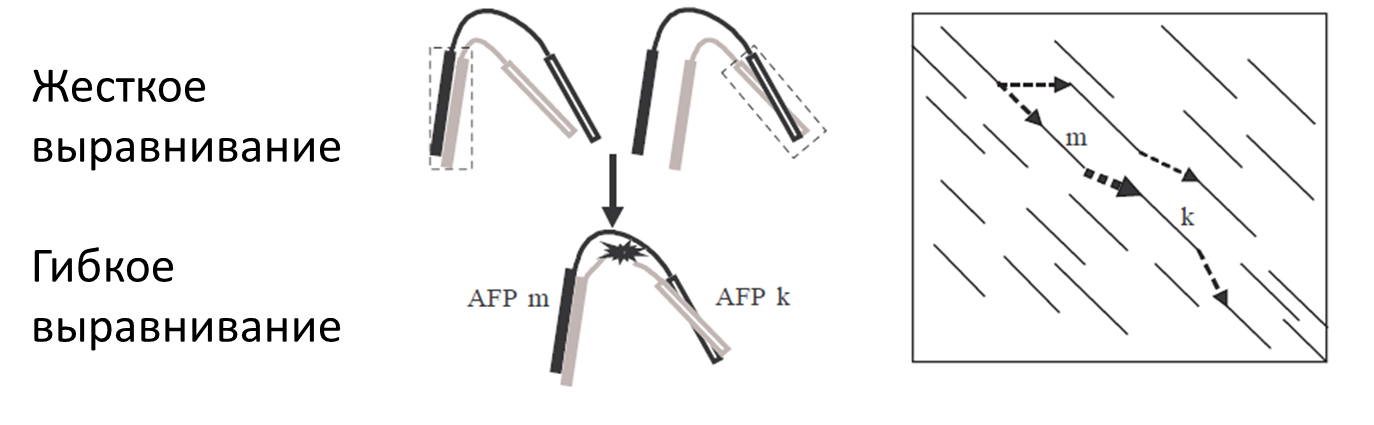


Рисунок 2. По [1]. Локальные выравнивания гипотетических структур (слева) и аналогичные локальные выравнивания, нанесенные в виде диагональных линий на матрицу выравнивания структур (см. рис. 1a). Пунктиром показаны способы получения глобального выравнивания.

Для понимания алгоритма обратимся к рисунку 2. На нем приведен пример, когда две структуры хорошо выравниваются двумя локальными участками, которые на точечной матрице выравнивания (рис. 1а) будут выглядеть как два диагональных отрезка. Для получения глобального выравнивания алгоритм гибкого выравнивания может сделать поворот между участками, что на точечной матрице будет выглядеть как горизонтальный или вертикальный отрезок. Нахождение пути из района N-конца последовательностей в область С-концов (не обязательно в самые углы), проходящего через наибольшее число выровненных остатков при наименьшем числе поворотов, и является задачей алгоритма.

Тут возникают первые уникальные для алгоритма FATCAT константы, отвечающие за определение локальной выровненной пары (ЛВП), это длина пары (в алгоритме это 8, но авторы на этой константе не настаивают [1]) и порог RMSD при котором ЛВП засчитывается. ЛВП находятся динамическим программированием без разрешения поворотов, и за порог принято 3Å, авторы FATCAT[1] ссылаются на алгоритм СЕ[3] при выборе этого порога.

После того, как все ЛВП найдены (рис 2, матрица) необходимо задать некоторую величину, характеризующую поворот, который возникнет для перехода от одной ЛВП к другой. В качестве такой меры авторы предлагают корень из суммы квадратов разностей расстояний между i-ми остатками двух участков структур, соответствующих двум ЛВП. Чем меньше различие положений участков ЛВП в структурах (то есть, чем меньше поворот, который возможно придется совершать), тем меньше величина D (формула из [1] и упрощенная мной схема на рис. 3).

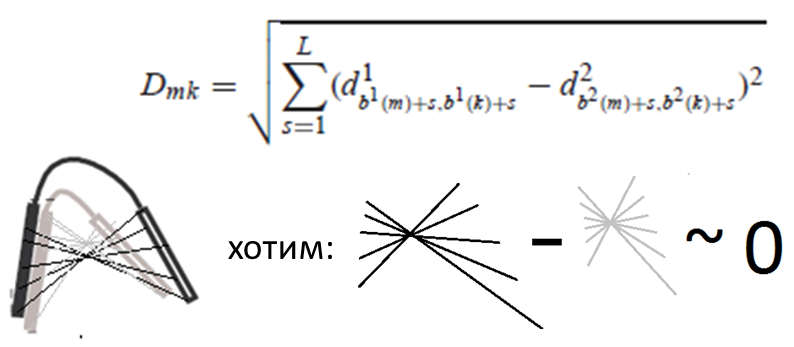


Рисунок 3. Проверка совместимости ЛВП по алгоритму FATCAT[1].

Воспользуюсь правом автора избавить читателя от разглядывания громоздких, но простых по своей сути формул со множеством переменных, постаравшись их обобщить (если читатель не согласен, формулы взятые из [1] почти полностью присутствовали в докладе, который читатель, смею надеяться, помнит). Итак, величина D, как мы показали связана с углом поворота между парой ЛВП, и логично предположить, что большие повороты менее вероятны, а очень малые вероятны. Еще три константы, одна формула, и у нас есть способ оценки перехода между ЛВК, а общая весовая функция (Score) будет в некотором приближении суммой весов соединенных ЛВП за вычетом штрафов за переходы. Эту сумму и нужно максимизировать. Внимательный наблюдатель может спросить, на каком же этапе вносятся повороты, ведь до сих пор мы ни разу их искусственно не вносили, а лишь подсчитывали веса возможных переходов по функции, которая обратно пропорциональна величине поворота. Действительно, это особенность алгоритма FATCAT – он не вносит поворот сразу, а уже выбрав наиболее вероятные участки для выравнивания, подбирает динамическим программированием способы перехода от ЛВП к ЛВП так, чтобы минимизировать число поворотов и RMSD. Кроме того, к уже подобранному выравниванию применяется некая операция подгонки, зависящая от выигрыша в RMSD и установленного порога на этот выигрыш, когда число поворотов может измениться в большую или меньшую сторону. По моим представлениям, этот шаг – главное преимущество алгоритма FATCAT над алгоритмом FlexProt. Алгоритм же FlexSnap[11], по данным его разработчиков, «не уступает … FATCAT по RMSD и длине выравнивания», что ж, этот алгоритм разработан на 7 лет позже, и не уступает старому FATCAT, выводы за читателем.

Как уже упоминалось выше, все формулы и этапы алгоритма FATCAT целесообразны и прозрачны, а для всех подобранных параметров (всего девять) указаны их использованные значения[1]. Создатели алгоритма FATCAT на большом количестве примеров, выбранных не ими, а используемыми в более ранних статьях, описывающих другие алгоритмы, показали, что новый алгоритм работает лучше других алгоритмов гибкого выравнивания и не хуже алгоритмов жесткого выравнивания. К сожалению, не удалось понять, как работает алгоритм в случае запретов поворотов, но судя по приведенным примерам, он не использует ни один из известных алгоритмов (мои подозрения пали на алгоритм СЕ[3], так как авторы FATCAT считают его лучшим из жестких алгоритмов выравнивания, но нет, это не он, см. табл. 1 в [1]). Смею предположить, что в данном случае проделываются все те же операции, что и при гибком выравнивании, но на последнем шаге все внесенные повороты запрещаются. Также возможно запретить большие повороты на шаге №2 с помощью порогов на функцию D (рис. 3).

Создав алгоритм гибкого выравнивания, логично:

1. Создать сервер на его основе [5,6]
2. Применить его к большим данным и дать статистические оценки[7]
3. Создать сервер множественного гибкого выравнивания[8,9]
4. Оптимизировать работу сервера, создать собственную базу данных[10]
5. Поддерживать работу сервера, заботиться о его функционале[12].

Все эти шаги были проделаны коллективом лаборатории – создателей FATCAT. Быстро пробежимся по всем этапам, обратив более пристальное внимание лишь на способ перехода от парного выравнивания к множественному[8].

Созданный сервер FATCAТ предоставлял широкий спектр возможностей для сравнения, на вход принимал идентификаторы структур PDB и SCOP или текстовые файлы, мог выдавать результаты на почту, причем его результаты были не только картинками, но имели и текстовый вариант выдачи (рис. 4 верх). Кроме того, сервер предлагал возможность сканирования выбранных баз данных против заданной структуры, что позволяло не только отыскать лучшие результаты, но и выявить наиболее частые точки внесения поворотов, что тоже имеет, на мой взгляд, большое значение (рис.4 низ).

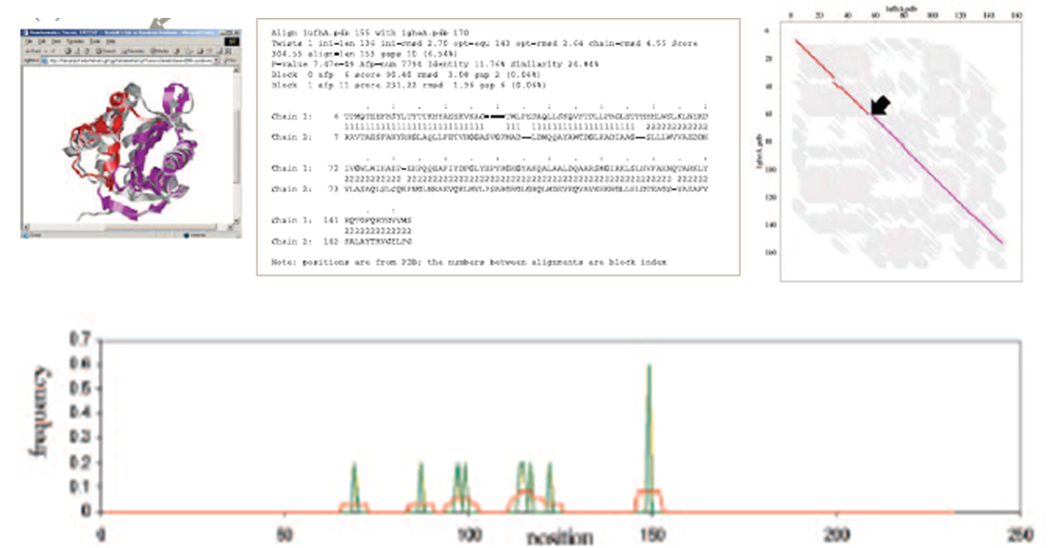


Рисунок 4. Результаты работы сервиса, реализующего алгоритм FATCAT, сверху представлены RasMol-скрипт или pdb-файл с двумя цепями выровненных белков, парное выравнивание и его визуализация в виде точечной матрицы. Снизу представлен результат модификации FATCAT-search, которая реализовывала поиск по базе данных, зеленым показаны места внесения поворота, оранжевым – сглаженная функция, характеризующая подвижность по этим поворотам.

*#Почему я использую прошедшее время в отношении сервиса FATCAT? Потому после создания сервиса множественного выравнивания, авторы обращали мало внимания на сервис парного выравнивания. Немного жаль модификацию FATCAT-search, остальные опции не были утрачены, а реализуются теперь в рамках сервиса POSA[9,12].*

В рамках крупной статистической работы[7], и более наглядного, чем в [1], сравнения методов гибкого и жесткого выравниваний, были получены статистические выводы, что гибкое выравнивание не хуже жестких, но существует небольшая доля случаев, когда оно дает значительные преимущества.

*#По-видимому, именно редкая необходимость применения гибкого выравнивания сказалась на том, что этот метод почти не прижился в практике.*

Вслед за парным выравниванием была предложена идея множественного выравнивания[8]. Опуская детали, идея очень проста: из заданных структур выровняем все со всеми. Наиболее схожие выровняем между собой попарно, затем выровняем наиболее схожие выравнивания и так далее до получения одного общего множественного выравнивания (рис 5).

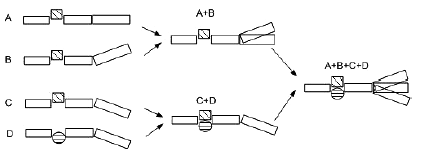


Рисунок 5. Идея множественного выравнивания с помощью парных выравниваний.

Изначально разработчики не настаивают на применении предложенного способа выбора порядка выравниваний, по сути представляющего собой дерево. Они (в более ранней модификации POSA) даже давали возможность задавать свое дерево для выравнивания, так как от этого шага значительно зависит полученный результат, но позднее эта возможность пропала[9].

*#на сегодня пользователь может управлять процессом несколько иначе, о чем речь ниже.*

Полезным для ряда целей является выдача сервиса в виде графа, который может иметь гораздо более сложный вид, чем на рисунке 6, однако, как минимум после усовершенствования сервиса в 2014 году, граф выдается в виде картинки.

Важным кажется, что время работы алгоритма O(L2N3), где L– средняя длина последовательностей, N– число последовательностей, большое время работы алгоритма POSA, а также большое время работы созданного незадолго до этого сервиса FATCAT-search, привело к созданию базы данных-хранилища результатов выравнивания всех структур со всеми[10]. Это колоссально ускорило работу FATCAT-search (так, если в [6] речь идет о 11 часах работы FATCAT по базе PDB в 2004 году, то сегодня, благодаря хранению данных, результат получается за секунды). Одна ремарка: база данных не обновлялась уже очень давно.

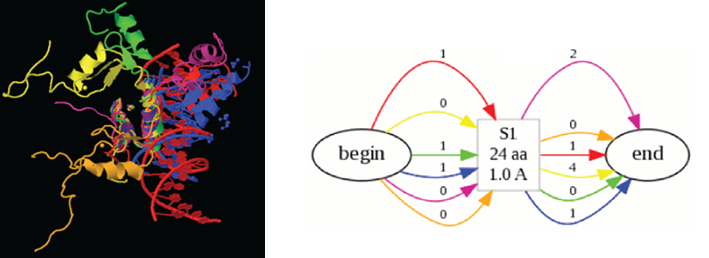


Рисунок 6. Множественное выравнивание JMol, реализуемое POSA[12], а также использованный при его создании небольшой участок последовательности каждой из выравненных молекул (Zn-палец в центре выравнивания), сопоставленный в виде графа.

На этом основные действия по улучшению и применению FATCAT были приостановлены до 2014 года, когда по предложениям некоторого пользователя был усовершенствован сервис POSA[12]. Изменения коснулись цветового оформления сервиса, скрипты и результаты были переделаны с PyMol под JMol (рис. 6). Из более важного для пользователей:

1. Применена возможность синхронно просматривать до 6 структур в отдельных окнах, а также в общем окне совместно (рис.7).
2. Можно удобно выбирать и задавать цепи белка (или ДНК), которые необходимо показать при выдаче, для возможности сопоставления наличия/отсутствия лигандов без отрыва от выравненных белков.
3. Предоставлена возможность задавать участок последовательности белка, на котором будет применяться алгоритм выравнивания.

Последнее усовершенствование позиционируется как уникальная возможность, благодаря которой пользователь может использовать свою интуицию и чутье для управления алгоритмом выравнивания, так как все эвристические подходы, как бы они ни старались, могут не найти важного результата.

*#почему при этом нельзя оставить возможность влиять на дерево выравнивания структур, а также на константы алгоритма, догадаться нельзя.*

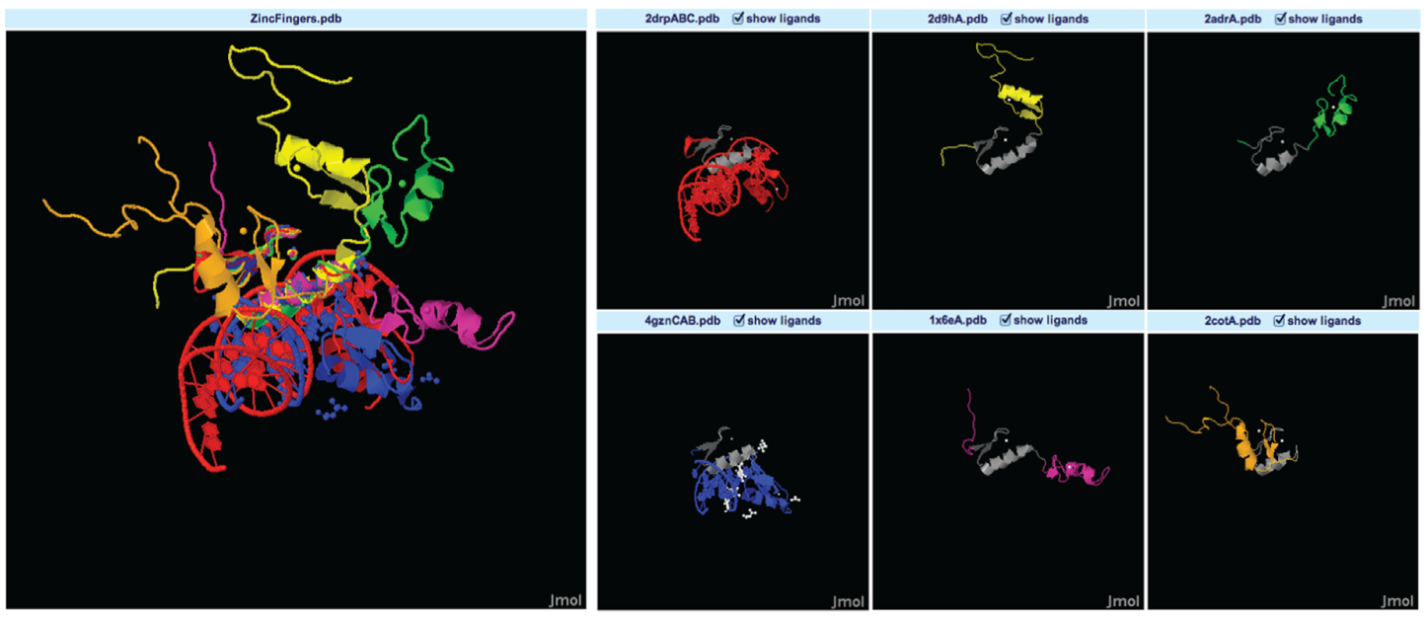


Рисунок 7. Окна Jmol с множественным выравниванием POSA[12]. Слева видно большое количество лигандов, показанных на выравнивании, а справа серым показана часть структуры, принимающая участие в выравнивании (Zn-палец).

В ходе работы с сервисом POSA мне удалось выявить его важные недостатки, достоинства и особенности.

Во-первых, воспользовавшись примером по умолчанию из [9,12], но очистив поля где указывалось, какие участки структур стоит выравнивать, а какие достроить в неизменном виде, я получил выравнивание (рис. 8), которое демонстрирует недостаток алгоритма POSA.

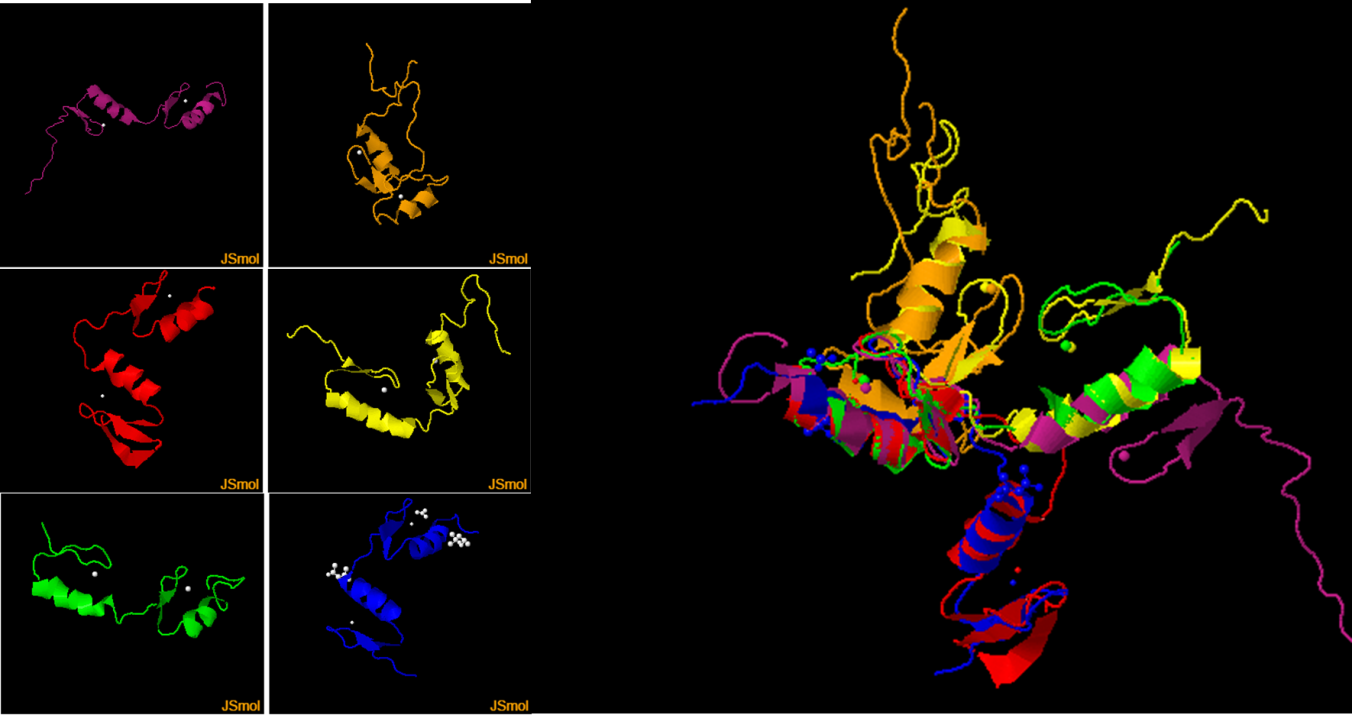


Рисунок 8. Множественное выравнивание POSA структур по умолчанию[9,12] по всем 6 структурам полностью.

Как мы видим, все 6 структур содержат по 2 домена Zn-пальца, и из-за этого множественное выравнивание «развалилось» на 4 выравниваемых узла и повороты между ними. Так, например, красная и синяя последовательность отлично выравниваются между собой, но желтая выравнивается частично с оранжевой, а частично с зеленой и фиолетовой, и так далее. Все это, очевидно, результат ошибки на этапе определения очередности выравнивания структур, то есть, на этапе дерева выравнивания. Также можно видеть несовершенство выравнивания правого домена фиолетовой структуры. В этом случае причину ошибки определить уже сложнее. Ясно, что хоть и выравнивание таких по-разному сложенных двудоменных Zn-пальцев с помощью FATCAT прошло хорошо, но вот где-то на этапе алгоритма POSA ошибки появились в большом количестве, и это уже недостаток эвристического подхода.

*#Ради справедливости я решил использовать те же белки для выравнивания другими сервисами: mulPBA[13,14] и PDBeFold[15].*

*#Вне всякого сомнения, это хорошие сервисы для задач жесткого выравнивания, но для выравнивания выбранных 6 структур двудоменных цинковых пальцев они не подходят вовсе. Эти сервисы не смогли выровнять хотя бы один домен Zn-пальца, если им подавать полные структуры.*

Заключение

Задача множественного выравнивания полностью включает в себя задачу парного выравнивания и в случае FATCAT и POSA различается незначительно.

Данный тип задачи NP-полный, поэтому использует различные эвристики, удачное применение которых сокращает время и вычисления, а неудача приводит к ошибкам. По этой причине все программы выравнивания структур допускают ошибки, поэтому возможность пользователя управлять местом выравнивания – преимущество алгоритма POSA. Такое же преимущество – возможность создания гибкого выравнивания, что правда, актуально лишь в небольшом ряде случаев, при этом время работы алгоритма POSA значительно превосходит аналогичные программы жесткого выравнивания.

Сервисы FATCAT и POSA удобны в использовании, просты и понятны, однако, причина их слабой используемости в том, что люди редко сталкиваются с необходимостью применять гибкое выравнивание, а в других случаях использовать медленные алгоритмы нет никакого смысла.

Список литературы

[1] Flexible structure alignment by chaining aligned fragment pairs allowing twists. Ye Y, Godzik A. Bioinformatics. 2003 Oct;19 Suppl 2:ii246-55. PMID: 14534198.

[2] Protein structure comparison by alignment of distance matrices. Holm L, Sander C. J Mol Biol. 1993 Sep 5;233(1):123-38. PMID: 8377180.

[3] Protein structure alignment by incremental combinatorial extension (CE) of the optimal path.

Shindyalov IN, Bourne PE. Protein Eng. 1998 Sep;11(9):739-47. PMID: 9796821.

[4] Flexible protein alignment and hinge detection. Shatsky M, Nussinov R, Wolfson HJ. Proteins. 2002 Aug 1;48(2):242-56. PMID: 12112693

[5] http://fatcat.burnham.org/

[6] FATCAT: a web server for flexible structure comparison and structure similarity searching. Ye Y, Godzik A. Nucleic Acids Res. 2004 Jul 1;32(Web Server issue):W582-5. PMID: 15215455

[7] Database searching by flexible protein structure alignment. Ye Y, Godzik A. Protein Sci. 2004 Jul;13(7):1841-50. PMID: 15215527

[8] Multiple flexible structure alignment using partial order graphs. Ye Y, Godzik A. Bioinformatics. 2005 May 15;21(10):2362-9. Epub 2005 Mar 3. PMID: 15746292

[9] http://posa.sanfordburnham.org/

[10] Flexible Structural Neighborhood--a database of protein structural similarities and alignments. Li Z, Ye Y, Godzik A. Nucleic Acids Res. 2006 Jan 1;34(Database issue):D277-80. PMID: 16381864.

[11] FlexSnap: flexible non-sequential protein structure alignment. Salem S, Zaki MJ, Bystroff C. Algorithms Mol Biol. 2010 Jan 4;5:12. doi: 10.1186/1748-7188-5-12. PMID: 20047669

[12] POSA: a user-driven, interactive multiple protein structure alignment server. Li Z, Natarajan P, Ye Y, Hrabe T, Godzik A. Nucleic Acids Res. 2014 Jul;42(Web Server issue):W240-5. doi: 10.1093/nar/gku394. Epub 2014 May 16. PMID: 24838569

[13] mulPBA: an efficient multiple protein structure alignment method based on a structural alphabet. Léonard S, Joseph AP, Srinivasan N, Gelly JC, de Brevern AG. J Biomol Struct Dyn. 2014 Apr;32(4):661-8. doi: 10.1080/07391102.2013.787026. Epub 2013 May 10. PMID: 23659291

[14] http://www.dsimb.inserm.fr/dsimb\_tools/mulpba/

[15] http://www.ebi.ac.uk/msd-srv/ssm/