Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра информационно-аналитических систем

Кривоносова Кристина Артуровна

Улучшение чувствительности поиска синтенных блоков при помощи постобработки

Бакалаврская работа

Допущена к защите.

Зав. кафедрой:

к.ф.-м.н., доцент Михайлова Е. Г.

Научный руководитель:

к. ф.-м.н., доцент Графеева Н.Г.

Научный консультант:

аспирант Минкин И.В.

Рецензент:

к.ф.м.н., доцент Евдокимова Т. О.

Санкт-Петербург 2017

Saint Petersburg State University

Department of Analytical Information Systems

Krivonosova Kristina

Sensitivity improvement of synteny block detection with postprocessing

Bachelor's Thesis

Admitted for defence.

Head of the chair:

Associate Professor Elena Mikhailova

Scientific supervisor:

Associate Professor Natalia Grafeeva

Scientific consultant:

PhD student Ilya Minkin

Reviewer:

Associate Professor Evdokimova Tatyana

Saint-Petersburg 2017

Содержание

Аннотация....3

Введение....3

1. Постановка задачи....6

1.1. Терминология....6

1.2. Формализованная постановка задачи....9

2. Обзор существующих методов кластеризации геномных последовательностей10

2.1. Сравнение последовательностей....10

2.2. Методы кластеризации....13

3. Выбор наиболее эффективного метода....17

3.1. Описание методов....17

3.1.1. Представление данных и сравнение блоков....17

3.1.2. Разбиение компонент связности....19

3.1.3. Кластеризация....19

3.2. Эксперименты....23

3.2.1. Выбор оптимальных параметров для каждого метода....24

3.2.2. Результаты....26

4. Сравнение выбранного метода с существующими инструментами....27

Заключение....28

Список литературы....30

Аннотация

На данный момент существует большое количество инструментов, производящих декомпозицию геномов на непересекающиеся синтенные блоки. Однако, существующие чувствительные и точные методы неприменимы к длинным последовательностям, а потенциально быстрые методы обладают худшей чувствительностью. В данной работе предлагаются методы постобработки синтенных блоков, основанные на кластеризации и призванные увеличить чувствительность быстрых инструментов.

Введение

В 1990 году был запущен проект «Геном человека», с целью получить нуклеотидную последовательность ДНК человека. Тем не менее, только спустя полтора десятка лет в 2004-ом году была опубликована окончательная версия генома [8].

Благодаря развитию технологий секвенирования [10], дальнейшие темпы секвенирования генома возрастали экспоненциально. В 2005-ом году был опубликован геном шимпанзе [3], который подтвердил близкое сходство между обезьянами и человеком. К 2008 году были полностью прочитаны геномы 32 позвоночных, 3 генома беспозвоночных вторичноротых, 15 геномов насекомых, 7 геномов червей и сотни геномов бактерий [9].

Каждый геном представляется последовательностью нуклеотидов, закодированных определённым набором символов (A, T, C, G). Длины этих последовательностей варьируются от нескольких тысяч до нескольких миллиардов нуклеотидов, в зависимости от организма. На данный момент в руках учёных скопился огромный запас секвенированных, но еще не проанализированных данных. Поэтому одной из самых актуальных задач биоинформатики на сегодняшний день является определение функциональности каждого участка ДНК. Так, например, сейчас науке известна функциональность всего лишь 5% ДНК в геноме человека (в них содержаться кодирующие участки), тогда как назначение остальных 95% пока не ясно.

Сравнение последовательностей ДНК различных организмов между собой оказалось очень плодотворным методом поиска новых функционально важных последовательностей в геноме: на основе информации о функциональности участков в одной ДНК, можно делать выводы о назначении схожих участках в другом ДНК.

Несмотря на внешнюю несхожесть различных организмов, их геномы могут различаются между собой всего на несколько процентов. В силу такого подобия геномов даже род мух Drosophila может быть использован для более полного понимания функций тех или иных человеческих генов, в частности, ответственных за возникновение и развитие некоторых заболеваний.

Подобие геномов различных организмов заключается в наличии в каждом из них некоторых схожих блоков, которые были унаследованы от одного общего предка. Такие блоки называются синтенными — это консервативные последовательности нуклеотидов, которые схожи в нескольких геномах с точностью до незначительных перестроек: небольших вставок, удалений, инверсий или дупликаций участков последовательности[27].

Задача декомпозиции геномов на не перекрывающиеся синтенные блоки очень важна для сравнения геномов. Данные блоки дают возможность сравнивать широкий диапазон геномов, анализируя перестройки внутри них: геномы представляются в виде перестановок блоков, между которыми можно затем высчитать расстояния и выбрать наиболее близких кандидатов для более глубокого изучения. Кроме того, идентификация синтенных регионов может помочь в изучении эволюции хромосом: по количеству общих блоков в геномах можно судить о близости видов между собой.

В настоящее время существует достаточно большое количество инструментов, позволяющих искать синтенные блоки: Sibelia [1], DRIMM-Synteny [6], Mugsy [2], Mauve [7] и многие другие.

Однако, при сравнении даже двух геномов, как правило, содержащих огромное количество нуклеотидов (вплоть до нескольких миллиардов), чувствительные[1] и точные методы становятся неприменимы из-за экспоненциальной сложности, а потенциально быстрые методы обладают худшей чувствительностью, которую, однако, можно улучшить при помощи постобработки.

Для улучшения чувствительности инструмента можно применять различные методы кластеризации блоков, используя уже найденные схожие участки в последовательностях.

---

[1] Чувствительность - доля гомологичных пар оснований, входящих в найденные блоки. Точное определение будет дано в главе 2.

Целью данной дипломной работы является улучшение чувствительности инструмента Sibelia [1] с помощью кластеризации найденных им синтенных блоков.

Для достижения этой цели в рамках работы были сформулированы следующие задачи:

1. Ознакомиться с предметной областью (синтенные блоки, отношения гомологии и выравнивания).
2. Исследовать и описать существующие методы кластеризации геномных последовательностей,
3. Адаптировать данные методы для улучшения чувствительности инструмента Sibelia и определить наиболее эффективный из них.
4. Провести сравнение полученных результатов с результатами работы существующих инструментов.

1. Постановка задачи

История эволюции биологических последовательностей в основном неизвестна, но может быть восстановлена из геномных последовательностей с определенным предположением об эволюционной модели.

Многие методы восстановления эволюционной истории предполагают нахождение участков геномов, унаследованных от общего предка [17].

Зачастую, инструменты при поиске родственных областей в последовательностях следуют предположению о том, что чем более похожи участки геномов, тем наиболее вероятно, что они были унаследованы от одного предка. Такие участки называют синтенными блоками.

1.1. Терминология

Множество последовательностей ДНК $S = \{s_1, \dots, s_n\}$ – это множество последовательностей в алфавите $\pi = \{A, T, C, G\}$. За $x[i]$ обозначаем $i-$ый символ в последовательности $x = x_1 \dots x_i \dots x_m$.

Обозначим за S' множество всех пар (i, j), где $i \in \{1, \dots, n\}$, а $j \in \{1, \dots, |s_i|\}$

Определим *отношение гомологии* $\sim$, действующее на $S' \times S'$ такое, что $(i_1, j_1) \in S'$, $(i_2, j_2) \in S'$ и $(i_1, j_1) \sim (i_2, j_2)$, если и только если нуклеотиды $s_{i_1}[j_1]$ и $s_{i_2}[j_2]$ произошли от одного предка.

Данное отношение является отношением эквивалентности, то есть оно рефлексивно, транзитивно и симметрично.

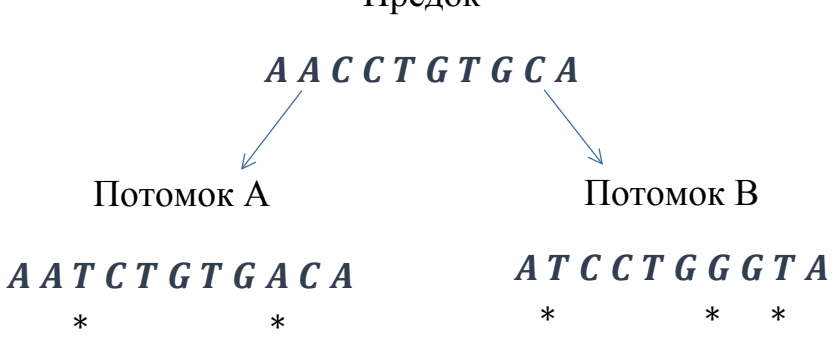

Рис.1 Две последовательности, произошедшие от общего предка, разделившиеся вследствие мутаций.

Выравнивание последовательностей - один из методов нахождения возможных гомологичных позиций нуклеотидов в геноме.

Для множества последовательностей строк $S = \{s_1, \dots, s_n\}$ в алфавите π *множественным выравниванием* называется множество строк $S^* = \{s_1^*, \dots, s_n^*\}$ в алфавите $\pi^* = \pi \cup \{-\}$ таких, что:

- каждая строка s_i^* получена вставками символа "$-$" в строку s_i
- хотя бы в одной из строк $s_1^*, \dots, s_n^*$ на месте j находится символ, отличный от "$-$"
- все строки $s_1^*, \dots, s_n^*$ имеют одинаковую длину $l : \max_i |s_i| \le l \le \sum_{i=1}^{n} |s_i|$

и определены неубывающие функции $\nu_i : \{1,\dots, |s_i|\} \to \{1, \dots, l\}$, отображающие позиции символов изначальных последовательностей в позиции непустых ($\neq " - "$) символов соответствующих строк в выравнивании:

1. $s_i^*[\nu_i(j)] = s_i[j]$ для $\forall j \in \{1, \dots, |s_i|\}$
2. если j из $\{1, \dots, l\}$ такое, что $\nu_i(k) \neq j \ \forall k \in \{1,\dots, |s_i|\}$, тогда $s_i^*[j] = " - "$

Тогда *отношение выравнивания* $\sim^*$, определенное на $S' \times S'$, задается следующим образом:

Пусть для множества строк $S = \{s_1, \dots, s_n\}$ определено некоторое выравнивание $S^* = \{s_1^*, \dots, s_n^*\}$, тогда:

$$s_i^*[j] \neq " - " \text{ и } s_k^*[j] \neq " - " \leftrightarrow \exists\, j_1, j_2 \ \nu_i(j_1) = \nu_k(j_2) = j \rightarrow (i, j_1) \sim^* (k, j_2)$$

Иными словами, пара (i, j_1), (k, j_2) попадет в отношение $\sim^*$, тогда и только тогда, когда образы позиций j_1 и j_2 совпадают, а в выравниваниях данных строк на j-ой позиции нет символа "$-$", где j общий образ для $\nu_i(j_1)$ и $\nu_k(j_2)$.

Например, на рис.4 в отношение выравнивания входят пары:

$$s_1[1] \sim^* s_2[1],\ s_1[2] \sim^* s_2[2], s_1[3] \sim^* s_2[3], \dots, s_1[10] \sim^* s_2[9], s_1[11] \sim^* s_2[10].$$

s_1	A A T C T G T G A C A	s_1^*	A A T C T G T G A C A
s_2	A T C C T G G G T A	s_2^*	A T C C T G G G - T A

Рис.2 Пример выравнивания для двух последовательностей

Отношение $\sim$ обычно неизвестно и его требуется найти. Один из способов восстановления отношения - поиск синтенных блоков и последующим построением для них отношения выравнивания.

Пусть дано *множество блоков* $B = \{b_1, \dots, b_m\}$ – результат работы одного из инструментов поиска синтенных блоков с входным множеством последовательностей $S = \{s_1, \dots, s_n\}$.

Каждый *блок* b_i представляет собой множество подстрок последовательностей из S: $b_i = \{c_1, \dots, c_{k_i}\}$

$c_j = s_t[h_j : h_j + l] = s_t[h_j]s_t[h_j + 1]\dots s_t[h_j + l]$ для некоторого $j \in \{1, \dots, k_i\}$ и $s_t \in S$.

Длина каждого c_j в блоке превышает некоторое пороговое значение τ.

"$AATCGGTA$"
"$ACATCGGTA$"
"$AATCGGTA$"

Рис.3 Пример блока b_i

Назовем *отношением выравнивания для некоторого множества блоков* B объединение всех отношений выравнивания его элементов. Иными словами, $\forall b_i \in B$ и его соответствующего выравнивания $\sim'_i$ (блоки состоящие из одного элемента имееют пустое отношение выравнивания), отношение выравнивания для B это $\sim' = \bigcup_{i=1}^{m} \sim'_i$.

Для отношения выравнивания $\sim'$ множества блоков $B = \{b_1, \dots, b_m\}$ можно определить показатели, сравнивающие его с отношением гомологии:

Чувствительность отношения $\sim'$ $sensitivity(\sim') = \frac{(|\sim| - |\sim \backslash \sim'|)}{|\sim|}$

Чувствительность является величиной, определяющей долю гомологичных пар позиций в отношении выравнивания среди всех существующих гомологичных позиций.

Точность отношения $\sim'$ $precision(\sim') = \frac{(|\sim'| - |\sim' \backslash \sim|)}{|\sim'|}$

Точность определяет отношение количества гомологичных пар позиций к количеству всех позиций, найденных в отношении выравнивания.

F- мера отношения $\sim'$ $F(\sim') = 2\,\frac{sensitivity(\sim') \times precision(\sim')}{sensitivity(\sim') + precision(\sim')}$

F-мера это показатель, совмещающий чувствительность и точность, необходимый для определения отношения между этими двумя показателями.

1.2 Формализованная постановка задачи

Задачей данной работы является поиск такого метода кластеризации множества блоков $B = \{b_1, \dots, b_m\}$, чтобы отношение выравнивания $\sim''$ для множества блоков $B'' = \{b_1'', \dots, b_s''\}$, получившегося объединением блоков из одного кластера, удовлетворяло следующим условиям:

1. $sensitivity(\sim'') > sensitivity(\sim')$
2. $F(\sim') \leq F(\sim'')$
3. $sensitivity(\sim'') = \max\limits_{\sim\sim \in U^{\sim}} (sensitivity(\sim\sim))$,

$U^{\sim}$ – множество отношений, которые удовлетворяют свойствам 1-2 и являются отношением выравнивания для некоторого B'', полученного с помощью объединения некоторых блоков в B

Первое и третье условие направлены на выбор метода кластеризации, который бы давал максимальное значение чувствительности. Однако, при резком увеличении чувствительности, точность может так же резко снижаться. Для предотвращения такой ситуации вводится ограничение 2, контролирующее величину F-меры отношения выравнивания, устанавливая для нее нижнюю границу.

2. Обзор существующих методов кластеризации геномных последовательностей.

В данном разделе будет приведен обзор некоторых методов сравнения последовательностей ([11], [12]) и кластеризации геномных последовательностей ([14-16], [18-20], [23])

2.1. Сравнение последовательностей

Результат любого метода кластеризации напрямую зависит от выбора метода сравнения последовательностей.

Методы сравнения геномных последовательностей разбиваются на две основные группы: основанные на алгоритмах выравнивания (alignment-based methods) [11] и использующие альтернативный подход (alignment-free methods) [12].

Сравнения последовательностей с помощью выравнивания

Выравнивание последовательностей - биоинформатический метод, основанный на размещении двух или более последовательностей мономеров ДНК, РНК или белков друг под другом таким образом, чтобы явно выделить сходные участки в этих последовательностях

В основном, все алгоритмы выравнивания используют динамическое программирование для пошагового преобразования (вставка, удаление или замена символов) одной строки в другую, применяя систему весов (score) и штрафов.

Значения веса для каждой операции замены определяется матрицей замен (substitution-score matrix). Для аминокислотных последовательностей это матрица 20×20 (для нуклеотидных 4×4). Значения ячеек матрицы высчитываются основываясь на частоте замены одной аминокислоты на другую в уже известных выравниваниях [21].

Штрафы взимаются за добавление символа "—" в выравнивание, другими словами, удаление или вставку символа.

Суммарное значение стоимости всех замен и штрафов является весом всего выравнивания, по которому можно судить о степени схожести двух последовательностей.

C Cys	12																			
S Ser	0	2																		
T Thr	-2	1	3																	
P Pro	-3	1	0	6																
A Ala	-2	1	1	1	2															
G Gly	-3	1	0	-1	1	5														
N Asn	-4	1	0	-1	0	0	2													
D Asp	-5	0	0	-1	0	1	2	4												
E Glu	-5	0	0	-1	0	0	1	3	4											
Q Gln	-5	-1	-1	0	0	-1	1	2	2	4										
H His	-3	-1	-1	0	-1	-2	2	1	1	3	6									
R Arg	-4	0	-1	0	-2	-3	0	-1	-1	1	2	6								
K Lys	-5	0	0	-1	-1	-2	1	0	0	1	0	3	5							
M Met	-5	-2	-1	-2	-1	-3	-2	-3	-2	-1	-2	0	0	6						
I Ile	-2	-1	0	-2	-1	-3	-2	-2	-2	-2	-2	-2	-2	2	5					
L Leu	-6	-3	-2	-3	-2	-4	-3	-4	-3	-2	-2	-3	-3	4	2	6				
V Val	-2	-1	0	-1	0	-1	-2	-2	-2	-2	-2	-2	-2	2	4	2	4			
F Phe	-4	-3	-3	-5	-5	-5	-4	-6	-5	-5	-2	-4	-5	0	1	2	-1	9		
Y Tyr	0	-3	-3	-5	-3	-5	-2	-4	-4	-4	0	-4	-4	-2	-1	-1	-2	7	10	
W Trp	-8	-2	-5	-6	-6	-7	-4	-7	-7	-5	-3	2	-3	-4	-5	-2	-6	0	0	17
	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W

Рис. 4 PAM (1-250) (Point Accepted Mutations) - матрица аминокислотных замен [22]

В качестве мер схожести и расстояний для сравнения последовательностей могут использоваться:

- максимальный вес локальных выравниваний (*score*)
- *E-value* - ожидаемое количество локальных выравниваний с заданным или большим весом, если рассматривать в качестве случайной величины вес локального выравнивая
- *P-value* - вероятность встретить (хоть одно) локальное выравнивание с заданным или большим весом, если рассматривать в качестве случайной величины вес локального выравнивая
- *p-distance* - процент не совпадающих символов в выравнивании последовательностей.

Альтернативный подход сравнения последовательностей.

Основной идеей в данном подходе является выявление признаков или шаблонов из последовательностей для представления их в виде векторов.

Данный подход удобен при использовании классических методов кластеризации (K-means, DBSCAN и др.), определенных для векторных пространств.

Один из методов представление последовательности в виде вектора - это использование частоты появления k-мера (подпоследовательности длины k) внутри последовательности.

Для ДНК последовательностей количество всех возможных k-меров равняется $t = 4^k$. Для каждой такой последовательности строится вектор $c = < c_1, ..., c_t >$, где значение каждого c_i определяется количеством встречаемости i-ого k-мера в лексикографической последовательности k-меров. Так же рассматривается вектор частот $f =< f_1, ..., f_t >$ определяющийся уравниванием:

$$f = \frac{c}{\sum_{i=1}^{t} c_i} \leftrightarrow f_i = \frac{c_i}{n - k + 1}$$

n - длина исходной последовательности.

Используя данные вектора, можно сравнивать последовательности с помощью известных метрик:

Для последовательностей X и Y в алфавите $\pi = \{A, C, G, T\}$ и соответствующих векторов $< c_{k,1}^X, ..., c_{k,t}^X >$, $< c_{k,1}^Y, ..., c_{k,t}^Y >$ и $< f_{k,1}^X, ..., f_{k,t}^X >, < f_{k,1}^Y, ..., f_{k,t}^Y >$

- Евклидово расстояние :

$$d_k^E(X, Y) = \sqrt{\sum_{i=1}^{t} \left(c_{k,i}^X - c_{k,i}^Y \right)^2}$$

- Взвешенное евклидово расстояние

$$d^2 = \sum_{L=k}^{u} \sum_{i=1}^{t} p_i^L \left(c_{L,i}^X - c_{L,i}^Y \right)^2$$

p_i^L - вес i-ого L-мера

- Угловая метрика (angle metrics)

$$d_L^{cos} = \theta_{X,Y} \text{ , где } cos(\theta_{X,Y}) = \frac{\sum_{i=1}^{t} c_{k,i}^X * c_{k,i}^X}{\sqrt{\sum_{i=1}^{t} \left(c_{k,i}^X \right)^2} + \sqrt{\sum_{i=1}^{t} \left(c_{k,i}^Y \right)^2}}$$

2.2 Методы кластеризации

Некоторые методы кластеризации представляют отношения между последовательностями в виде графа. Вершинами графа являются сами последовательности, а в качестве весов ребер рассматривается расстояние или мера схожести между ними.

Для ограничения количества рассматриваемых ребер используется порог для отбрасывания связей с незначительным (или наоборот большим) значением веса.

После построения графа выбирается алгоритм кластеризующий вершины в нем.

Далее будут приведены некоторые алгоритмы и их описание.

Марковская кластеризация (MCL)

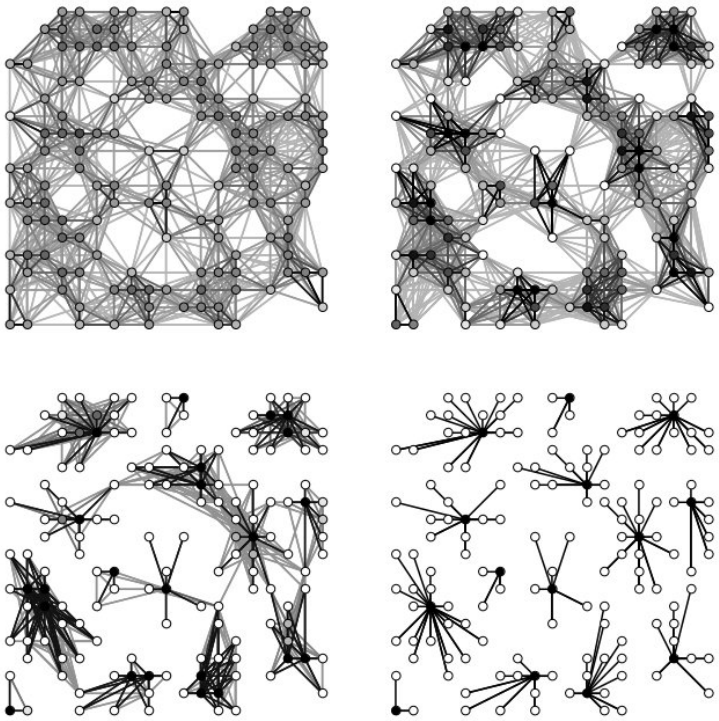

Рис. 5 Последовательные этапы моделирования случайных блужданий в процессе MCL [16].

Кластер, в данном случае, в графе характеризуется наличием большого количества связей между множеством, принадлежащих ему вершин. Причем

множество ребер между вершинами из разных кластеров должно быть значительно меньше.

Другими словами, случайные блуждания изредка будут переходить из одного кластера в другой, основываясь на оценках вероятностей переходов графа.

Алгоритм MCL находит кластерную структуру в графах путём итеративной математической процедуры. Этот процесс заключается в детерминированном вычислении вероятностей случайных блужданий в графе, применяя две алгебраических операции: распространение (expansion) и накачивание (inflation). Распространение соответствует стандартному умножению стохастических матриц[2] В основе накачивания лежит умножению Адамара.

Таким образом, алгоритм моделирует прохождение по случайному пути в графе. Распространение соответствует длинному пути в графе, а длинный путь с большой вероятностью проходит в пределах одного кластера. В свою очередь операция накачивания будет уменьшать маленькие вероятности, соответствующие межкластерным переходам, делая их практически невозможными.

В результате алгоритм будет изменяться до тех пор, пока не наступит равновесие. Состояние равновесия соответствует матрице, у которой нет путей между кластерами, а присутствуют только внутрикластерные связи.

Данный алгоритм часто используется для определения семейства белков [14] и нахождения ортологов[3] и паралогов[4] [15]

K-clique communities (Clique Percolation Method)

В работах [18], [19], [20] обнаружение кластеров в графе основано на идее выделении k-клика[5] плотных участков.

---

[2] Матрица называется стохастической, если каждый ее столбец в сумме дает 1

[3] Ортологами называют гены двух разных видов, произошедшие от одного предка. Такие гены обычно играют одну и ту же биологическую роль.

[4] Паралогами называют схожие последовательности внутри одного вида, получившиеся в результате удвоения гена

[5] K-клика - это полный граф на k вершинах или граф, в котором между каждыми двумя вершинами есть ребро.

K-клика кластеры представляют из себя максимально k-клика связных подграфов. Две клики считаются k-клика связными если они находятся в одном объединение, которое получается объединением k-клик имеющих k-1 общую вершину.

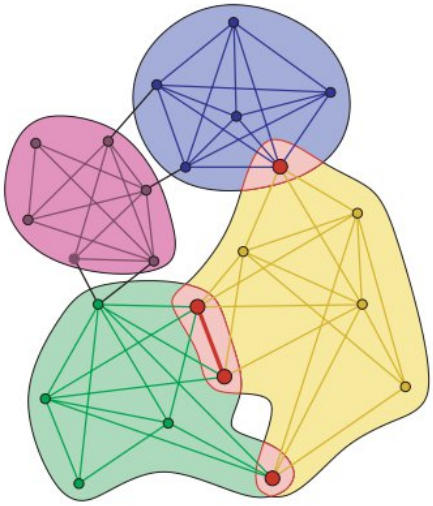

Рис. 6 Пример k-clique communities, для k = 4 [18]

В таком подходе кластеры могут пересекаться по вершинам.

Данный алгоритм используется для обнаружения протеиновых комплексов или определения функций и семейства белков [18].

K-Comm

В основе данного алгоритма лежит понятие k-community - связного графа, в котором каждая пара вершин имеет по крайней мере k общих соседей

Основная идея подхода описанного в [23] заключается в итеративном обнаружении k-communities. В начале работы алгоритма k принимает некоторое большое значение, после чего происходит поиск всех имеющихся k-communities и извлечение их из графа. Перед переходом на следующую итерацию алгоритма, значение k уменьшается на единицу и алгоритм продолжает работу с новым значением k-1.Условием окончания работы алгоритма является значение k = 0.

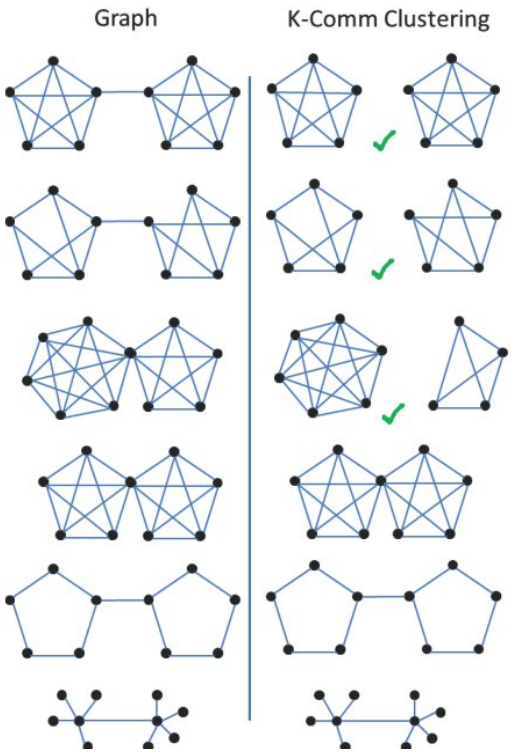

Рис. 6 Нахождение кластеров с помощью k-comm для различных графов.

3 Выбор наиболее эффективного метода

В данном разделе будет приведено описание методов улучшения чувствительности с последующим выбором наиболее эффективного метода.

3.1 Описание методов

Первоначально известно множество блоков $B = \{b_1, \dots, b_m\}$.

Каждый метод кластеризации синтенных блоков состоит из 3 частей:

- представление данных и сравнение блоков
- разбиение компонент связности на более мелкие части
- кластеризация

3.1.1 Представление данных и сравнение блоков

Перед тем как перейти к описанию алгоритмов кластеризации, необходимо выбрать способ сравнения блоков и метод их представления. Так как блоки являются множеством последовательностей, то к ним можно применить методы сравнения описанные в разделе 2.1.

По аналогии с последовательностями для каждого блока можно определить множество k-меров, встречаемых в нем, и представить блок в виде вектора. Однако, такой способ требует много ресурсов для хранения векторов длинной 4^k. Помимо этого, сравнение таких векторов так же требует большого количества вычислительных ресурсов.

Альтернативной формой представления блоков является граф. Вершинами такого графа являются блоки, а ребра определяют отношения между ними.

Для множества блоков $B = \{b_1, \dots, b_N\}$ определяется граф $G = (V, E, w)$. Множество вершин $V = \{0, \dots, N-1\}$, а весовая функция для ребер w определяется через расстояние (меру схожести) между блоками ρ:

$$w(i, j) = \rho(b_i, b_j)$$

Для ограничения количества ребер в графе, используется порог τ. Все ребра с весом $< \tau$ ($> \tau$) не рассматриваются.

В качестве ρ рассматриваются несколько возможных функций:

Использование выравнивания для сравнения блоков.

Так как каждый блок может состоять из нескольких последовательностей, то два блока можно выравнить только с помощью алгоритмов множественного выравнивания. Однако, нахождение оптимального множественного выравнивания для более чем двух последовательностей является NP-полной задачей [24].

Поэтому в качестве сравнения удобней использовать парное локальное выравнивание между последовательностями в блоках. Так как таких выравниваний между двумя блоками может быть много, то в качестве меры используется их агрегированное значение.

Для двух блоков $b_i = \{c_1^i, ..., c_{k_i}^i\}$ и $b_j = \{c_1^j, ..., c_{k_j}^j\}$ ρ определяется следующим образом:

1. $\rho = \max\limits_{n,k} score(c_n^i, c_k^j)$, где $score(c_n^i, c_n^j)$ - максимальный вес локальных выравниваний между c_n^i, и c_k^j

   ρ в данном случае может принимать значение от установленного порога τ - минимального значения выравнивания - и выше. Чем выше это значение, тем меньше различаются блоки между собой. Однако, данная мера имеет недостаток - она зависит от длины сравниваемых последовательностей внутри блоков. Чем больше их длины, тем выше значение веса, несмотря на то, что схожесть последовательностей может быть ниже.

2. $\rho = \max\limits_{n,k} \dfrac{\sum_l (c_{i,n}^* == c_{j,k}^*)}{\left|c_{i,n}^*\right|}$, где $\{c_{i,n}^*, c_{j,k}^*\}$ - максимальное по весу локальное выравнивание для c_n^i, и c_k^j

   ρ лежит в полуинтервале (0, 1]. Чем ближе значение к 1, тем более схожи блоки.

3.1.2 Разбиение компонент связности

В результате построения графа некоторые из компонент связности получаются намного больше остальных. Для упрощения работы алгоритмов кластеризации, необходимо разбить слишком большие компоненты связности.

Формирование объединений.

В основе данного метода лежит принцип, схожий с идеей алгоритма Крускала [26] и заключающийся в последовательном добавлении в граф тех ребер, которые удовлетворяют некоторому условию. В данном случае оно ограничивает компоненты связности от чрезмерного разрастания.

Так как все веса для ребер определяются мерой схожести, то, в отличии от алгоритма Крускала, ребра в граф будут добавляться в порядке уменьшения их веса.

Для каждой вершины в графе создаётся множество, которое изначально содержит лишь эту единственную вершину, меткой которой и будет идентифицироваться в дальнейшем.

В дальнейшем, при добавлении ребра $\{u, v\}$ проверяется условие:

$$|set(u)| + |set(v)| < \lambda \ (*)$$

где $set(u)$ - это множество с меткой u, $|set(u)|$ - количество элементов в этом множестве и λ - параметр, определяемый заранее.

Если условие выполняется, тогда множества объединяются:

$$set(u) = set(u) \cup set(v)$$

$$set(v) = set(u) \cup set(v)$$

3.1.3 Кластеризация

По причине того, что отношение гомологии имеет свойства транзитивности, алгоритмы кластеризации так же должны обладать этим свойством. Иными словами, получившиеся в результате работы этих алгоритмов кластеры не должны иметь общих вершин.

Также некоторые вершины могут не принадлежать ни одному кластеру.

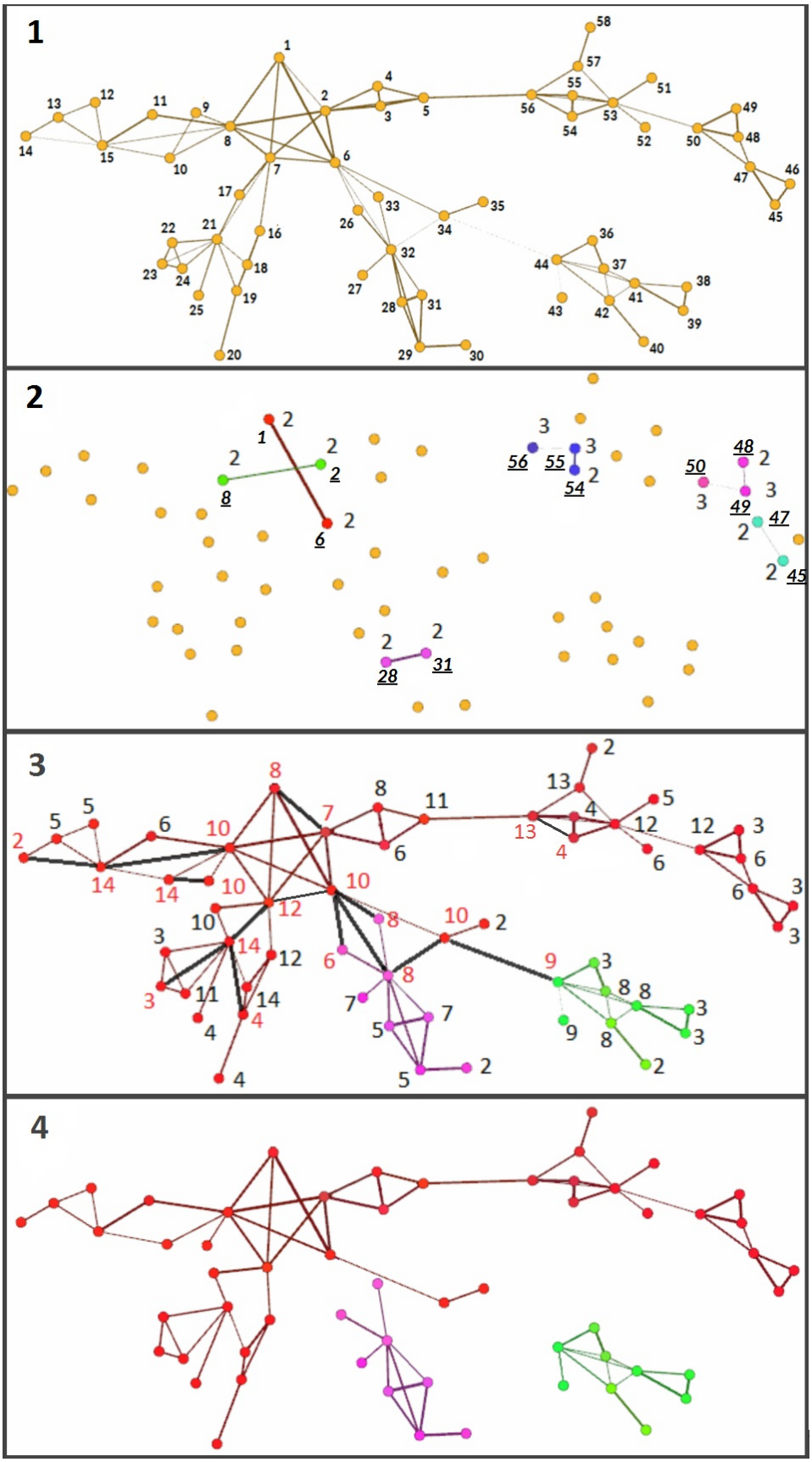

Рис. 7 Пример формирования объединений при $\lambda = 15$

1: Изначальный граф

На рисунках 2, 3 цифры с подчеркиванием обозначены метки множеств, а числами без подчеркивание - размер этих множеств.

2:Добавление ребер:
{1,6}; {28, 31}; {8, 2}; {55, 54}; {48, 49}; {47, 45}; {49, 50}; {55, 56}.
Пошаговое добавление ребер и изменение множеств:
{1,6} → $set(1) = set(6) = \{1, 6\}$
{28, 31}→ $set(28) = set(31) = \{28, 31\}$
{8, 2}→ $set(8) = set(2) = \{8, 2\}$
{55, 54}→ $set(55) = set(54) = \{55, 54\}$
{48, 49}→ $set(48) = set(49) = \{49, 48\}$
{47, 45}→ $set(47) = set(45) = \{47, 45\}$
{49, 50}→ $set(49) = set(50) = \{49, 48, 50\}$
{55, 56}→ $set(56) = set(55) = \{56, 55, 54\}$

3: Конечный шаг, после добавления всех ребер. Черным указаны ребра, не удовлетворяющие условию (*), поэтому они исключаются из графа.

4: Результирующий граф

Взвешенные полные подграфы

Данный метод является итерационным. Изначально для каждой компоненты связности находится множество всех возможных полных её подграфов максимального размера. Для каждого такого подграфа считается вес - сумма весов всех входящих в него ребер. Из данного множества выбирается подграф максимального веса и извлекается из графа со всеми связными с ним ребрами. Этот исключенный из рассмотрения подграф будем считать очередным найденным кластером. Процесс повторяется, пока в графе не остается ребер, не принадлежащий ни одному кластеру.

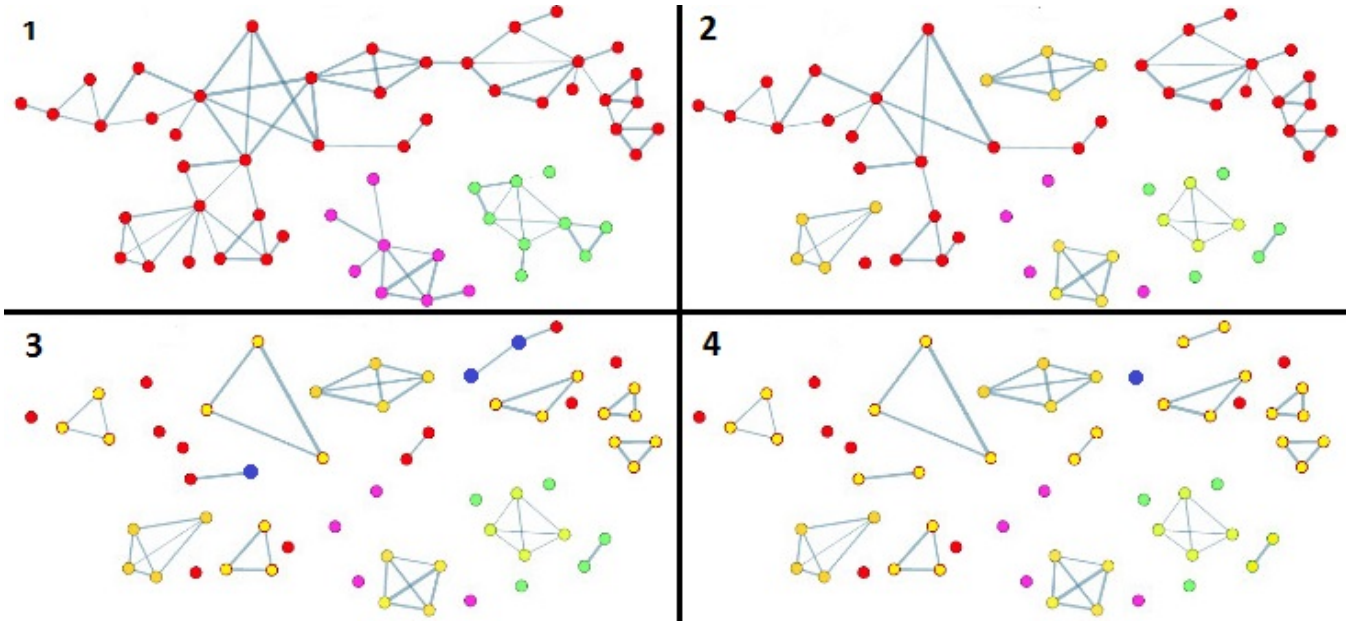

Рис. 8 Пример работы алгоритма - взвешенные полные подграфы. Желтым цветом указаны кластеры.(1)Изначальный граф. (2) Граф после исключения клик размера 4. (3) Граф после исключения клик размера 3. (4) Результирующий граф.

Clique Percolation Method

Основная идея данного алгоритма была описана в разделе 2.2.

В данном методе предполагается, что k-clique community могут пересекаться по вершинам. Для исключения таких случаев в модифицированном алгоритме при пересечении k-clique communities в качестве кластера выбирается то k-clique community, которое имеет больший вес. Вес community, так же как и в предыдущем алгоритме, определяется как сумма весов всех ребер внутри этого community.

Так же как и в алгоритме k-comm, сначала происходит поиск k-clique community с максимальным значением k, для которого оно не пустое, затем с каждой итерацией происходит уменьшение k на единицу. Конечное значение k = 3.

Для k = 2 алгоритм работает как предыдущий.

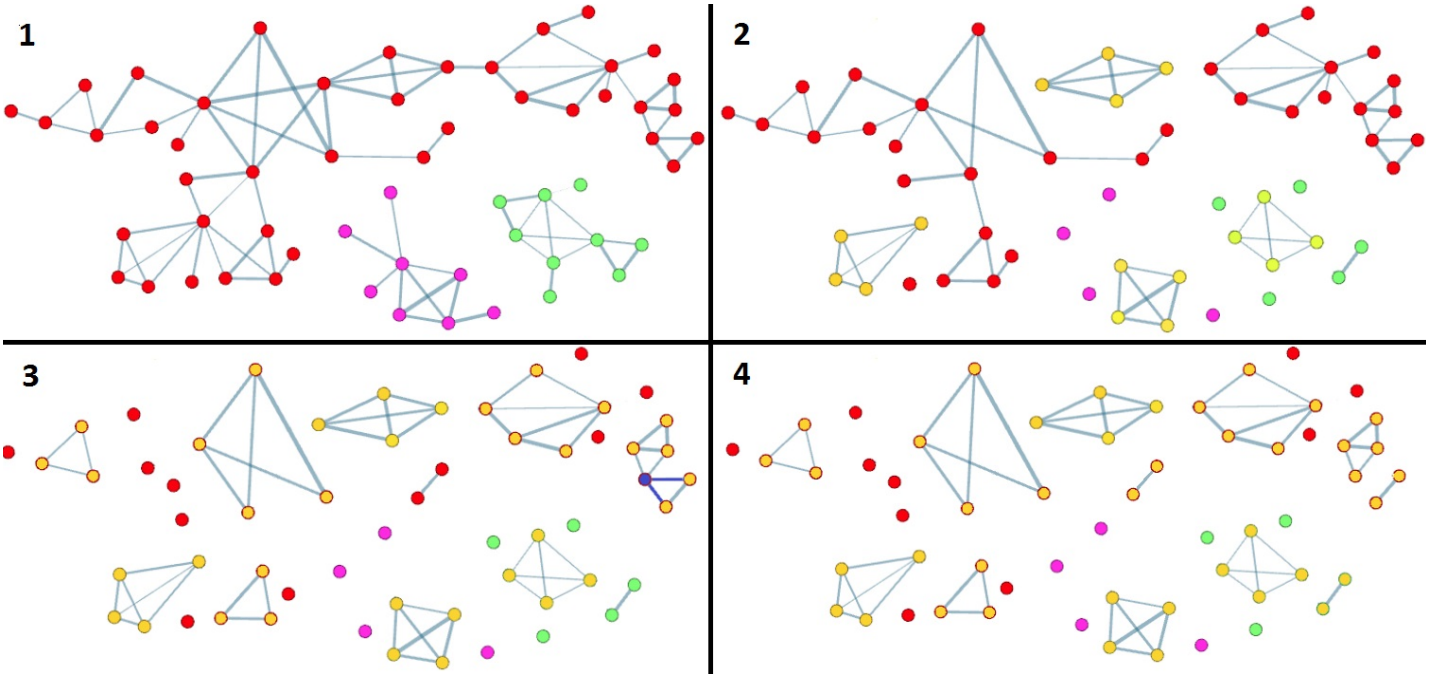

Рис. 9 Пример работы модифицированного алгоритма Clique Percolation Method. Желтым цветом указываются кластеры. (1) Изначальный граф. (2) Поиск и извлечение 4-clique-communities. (3) Поиск и извлечение 3-clique communities. Синим цветом обозначено пересечение 3-clique communities. (4) Поиск и извлечение 3-clique communities

Марковская кластеризация (MCL)

Краткое описание данного алгоритма было приведено в разделе 2.2.

Перед кластеризацией выбираются 2 параметра r ($r > 1$) и e необходимые для операций накачивания и распространения соответственно.

При увеличении параметра r, увеличивается плотность кластеров. Автор алгоритма рекомендует рассмотреть следующие начальные значения r : 1.4, 2, 4 и 6. [25]

Значения параметра e рассматриваются небольшие (< 3), так как с его увеличением, увеличивается и размер кластеров, что ведет к уменьшению точности метода.

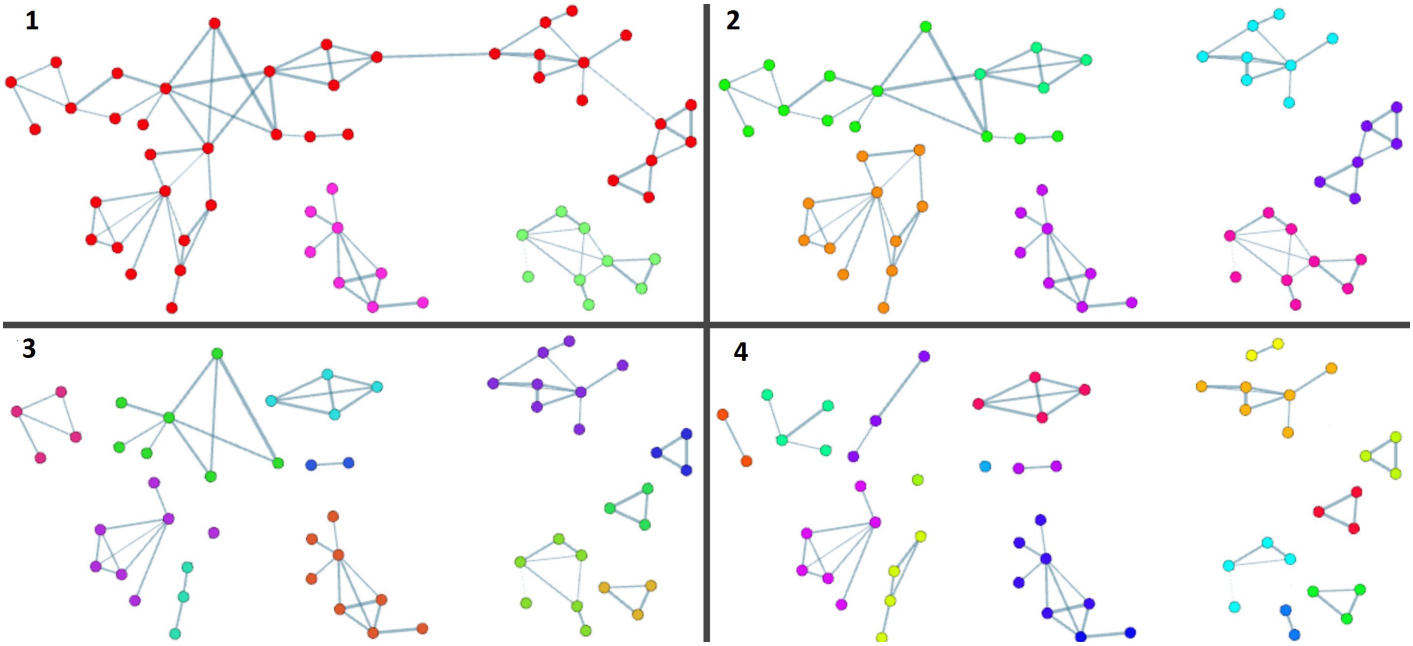

Рис.10 Пример работы алгоритма MCL. (1) Изначальный граф. (2) Результат работы алгоритма с параметрами $r = 1.4$ $e = 2$. (3) Результат работы алгоритма с параметрами $r = 2$ $e = 2$. (4) Результат работы алгоритма с параметрами $r = 4$ $e = 2$.

3.2 Эксперименты

Для экспериментов были использованы две метрики описанные в пункте "Представление данных и сравнение блоков" текущей главы. Параметр τ - нижняя граница для веса выравниваний последовательностей в блоке. Параметр λ - ограничивает формирование объединений в уравнении (*).

Рассмотренные алгоритмы были реализованы на языке программирования Python; для выявления синтенных блоков использовалась система Sibelia, а для локального выравнивания был задействован инструмент lastz. В качестве входных данных рассматривались четыре ДНК последовательности, каждая длиной приблизительно в 1.5 миллиона символов.

Таблица 1 Результат работы Sibelia без кластеризации

sensitivity	precision	F-measure
0,693	0,874	0,773

3.2.1 Выбор оптимальных параметров для каждого метода

Таблица 2 Результаты работы алгоритма - Взвешенные полные подграфы

metric	τ	λ	sensitivity	precision	F-measure
1	2400	15	0,767	0,569	0,653
		25	0,770	0,537	0,633
		50	0,770	0,537	0,633
2	2400	15	0,731	0,835	0,780
		25	0,741	0,818	0,777
		50	0,750	0,669	0,707
	2000	15	0,771	0,843	0,805
		25	0,784	0,827	0,805
		50	**0,793**	**0,800**	**0,796**

В методе использовались низкие значения τ (2000 и 2400), так как при увеличении количества ребер в графе, увеличивается вероятность его превращения в полный граф. Если в подграфе не хватает всего одного ребра для образования кластера, то алгоритм разобьет его на более мелкие части, что приведёт к уменьшению чувствительности.

Параметр λ выбирался в диапазоне от 15 до 50, так как при меньших значениях во время разбиения компонент связности исключается много ребер, что так же уменьшает чувствительность. При $\lambda > 50$ резко начинает падать точность: в один кластер могут попадать блоки, отношения выравнивания которых почти не содержит гомологичных пар.

Для данного метода наилучший результат, удовлетворяющий условиям 1-2 из пункта 1.2 "Формализованная постановка задачи", достигается при использовании 2-й меры схожести и значениях $\tau = 2000$, $\lambda = 50$

Для Clique Percolation Method значения τ были увеличены, поскольку алгоритм обладает противоположным свойством: при появлении лишних ребер кластер увеличивается в размерах, добавляя к себе несхожие вершины, что влияет на точность метода. Верхняя граница параметра λ была уменьшена по аналогичной причине.

Так же как и в предыдущем методе, лучшее результаты достигаются при использовании 2-ой меры схожести и значениях $\tau = 2400$, $\lambda = 25$.

Таблица 3 Результаты работы алгоритма - Clique Percolation Method

metric	τ	λ	sensitivity	precision	F-measure
1	3000	15	0,809	0,535	0,644
		20	0,814	0,503	0,622
		25	0,816	0,509	0,627
2	2400	15	0,770	0,828	0,798
		20	0,777	0,836	0,805
		25	**0,783**	**0,784**	**0,783**
	2600	15	0,769	0,838	0,802
		20	0,775	0,835	0,804
		25	0,781	0,796	0,788
	3000	15	0,765	0,837	0,799
		20	0,772	0,812	0,791
		25	0,778	0,788	0,783

Таблица 4 Результаты работы алгоритма - MCL

metric	τ	λ	r	sensitivity	precision	F-measure
1	3000	15	2	0,853	0,563	0,678
			4	0,789	0,676	0,728
			6	0,817	0,706	0,758
2	2400	15	2	0,852	0,701	0,769
			4	0,809	0,761	0,785
			6	0,809	0,779	0,794
		25	2	0,862	0,670	0,754
			4	0,823	0,732	0,775
			6	0,820	0,773	0,796
	3000	15	2	0,834	0,784	0,808
			4	**0,811**	**0,805**	**0,808**
			6	0,804	0,808	0,806
		25	2	0,849	0,732	0,786
			4	0,821	0,770	0,795
			6	0,816	0,774	0,794

Для алгоритма MCL значение параметра r выбирались по рекомендации автора [25]. Параметр e, описанный в пункте 3.1.3, рассматривается во всех случаях равным 2 и поэтому опускается в таблице.

Лучший результат получается при использовании 2-ой метрики $\tau = 3000$, $\lambda = 15$ и $r = 4$.

3.2.2 Результаты

Таблица 5 Наилучшие результаты всех методов

	sensitivity	**precision**	**F-measure**
MCL	0,811	0,805	0,808
Максимально взвешенные клики	0,793	0,800	0,796.1
Clique Percolation Method	0,783	0,784	0,783

В результате проделанных экспериментов наиболее эффективным методом получился последний рассмотренный алгоритм MCL при значениях параметров $\tau = 3000$, $\lambda = 15$ и $r = 4$ и использовании 2-ой метрики.

4 Сравнение выбранного метода с существующими инструментами

В данной главе будет приведена сравнительная таблица метода описанного в работе с инструментами Mauve [7] и Mugsy [2].

Параметрами для сравнения являются чувствительность и точность, описанные в разделе 1.1.

Таблица 6 Сравнение метода описанного в работе с существующими инструментами

tools	sensitivity	precision
mugsy	0,409	0,935
mauve	0,304	0,532
sibelia + new_method	0,811	0,805

Как можно увидеть, чувствительность у рассмотренных инструментов ниже, чем при использование метода описанного в данной работе.

Плохие результаты инструмента mauve могут быть обусловлены тем, что данный инструмент неприспособлен на работу с последовательностями, в которых есть повторы.

Заключение

В результате выполнения данной работы были решены все поставленные задачи: изучена необходимая предметная область (синтенные блоки, отношения гомологии и выравнивания); исследованы существующие методы кластеризации и сравнения геномных последовательностей, которые в последствии были адаптированы для улучшения чувствительности инструмента Sibelia и реализованы на языке Python. Экспериментальным путем был определен наиболее эффективный метод и проведено сравнение данного метода с существующими инструментами поиска синтенных блоков.

В дальнейшем планируется рассмотреть работу данного алгоритма на других наборах входных данных и использовать его для улучшения результатов работы других инструментов**;** исследовать альтернативные методы кластеризации и представления геномных данных.

Список Литературы

[1] Minkin I., Patel A., Kolmogorov M., Vyahhi N., Pham S. Sibelia: A scalable and comprehensive synteny block generation tool for closely related microbial genomes. Proceedings of Algorithms in Bioinformatics Springer (2013), 215-229.

[2] Angiuoli S., Salzberg S. Mugsy: fast multiple alignment of closely related whole genomes. Bioinformatics (2011) 27(3), 334–342.

[3] Initial sequence of the chimpanzee genome and comparison with the human genome. Nature (2005) 437(7055), 69-87.

[4] Cristina G., Ghiurcuta and Bernard M., Moret E. Evaluating synteny for improved comparative studies. Bioinformatics (2014) 30(12), 9-18.

[5] Тарантул В.З. Все познается в сравнении (сравнительная геномика). Геном человека (Энциклопедия, написанная четырьмя буквами), 274-276.

[6] Pham S., Pevzner P. DRIMM-Synteny: decomposing genomes intoevolutionary conserved segments. Bioinformatics (2010) 26, 2509–2516.

[7] Darling A., Mau B., Blattner F., Perna N.: Mauve: multiple alignment of conserved genomic sequence with rearrangements. G.R. (2004) 14(7), 1394–1403.

[8] Finishing the euchromatic sequence of the human genome. Nature (2004) 431(7011), 931-945.

[9] История геномики (Часть 1: геномные проекты) http://scinquisitor.livejournal.com/8981.html

[10] https://en.wikipedia.org/wiki/DNA_sequencing

[11] Vingron M., Waterman MS. Sequence alignment and penalty choice: review of concepts, case studies and implications. J. Mol. Biol. (1994) 235, 1–12.

[12] Vinga S., Almeida J.: Alignment-free sequence comparison - a review. Bioinformatics (2003) 19(4), 513–523.

[13] Remm M., Storm C., Sonnhammer E. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. J Mol Biol (2001) 314, 1041–1052.

[14] Enright A., van Dongen S., Ouzounis C. An Efficient Algorithms for Large Scale Protein Families, Nucleic Acids Research (2002) 30, 1575-1584.

[15] Li L., Stoeckert C., Roos D. OrthoMCL: identification of ortholog groups for eukaryotic genomes. Genome Res (2003) 13: 2178–2189

[16] Van Dongen S. . Graph clustering by flow simulation. Ph.D thesis, University of Utrecht, The Netherlands (2000).

[17] Dowell K. Molecular Phylogenetics: An introduction to computational methods and tools for analyzing evolutionary relationships. Math (2008) 500.

[18] Palla G., Dere´nyi I., Farkas I. Vicsek T. Uncovering the overlapping community structure of complex networks in nature and society. Nature (2005) 435, 814–818.

[19] Dere´nyi I., Palla G., Vicsek T. Clique percolation in random networks. Phys. Rev. Lett. (2005) 94.

[20] Palla G., Baraba´si A., Vicsek, T. Quantifying social group evolution. Nature (2007) 446, 664–667.

[21] Eddy R. Where did the BLOSUM62 alignment score matrix come from. Nat. Biotechnol. (2004) 22, 1035–1036.

[22] Dayhoff O., Schwartz R., Orcutt C. A model of Evolutionary Change in Proteins. Atlas of protein sequence and structure. Nat. Biomed. Res. Found (1978) 5(3), 345–358.

[23] Verma A., Butenko. S. Network clustering via clique relaxations: A community based approach. Contemporary Mathematics (2013) 588.

[24] Wang L., Jiang T. On the complexity of multiple sequence alignment. J. Comput. Biol. (1994) 1, 337–348.

[25] https://micans.org/mcl/

[26] https://en.wikipedia.org/wiki/Kruskal%27s_algorithm#Algorithm

[27] https://ru.wikipedia.org/wiki/Хромосомные_перестройки