



Математическое сочинение:

“Индекс Винера в растущих графах”

Чудопалов Денис

ВМК МГУ

Февраль 2026

Содержание

1 Введение	4
1.1 Постановка задачи	4
2 Индекс Винера	5
2.1 Пример вычисления индекса Винера	5
3 Аналитические решения	7
3.1 Полный граф K_n	7
3.2 Путь P_n	7
3.3 Цикл C_n	8
3.4 Звезда $S_n = K_{1,n-1}$	9
3.5 Полный двудольный граф $K_{a,b}$	9
3.6 Полный многодольный граф K_{n_1,\dots,n_k}	10
3.7 Колесо W_n (цикл на $n - 1$ вершинах + центр)	10
3.8 Двойная звезда $DS_{a,b}$	11
3.9 Универсальная формула для графов диаметра 2	11
3.10 Итог	12
4 Случай деревьев	13
4.1 Прямые формулы для деревьев	13
4.2 рекуррентные формулы для деревьев	14
4.3 Итог	17
5 Теоретические основы и реализация итерационного вычисления индекса Винера на растущем графе	18
5.1 Добавление leaf	18
5.2 Добавление edge	18
5.3 Общая схема модулей	18
5.4 Программная реализация на python	18
5.5 Устройство кода генерации растущего графа	19
5.5.1 Дополнительные рёбра (не входящие в BFS-остов)	20
5.5.2 Почему все выполняется корректно	20
5.5.3 Случайное чередование leaf/edge	20
5.5.4 Итог	20
6 Результаты	22
7 Индекс Винера для типовых топологий компьютерных сетей	24
7.1 Шина (bus): граф-путь P_n	24
7.2 Top (2D torus): квадратная решётка $C_k \square C_k$	25
7.2.1 Декартово произведение графов	25
7.2.2 Формула для индекса Винера декартова произведения	26
7.2.3 Применение к квадратному тору $C_k \square C_k$	27
7.3 Fat-tree (толстое дерево): иерархическая топология	27
7.4 Сравнение “число рёбер — средние расстояния”	28
7.5 Вычисление $W(G)$: формулы и алгоритмы	28
7.5.1 Интерпретация для МРР-систем	29

8 Вывод	30
9 Дальнейшие направления для развития	30
10 Список используемой литературы	31

1 Введение

В данной работе рассматриваются способы эффективного расчета индекса Винера для растущего графа, в случае присоединения к нему узла или рёбра. Основная цель настоящей работы — исследовать методы поэтапного (рекуррентного) пересчёта индекса Винера при добавлении в граф новых элементов (вершин или рёбер). Такие алгоритмы позволяют обновлять значение индекса без полного перебора всех пар вершин после каждого изменения графа.

Индекс Винера эффективно используется в химии для корреляции геометрии молекул с их физико-химическими свойствами, а также применяется при анализе сетевых структур различной природы. Так, он служит мерой связности и компактности графовых структур (молекулярных графов, компьютерных и транспортных сетей).

Я захотел разобраться в данной теме, поскольку в условиях роста графа полный пересчёт индекса Винера с нуля при каждом обновлении графа требует вычисления матрицы расстояний и имеет сложность $O(n^2)$ по числу вершин, что крайне неэффективно.

1.1 Постановка задачи

Дано:

Неориентированный граф $G(V, E)$, где V - множество вершин, E - множество ребер.

Задача:

- Дать определение индексу Винера
- Найти аналитические решения
- Разобрать удобные частные случаи (случай деревьев)
- Написать программу для вычисления индекса Винера
- Тестирование программы и получение результатов

2 Индекс Винера

На протяжении всей работы все графы будут неориентированными

Определение 1. Пусть G — граф с множеством вершин $V(G)$ и множеством рёбер $E(G)$. Расстояние $d_G(u, v)$ между двумя вершинами $u, v \in V(G)$ — это минимальное число рёбер на пути в G между u и v .

Определение 2. Пусть G удовлетворяет определению 1. *Винеровский индекс* $W(G)$ графа G определяется формулой

$$W(G) = \sum_{\{u,v\} \subseteq V(G)} d_G(u, v),$$

а *среднее расстояние* $\mu(G)$ между вершинами графа G — формулой

$$\mu(G) = \frac{W(G)}{\binom{|V(G)|}{2}}.$$

Определение 3. Пусть G удовлетворяет определению 1. *Расстояние вершины* v обозначим через $d_G(v)$ и определим как сумму расстояний между v и всеми остальными вершинами графа G :

$$d_G(v) = \sum_{u \in V(G) \setminus \{v\}} d_G(u, v).$$

Таким образом, индекс Винера можно определить немного иначе:

$$W(G) = \frac{1}{2} \sum_{v \in V(G)} d_G(v),$$

где множитель $\frac{1}{2}$ компенсирует тот факт, что каждый путь между u и v учитывается в $d_G(u)$, а также в $d_G(v)$.

Определение 4. Пусть G — граф. *Диаметр* $d(G)$ графа G определяется как

$$d(G) = \max_{\{u,v\} \subseteq V(G)} d_G(u, v).$$

2.1 Пример вычисления индекса Винера

Рассмотрим граф T на 10 вершинах (рис. 1):

$$E(T) = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_6), (v_3, v_7), (v_3, v_8), (v_5, v_9), (v_5, v_{10})\}.$$

Индекс Винера считаем по формуле

$$W(T) = \frac{1}{2} \sum_{v \in V(T)} d_T(v), \quad d_T(v) = \sum_{u \neq v} d_T(u, v).$$

Для данного графа получаем:

$$\begin{aligned} d_T(v_1) &= 31, & d_T(v_2) &= 23, & d_T(v_3) &= 17, & d_T(v_4) &= 19, & d_T(v_5) &= 19, \\ d_T(v_6) &= 27, & d_T(v_7) &= 25, & d_T(v_8) &= 25, & d_T(v_9) &= 27, & d_T(v_{10}) &= 27. \end{aligned}$$

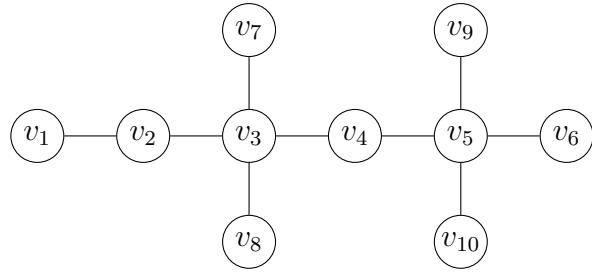


Рис. 1: граф T на 10 вершинах.

Тогда

$$\sum_{i=1}^{10} d_T(v_i) = 240 \quad \Rightarrow \quad W(T) = \frac{1}{2} \cdot 240 = 120.$$

Среднее расстояние между вершинами:

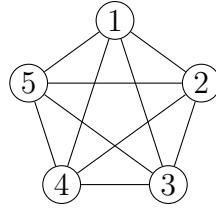
$$\mu(T) = \frac{W(T)}{\binom{10}{2}} = \frac{120}{45} = \frac{8}{3}.$$

3 Аналитические решения

Первое что приходит в голову после постановки задачи для растущих графов, возможно ли вычислять Индекс Винера с нимиальными затратами? На этот вопрос есть ответ - если индекс можно выразить аналитически, то существует конечная формула в которую мы подставляем значение и получаем результат. Далее будем рассматривать такие графы.

3.1 Полный граф K_n

Рисунок (пример K_5)



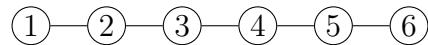
Вывод формулы

В полном графе K_n любая пара различных вершин соединена ребром, значит $\text{dist}(x, y) = 1$ для всех $x \neq y$. Число неупорядоченных пар равно $\binom{n}{2}$, поэтому

$$W(K_n) = \binom{n}{2} = \frac{n(n-1)}{2}.$$

3.2 Путь P_n

Рисунок (пример P_6)



Вывод формулы

Пронумеруем вершины $1, 2, \dots, n$ вдоль пути. Тогда

$$\text{dist}(i, j) = |i - j|.$$

Для каждого расстояния $d \in \{1, \dots, n-1\}$ число пар вершин на расстоянии d равно $n-d$ (это пары $(i, i+d)$, где $i = 1, \dots, n-d$). Тогда

$$W(P_n) = \sum_{d=1}^{n-1} d(n-d) = n \sum_{d=1}^{n-1} d - \sum_{d=1}^{n-1} d^2.$$

Используем известные суммы:

$$\sum_{d=1}^{n-1} d = \frac{(n-1)n}{2}, \quad \sum_{d=1}^{n-1} d^2 = \frac{(n-1)n(2n-1)}{6}.$$

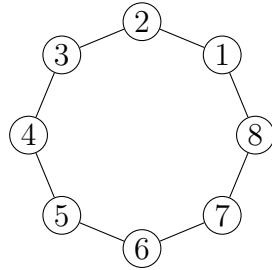
Подставляя, получаем

$$W(P_n) = n \cdot \frac{(n-1)n}{2} - \frac{(n-1)n(2n-1)}{6} = \frac{(n-1)n(n+1)}{6} = \binom{n+1}{3}.$$

$$W(P_n) = \binom{n+1}{3} = \frac{(n-1)n(n+1)}{6}.$$

3.3 Цикл C_n

Рисунок (пример C_8)



Выход формулы

В цикле расстояние между двумя вершинами равно меньшей длине дуги между ними. Положим $k = \lfloor n/2 \rfloor$.

Случай 1: $n = 2k + 1$ (**нечётный**). Для каждого $d = 1, \dots, k$ каждая вершина имеет ровно 2 вершины на расстоянии d , поэтому число неупорядоченных пар на расстоянии d равно n . Тогда

$$W(C_n) = \sum_{d=1}^k d \cdot n = n \cdot \frac{k(k+1)}{2}.$$

Так как $n = 2k + 1$, то $k(k+1) = \frac{(n^2-1)}{4}$,

$$W(C_{2k+1}) = \frac{(2k+1)k(k+1)}{2} = \frac{n(n^2-1)}{8}.$$

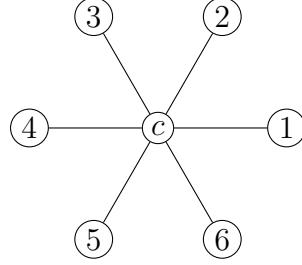
Случай 2: $n = 2k$ (**чётный**). Для $d = 1, \dots, k-1$ число пар на расстоянии d равно n . Для $d = k$ (противоположные вершины) число пар равно $n/2$. Значит

$$W(C_n) = \sum_{d=1}^{k-1} d \cdot n + k \cdot \frac{n}{2} = n \cdot \frac{(k-1)k}{2} + k \cdot k = k^3 = \frac{n^3}{8}.$$

$$W(C_{2k}) = k^3 = \frac{n^3}{8}.$$

3.4 Звезда $S_n = K_{1,n-1}$

Рисунок (пример S_7)



Вывод формулы

В звезде есть центр c и $n - 1$ листьев.

- Пар *центр–лист* ровно $n - 1$, расстояние 1.
- Пар *лист–лист* ровно $\binom{n-1}{2}$, расстояние 2 (через центр).

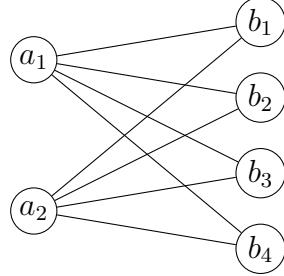
Следовательно,

$$W(S_n) = (n - 1) \cdot 1 + \binom{n - 1}{2} \cdot 2 = (n - 1) + (n - 1)(n - 2) = (n - 1)^2.$$

$$\boxed{W(S_n) = (n - 1)^2.}$$

3.5 Полный двудольный граф $K_{a,b}$

Рисунок (пример $K_{2,4}$)



Вывод формулы

В $K_{a,b}$ любые две вершины из разных долей имеют расстояние 1, а любые две вершины из одной доли — расстояние 2.

- Междольных пар ab , вклад ab .
- Внутри доли размера a пар $\binom{a}{2}$, вклад $2\binom{a}{2}$.
- Внутри доли размера b пар $\binom{b}{2}$, вклад $2\binom{b}{2}$.

Итого

$$\boxed{W(K_{a,b}) = ab + 2 \left(\binom{a}{2} + \binom{b}{2} \right) = ab + a(a - 1) + b(b - 1).}$$

3.6 Полный многодольный граф K_{n_1, \dots, n_k}

Полный вывод формулы

В полном k -дольном графе:

- если вершины в разных долях, расстояние 1;
- если в одной доле, расстояние 2 (через любую вершину другой доли).

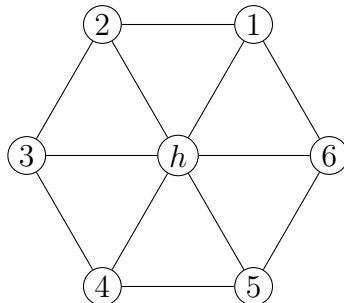
Пусть $n = \sum_{i=1}^k n_i$. Тогда число пар из разных долей равно $\sum_{i < j} n_i n_j$ (вклад 1), а число пар внутри долей равно $\sum_i \binom{n_i}{2}$ (вклад 2). Поэтому

$$W(K_{n_1, \dots, n_k}) = \sum_{1 \leq i < j \leq k} n_i n_j + 2 \sum_{i=1}^k \binom{n_i}{2}.$$

Также можно переписать $\sum_{i < j} n_i n_j = \binom{n}{2} - \sum_i \binom{n_i}{2}$.

3.7 Колесо W_n (цикл на $n - 1$ вершинах + центр)

Рисунок (пример W_7)



Вывод формулы

В колесе W_n :

- Центр h соединён со всеми $n - 1$ вершин обода: вклад $n - 1$ пар на расстоянии 1.
- На ободе есть $(n - 1)$ соседних пар (ребра цикла), расстояние 1, вклад $n - 1$.
- Любые две *несоседние* вершины обода имеют расстояние 2 (через центр). Число всех пар на ободе: $\binom{n-1}{2}$, значит несоседних пар: $\binom{n-1}{2} - (n - 1)$.

Итак,

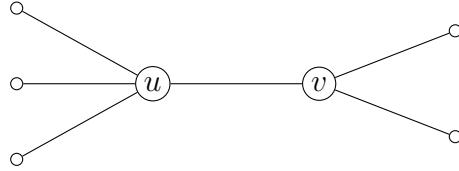
$$W(W_n) = (n - 1) + (n - 1) + 2 \left(\binom{n-1}{2} - (n - 1) \right) = 2 \binom{n-1}{2} = (n - 1)(n - 2).$$

$$W(W_n) = (n - 1)(n - 2).$$

3.8 Двойная звезда $DS_{a,b}$

Двойная звезда $DS_{a,b}$: две центральные вершины u и v соединены ребром, к u подвешено a листьев, к v — b листьев. Всего вершин $n = a + b + 2$.

Рисунок



Выход формулы

Считаем вклад по типам пар:

- (u, v) : одна пара на расстоянии 1.
- $(u, \text{лист у } u)$: a пар на расстоянии 1; $(v, \text{лист у } v)$: b пар на расстоянии 1.
- Два листа у u : $\binom{a}{2}$ пар на расстоянии 2; два листа у v : $\binom{b}{2}$ пар на расстоянии 2.
- Лист у u и вершина v : a пар на расстоянии 2; лист у v и вершина u : b пар на расстоянии 2.
- Лист у u и лист у v : ab пар на расстоянии 3.

Итого

$$W(DS_{a,b}) = 1 + (a + b) + 2 \left(\binom{a}{2} + \binom{b}{2} \right) + 2(a + b) + 3ab.$$

Упростим:

$$2 \binom{a}{2} = a(a - 1), \quad 2 \binom{b}{2} = b(b - 1),$$

поэтому

$$W(DS_{a,b}) = 1 + 3(a + b) + a(a - 1) + b(b - 1) + 3ab.$$

3.9 Универсальный формула для графов диаметра 2

Пусть G — связный граф диаметра 2, то есть любые две вершины находятся на расстоянии 1 или 2. Пусть $n = |V(G)|$, $m = |E(G)|$. Тогда пар на расстоянии 1 ровно m (это рёбра), а остальных пар $\binom{n}{2} - m$, и они на расстоянии 2. Следовательно,

$$W(G) = 1 \cdot m + 2 \left(\binom{n}{2} - m \right) = 2 \binom{n}{2} - m = n(n - 1) - m.$$

$$\boxed{\text{Если } \text{diam}(G) = 2, \ W(G) = n(n - 1) - m.}$$

Этот формула мгновенно даёт решения для многих классов (включая колёса, многие двудольные графы при $a, b \geq 2$ и т. п.).

3.10 Итог

Рассмотренные частные случаи делают вычисление индекса Винера элементарной задачей: при каждом росте графа, нужно всего лишь подставлять значения и получать результат. Однако таких случаев не так много, поэтому далее рассмотрим чуть более обширный класс графов, который может представлять для нас интерес.

4 Случай деревьев

Особый интерес для нас представляют деревья. Поскольку в дереве путь, а значит и расстояние, между двумя вершинами единственен, индекс Винера для дерева вычислять существенно проще, чем для произвольного графа. Далее мы приведём различные формулы для вычисления индекса Винера: в первой части — прямые формулы, во второй — рекуррентные, которые требуют выполнения некоторых условий на деревья, но при их выполнении могут значительно упростить вычисления.

4.1 Прямые формулы для деревьев

Первая формула, которую мы приведём, является базовой и была получена Х. Винером в 1947 году. Если определение индекса Винера акцентирует внимание на сумме расстояний от каждой вершины до всех остальных, то данная формула подсчитывает, сколько раз в суммарных кратчайших путях используется каждое ребро.

Определение 5. Пусть $e = (u, v) \in E(T)$ — ребро дерева T . Поддеревья T_u и T_v определим как связные компоненты графа $T \setminus e$, содержащие соответственно вершины u и v . Числа вершин этих поддеревьев обозначим

$$n_u(e) = |V(T_u)|, \quad n_v(e) = |V(T_v)|.$$

Теорема 1. Пусть T — дерево. Тогда

$$W(T) = \sum_{e=(u,v) \in E(T)} n_u(e) n_v(e). \quad (1)$$

Доказательство. Так как T — дерево, путь между любой вершиной $x \in V(T_u)$ и любой вершиной $y \in V(T_v)$ единственен и обязательно содержит ребро e . Если же вершины x и y выбираются из одной и той же компоненты, то ребро e в путь между ними не входит. Следовательно, величина $n_u(e) n_v(e)$ в точности равна числу пар вершин, кратчайший путь между которыми проходит через e . Суммируя $n_u(e) n_v(e)$ по всем рёбрам $e \in E(T)$, получаем индекс Винера $W(T)$. \square

Определение 6. Пусть $e = (u, v) \in E(G)$ — ребро графа G . Определим множества вершин

$$B_u(e) = \{x \in V(G) : d_G(x, u) < d_G(x, v)\}, \quad B_v(e) = \{y \in V(G) : d_G(y, v) < d_G(y, u)\}.$$

Мощности этих множеств обозначим

$$n_u(e) = |B_u(e)|, \quad n_v(e) = |B_v(e)|.$$

Пример 1. Рассмотрим дерево T на рис. 2. Поскольку дерево не содержит циклов, каждое ребро является мостом: при удалении ребра e дерево распадается на две компоненты размерностей a и $10 - a$, и вклад ребра в индекс Винера равен $a(10 - a)$ (это частный случай формулы $W(T) = \sum_e n_u(e)n_v(e)$).

Посчитаем размеры компонент для каждого ребра (достаточно указать меньшую из двух компонент):

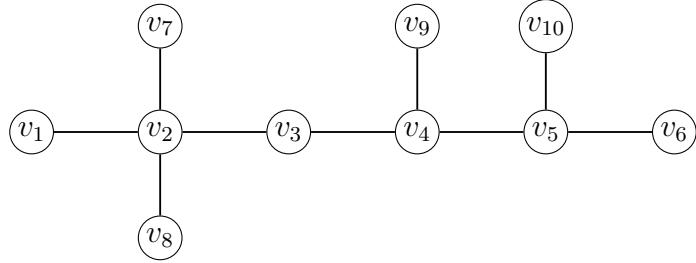


Рис. 2: Дерево T порядка 10.

ребро e	a	$a(10 - a)$
(v_1, v_2)	1	$1 \cdot 9 = 9$
(v_2, v_3)	4	$4 \cdot 6 = 24$
(v_3, v_4)	5	$5 \cdot 5 = 25$
(v_4, v_5)	7	$7 \cdot 3 = 21$
(v_5, v_6)	1	$1 \cdot 9 = 9$
(v_2, v_7)	1	$1 \cdot 9 = 9$
(v_2, v_8)	1	$1 \cdot 9 = 9$
(v_4, v_9)	1	$1 \cdot 9 = 9$
(v_5, v_{10})	1	$1 \cdot 9 = 9$

Суммируя вклады, получаем

$$W(T) = 9 + 24 + 25 + 21 + 9 + 9 + 9 + 9 = 124.$$

4.2 рекуррентные формулы для деревьев

В этом случае мы будем рассматривать специальные поддеревья, чтобы получить индекс Винера всего дерева как комбинацию индексов этих поддеревьев. Например, когда дерево T получается соединением нескольких копий дерева T' в одной вершине u , причём каждый раз используется одна и та же вершина $v' \in V(T')$. Далее будут приведены некоторые рекуррентные формулы.

Первая идея, которая возникает при желании вычислять индекс Винера рекурсивно — взять дерево, удалить лист и вычислить индекс Винера оставшегося поддерева.

Теорема 2. Пусть T — дерево на $n \geq 2$ вершинах, а $v \in V(T)$ — лист дерева T . Пусть также $(u, v) \in E(T)$ и $T' = T - v$ — подграф дерева T , полученный удалением вершины v (и инцидентного ей ребра). Тогда

$$W(T) = W(T') + d_{T'}(u) + n - 1.$$

Доказательство. Рассмотрим две вершины x и y дерева T . Если $v \neq x$ и $v \neq y$, то расстояние между x и y не изменяется после удаления v , поэтому сумма всех таких расстояний равна $W(T')$.

Если одна из вершин равна v , без ограничения общности пусть $x = v$. Тогда для любой вершины $y \in V(T')$ имеем

$$d_T(x, y) = d_T(v, y) = d_{T'}(u, y) + 1.$$

Суммируя по всем $y \in V(T')$ (таких вершин $n - 1$), получаем вклад

$$\sum_{y \in V(T')} d_T(v, y) = \sum_{y \in V(T')} (d_{T'}(u, y) + 1) = d_{T'}(u) + (n - 1).$$

Складывая с $W(T')$, получаем требуемую формулу. \square

Пример 2. Рассмотрим дерево T на n вершинах, полученное из звезды S_{n-1} добавлением ещё одной вершины v , соединённой с листом u звезды S_{n-1} . По Теореме 2 имеем

$$W(T) = W(S_{n-1}) + d_{S_{n-1}}(u) + n - 1.$$

Для звезды S_{n-1} известно, что

$$W(S_{n-1}) = (n - 2)^2,$$

а для листа u : расстояние до центра равно 1, а до остальных $n - 3$ листьев равно 2, поэтому

$$d_{S_{n-1}}(u) = 1 + 2(n - 3) = 2n - 5.$$

Следовательно,

$$W(T) = (n - 2)^2 + (2n - 5) + n - 1 = n^2 - n - 2.$$

Так как в доказательстве того, что T является деревом, использовался лишь подсчёт числа пар вершин $\{v, y\}$, очевидно, что Теорему 2 можно обобщить и на связные графы. Поэтому отдельно запишем обобщенную теорему.

Теорема 3. Пусть G — связный граф, а $v \in V(G)$ — лист (вершина степени 1). Пусть также $(v, u) \in E(G)$ и $G' = G - v$ — подграф графа G , полученный удалением вершины v (и инцидентного ей ребра). Тогда

$$W(G) = W(G') + d_{G'}(u) + |V(G')|.$$

Теорема 4. Пусть T — дерево порядка $n \geq 2$, полученное следующим образом: берём вершину v и поддеревья T_1, \dots, T_m , попарно не пересекающиеся по вершинам, и соединяем v с вершиной $u_i \in V(T_i)$ для каждого $i = 1, \dots, m$. Тогда

$$W(T) = \sum_{i=1}^m \left[W(T_i) + (n - |V(T_i)|) d_{T_i}(u_i) - |V(T_i)|^2 \right] + n(n - 1). \quad (2)$$

Доказательство. Зафиксируем i и рассмотрим пары вершин (x, y) , где $x \in V(T_i)$.

Случай 1: $y \in V(T_i)$. Тогда $d_T(x, y) = d_{T_i}(x, y)$, а сумма по всем таким парам равна $W(T_i)$.

Случай 2: $y \in V(T_j)$, $j \neq i$, или $y = v$. Пусть $y \in V(T_j)$, $j \neq i$ (случай $y = v$ получается аналогично и включится в итоговую формулу). Единственный путь в дереве проходит через вершины u_i , затем v , затем u_j , поэтому

$$d_T(x, y) = d_{T_i}(x, u_i) + d_T(u_i, v) + d_T(v, u_j) + d_{T_j}(u_j, y).$$

Поскольку u_i соединена с v одним ребром, имеем $d_T(u_i, v) = 1$ и $d_T(v, u_j) = 1$, откуда

$$d_T(x, y) = d_{T_i}(x, u_i) + 2 + d_{T_j}(u_j, y).$$

Суммируя по всем $y \notin V(T_i)$, заметим, что для фиксированного $x \in V(T_i)$ величина $d_{T_i}(x, u_i)$ повторяется ровно $n - |V(T_i)|$ раз, поэтому

$$\sum_{y \notin V(T_i)} d_T(x, y) = (n - |V(T_i)|) d_{T_i}(x, u_i) + \sum_{y \notin V(T_i)} (2 + d_{\text{в своей компоненте}}).$$

После суммирования по всем $x \in V(T_i)$ и приведения подобных членов получаем (в стандартной записи через $d_{T_i}(u_i) = \sum_{x \in V(T_i)} d_{T_i}(x, u_i)$) цепочку преобразований:

$$\begin{aligned} W(T) &= \sum_{i=1}^m \left[W(T_i) + \frac{1}{2} \sum_{x \in V(T_i)} \sum_{\substack{y \in V(T) \\ y \notin V(T_i)}} (d_{T_i}(x, u_i) + d_T(u_i, v)) \right] \\ &= \sum_{i=1}^m \left[W(T_i) + \sum_{x \in V(T_i)} (d_{T_i}(x, u_i) + 1) (n - |V(T_i)|) \right] \\ &= \sum_{i=1}^m \left[W(T_i) + (n - |V(T_i)|)(d_{T_i}(u_i) + |V(T_i)|) \right] \\ &= \sum_{i=1}^m \left[W(T_i) + (n - |V(T_i)|) d_{T_i}(u_i) - |V(T_i)|^2 \right] + n \sum_{i=1}^m |V(T_i)|. \end{aligned}$$

Так как вершины T_1, \dots, T_m вместе дают все вершины дерева, кроме v , имеем

$$\sum_{i=1}^m |V(T_i)| = n - 1,$$

и потому последний член равен $n(n - 1)$. Получаем формулу (2). \square

Пример 3. Рассмотрим дерево T на n вершинах, полученное следующим образом: берём путь P_{l+1} и к одному из его концов v присоединяем ещё $n - l - 1$ висячих вершин (то есть добавляем $n - l - 1$ листьев, смежных с v). Покажем, как вычислить $W(T)$ с помощью Теоремы 4.

В качестве разделяющей вершины возьмём вершину v . Тогда поддеревья T_i имеют вид:

- $n - l - 1$ поддеревьев, состоящих из одной вершины (лист), для каждого из них $|V(T_i)| = 1$, $W(T_i) = 0$, $d_{T_i}(u_i) = 0$;
- одно поддерево, являющееся путём на l вершинах (это P_l), причём u — его конец, смежный с v .

Для одиночной вершины вклад в сумму из (2) равен

$$W(T_i) + (n - |V(T_i)|) d_{T_i}(u_i) - |V(T_i)|^2 = 0 + (n - 1) \cdot 0 - 1^2 = -1,$$

поэтому суммарный вклад всех $n - l - 1$ листьев равен $-(n - l - 1)$.

Для пути P_l используем известные формулы:

$$W(P_l) = \binom{l+1}{3}, \quad d_{P_l}(u) = \sum_{k=0}^{l-1} k = \binom{l}{2}, \quad |V(P_l)| = l.$$

Тогда вклад этого поддерева в (2) равен

$$W(P_l) + (n - l) d_{P_l}(u) - l^2 = \binom{l+1}{3} + (n - l) \binom{l}{2} - l^2.$$

С учётом последнего слагаемого $n(n - 1)$ из (2) получаем

$$\begin{aligned} W(T) &= \left(\binom{l+1}{3} + (n - l) \binom{l}{2} - l^2 \right) - (n - l - 1) + n(n - 1) \\ &= \binom{l+1}{3} + (n - l) \binom{l}{2} - (n - l - 1) - l^2 + n(n - 1). \end{aligned}$$

Упрощая выражение, получаем компактную форму:

$$W(T) = \binom{l}{3} + (n - l - 1) \binom{l}{2} + (n - 1)^2.$$

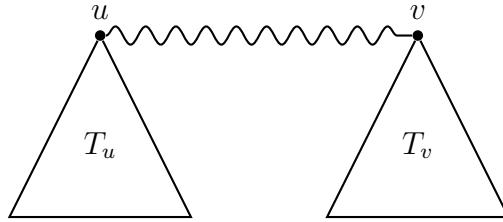


Рис. 3: Деревья T_u и T_v , соединённые путём из k новых вершин.

4.3 Итог

В этой главе мы показали основные прямые и рекуррентные методы вычисления индекса Винера в деревьях. С помощью итерационных методов вычислять индекс Винера в растущих деревьях можно с помощью нехитрых формул, что минимизирует процесс вычислений и делает их максимально быстрыми.

5 Теоретические основы и реализация итерационного вычисления индекса Винера на растущем графе

Была поставлена задача реализовать конкретную процедуру по вычислению индексов Винера для растущих графов эффективнее, чем в готовом пакете networkX. В пакете существует только одна функция вычисления индекса Винера, которая реализована для статичного графа. Поэтому вычисления с помощью готовой функции будут крайне не эффективными. Следовательно логично реализовать ее самим. Для начала надо понять как это можно сделать для случая `leaf` и `edge`.

5.1 Добавление `leaf`

Рассмотрим операцию роста `leaf`: к графу G добавляется новая вершина $x \notin V$ и ровно одно ребро (x, p) , где $p \in V$ — уже существующая вершина. Обозначим новый граф через

$$G' = G + \{x\} + \{(x, p)\}.$$

Добавление листа не изменяет расстояния между старыми вершинами, так как возникнуть новых более коротких маршрутов не может. Следовательно, изменения индекса Винера возникают только за счёт новых пар (x, v) , $v \in V$:

$$W(G') = W(G) + \sum_{v \in V} d_{G'}(x, v). \quad (3)$$

Так как граф невзвешенный, величина $\sum_{v \in V} d_{G'}(x, v)$ вычисляется одним обходом в ширину (BFS) из источника x за время $O(|V| + |E|)$ (т.е можно считать что за линейное $O(1)$ время) на текущем графе.

5.2 Добавление `edge`

Рассмотрим операцию роста `edge`: добавляется ребро между двумя уже существующими вершинами $a, b \in V$:

$$G' = G + \{(a, b)\}.$$

Добавление ребра может сократить кратчайшие пути для большого числа пар вершин. Поэтому точное итерационное обновление $W(G)$ при вставках рёбер неочевидно; поэтому потребуется точный пересчёт $W(G')$.

5.3 Общая схема модулей

Реализация разделена на два файла:

- `wiener.py` — класс `WienerGrowingUnweighted`, вычисление индекса Винера при росте графа;
- `main.py` — загрузка датасета, построение графа, рандомизированное выполнение операций роста, измерение времени и сохранение графика.

5.4 Программная реализация на python

Реализуем Класс и соответствующие ему функции.

```

1 class WienerGrowingUnweighted:
2
3
4     def __init__(self) -> None:
5         self.G = nx.Graph()
6         self.W = 0.0
7
8     def add_initial_node(self, u: Any) -> None:
9         self.G.add_node(u)
10        self.W = 0.0
11
12    def add_leaf(self, u: Any, attach_to: Any) -> float:
13        self.G.add_node(u)
14        self.G.add_edge(u, attach_to)
15        dist = \
16        = nx.single_source_shortest_path_length(self.G, u)
17        self.W += sum(dist.values())
18        return self.W
19
20    def add_edge(self, a: Any, b: Any) -> float:
21        if a == b:
22            return self.W
23
24        self.G.add_edge(a, b)
25        self.W = float(nx.wiener_index(self.G))
26        return self.W

```

5.5 Устройство кода генерации растущего графа

Далее часть `main.py`, отвечающая за генерацию *растущего графа*.

На вход подаётся файл со списком рёбер (формат *edgelist*). Функция загрузки читает файл в байтовом режиме (без декодирования в UTF-8), формируя неориентированный граф G (объект `networkx.Graph`).

Если исходный граф несвязен, выбирается крупнейшая связная компонента H :

$$H = \text{LargestConnectedComponent}(G).$$

Дальнейший рост происходит так, чтобы текущий граф на каждом шаге оставался связным.

Функция `bfs_tree_edges_sorted(H, root)` строит BFS-остов из вершины *root*.

Результатом является список рёбер BFS-дерева в порядке открытия:

$$T = ((p_1, c_1), (p_2, c_2), \dots, (p_{n-1}, c_{n-1})),$$

где каждое ребро (p_i, c_i) означает: вершина c_i впервые обнаружена из p_i .

На основе T строится отображение

$$\text{children_by_parent}[p] = \{c : (p, c) \in T\},$$

которое в коде реализовано как словарь `parent → deque(children)`. Эта структура позволяет добавлять новые вершины только тогда, когда их родитель уже присутствует в текущем графе.

5.5.1 Дополнительные рёбра (не входящие в BFS-остов)

Все рёбра исходного графа H , не входящие в BFS-остов, образуют множество *дополнительных рёбер*:

$$E_{\text{extra}} = E(H) \setminus E(T).$$

В коде это формируется в виде списка `extra_edges` (с нормализацией порядка концов ребра и сортировкой), чтобы вход был определен.

5.5.2 Почему все выполняется корректно

Для корректного роста вводится множество активных вершин A — вершин, которые уже добавлены в текущий граф роста. В начале:

$$A = \{\text{root}\}.$$

Операция `leaf(c, p)` выполнима только если $p \in A$. Операция `edge(u, v)` выполнима только если $u \in A$ и $v \in A$.

Чтобы эффективно находить `edge`-операции, которые становятся выполнимыми после добавления новой вершины, строится индекс инцидентности:

$$\text{incident}[x] = \{ i \mid \text{extra_edges}[i] \text{ инцидентно вершине } x \}.$$

После добавления вершины x активируются все рёбра (x, y) , для которых $y \in A$. В коде это множество хранится как `eligible_edges` (индексы в `extra_edges`).

5.5.3 Случайное чередование `leaf`/`edge`

Функция `make_randomized_ops(...)` строит последовательность шагов роста (до `max_ops` шагов), случайно чередуя `leaf` и `edge` при соблюдении всех условий.

Если доступны оба типа операций, то `edge` выбирается с вероятностью p_{edge} , иначе выбирается `leaf`. Если доступен только один тип, выбирается он.

Роль `seed`. Для воспроизводимости используется генератор `random.Random(seed)`. При фиксированном `seed` последовательность случайных выборов повторяется, то есть траектория роста (порядок добавления вершин и рёбер) будет одинаковой при каждом запуске.

5.5.4 Итог

Итоговая логика генерации роста:

1. загрузить граф G и выделить крупнейшую компоненту H ;
2. выбрать корень root ;
3. построить BFS-остов T и структуру `children_by_parent`;
4. построить список дополнительных рёбер `extra_edges`;
5. начиная с $A = \{\text{root}\}$, итеративно выполнять шаги:
 - `leaf`: добавить новую вершину из `children_by_parent` и обновить A ;

- после добавления вершины активировать допустимые дополнительные рёбра в `eligible_edges`;
- `edge`: выбрать одно допустимое ребро из `eligible_edges` и добавить его;

пока не выполнено `max_ops` шагов или пока операции не исчерпаны.

Такая конструкция обеспечивает, что на каждом шаге строится корректный промежуточный граф (все рёбра добавляются только между уже существующими вершинами), при этом порядок роста может быть случайно перемешан, но остаётся воспроизводимым при фиксированном `seed`.

6 Результаты

Написанная мной программа была протестирована на трех датасетах, взятых из Stanford Large Network Dataset Collection(SNAP). Результат оказался предсказуемым: времени потребовалось почти в 2 раза меньше, чем у стандартной функции networkX. Оценка сложности для операции типа `leaf` не $O(n^2)$, а $O(n + n)$ (т.е $O(1)$). Даже чисто интуитивно можно заметить, что ветвь синей линии напоминает ветви параболы (функция $f(x) = x^2$), а оранжевая $f(x) = \frac{x^2}{2}$

Графики

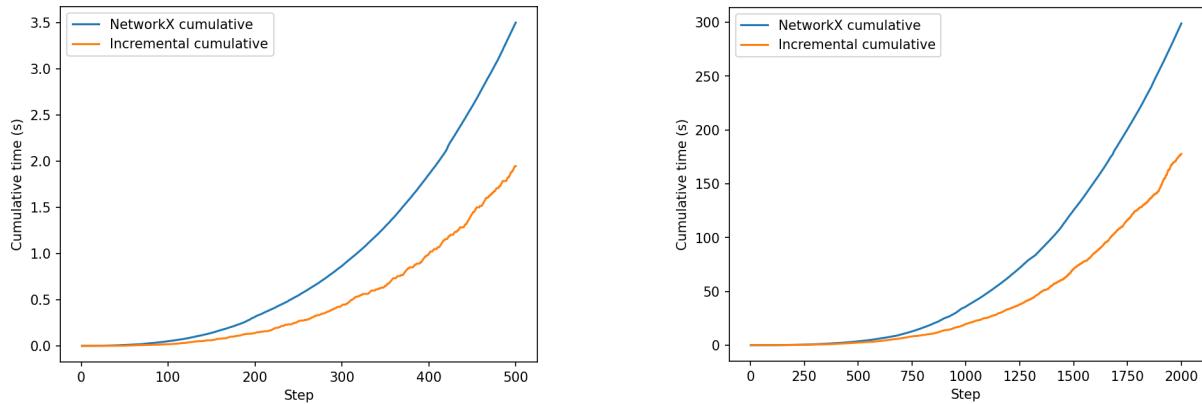


Рис. 4: Сеть сотрудничества Arxiv в области физики высоких энергий

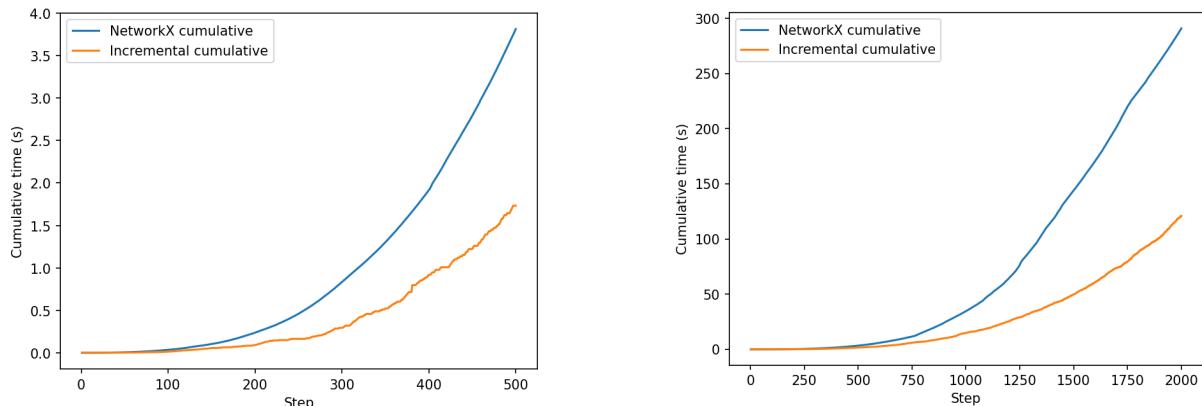


Рис. 5: Сеть электронной переписки компании Enron

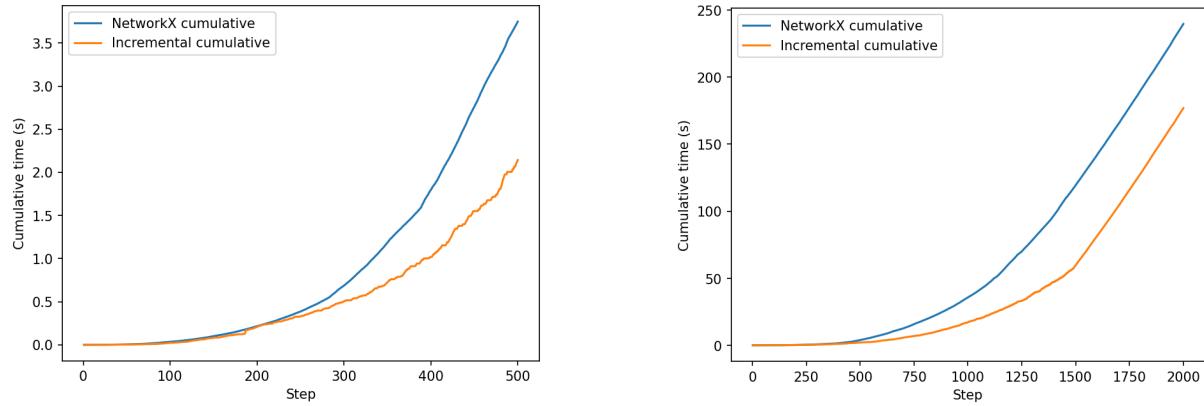


Рис. 6: Социальные круги из Facebook (анонимизированные)

7 Индекс Винера для типовых топологий компьютерных сетей

Кроме химии, индекс Винера можно применять в оценке эффективности различных топологий сетей, мультипроцессорных взаимодействий и топологии межсоединений в вычислительных кластерах. Граф $G = (V, E)$ естественно интерпретируется как модель коммуникационной структуры: вершины соответствуют узлам, а рёбра — физическим/логическим соединениям. В этом контексте *индекс Винера* $W(G)$

$$W(G) = \sum_{\{u,v\} \subseteq V} d_G(u, v)$$

характеризует *суммарную* удалённость узлов, а нормированная величина

$$\mu(G) = \frac{W(G)}{\binom{|V|}{2}}$$

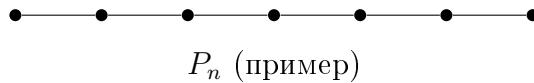
есть *средняя длина кратчайшего пути* между случайной парой узлов. При прочих равных, меньшие значения $\mu(G)$ означают меньшую среднюю “маршрутную задержку” (в числе переходов).

Далее рассмотрены три типовые топологии: *шина* (bus), *top* (2D torus), *fat-tree* (толстое дерево). Цель данного раздела — показать, как масштабируются $W(G)$ и $\mu(G)$ при росте числа вершин и при каком числе рёбер достигается тот или иной компромисс.

7.1 Шина (bus): граф-путь P_n

Линейная топология “шина” моделируется путём P_n из n вершин, соединённых последовательно. Число рёбер минимально для связности: $|E| = n - 1$.

Рисунок



Вывод формулы

Для пути P_n расстояние между вершинами i и j равно $|i - j|$, поэтому индекс Винера вычисляется аналитически:

$$W(P_n) = \sum_{1 \leq i < j \leq n} (j - i) = \sum_{d=1}^{n-1} d(n - d) = \binom{n+1}{3} = \frac{(n-1)n(n+1)}{6}.$$

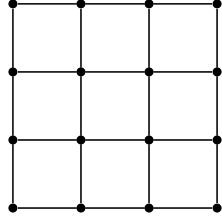
Отсюда средняя длина пути

$$\mu(P_n) = \frac{W(P_n)}{\binom{n}{2}} = \frac{n+1}{3} \sim \frac{n}{3}.$$

Таким образом, при минимальном числе рёбер путь даёт наихудшую масштабируемость расстояний: μ растёт линейно.

7.2 Top (2D torus): квадратная решётка $C_k \square C_k$

Двумерный *квадратный тор* — это решётка $k \times k$ с замыканием по горизонтали и вертикали. Формально это декартово произведение циклов $C_k \square C_k$. Число вершин $N = k^2$, число рёбер $|E| = 2k^2 = 2N$ (каждая вершина степени 4).



$C_4 \square C_4$ (пример квадратного тора)

Ниже приведён подробный вывод формул, используемых для вычисления индекса Винера тора.

7.2.1 Декартово произведение графов

Пусть $G = (V(G), E(G))$ и $H = (V(H), E(H))$ — неориентированные связные графы. *Декартовым произведением* $G \square H$ называется граф со следующими свойствами:

- множество вершин равно декартову произведению множеств вершин:

$$V(G \square H) = V(G) \times V(H);$$

- две вершины (g, h) и (g', h') смежны тогда и только тогда, когда выполняется одно из условий:

1. $g = g'$ и $(h, h') \in E(H)$;
2. $h = h'$ и $(g, g') \in E(G)$.

Другими словами, за один шаг по ребру в $G \square H$ меняется ровно одна координата: либо координата в G , либо координата в H .

Теорема 5. Для любых $(g, h), (g', h') \in V(G \square H)$ выполняется равенство

$$d_{G \square H}((g, h), (g', h')) = d_G(g, g') + d_H(h, h').$$

Доказательство. Докажем сначала неравенство “ \geq ”. Возьмём произвольный путь P в $G \square H$, соединяющий (g, h) и (g', h') , и пусть его длина равна L (число рёбер в пути). На каждом ребре пути изменяется ровно одна координата: либо g (движение в G), либо h (движение в H). Обозначим через L_G число шагов пути, на которых менялась координата g , а через L_H — число шагов, на которых менялась координата h . Тогда

$$L = L_G + L_H.$$

Чтобы перейти от g к g' внутри графа G , требуется как минимум $d_G(g, g')$ шагов, следовательно $L_G \geq d_G(g, g')$. Аналогично $L_H \geq d_H(h, h')$. Поэтому

$$L = L_G + L_H \geq d_G(g, g') + d_H(h, h').$$

Так как это верно для любого пути P , то верно и для кратчайшего пути:

$$d_{G \square H}((g, h), (g', h')) \geq d_G(g, g') + d_H(h, h').$$

Теперь докажем неравенство “ \leq ”. Возьмём кратчайший путь в G от g к g' длины $d_G(g, g')$. В графе $G \square H$ при фиксированном h можно повторить этот путь, не изменяя вторую координату:

$$(g, h) \rightarrow (\cdot, h) \rightarrow \cdots \rightarrow (g', h).$$

Затем возьмём кратчайший путь в H от h к h' длины $d_H(h, h')$ и пройдём его при фиксированном g' :

$$(g', h) \rightarrow (g', \cdot) \rightarrow \cdots \rightarrow (g', h').$$

Полученный путь в $G \square H$ имеет длину $d_G(g, g') + d_H(h, h')$, откуда

$$d_{G \square H}((g, h), (g', h')) \leq d_G(g, g') + d_H(h, h').$$

Совместно с уже доказанным неравенством “ \geq ” получаем равенство. \square

7.2.2 Формула для индекса Винера декартова произведения

Индекс Винера графа X определяется как сумма расстояний по всем неупорядоченным парам вершин:

$$W(X) = \sum_{\{x,y\} \subseteq V(X)} d_X(x, y).$$

Теорема 6. Для любых связных графов G и H выполняется формула

$$W(G \square H) = |V(H)|^2 W(G) + |V(G)|^2 W(H).$$

Доказательство. Обозначим $n_G = |V(G)|$ и $n_H = |V(H)|$. По определению и по Теореме 5 имеем

$$\begin{aligned} W(G \square H) &= \sum_{\{(g,h),(g',h')\} \subseteq V(G) \times V(H)} d_{G \square H}((g, h), (g', h')) \\ &= \sum_{\{(g,h),(g',h')\}} (d_G(g, g') + d_H(h, h')) \\ &= \sum_{\{(g,h),(g',h')\}} d_G(g, g') + \sum_{\{(g,h),(g',h')\}} d_H(h, h'). \end{aligned}$$

Рассмотрим первую сумму. Зафиксируем пару вершин $\{g, g'\} \subseteq V(G)$. Тогда h можно выбрать n_H способами, и h' также n_H способами, то есть каждой паре $\{g, g'\}$ соответствует ровно n_H^2 пар $\{(g, h), (g', h')\}$ в $G \square H$. Следовательно,

$$\sum_{\{(g,h),(g',h')\}} d_G(g, g') = n_H^2 \sum_{\{g,g'\} \subseteq V(G)} d_G(g, g') = n_H^2 W(G).$$

Аналогично для второй суммы:

$$\sum_{\{(g,h),(g',h')\}} d_H(h, h') = n_G^2 W(H).$$

Складывая, получаем

$$W(G \square H) = n_H^2 W(G) + n_G^2 W(H),$$

что и требовалось. \square

7.2.3 Применение к квадратному тору $C_k \square C_k$

Квадратный тор определяется как $C_k \square C_k$, где C_k — цикл длины k . Поскольку $|V(C_k)| = k$, из Теоремы 6 получаем

$$W(C_k \square C_k) = k^2 W(C_k) + k^2 W(C_k) = 2k^2 W(C_k).$$

Для цикла C_k индекс Винера известен в явном виде:

$$W(C_k) = \begin{cases} \frac{k^3}{8}, & k \text{ чётное,} \\ \frac{k(k^2 - 1)}{8}, & k \text{ нечётное.} \end{cases}$$

Следовательно

$$W(C_k \square C_k) = \begin{cases} 2k^2 \cdot \frac{k^3}{8} = \frac{k^5}{4}, & k \text{ чётное,} \\ 2k^2 \cdot \frac{k(k^2 - 1)}{8} = \frac{k^3(k^2 - 1)}{4}, & k \text{ нечётное.} \end{cases}$$

Пусть $N = |V(C_k \square C_k)| = k^2$. Тогда $k = \sqrt{N}$ и при больших N справедлива асимптотика

$$W(C_k \square C_k) \sim \frac{k^5}{4} = \frac{1}{4}(\sqrt{N})^5 = \frac{1}{4}N^{5/2}, \quad \mu = \frac{W}{\binom{N}{2}} = \Theta(\sqrt{N}).$$

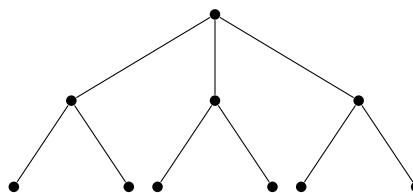
7.3 Fat-tree (толстое дерево): иерархическая топология

Под *fat-tree* понимают многоуровневую иерархическую топологию сети, в которой узлы нижнего уровня (leaf/edge) подключены к промежуточным (aggregation), а те — к верхнему уровню (core); при движении вверх число параллельных соединений увеличивается, поэтому сеть не образует “узких мест” и допускает несколько альтернативных маршрутов между узлами. Основная идея: обеспечить малый диаметр и большое число альтернативных кратчайших маршрутов, избегая “узких мест” у корня.

Чтобы иметь строгую математическую модель для оценок, удобно рассматривать приближение fat-tree в виде *сбалансированного* бинарного дерева высоты h (перфектное дерево). Тогда

$$N = |V| = \frac{b^{h+1} - 1}{b - 1}, \quad |E| = N - 1.$$

Глубина h при этом масштабируется как $h = \Theta(\log_b N)$, а характерная длина маршрута между вершинами пропорциональна h , поскольку пути между листьями проходят через их наименьшего общего предка.



пример сбалансированного дерева (модель fat-tree)

Масштабирование W и μ . Для сбалансированных иерархий типичное расстояние между вершинами имеет порядок $O(\log N)$, а число пар вершин имеет порядок N^2 . Поэтому для fat-tree-структур (в приближении сбалансированного дерева) естественна оценка

$$W(G) = \Theta(N^2 \log N), \quad \mu(G) = \Theta(\log N).$$

Эта зависимость принципиально лучше, чем $\Theta(N)$ у шины и лучше, чем $\Theta(\sqrt{N})$ у двумерного тора. В реальных fat-tree топологиях число рёбер обычно также линейно по N , однако с заметным коэффициентом, что связано с необходимостью множества параллельных маршрутов.

7.4 Сравнение “число рёбер — средние расстояния”

Сравнение трёх топологий можно суммировать по порядкам роста при увеличении числа вершин N :

Топология	$ E $	$W(G)$	$\mu(G)$
Шина (P_N)	$N - 1$	$\Theta(N^3)$	$\Theta(N)$
Квадратный тор ($C_{\sqrt{N}} \square C_{\sqrt{N}}$)	$\approx 2N$	$\Theta(N^{5/2})$	$\Theta(\sqrt{N})$
fat-tree	$\approx N - 1$	$\Theta(N^2 \log N)$	$\Theta(\log N)$

С практической точки зрения, при фиксированном N и ограничениях на стоимость связей (число рёбер) “хорошей” можно считать топологию, обеспечивающую меньшую $\mu(G)$ при сопоставимом $|E|$.

7.5 Вычисление $W(G)$: формулы и алгоритмы

(1) Аналитические формулы. Для шины (пути) и тора (как декартова произведения) индекс Винера можно вычислять по формулам:

$$W(P_n) = \binom{n+1}{3}, \quad W(C_k \square C_k) = 2k^2 W(C_k),$$

где $W(C_k)$ вычисляется по известной формуле (в зависимости от чётности k) или по асимптотике при больших k .

(2) Деревья (включая модель fat-tree): эффективный подсчёт по рёбрам-разрезам. Для любого дерева T справедлива формула

$$W(T) = \sum_{e \in E(T)} s_e (N - s_e),$$

где s_e — размер одной из компонент после удаления ребра e (вторая имеет размер $N - s_e$). Практически это удобно считать через размеры поддеревьев.

(3) Общие графы: замечание о добавлении рёбер. Для произвольных графов точное инкрементальное обновление $W(G)$ при добавлении рёбер сложнее, поскольку одно новое ребро может уменьшить расстояния у большого числа пар вершин. В этом случае на практике применяются либо пересчёт кратчайших путей (например, BFS от каждой вершины для невзвешенных графов), либо приближённые методы (выборка пар, опорные вершины), либо более сложные динамические структуры для поддержки кратчайших путей.

Algorithm 1 Вычисление индекса Винера для дерева по размерам поддеревьев

Require: Дерево $T = (V, E)$, $|V| = N$

Ensure: $W(T)$

- 1: Выбрать корень r
 - 2: Одним DFS посчитать $sub[v]$ — размер поддерева вершины v
 - 3: $W \leftarrow 0$
 - 4: **for** каждое ребро (p, v) , где p — родитель v **do**
 - 5: $s \leftarrow sub[v]$
 - 6: $W \leftarrow W + s \cdot (N - s)$
 - 7: **end for**
 - 8: **return** W
-

7.5.1 Интерпретация для МПР-систем

В контексте МПР (massively parallel processing) граф $G = (V, E)$ естественно рассматривать как модель межсоединительной сети: вершины соответствуют вычислительным узлам и/или коммутаторам, а рёбра — физическим или логическим каналам связи. В этой постановке длина кратчайшего пути $d_G(u, v)$ интерпретируется как число *переходов* (hop count) между узлами u и v , а нормированная величина

$$\mu(G) = \frac{W(G)}{\binom{|V|}{2}}$$

характеризует *среднюю длину маршрута* между случайной парой узлов. При прочих равных меньшие значения $\mu(G)$ соответствуют меньшей средней “геодезической” задержке, поскольку каждое дополнительное ребро на маршруте обычно означает прохождение дополнительного сетевого элемента, например буферизации.

Однако, производительность межсоединений в МПР определяется не только геометрией кратчайших путей. Много факторов влияет на производительность, а топология, только один из них. Поэтому индекс Винера следует трактовать как *первичную метрическую характеристику*, отражающую среднюю длину маршрута, но не полностью описывающую пропускные свойства сети.

С этой точки зрения типовые топологии демонстрируют различный компромисс между числом рёбер и ростом $\mu(G)$. Линейная топология “шина” (путь P_N) минимальна по числу рёбер ($|E| = N - 1$), однако имеет $\mu(P_N) = \Theta(N)$, что делает её непригодной для масштабируемых МПР-систем из-за быстро растущих задержек. Двумерный тор $C_k \square C_k$ (где $N = k^2$ и $|E| \approx 2N$) обеспечивает $\mu = \Theta(\sqrt{N})$, что существенно лучше по расстояниям и при этом сохраняет регулярную структуру и степень вершин, удобные для локальных коммуникаций. Иерархические топологии fat-tree (в сбалансированном дереве) дают существенно меньшие средние расстояния $\mu = \Theta(\log N)$, а также предоставляют множество альтернативных маршрутов, что особенно важно для коллективных операций и смешанных коммуникационных нагрузок, однако обычно требуют большего числа соединений и более сложной коммутационной инфраструктуры.

Таким образом, в терминах индекса Винера и средней длины маршрута $\mu(G)$ “хорошей” для МПР можно считать топологию, у которой $\mu(G)$ растёт как можно медленнее при сопоставимой стоимости связей (числе рёбер и портов). При этом для корректного практического сравнения необходимо дополнительно учитывать так же характеристики пропускной способности.

8 Вывод

В данной работе я рассмотрел различные способы вычисления индекса Винера для растущего графа. Были предложены аналитические решения некоторых случаев роста графа. Проанализировал частный случай дерева и усовершенствована функция вычисления индекса Винера для растущих графов из библиотеки networkX и представлены графики скорости вычислений. В итоге время вычислений сократилось почти в 2 раза. Рассмотрел случаи применения индекса Винера в топологии компьютерных сетей.

9 Дальнейшие направления для развития

Индекс Винера применяется во многих сферах, поэтому было бы логично для будущих исследований искать новые способы эффективного вычисления, а так же расширять и обобщать рекуррентные формулы на более широкий класс графов, не ограничиваясь деревьями.

Так же интерес могут представлять исследование погрешностей при приближённом обновлении индекса Винера — оценка, насколько точно можно обновлять $W(G)$ на больших графах, жертвуя частью точности ради выигрыша в скорости.

Полученные алгоритмы и подходы можно применять к реальным различным задачам: от анализа сетей различной природы (социальных, транспортных, коммуникационных) до исследования молекулярных структур.

10 Список используемой литературы

- [1] A. Altassan и H. Ahmad. “Computation of Distance-Based Polynomials and Topological Indices for Generalized Friendship Graphs ($G(n, k)$): A Python-Based Approach”. B: *Malaysian Journal of Mathematical Sciences* 19.4 (2025), с. 1453—1470. DOI: 10.47836/mjms.19.4.15.
- [2] Bojan Mohar и Tomaž Pisanski. “How to compute the Wiener index of a graph”. B: *Journal of Mathematical Chemistry* 2 (1988), с. 267—277. DOI: 10.1007/BF01167206.
- [3] Индекс Винера. Википедия. URL: <https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81%D0%92%D0%B8%D0%BD%D0%B5%D1%80%D0%B0>.
- [4] Aric A. Hagberg, Daniel A. Schult и Pieter J. Swart. “Exploring network structure, dynamics, and function using NetworkX”. B: *Proceedings of the 7th Python in Science Conference (SciPy 2008)*. Под ред. Gaël Varoquaux, Travis Vaught и Jarrod Millman. Pasadena, CA, USA, авг. 2008, с. 11—15.
- [5] *NetworkX: Software for Complex Networks*. Официальная документация. URL: <https://networkx.org/en/>.