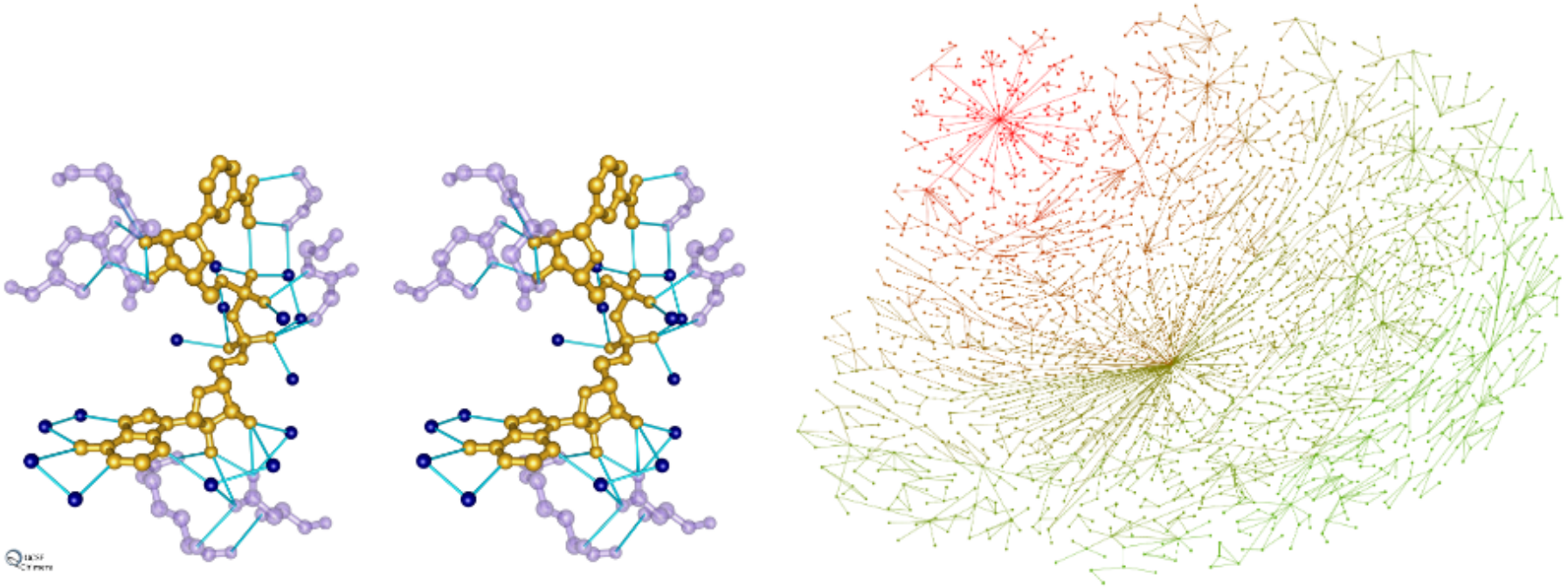
# Metody kernelowe na grafach

Maciej Brzeski

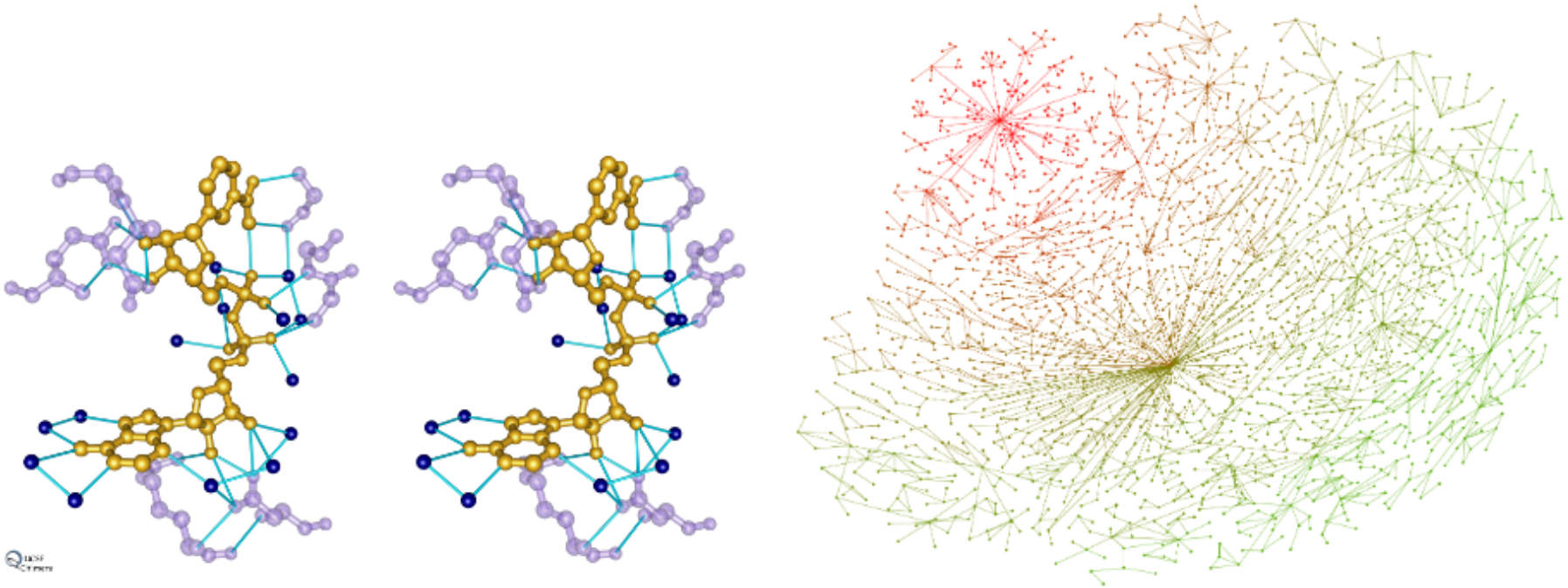
Seminarium: Matematyka dyskretna

22 kwietnia 2020

# Agenda



Rysunek: Przykłady użycia grafów do zamodelowania molekuł białkowych (z lewej) lub sieci internetowej (z prawej)



Rysunek: Przykłady użycia grafów do zamodelowania molekuł białkowych (z lewej) lub sieci internetowej (z prawej)

- jak podobne są dwa grafy?
- jak podobne są dwa wierzchołki w grafie?

# Definicja funkcji podobieństwa

# Definicja funkcji podobieństwa

Dla danych dwóch grafów $G$ i $G'$ chcemy znaleźć funkcję:

$$s\colon G_1 \times G_2 \mapsto \mathbb{R},$$

która mierzy podobieństwo pomiędzy grafami $G$ i $G'$.

# Techniki porównywania grafów

# Techniki porównywania grafów

- algorytmy matchujące
- izomorfizm grafów, izomorfizm podgrafu
- największy wspólny podgraf, najmniejszy wspólny nadgraf, maksymalne wspólne podgrafy
- algorytmy niepełnego dopasowania, odległość edycyjna
- deskryptory topologiczne

# Metody kernelowe

Mamy pewien zbiór obiektów $\mathcal{X}$. Chcemy znaleźć przestrzeń Hilberta $\mathcal{F}$ funkcję mapującą $\phi : \mathcal{X} \mapsto \mathcal{F}$, dla której nasza funkcja podobieństa będzie wyrażona:

$$k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{F}} \tag{1}$$

# Dodatnia półokreśloność

## Definition (Dodatnia półokreśloność)

Macierz rzeczywistą K, o wymiarach $n \times n$, dla której zachodzi $v^T K v \geqslant 0$ dla dowolnego wektora $v \in \mathbb{R}$ nazywamy **dodatnio półokreśloną**.

Funkcję $k\colon \mathcal{X} \mapsto \mathbb{R}$ nazywamy dodatnio półokreśloną, jeśli dla dowolnych $x_1, \ldots, x_n \in \mathcal{X}$ macierz $K \in \mathbb{R}^{n \times n}$ zdefiniowaną przez $K_{ij} = k(x_i, x_j)$ jest dodatnio półokreślona.

# Twierdzenie Mercera

Jeśli żądamy, by istniała przestrzeń Hilberta $\mathcal{F}$, oraz funkcja mapująca $\phi$, to nasza funkcja podobieństwa (kernel) była symetryczna oraz dodatnio półokreślona.

# Twierdzenie Mercera

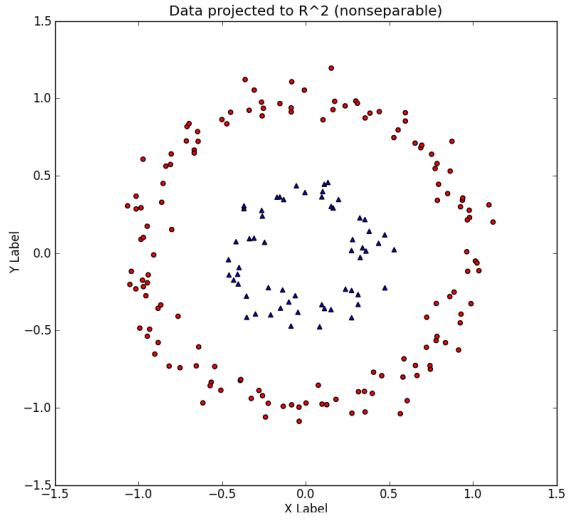
Jeśli żądamy, by istniała przestrzeń Hilberta $\mathcal{F}$, oraz funkcja mapująca $\phi$, to nasza funkcja podobieństwa (kernel) była symetryczna oraz dodatnio półokreślona.

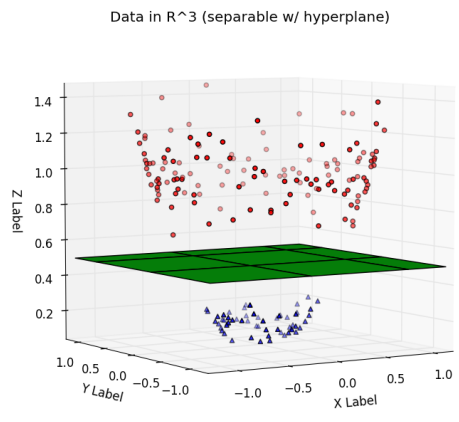
Okazuje się, że są to warunki wystarczające:

**Theorem (Twierdzenie Mercera)**

*Jeśli* $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ *jest symetryczna i dodatnio półokreślona, to istnieje przetrzeń Hilberta* $\mathcal{F}$ *oraz funkcja mapująca* $\phi$*, dla których zachodzi:* $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$

# Maszyna wektorów nośnych



Zalety liczenia bezpośrednio na funkcji kernelowej:

- nie musimy liczyć (ani nawet szukać) przetrzeni Hilberta i funkcji mapującej - liczenie kernela jest bezpośrednie i przez to efektywniejsze
- dla niektórych kerneli szukane przetrzenie Hilberta mogą być nieskończenie wymiarowe
- czasem jest prościej podać kernel, niż funkcję mapującą
- * zdarza się, że kernel nie jest zawsze dodatnio półokreślony - wystarczy, by był dla naszych danych

# Przykłady

Oznaczenia:
$n_i = |V_i|$, $m_i = |E_i|$
$A$, $B$ - zbiory etykiet dla wierzchołków i krawędzi
$\phi_{count} : \mathcal{G} \mapsto \mathbb{R}^{|A|+|B|}$

Przykładowe kernele:

- $k(G_1, G_2) = n_1 \cdot n_2 + m_1 \cdot m_2$, $n_i = |V_i|$, $m_i = |E_i|$
- $k(G_1, G_2) = \langle \phi_{count(G_1)}, \phi_{count(G_2)} \rangle$,

# Odległość edycyjna

# Odległość edycyjna

Ciekawszym przykładem może być funkcja kernelowa oparta na odległości edycyjnej. Niech $x_0$ będzie ustalonym „wzorcem” (elementem z przestrzeni grafów), a $d$ odległością edycyjną. Zdefiniujmy kernel:

$$k(x, x') = k_{x_o}(x, x') = \frac{1}{2}(d(x, x_0)^2 + d(x_0, x')^2 - 2 \cdot d(x, x')^2) \qquad (2)$$

# Odległość edycyjna

Ciekawszym przykładem może być funkcja kernelowa oparta na odległości edycyjnej. Niech $x_0$ będzie ustalonym „wzorcem" (elementem z przestrzeni grafów), a $d$ odległością edycyjną. Zdefiniujmy kernel:

$$k(x, x') = k_{x_o}(x, x') = \frac{1}{2}(d(x, x_0)^2 + d(x_0, x')^2 - 2 \cdot d(x, x')^2) \tag{2}$$

Kernel jest dobrze zdefiniowany, wobec tego istnieje przestrzeń Hilberta $\mathcal{F}$ i funkcja mapująca $\phi$. Zauważmy, że dla dowolnych wektorów $x, x' \in \mathcal{F}$ zachodzi:

$$\begin{aligned} \|\phi(x) - \phi(x')\|^2 &= \langle \phi(x) - \phi(x'), \phi(x) - \phi(x') \rangle \\ &= \langle \phi(x), \phi(x) \rangle + \langle \phi(x'), \phi(x') \rangle - 2 \cdot \langle \phi(x), \phi(x') \rangle \\ &= k(x, x) + k(x', x') - 2k(x, x') \\ &= d(x, x')^2 - \frac{1}{2}(d(x, x)^2 + d(x', x')^2) \\ &= d(x, x')^2 \end{aligned} \tag{3}$$

**Definition 1** *Given real matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{p\times q}$, the Kronecker product $A \otimes B \in \mathbb{R}^{np\times mq}$ and column-stacking operator $\mathrm{vec}(A) \in \mathbb{R}^{nm}$ are defined as*

$$A \otimes B := \begin{bmatrix} A_{11}B & A_{12}B & \dots & A_{1m}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{n1}B & A_{n2}B & \dots & A_{nm}B \end{bmatrix}, \quad \mathrm{vec}(A) := \begin{bmatrix} A_{*1} \\ \vdots \\ A_{*m} \end{bmatrix},$$

*where $A_{*j}$ denotes the $j^{\text{th}}$ column of $A$.*

The Kronecker product and vec operator are linked by the well-known property (e.g., Bernstein, 2005, Proposition 7.1.9):

$$\mathrm{vec}(ABC) = (C^\top \otimes A)\,\mathrm{vec}(B). \tag{1}$$

Another well-known property of the Kronecker product which we make use of is (Bernstein, 2005, Proposition 7.1.6):

$$(A \otimes B)(C \otimes D) = AC \otimes BD. \tag{2}$$

Finally, the Hadamard product of two real matrices $A, B \in \mathbb{R}^{n\times m}$, denoted by $A \odot B \in \mathbb{R}^{n\times m}$, is obtained by element-wise multiplication. It interacts with the Kronecker product via

$$(A \otimes B) \odot (C \otimes D) = (A \odot C) \otimes (B \odot D). \tag{3}$$

All the above concepts can be extended to a Reproducing Kernel Hilbert Space (RKHS) (See Appendix A for details).

# Oznaczenia

Przez graf $G$ rozumiemy uporządkowany zbiór $n$ wierzchołków $V = \{v_1, \ldots, v_n\}$, i zbiór skierowanych krawędzi $E \subset V \times V$, przy czym nie dopuszczamy krawędzi do tego samego wierzchołka.

Spacer o długości $k$ na $G$ rozumiemy jako ciąg takich indeksów $i_0, \ldots, i_k$, że $v_{i_{r-1}} \sim v_{i_r}$.

Dopuszczamy wagi krawędzi, tzn z każdą krawędzią skojarzona jest waga $w_{ij}$, interpretowana jako „siła" połączenia. Jeśli krawędź nie istnieje, waga musi być równa zero.

Przez $\tilde{A}$ oznaczamy macierz sąsiedztwa. Dla grafów nieważonych $\tilde{A}_{ij} = 1$ jeśli krawędź istnieje (i 0 w przeciwny przypadku). Dla ważonych przyjmujemy $\tilde{A}_{ij} = w_{ij}$. Z kolei przez $A$ oznaczamy znormalizowaną macierz sąsiedztwa - $A = \tilde{A}D^{-1}$, gdzie $D$ jest diagonalną macierzą stopni wierzchołka.

# Random Walk

Spacer losowy rozumiemy jako ciąg wierzchołków generowany zgodnie z prawdopodobieństwem znormalizowanej macierzy przejścia $A$. Wówczas $A^t$ opisuje spacery o długości $t$. Jeśli $p_0$ jest początkowym rozkładem prawdopodobieństwa na wierzchołkach, to $p_t = A^t p_0$ opisuje rozkład położenia po $t$ krokach.

Spacer losowy nie musi iść w nieskończoność - aby zamodelować to z każdym wierzchołkiem $v_{i_k}$ skojarzmy prawdopodobieństwo zatrzymania $q_{i_k}$. Nasz losowy spacer uwzględniając możliwość zatrzymania wyraża się teraz przez $q^\top p_t$

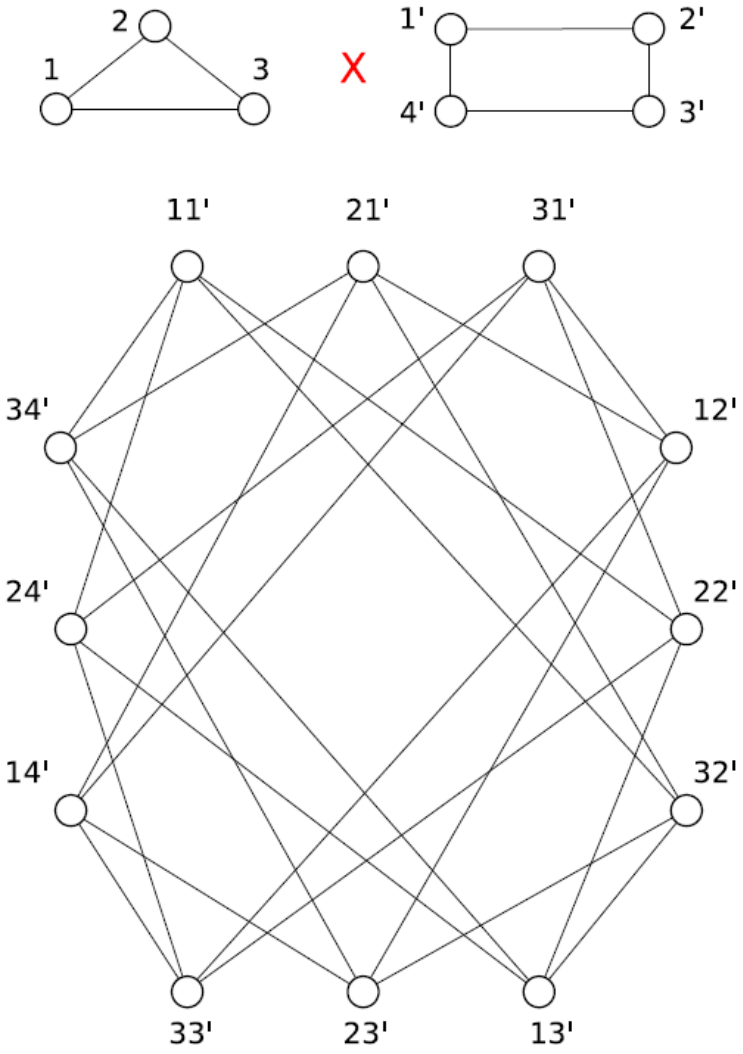
Niech $\mathcal{X}$ będzie zbiorem etykiet, se specjalną etykietą $\zeta$. Dla grafu etykietowanego (rozważamy wyłącznie etykiety na krawędziach) kojarzymi z nim macierz $X \in \mathcal{X}^{n \times n}$, gdzie wartości macierzy odpowiadają etykiecie krawędzi (lub $\zeta$, gdy tej krawędzi nie ma). Niech $k \colon \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ będzie funkcją kernelową dla etykiet. Wówczas przez $\phi$ oznaczamy funkcję mapującą $\mathcal{X}$ na pewną przeztrzeń Hilberta (dla której $\zeta$ przechodzi na 0). Z kolei przez $\Phi(X)$ oznaczamy macierz cech dla grafu $G$.

Mając dwa grafy $G(V, E)$ i $G'(V', E')$ zdefiniujmy ich iloczyn $G_\times$ jako graf ze zbiorem wierzchołków:

$$V_\times = \{(v_i, v'_r) : v_i \in V, v'_r \in V'\} \tag{4}$$

i zbioru krawędzi:

$$E_\times = \{((v_i, v'_r), (v_j, v'_s)) : (v_i, v_j) \in E, (v'_r, v'_s) \in E'\} \tag{5}$$


2
1
3
X
1'
2'
4'
3'
11'
21'
31'
34'
12'
24'
22'
14'
32'
33'
23'
13'

Dla $\bar{A}$ i $\bar{A}'$ będących macierzami sąsiedztwa dla odpowiednio $G$ i $G'$ definiujemy macierz sąsiedztwa $G_\times$ - $\bar{A_\times} = \bar{A} \otimes \bar{A}'$. Analogicznie dla znormalizowanej macierzy sąsiedztwa $A_\times = A \otimes A'$

Dla $\bar{A}$ i $\bar{A}'$ będących macierzami sąsiedztwa dla odpowiednio $G$ i $G'$ definiujemy macierz sąsiedztwa $G_\times$ - $\bar{A_\times} = \bar{A} \otimes \bar{A}'$. Analogicznie dla znormalizowanej macierzy sąsiedztwa $A_\times = A \otimes A'$

Wykonywanie spaceru losowego na grafie $G_\times$ jest równoważne wykonaniu niezależnie spaceru losowego na każdym z tych grafów. Wobec tego zdefiniujmy $p_\times = p \otimes p'$ oraz $q_\times = q \otimes q'$

W przypadku grafów etykietowanych możemy skojarzyć macierz wag $W_\times \in \mathbb{R}^{nn' \times nn'}$ z $G_\times$ rozszerzając iloczyn Kroneckera:

$$W_\times = \Phi(X) \otimes \Phi(X') \tag{6}$$

W przypadku, gdy zbiór etykiet jest skończony, bez straty ogólności $\{1, 2, \ldots, d\}$, możemy pozwolić, by przestrzeń Hilberta $\mathcal{H}$ była przestrzenią $R^d$ ze standardowym iloczynem skalarnym. Dla każdej krawędzi $(v_j, v_i) \in E$ ustalamy $\phi(X_{ij}) = \frac{e_l}{d_i}$, jeśli krawędź ma etykietę $l$, pozostałe wartości ustawiamy na 0. Ostatecznie (pomijając dowód) możemy macierz wag iloczynu grafów przedstawić jako:

$$W_\times = \sum_{l=1}^{d} {}^lA \otimes {}^lA', \tag{7}$$

gdzie ${}^lA = A_{ij}$ gdy $X_{ij} = l$ i zero w przeciwnym przypadku.

# Random Walk Kernel

$W_\times^k$ reprezentuje podobieństwo pomiędzy ścieżkami (o długości $k$). Dodając początkowe prawdopodobieństwo i prawdopodonieństwo zatrzymania możemy obliczyć oczekiwane podobieństwo ścieżek $q_\times^\top W_\times^k p_\times$. Ostetcznie pomysłem na kernel jest zsumowanie dla wszystkich długości ścieżek:

$$k(G, G') = \sum_{k=0}^{\infty} \mu(k) q_\times^\top W_\times^k p_\times \tag{8}$$

Jednak zwykłe zsumowanie mogłoby nie być zbieżne, dlatego dodaje się nieujmeny współczynnik $\mu(k)$. Dzięki niemu można zagwarantować sobie zbieżność, ale też daje elastyczność - możemy bardziej podkreślić znaczenie znaczenie ścieżek o danej długości.

# vec-trick

By udowodnić, że kernel jest dobrze zdefiniowany, pokażmy najpierw:

**Lemma 12** *If $A \in \mathcal{X}^{n\times m}$, $B \in \mathbb{R}^{m\times p}$, and $C \in \mathcal{X}^{p\times q}$, then*

$$\operatorname{vec}(\Phi(A)\,B\,\Phi(C)) = (\Phi(C)^\top \otimes \Phi(A))\operatorname{vec}(B) \;\in \mathbb{R}^{nq\times 1}.$$

**Proof** We begin by rewriting the $k^{\text{th}}$ column of $\Phi(A)B\Phi(C)$ as

$$\begin{aligned}
[\Phi(A)B\Phi(C)]_{*k} &= \Phi(A)\sum_j B_{*j}\,\phi(C_{jk}) = \sum_j \phi(C_{jk})\Phi(A)B_{*j} \\
&= [\phi(C_{1k})\Phi(A), \phi(C_{2k})\Phi(A), \ldots \phi(C_{nk})\Phi(A)] \underbrace{\begin{bmatrix} B_{*1} \\ B_{*2} \\ \vdots \\ B_{*n} \end{bmatrix}}_{\operatorname{vec}(B)} \\
&= ([\phi(C_{1k}), \phi(C_{2k}), \ldots \phi(C_{nk})] \otimes \Phi(A))\operatorname{vec}(B). \qquad (70)
\end{aligned}$$

To obtain Lemma 12 we stack up the columns of (70):

$$\begin{aligned}
\operatorname{vec}(\Phi(A)\,B\,\Phi(C)) &= \left( \begin{bmatrix} \phi(C_{11}) & \phi(C_{21}) & \ldots & \phi(C_{n1}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(C_{1n}) & \phi(C_{2n}) & \ldots & \phi(C_{nn}) \end{bmatrix} \otimes \Phi(A) \right) \operatorname{vec}(B) \\
&= (\Phi(C)^\top \otimes \Phi(A))\operatorname{vec}(B). \qquad \blacksquare
\end{aligned}$$

Używając powyższego lematu, pokażemy, że:

$$\forall k \in \mathbb{N}\colon W_{\times}^{k} p_{\times} = vec[\Phi(X')^{k} p' (\Phi(X)^{k} p)^{\top}] \tag{9}$$

Dowód przeprowadzimy indukcyjny. Dla $k = 0$ mamy:

$$W_{\times}^{0} p_{\times} = p_{\times} = (p \otimes p') vec(1) = vec(p' 1 p^{\top}) = vec[\Phi(X')^{0} p' (\Phi(X)^{0} p)^{\top}] \tag{10}$$

Załóżmy, że $W_{\times}^{k} p_{\times} = vec[\Phi(X')^{k} p' (\Phi(X)^{k} p)^{\top}]$. Używając vec-tricku:

$$\begin{aligned} W_{\times}^{k+1} p_{\times} = W_{\times} W_{\times}^{k} p_{\times} &= (\Phi(X) \times \Phi(X')) vec[\Phi(X')^{k} p' (\Phi(X)^{k} p)^{\top}] \\ &= vec[\Phi(X') \Phi(X')^{k} p' (\Phi(X)^{k} p)^{\top} \Phi(X)^{\top}] \\ &= vec[\Phi(X')^{k+1} p' (\Phi(X)^{k+1} p)^{\top}] \end{aligned} \tag{11}$$

Teraz możemy pokazać, że jeśli w naszej definicji kernela $\mu(k)$ jest takie, że formuła jest zbieżna, to jest to dobrze zdefiniowany kernel:

$$\begin{aligned} q_\times^\top W_\times^k p_\times &= (q \otimes q')^\top vec[\Phi(X')^k p' (\Phi(X)^k p)^\top] \\ &= vec[q'^\top \Phi(X')^k p' (\Phi(X)^k p)^\top q] \\ &= (q^\top \Phi(X)^k p)^\top (q'^\top \Phi(X')^k p') \\ &= f_k(G)^\top f_k(G') \end{aligned} \tag{12}$$

Otrzymujemy, że pojedynczy składnik jest symetryczny i dodatnio półokreślony. Ponieważ funkcje kernelowe są zamknięte ze względu na liniowe kombinacje i granice punktowe, otrzymujemy tezę.

# Szczególne przypadki

Możemy spróbować zdefiniować kernel na grafach etykietowanych (krawędziowo), ale poprzez spacery i ich ciągi etykiet. Spacerem o długości $t$ na grafie $G$ jest ciąg $i_1, \ldots, i_{t+1}$ taki, że $v_k \sim v_{k+1}$. Niech $h = h_1, \ldots, h_t$ będzie ciągiem etykiet krawędzi, natomiast niech $P_{ij}$ będzie macierzą przejścia z wierzchołka $v_i$ do wierzchołka $v_j$, a $p$ i $q$ niech będą prawdopodobieństwami początkowym i stopu. Wówczas prawdopdobieństwo ścieżki $h$ jest równe:

$$p(h|G) = q_{i_{t+1}} \prod_{j=1}^{t} P_{i_j, i_{j+1}} p_{i_1} \tag{13}$$

Następnie potrzebujemy zdefiniować kernel dla dwóch różnych ścieżek (gdzie $\phi$ oznacza funkcję przypisującą etykiety krawędziom):

$$k(h, h') = \prod_{i=1}^{t} k(h_i, h'_i) = \prod_{i=1}^{t} \langle \hat{\phi}(h_i), \hat{\phi}(h'_i) \rangle \tag{14}$$

jeśli $h$ i $h'$ mają tę samą długość (i 0 w przeciwnym przypadku).

Teraz możemy nasz kernel zdefiniować jako:

$$k(G, G') = \sum_{h} \sum_{h'} \mathcal{k}(h, h') p(h|G) p(h'|G') \tag{15}$$

Ostatecznie można zapisać to w formie (bez dowodu):

$$k(G, G') = q_{\times}^{\top} (I - T_{\times})^{-1} p_{\times}, \tag{16}$$

gdzie $T_{\times} = [vec(P)vec(P')^{\top}] \odot [\hat{\Phi}(X) \otimes \hat{\Phi}(X')]$.

Okazuje się, że również tę formułę można zapisać przy pomocy zdefiniowanego wcześniej kernela. Załóżmy, że $\mu(k) = \lambda^k$ dla pewnego $\lambda > 0$. Wówczas możemy zapisać:

$$k(G, G') = \sum_{k=0}^{\infty} \lambda^k q_{\times}^{\top} W_{\times}^k p_{\times} = q_{\times}^{\top} (I - \lambda W_{\times})^{-1} p_{\times} \tag{17}$$

Przyjmując $\lambda = 1$ i definiując $\Phi(X_{ij}) = P_{ij} \Phi(\hat{X}_{ij})$, (co oznacza, że $W_{\times} = T_{\times}$) otrzymujemy równoważność definicji.

Innym szczególnym przypadkiem jest kernel zliczający identyczne ścieżki:

$$k(G, G') = \sum_{i=1}^{n} \sum_{j=1}^{n'} \sum_{k=0}^{\infty} \lambda^k [\tilde{A}_{\times}^k]_{ij} \tag{18}$$

Aby uzyskać ten kernel w naszym frameworku podstawiamy $\mu(k) = \lambda^k$, zakładamy jednostajny rozkład startu i stopu ($p_i = q_i = \frac{1}{n}$), i niech $\Phi(X) = \tilde{A}$ oraz $\Phi(X') = \tilde{A}'$. Ostatecznie otrzymujemy:

$$k(G, G') = q_{\times}^{\top} (I - T_{\times})^{-1} p_{\times} = \frac{1}{n^2 \cdot n'^2} \sum_{i=1}^{n} \sum_{j=1}^{n'} \sum_{k=0}^{\infty} \lambda^k [\tilde{A}_{\times}^k]_{ij} \tag{19}$$

Innym szczególnym przypadkiem jest kernel zliczający identyczne ścieżki:

$$k(G, G') = \sum_{i=1}^{n} \sum_{j=1}^{n'} \sum_{k=0}^{\infty} \lambda^k [\tilde{A}_{\times}^k]_{ij} \tag{18}$$

Aby uzyskać ten kernel w naszym frameworku podstawiamy $\mu(k) = \lambda^k$, zakładamy jednostajny rozkład startu i stopu ($p_i = q_i = \frac{1}{n}$), i niech $\Phi(X) = \tilde{A}$ oraz $\Phi(X') = \tilde{A}'$. Ostatecznie otrzymujemy:

$$k(G, G') = q_{\times}^{\top} (I - T_{\times})^{-1} p_{\times} = \frac{1}{n^2 \cdot n'^2} \sum_{i=1}^{n} \sum_{j=1}^{n'} \sum_{k=0}^{\infty} \lambda^k [\tilde{A}_{\times}^k]_{ij} \tag{19}$$

W pracy wyróżnione są dwa przypadki doboru $\mu(k)$:

- $\mu(k) = \lambda^k$
- $\mu(k) = \frac{\lambda^k}{k!}$

# Złożoność

Naiwne podejście dla kernelu geometrycznego ($\mu(k) = \lambda^k$) wymaga odwrócenia macierzy $(I - \lambda W_\times)$. Złożoność takiej operacji jest sześcienna od wymiaru macierzy. Jako, że nasza wyjściowa macierz jest wymiaru $n^2 \times n^2$, to całość obliczeń ma złożoność $O(n^6)$.

# Metoda równania Sylvestera

Równanie Sylvestera jest postaci:

$$M = SMT + M_0 \tag{20}$$

gdzie w naszym przypadku $S, T, M_0 \in \mathbb{R}^{n \times n}$. Takie równania są rozwiązywane w czasie $O(n^3)$. Niestety uogólnione równanie Sylvestera:

$$M = \sum_{i=1}^{d} S_i M T_i + M_0 \tag{21}$$

wymaga rozkładu Shura $d$ macierzy, co jest bardziej kosztowne (złożoność nie jest znana).

Nasz problem możemy sprowadzić do powyższego problemu:

$$M = \sum_{i=1}^{d} \lambda^i A' M^i A^\top + M_0, \tag{22}$$

gdzie $vec(M_0) = p_\times$. Dalej mamy:

$$vec(M) = \lambda \sum_{i=1}^{d} vec({}^i A' M^i A^\top) + p_\times \tag{23}$$

Używając vec-tricku otrzymujemy:

$$(I - \lambda \sum_{i=1}^{d} {}^i A \otimes {}^i A') vec(M) = p_\times \tag{24}$$

oraz podstawiając $W_\times = \sum_{l=1}^{d} {}^l A \otimes {}^l A'$:

$$\begin{aligned} vec(M) &= (I - \lambda W_\times)^{-1} p_\times \\ q_\times^\top vec(M) &= q_\times^\top (I - \lambda W_\times)^{-1} p_\times \end{aligned} \tag{25}$$

Dla grafów nieetykietowanych otrzymujemy złożoność $O(n^3)$.

# Metoda gradientu sprzężonego

Używając gradientu sprzężonego umiemy rozwiązywać równania

$$Mx = b \tag{26}$$

o ile macierz $M$ jest symetryczna i dodatnio określona.

Nasz problem można rozwiązać wpierw rozwiązując:

$$(I - \lambda W_\times)x = p_\times, \tag{27}$$

a następnie obliczając $xq_\times^\top$. Naiwne rozwiązanie powyższego równania jednak nadal wymaga $O(n^4)$ operacji, jako że $W_\times$ ma wymiar $n^2 \times n^2$.

Wprowadzając macierz $Y \in \mathbb{R}^{n \times n}$ i $y = vec(Y)$, oraz używając vec-tricku możemy zapisać:

$$W_{\times} y = (\Phi(X) \otimes \Phi(X'))vec(Y) = vec(\Phi(X')Y \otimes \Phi(X)^{\top}) \tag{28}$$

Wówczas jeśli $\phi(\cdot) \in \mathbb{R}^d$ możemy policzyć powyższy iloczyn w czasie $O(dn^3)$. Jeśli $Phi(X)$ i $Phi(X')$ są rzadkie, wówczas można policzyć jeszcze szybciej ($O(n^2)$, jeśli $O(n)$ jest niezerowych elementów w $Phi(X)$ i $Phi(X')$).

# Metoda punktu stałego

Nasz kernel można zapisać do postaci:

$$x = p_\times + \lambda W_\times x \tag{29}$$

Znalezienie $x$ jest równoważne znalezieniu punktu stałego.
Ustawiamy $x_0 = p_\times$, a następnie obliczamy:

$$x_{t+1} = p_\times + \lambda W_\times x_t, \tag{30}$$

dopóki $\|x_{t+1} - x_t\| > \epsilon$

# Metoda dekompozycji spektralnej

Niech $W_\times = P_\times D_\times P_\times^{-1}$ będzie takim rozkładem $W_\times$, że kolumny $P_\times$ są jej wektorami własnymi, a $D_\times$ macierzą diagonalną o wartościach będącymi wartościami własnymi. Kernel błądzenia losowego możemy zapisać:

$$k(G, G') = \sum_{k=0}^{\infty} \mu(k) q_\times^\top (P_\times D_\times P_\times^{-1})^k p_\times = q_\times^\top P_\times (\sum_{k=0}^{\infty} \mu(k) D_\times^k) P_\times^{-1} p_\times \tag{31}$$

Co oznacza, że do policzenia kernelu wystarczy policzyć potęgę diagonalnej macierzy. Biorąc odpowiednie $\mu(k)$ obliczenia się trywializują

Dla $\mu(k) = \lambda^k$ otrzemujemy:

$$k(G, G') = \sum_{k=0}^{\infty} \mu(k) q_{\times}^{\top} (P_{\times} D_{\times} P_{\times}^{-1})^k p_{\times} = q_{\times}^{\top} P_{\times} (\sum_{k=0}^{\infty} \mu(k) D_{\times}^k) P_{\times}^{-1} p_{\times} \tag{32}$$

Podobny wynik otrzymujemy dla wykładniczego kernela $\mu(k) = \frac{\lambda^k}{k!}$.
Problemem jest to, że samo znalezienie rozkładu gęstej macierzy $W_{\times}$, która jest o wymiarach $n^2 \times n^2$, zajmuje $O(n^6)$ czasu.
Powyższy wynik można jednak poprawić dla grafów nieetykietowanych.

# Metoda przybliżenia iloczynu Kroneckera

Ideą jest znalezienie macierzy $S$ i $T$ takich, że $W_\times \approx S \otimes T$, a następnie użyć jednej z powyższych metod dla $S \otimes T$ przyjmując, że jest to macierz sąsiedztwa grafu nieetykietowanego.
Do tego celu liczy się normę Frobeniusa $\|W_\times - S \otimes T\|_F$ licząc największe wartości własne permutacji $W_\times$.

# Porównanie efektywności

# Porównanie efektywności

| sparsity | dense | | | | sparse |
|---|---|---|---|---|---|
| edge labels | none/scalar | finite set | finite-dim. | ∞-dim. | any |
| Method (Section) $W_\times =$ | $A \otimes A'$ | (7) | kernel (6) | | |
| Sylvester Equation (4.1) | $m^2n^3$ | unknown | — | | — |
| Conjugate Gradient (4.2) | $m^2rn^3$ | $m^2rdn^3$ | | $m^2rn^4$ | $m^2rn^2$ |
| Fixed-Point Iterations (4.3) | $m^2kn^3$ | $m^2kdn^3$ | | $m^2kn^4$ | $m^2kn^2$ |
| Spectral Decomposition (4.4) | $(m+n)mn^2$ | $m^2n^6$ | | | — |
| Nearest Kron. Product (4.5) | 1 | $m^2k'dn^2$ | $m^2k'n^4$ | | $m^2k'n^2$ |

Table 1: Worst-case time complexity (in $O(\cdot)$ notation) of our methods for an $m \times m$ graph kernel matrix, where $n$ = size of the graphs (number of nodes), $d$ = size of label set *resp.* dimensionality of feature map, $r$ = effective rank of $W_\times$, $k$ = number of fixed-point iterations (31), and $k'$ = number of power iterations (37).

# Porównanie efektywności

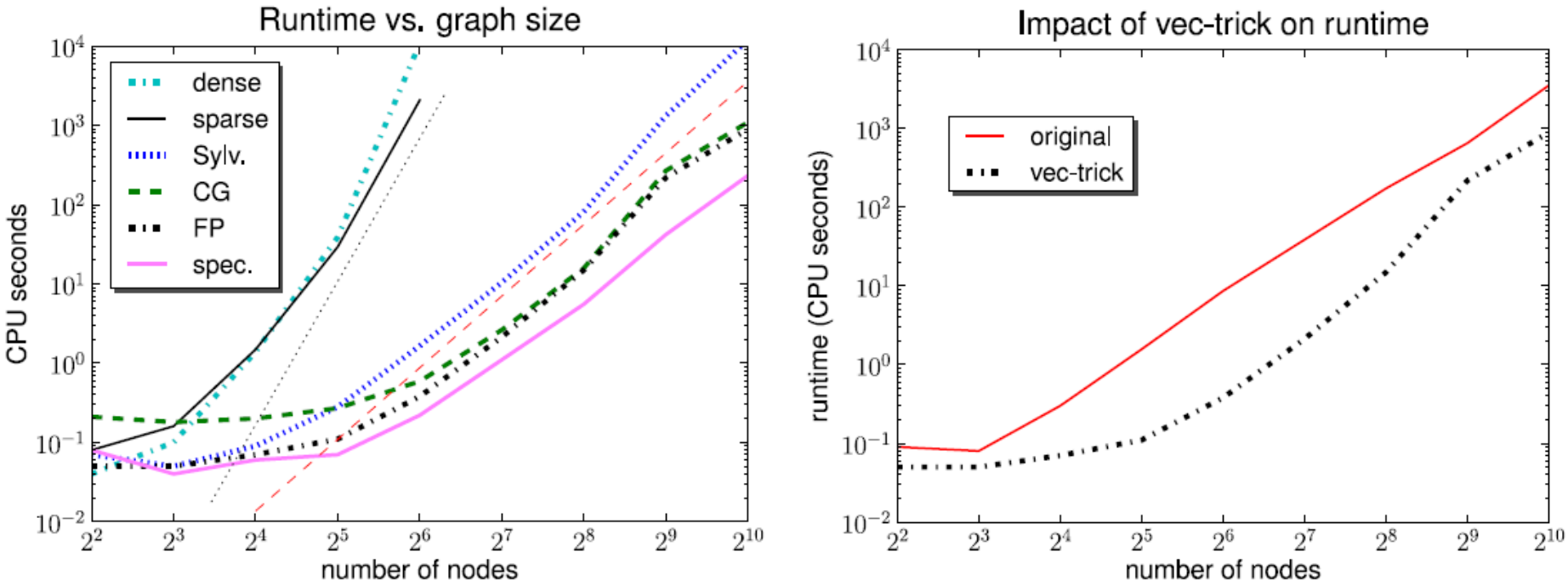



Figure 3: Time to compute a $10 \times 10$ kernel matrix on random graphs with $n$ nodes and $3n$ edges as a function of the graph size $n$. Left: The Sylvester equation (Sylv.), conjugate gradient (CG), fixed-point iteration (FP), and spectral decomposition (spec.) approaches, compared to the dense and sparse direct method. Thin straight lines indicate $O(n^6)$ (black dots) *resp.* $O(n^3)$ (red dashes) scaling. Right: Kashima et al.'s (2004) fixed-point iteration (original) compared to our version, which exploits Lemma 12 (vec-trick).

# Porównanie efektywności

VISHWANATHAN, SCHRAUDOLPH, KONDOR AND BORGWARDT

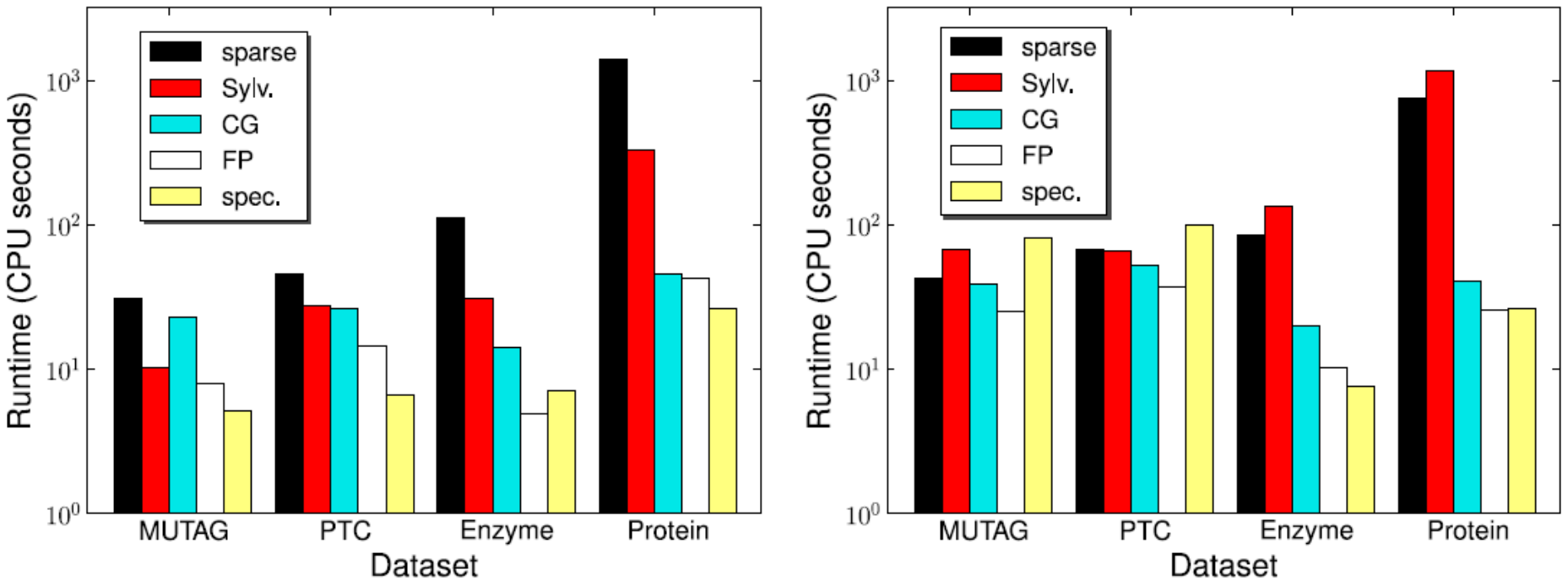



Figure 5: Time (in seconds on a log-scale) to compute $100 \times 100$ kernel matrix for unlabeled (left) *resp.* labeled (right) graphs from several data sets, comparing the conventional sparse method to our fast Sylvester equation, conjugate gradient (CG), fixed-point iteration (FP), and spectral approaches.

Dziękuję za uwagę.

Dziękuję za uwagę.

Literatura:


- S.V.N. Vishwanathan i in. - „Graph Kernels”, 2008
- Swarnendu Ghosh i in. - „The journey of graph kernels through two decades”, 2018