Подход к преобразованиям матриц расстояний ДНК с целью улучшения исходных алгоритмов расчёта расстояний

В.А. Дудников[1], Б.Ф. Мельников[1,2]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086
[2]Университет МГУ-ППИ в Шэньчжэне, Гоцзидасюеюань 1, Шэньчжэнь, КНР, 517182

Аннотация. Рассмотрена проблема современных алгоритмов расчёта расстояния между последовательностями ДНК, способы их улучшения с помощью алгоритмов оптимизации. Сформулирована оптимизационная задача, предназначенная для улучшения алгоритмов расчёта расстояния. Предложено решение этой задачи с помощью одного из вариантов генетического алгоритма, описаны все этапы построения генетического алгоритма для поставленной задачи. Приведены результаты улучшения.

1. Введение

На практике часто бывает необходимо рассчитать расстояния между последовательностями различной природы. Подобные алгоритмы часто применяются в биоинформатике; они представляют собой отдельный и очень важный вид проблемы, т. е. нахождение расстояния между генетическими последовательностями. Основная трудность, возникающая при расчёте расстояния между генетическими последовательностями, – очень большая длина такой последовательности. Например, даже для очень короткой митохондриальной ДНК (мДНК) человека длина последовательности превышает 16 000 символов, а для обычной ДНК может превышать 3 000 000 000 символов, см. [1].

Таким образом, точные алгоритмы сравнения на практике не могут использоваться. Для прикладных задач сравнения генетических последовательностей используют, например, алгоритм Нидлмана-Вунша – алгоритм выравнивания двух аминокислотных или нуклеотидных последовательностей. Однако понятно, что использования неточных алгоритмов связано с потерей качества результата, так как они как раз разработаны так, чтобы уменьшить точность, но увеличить скорость работы.

Но возникает вопрос. Как можно улучшить качество таких алгоритмов? Этот вопрос связан со следующими соображениями, которые мы уже опубликовали в наших предыдущих работах (мы не нашли аналогов в литературе) [1, 2, 3, 4, 5].

Изложим наши основные размышления по поводу задачи, которые уже были опубликованы, например, в [5]. Рассмотрим, например, три следующих вида: человека (H), шимпанзе (C) и бонобо (B). Согласно биологам и различным оценкам, C и B разошлись (имели общего предка) примерно от 2 до 2,5 миллионов лет назад, а H разошлись с обоими другими видами около 5,5-7 миллионов лет назад. Поэтому возникает следующий вопрос:

почему H будет ближе к B, сравнивая с C? Или наоборот: почему он должен быть ближе к C, сравнивая с B? Очевидно, что ответ на оба этих вопроса отрицательный, т. е., иначе говоря, объяснения большей близости нет. Поэтому в матрице расстояний между геномами все полученные треугольники должны быть близкие к равнобедренным. Для этого мы вводим оценку плохого значения для каждого треугольника (badness) особым образом, мы формируем матрицы расстояний по одному выбранному алгоритму и рассматриваем badness всех треугольников как сумму.

Рассмотрим матрицу расстояний по некоторым алгоритмам для нахождения расстояний между геномами. Для большей наглядности рассмотрим конкретный пример. Допустим, мы рассматриваем три отряда млекопитающих, и выберем по одному виду из каждого отряда. Теперь, допустим, мы рассчитали с помощью алгоритма Нидлмана-Вунша для каждой пары из тройки расстояния между их ДНК последовательностями, то есть получили матрицу расстояний:

$$\begin{pmatrix} 0 & a_{12} & a_{13} \\ a_{21} & 0 & a_{23} \\ a_{31} & a_{32} & 0 \end{pmatrix},$$

где a_{ij} – расстояние между i-м и j-м отрядом. Любые три элемента матрицы i, j, k образуют треугольник со сторонами a_{ij}, a_{ik}, a_{jk}, для $i \neq j \neq k$. В данном случая можно получить только один треугольник. Гипотеза исследования заключается в том, что такой треугольник должен быть равнобедренным остроугольным. Таким образом в соответствии с гипотезой, чтобы получить «правильную» матрицу, нужно изменять значение a_{ij} так, чтобы треугольник со сторонами a_{ij}, a_{ik}, a_{jk} стал равнобедренным остроугольным.

Исходя из вышесказанного, мы вводим критерий [5], который будет оцениваться в виде углового равнобедренного треугольника, то есть:

$$\frac{\alpha - \beta}{\alpha} \tag{1}$$

где α – наибольший угол, β следующий по величине угол.

Общий badness матрицы считается суммой всех badness для каждого треугольника. И мы считаем, что нужно уменьшить эту сумму, то есть общий badness (далее просто badness). Чтобы уменьшить значение badness, нам нужно изменять элементы матрицы. Т.е. нужно изменять расстояния в матрице расстояний так, чтобы получившиеся треугольники были близки к равнобедренным остроугольным. Понятно, что конечная матрица должна быть «похожа» на исходную по какому-то критерию, иначе может случится, что конечная матрица будет иметь хороший badness, но не иметь никакого отношения к исходным расстояниям в матрице. Чтобы избежать этого, дальше мы воодим критерий останова, который указывает насколько конечная матрица должна быть «похожа» на исходную.

Рассмотрим для начала полностью функцию оценивания треугольника со сторонами a, b, c насколько сильно треугольник похож на равнобедренный остроугольный. Для этого воспользуемся формулой (1), но в ней используются углы, поэтому перепишем эту формулу так, чтобы её значение зависело от сторон треугольника. Для вычисления угла в треугольнике между любыми сторонами воспользуемся формулой:

$$\phi(a, b, c) = \arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right).$$

Если $a > b > c$, то $\phi(c, b, a)$ – наибольший угол, а $\phi(c, a, b)$ – следующий за ним по величине угол. Если $M = \max(a, b, c)$, $I = \min(a, b, c)$ и $D = (a + b + c) - (M + I)$, то $M > D > I$. Тогда итоговая формула имеет вид:

$$B(a, b, c) = \frac{\phi(I, D, M) - \phi(I, M, D)}{\phi(I, D, M)}. \tag{2}$$

Причем

$$\max(a, b, c) = \max(a, \max(b, c)),$$

а

$$\max(a, b) = a \cdot \theta(a - b) + b \cdot \theta(b - a),$$

где θ – функция Хевисайда. Функцию min можно определить через max так:

$$\min(a, b) = -\max(-a, -b),$$

и, соответственно,

$$\min(a, b, c) = \min(a, \min(b, c)).$$

В работе рассматривается матрица расстояний между 28 отрядами млекопитающих, которая получена с помощью алгоритма Нидлмана-Вунша. Все дальнейшие размышления будут относиться именно к этому алгоритму. Однако как и было сказано во введении, необязательно ограничиваться именно этим алгоритмом, можно рассматривать матрицы расстояний, полученные и другими алгоритмами поиска расстояния между ДНК последовательностями. Сама матрица расстояний приведена в приложении А. Значение матрицы равно 0, если геномы двух отрядов совпадают.

2. Постановка задачи

Дана матрица $A \in R^{n \times n}$, где $R^{n \times n}$ – множество всех симметричных матриц размера $n \times n$ (в нашем случае $n = 28$).

Найти матрицу $A \in R^{n \times n}$ такую, что

$$d(A, A^*) \leq b \tag{3}$$

и $f(A^*) = \min_{A \in R^{n \times n}} f(A)$. Здесь

$$d(A, B) = \sqrt{\sum_{i=1}^{n} \sum_{j=i+1}^{n} (A_{ij} - B_{ij})^2}, \tag{4}$$

b – максимально допустимое расстояние между матрицами (задается в качестве параметра), функция $f(A)$ определяется так:

$$f(A) = \sum_{(i,j,k) \in C} B(A_{ij}, A_{jk}, A_{ik}), \tag{5}$$

где C – множество всех сочетаний из $\binom{n}{3}$. То есть $\binom{n}{3}$ – максимальное количество треугольников, которое можно получить из матрицы размера $n \times n$.

3. Решение задачи

Для решении задачи мы решили использовать классическую схему генетического алгоритма [6, 7], дадим её краткое описание.

Классическая схема генетического алгоритма.

1. Создать начальную популяцию.
2. Применить операторы скрещивания и мутации (порядок выполнения операторов вариативный).
3. Применить оператор селекции.
4. Сформировать новое поколение.

5. Если результат достигнут, то закончить выполнение, если нет, то перейти к шагу 2.

Видим, что в схеме есть различные этапы, во всех этапах можно применять разные методы для достижения желаемого результата. Есть большой выбор операторов скрещивания и мутации, их выбирают исходя из структуры данных особей, и от ограничений, которые на них накладываются [7]. Также существуют разные операторы селекции, которые имеют свои преимущества и недостатки, и выбираются исходя из поведения целевой функции [7].

Для применения генетических алгоритмов к задаче нужно выполнение нескольких условий. Во-первых, решения задачи должны быть представимы в виде векторов. Во-вторых, чтобы существовали операторы скрещивания и мутации для естественных ограничений на структуру векторов (когда решением задачи, например, является подстановка, то нельзя допускать повторение элементов).

Представим решения задачи в виде векторов $\langle x_1, x_2, ..., x_{n(n-1)/2} \rangle, x_i \in \mathbb{R}$, которые являются множителями исходной матрицы. То есть каждый элемент вектора умножается на соответствующий элемент верхнего и нижнего треугольника исходной матрицы. Обозначим такое умножение вектора x и матрицы A как $x \circ A$. Тогда нужно найти такой вектор x, что $d(A, x \circ A) \leq b$ и $f(x \circ A) = \min_{A \in R^{n \times n}} f(A)$. Сразу введём ограничение на все элементы вектора $\frac{2}{3} \leq x_i \leq \frac{3}{2}$. Такое ограничение необходимо, чтобы обеспечить не слишком сильное изменение исходной матрицы. Таким образом, для выполнения второго условия необходимо найти операторы, которые бы не нарушали это ограничение. К счастью, заданное ограничение не является необычными, и существуют операторы, которые не нарушают его. Далее будут рассмотрены все этапы генетического алгоритма с описанием конкретных операторов и методов, которые мы применяли.

3.1. Целевая функция

Первым делом нужно определить целевую функцию, удовлетворяющую заданным в задаче условиям. Так как необходимо найти решение, которое бы улучшило badness матрицы и удовлетворяло неравенству (3), целевая функция является суммой двух функций оценивания:

$$g(x) = \frac{f(x \circ A)}{f(A)} + k \left| 1 - \frac{d(A, x \circ A)}{b} \right|, \tag{6}$$

где A – исходная матрица, k – коэффициент масштабирования. Значение k выбиралось нами специально таким, чтобы увеличить влияние второго члена суммы, для того, чтобы генетический алгоритм быстрее сходился к нужному значению b. То есть, коэффициент b должен увеличивать второй член суммы настолько, чтобы он имел влияние больше, чем первый. В нашей работе мы взяли $k = 5$.

3.2. Начальная популяция

Задание начальной популяции во множестве задач является важным этапом, так как он определяет «генетический материал», с которым будет производиться дальнейшая работа; однако существует не много способов, которые применяют на практике. Наверное, самый популярный способ это взять векторы случайным образом из множества допустимых значений. Существует, конечно, и множество других способов, но их рассмотрение не является предметом работы.

В рассматриваемой задаче с таким способом возникла проблема: получаемые элементы слишком сильно изменяли исходную матрицу, и расстояние между такой матрицей и исходной получалось чересчур большим, из-за чего алгоритму нужно было много времени для достижения нужных результатов. Такое поведение возникало тогда, когда элементы

векторов выбирались в большом промежутке (в пределах ограничений), поэтому каждый элемент вектора выбирался алгоритмом в небольшом промежутке $[0,99;1,01]$.

3.3. Оператор скрещивания

Существует большое количество различных операторов скрещивания для вещественных векторов, но часто они не учитывают естественных ограничений на элементы вектора (как в нашем случае элементы должны входить в промежуток), для решения такой проблемы вводят штрафные функции. Чтобы не усложнять алгоритм было решено воспользоваться классическим одноточечным кроссинговером, который может быть с легкостью применён к рассматриваемой задаче без каких-либо изменений.

3.4. Оператор мутации

В качестве оператора мутации, который учитывает наши ограничения можно было взять, например, обмен произвольной пары элементов. Однако поскольку подобным образом работает выбранный оператор скрещивания (обменивает некоторые элементы двух векторов), возникли бы проблемы с «генетическим материалом». Новые элементы не появлялись бы, поэтому такая популяция могла бы преждевременно сойтись. В качестве оператора мутации был выбран следующий. Сгенерируем датчиком случайных чисел с равномерным распределением число $r \in [-0,01;0,01]$, тогда мутацией числа a будет число $b = a + r$, если $\frac{2}{3} \leq a + r \leq \frac{3}{2}$, но если $a + r > \frac{3}{2}$, то $b = \frac{3}{2}$, а если $a + r < \frac{2}{3}$, то $b = \frac{2}{3}$. Промежуток $[-0,01;0,01]$ был выбран опытным путём (как и ранее для создания начальной популяции).

3.5. Оператор селекции и формирования новой популяции

В качестве оператора селекции выбран оператор элитарной селекции. Смысл оператора заключается в том, что после того как к популяции были применены операторы скрещивания и мутации формируется промежуточная популяция, состоящая из родителей и их потомков, далее отбираются N (размер исходной популяции) лучших потомков, которые и составляют новую популяцию.

3.6. Критерий останова

Критериев останова может быть два: остановка после фиксированного количества итераций или остановка после достижения нужного значения целевой функции. Поскольку нам неизвестно какой результат мы хотим получить был выбран первый вариант.

3.7. Параметры генетического алгоритма

Параметры генетического алгоритма:

- размер популяции – 100;
- количество итераций алгоритма – $50000b + 2500$, где b – ограничение на расстояние между матрицами по метрике Евклида;
- вероятность скрещивания – $0,9$;
- вероятность мутации – $0,15$.

Видно, что вероятность мутации выбрана достаточно нетипичной (большой). Такой выбор связан с тем, чтобы постоянно поддерживать в популяции многообразие, а оператор элитарной селекции, в свою очередь, сохраняет лучших представителей популяции (в случае, если оператор мутации будет порождать «плохие» решения).

4. Вычислительные эксперименты

В таблице 1 располагаются результаты работы генетического алгоритма с применением описанных выше параметров улучшения badness (общей суммы по всем треугольникам). Таблица 1 показывает насколько улучшился badness в зависимости от параметра b, который, напомним, означает максимальное допустимое расстояние от исходной матрицы по метрике Евклида. Невооружённым глазом видно, что есть обратная корреляция между параметром b и $f(A \cdot x)$, а линейный коэффициент Пирсона равен -0.97, что говорит о сильной, почти линейной корреляции. Сильная обратная корреляция говорит о том, что увеличение параметра b будет улучшать badness линейно, однако это повлечёт то, что конечная матрица будет сильнее отличаться от исходной.

Таблица 1. Таблица улучшения badness

b	$f(A \circ x)$
0.05	214.4
0.10	168.9
0.15	135.9
0.2	98.9
0.25	93.8
0.30	74.0
0.35	57.8

5. Заключение

Полученные результаты мы предполагаем применять в следующих задачах – не связанных алгоритмически с рассматриваемой в настоящей статье. (Мы пишем «задачи», а не «задача», поскольку, в отличие от рассматриваемой здесь, эти задачи совершенно различны для каждого из алгоритмов подсчёта расстояний.) В каждом из алгоритмов подсчёта расстояний между цепочками ДНК (именно на основе этих алгоритмов и формируются матрицы) имеются некоторые параметры (например, в алгоритме Нидлмана-Вунша – набор штрафов), которые разработчик программ подсчёта расстояний может корректировать. *Выходные* данные проведённых нами вычислительных экспериментов дают *входные* данные для будущей работы, связанной с алгоритмизацией изменённых версий алгоритмов расчёта расстояний. Эти изменённые версии должны дать результаты (матрицы расстояний), более похожие на матрицы выходных данных алгоритма, рассматриваемого в настоящей статье (т.е. более близкие к ним по какой-либо естественной метрике). Специально отметим ещё раз, что аналогичные вспомогательные алгоритмы (и соответствующие задачи) возможны для любого исходного алгоритма расчёта расстояний.

6. Литература

[1] Мельников, Б.Ф. Подход к улучшению алгоритмов расчёта расстояний между цепочками ДНК (на примере алгоритма Нидлмана-Вунша) / Б.Ф. Мельников, М.А. Тренина, А.С. Кочергин // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. – 2018. – № 1. – С. 46-49. DOI: 10.21685/2072-3040-2018-1-4.
[2] Мельников, Б.Ф. Оценка алгоритмов расчёта расстояния строк ДНК / Б.Ф. Мельников, С.В. Пивнева, М.А. Трифонов // Известия высших учебных заведений. Поволжский регион. Физико-математические науки. – 2015. – № 2. – С. 89-96.
[3] Мельников, Б.Ф. Об одной задаче восстановления матриц расстояний между цепочками ДНК / Б.Ф. Мельников, М.А. Тренина // Inteinational Jouinal of Open Infoimation Technologies. – 2018. – № 6. – С. 1-13.
[4] Мельников, Б.Ф. Применение метода ветвей и границ в задаче восстановления матрицы расстояний между цепочками ДНК / Б.Ф. Мельников, М.А. Тренина // Inteinational Jouinal of Open Infoimation Technologies. – 2018. – № 8. – С. 1-13.

[5] Мельников, Б.Ф. Мультиэвристический подход к сравнению качества определяемых метрик на множестве последовательностей ДНК / Б.Ф. Мельников, С.В. Пивнева, М.А. Трифонов // Современные информационные технологии и ИТ-образование. – 2017. – № 2. – С. 89-96.
[6] Holland, J.N. Adaptation in Natural and Artificial Systems // Ann Arrbor, Michigan: Univ. Michigan Press, 1975.
[7] Luke, S. Essentials of Metaheuristics – Lulu, 2013.

An approach to transforming DNA distance matrices in order to improve the original distance calculation algorithms

V.A. Dudnikov[1], B.F. Melnikov[1,2]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086
[2]Shenzhen MSU-BIT University, International University Park Road 1, Shenzhen, PRC, 518172

Abstract. The research addressed the problem of the modern distance calculation algorithms between DNA sequences and ways of their improvements through the algorithm enhancements. The problem of improving distance calculation algorithms has been introduced. A solution to this problem had been proposed using one of the variants of the genetic algorithm. All stages of genetic algorithm development have been described. The results of the improvement have been given.

Приложение А

Таблица 2. Матрица расстояний для 28 видов из разных отрядов млекопитающих, полученная алгоритмом Нидлмана-Вунша (значения округлённые до четырёх знаков после запятой)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.0000	0.2992	0.2585	0.2695	0.3149	0.3241	0.2850	0.2947	0.5034	0.3273	0.2682	0.2707	0.3023	0.2934
1	0.2992	0.0000	0.3691	0.2979	0.2399	0.2089	0.3014	0.2220	0.5050	0.3267	0.3034	0.3016	0.3213	0.3074
2	0.2585	0.3691	0.0000	0.2921	0.3432	0.3390	0.3580	0.3717	0.5837	0.3976	0.3581	0.2715	0.3762	0.3734
3	0.2695	0.2979	0.2921	0.0000	0.2936	0.3482	0.2924	0.2984	0.5002	0.3062	0.2731	0.2793	0.2835	0.2785
4	0.3149	0.2399	0.3432	0.2936	0.0000	0.2168	0.3177	0.2359	0.5064	0.3161	0.3181	0.3097	0.3147	0.3122
5	0.3241	0.2089	0.3390	0.3482	0.2168	0.0000	0.3411	0.1976	0.5104	0.3584	0.3513	0.3365	0.3673	0.3569
6	0.2850	0.3014	0.3580	0.2924	0.3177	0.3411	0.0000	0.3041	0.5071	0.3180	0.2924	0.2902	0.3088	0.2923
7	0.2947	0.2220	0.3717	0.2984	0.2359	0.1976	0.3041	0.0000	0.5053	0.3225	0.3143	0.2968	0.3237	0.3163
8	0.5034	0.5050	0.5837	0.5002	0.5064	0.5104	0.5071	0.5053	0.0000	0.5110	0.5018	0.5046	0.5062	0.5018
9	0.3273	0.3267	0.3976	0.3062	0.3161	0.3584	0.3180	0.3225	0.5110	0.0000	0.3262	0.3222	0.3018	0.3112
10	0.2682	0.3034	0.3581	0.2731	0.3181	0.3513	0.2924	0.3143	0.5018	0.3262	0.0000	0.2823	0.2949	0.2870
11	0.2707	0.3016	0.2715	0.2793	0.3097	0.3365	0.2902	0.2968	0.5046	0.3222	0.2823	0.0000	0.2936	0.3028
12	0.3023	0.3213	0.3762	0.2835	0.3147	0.3673	0.3088	0.3237	0.5062	0.3018	0.2949	0.2936	0.0000	0.3000
13	0.2934	0.3074	0.3734	0.2785	0.3122	0.3569	0.2923	0.3163	0.5018	0.3112	0.2870	0.3028	0.3000	0.0000
14	0.3050	0.2286	0.3782	0.2975	0.2496	0.2670	0.3120	0.2215	0.5014	0.3301	0.3079	0.3058	0.3178	0.3114
15	0.2833	0.2960	0.3109	0.2707	0.3121	0.3346	0.2752	0.3006	0.5017	0.3106	0.2807	0.2851	0.2963	0.2769
16	0.2621	0.2969	0.3443	0.2749	0.3029	0.3460	0.2876	0.2965	0.5057	0.3105	0.2839	0.2832	0.2973	0.2936
17	0.2606	0.2874	0.2988	0.2525	0.3071	0.3325	0.2795	0.2943	0.5029	0.3183	0.2733	0.2739	0.2986	0.2848
18	0.3025	0.2652	0.3407	0.3265	0.2896	0.2516	0.3204	0.2610	0.5040	0.3636	0.3299	0.3228	0.3558	0.3349
19	0.3176	0.2986	0.3862	0.3095	0.3037	0.3200	0.3169	0.3009	0.5102	0.3290	0.3256	0.3147	0.3205	0.3230
20	0.2665	0.2923	0.3435	0.2571	0.2967	0.3370	0.2698	0.2946	0.5012	0.3040	0.2671	0.2742	0.2805	0.2642
21	0.2725	0.3163	0.3563	0.2894	0.3347	0.3543	0.3102	0.3191	0.5044	0.3429	0.2754	0.2896	0.3131	0.3126
22	0.3279	0.2693	0.4005	0.3140	0.2722	0.2240	0.3332	0.2685	0.5082	0.3291	0.3323	0.3353	0.3210	0.3279
23	0.2665	0.2984	0.3473	0.2702	0.3114	0.3458	0.2791	0.3041	0.5001	0.3200	0.2720	0.2902	0.2969	0.2642
24	0.2418	0.2881	0.2733	0.2601	0.3088	0.3361	0.2799	0.2981	0.5005	0.3008	0.2670	0.2773	0.2878	0.2745
25	0.2679	0.2961	0.2379	0.3106	0.3157	0.3551	0.2846	0.3314	0.5037	0.3548	0.3120	0.2715	0.3004	0.2959
26	0.2742	0.2990	0.3036	0.2637	0.3116	0.3505	0.2756	0.3104	0.5032	0.3107	0.2681	0.2901	0.2867	0.2429
27	0.2641	0.2907	0.2952	0.2585	0.3017	0.3337	0.2909	0.2947	0.5048	0.3115	0.2628	0.2759	0.2799	0.2847

	14	15	16	17	18	19	20	21	22	23	24	25	26	27
0	0.3050	0.2833	0.2621	0.2606	0.3025	0.3176	0.2665	0.2725	0.3279	0.2665	0.2418	0.2679	0.2742	0.2641
1	0.2286	0.2960	0.2969	0.2874	0.2652	0.2986	0.2923	0.3163	0.2693	0.2984	0.2881	0.2961	0.2990	0.2907
2	0.3782	0.3109	0.3443	0.2988	0.3407	0.3862	0.3435	0.3563	0.4005	0.3473	0.2733	0.2379	0.3036	0.2952
3	0.2975	0.2707	0.2749	0.2525	0.3265	0.3095	0.2571	0.2894	0.3140	0.2702	0.2601	0.3106	0.2637	0.2585
4	0.2496	0.3121	0.3029	0.3071	0.2896	0.3037	0.2967	0.3347	0.2722	0.3114	0.3088	0.3157	0.3116	0.3017
5	0.2670	0.3346	0.3460	0.3325	0.2516	0.3200	0.3370	0.3543	0.2240	0.3458	0.3361	0.3551	0.3505	0.3337
6	0.3120	0.2752	0.2876	0.2795	0.3204	0.3169	0.2698	0.3102	0.3332	0.2791	0.2799	0.2846	0.2756	0.2909
7	0.2215	0.3006	0.2965	0.2943	0.2610	0.3009	0.2946	0.3191	0.2685	0.3041	0.2981	0.3314	0.3104	0.2947
8	0.5014	0.5017	0.5057	0.5029	0.5040	0.5102	0.5012	0.5044	0.5082	0.5001	0.5005	0.5037	0.5032	0.5048
9	0.3301	0.3106	0.3105	0.3183	0.3636	0.3290	0.3040	0.3429	0.3291	0.3200	0.3008	0.3548	0.3107	0.3115
10	0.3079	0.2807	0.2839	0.2733	0.3299	0.3256	0.2671	0.2754	0.3323	0.2720	0.2670	0.3120	0.2681	0.2628
11	0.3058	0.2851	0.2832	0.2739	0.3228	0.3147	0.2742	0.2896	0.3353	0.2902	0.2773	0.2715	0.2901	0.2759
12	0.3178	0.2963	0.2973	0.2986	0.3558	0.3205	0.2805	0.3131	0.3210	0.2969	0.2878	0.3004	0.2867	0.2799
13	0.3114	0.2769	0.2936	0.2848	0.3349	0.3230	0.2642	0.3126	0.3279	0.2642	0.2745	0.2959	0.2429	0.2847
14	0.0000	0.3007	0.3047	0.2973	0.2642	0.3006	0.3010	0.3134	0.2656	0.2996	0.2950	0.3026	0.2992	0.2987
15	0.3007	0.0000	0.2848	0.2670	0.3210	0.3152	0.2554	0.2943	0.3271	0.2562	0.2626	0.3123	0.2598	0.2694
16	0.3047	0.2848	0.0000	0.2722	0.3260	0.3120	0.2729	0.2941	0.3275	0.2779	0.2572	0.2972	0.2857	0.2716
17	0.2973	0.2670	0.2722	0.0000	0.3071	0.3178	0.2614	0.2867	0.3262	0.2647	0.2570	0.2980	0.2728	0.2645
18	0.2642	0.3210	0.3260	0.3071	0.0000	0.3263	0.3172	0.3273	0.2967	0.3216	0.3145	0.3327	0.3263	0.3204
19	0.3006	0.3152	0.3120	0.3178	0.3263	0.0000	0.3035	0.3340	0.3219	0.3181	0.3131	0.3183	0.3163	0.3103
20	0.3010	0.2554	0.2729	0.2614	0.3172	0.3035	0.0000	0.2884	0.3124	0.2586	0.2521	0.3055	0.2397	0.2561
21	0.3134	0.2943	0.2941	0.2867	0.3273	0.3340	0.2884	0.0000	0.3379	0.2870	0.2807	0.2961	0.2895	0.2807
22	0.2656	0.3271	0.3275	0.3262	0.2967	0.3219	0.3124	0.3379	0.0000	0.3256	0.3216	0.3637	0.3222	0.3200
23	0.2996	0.2562	0.2779	0.2647	0.3216	0.3181	0.2586	0.2870	0.3256	0.0000	0.2517	0.3078	0.2420	0.2640
24	0.2950	0.2626	0.2572	0.2570	0.3145	0.3131	0.2521	0.2807	0.3216	0.2517	0.0000	0.2437	0.2502	0.2584
25	0.3026	0.3123	0.2972	0.2980	0.3327	0.3183	0.3055	0.2961	0.3637	0.3078	0.2437	0.0000	0.2755	0.2701
26	0.2992	0.2598	0.2857	0.2728	0.3263	0.3163	0.2397	0.2895	0.3222	0.2420	0.2502	0.2755	0.0000	0.2675
27	0.2987	0.2694	0.2716	0.2645	0.3204	0.3103	0.2561	0.2807	0.3200	0.2640	0.2584	0.2701	0.2675	0.0000